Основы параллельного программирования с использованием MPI

Лекция 2

Немнюгин Сергей Андреевич

Санкт-Петербургский государственный университет

физический факультет

кафедра вычислительной физики

snemnyugin@mail.ru

Интернет-Университет
Суперкомпьютерных Технологий

High-Performance Computing University

Лекция 2

Аннотация

Во второй лекции рассматривается «архитектура» программной реализации MPI на примере MPICH. Объясняется роль демона mpd. Вводятся основные понятия и терминология. Приводятся типовые схемы организации параллельных MPI-программ, их структура. Рассматриваются привязки к языкам программирования C и Fortran, а также способы компиляции и запуска MPI-программ.

План лекции

- Спецификация MPI. История создания, версии.
- Основные понятия, терминология.
- Состав пакета MPICH.
- Компиляция и запуск MPI-программ в MPI-1. Файлы конфигурации.
- Компиляция и запуск MPI-программ в MPI-2. Демон mpd. Файлы конфигурации.
- Структура MPI-программы.
- Простейшая MPI-программа.

Спецификация MPI. История создания. Версии

Спецификация MPI

Большой интерес к системам передачи сообщений возник в 1980-е годы. Его следствием стало появление достаточно большого количества реализаций, создававшихся разными коллективами разработчиков и предназначенных для разных вычислительных систем.

Примеры:
p4, PICL, PVM, TCGMSG, Express и другие

Возникла необходимость координировать и стандартизировать этот процесс, поэтому в рамках конференции Supercomputing’92 состоялось совещание, на котором было принято решение о разработке стандарта программного интерфейса обмена сообщениями.

Спецификация MPI

Процесс стандартизации был поддержан индустрией:
Convex, Cray, IBM, Intel, Meiko, nCUBE, NEC and Thinking Machines

Индустрия была заинтересована в средстве разработки эффективных программ для высокопроизводительных вычислительных систем. В жертву производительности были принесены устойчивость и синхронизация процессов.

Спецификация MPI

Сайт, на котором можно найти официальные документы MPI:
http://www.mpi-forum.org

Версия MPI-1 вышла в 1994 году

Версия MPI-2 вышла в 1998 году, первая реализация появилась в 2002 году.

Версия MPI-2.1 вышла в начале сентября 2008 года

Спецификация MPI-1

Содержит описание стандарта программного интерфейса обмена сообщениями. Спецификация учитывает опыт предшествующих разработок и ориентирована на большую часть аппаратных платформ. Несмотря на то, что MPI рассчитано на использование с языками C/C++ и Fortran, семантика в значительной степени не зависит от языка.

В MPI-1 описываются интерфейсы процедур двухточечного и коллективного обмена, сбора информации, организации обменов в группах процессов, синхронизации процессов, виртуальные топологии, привязки к языкам программирования и т. д.

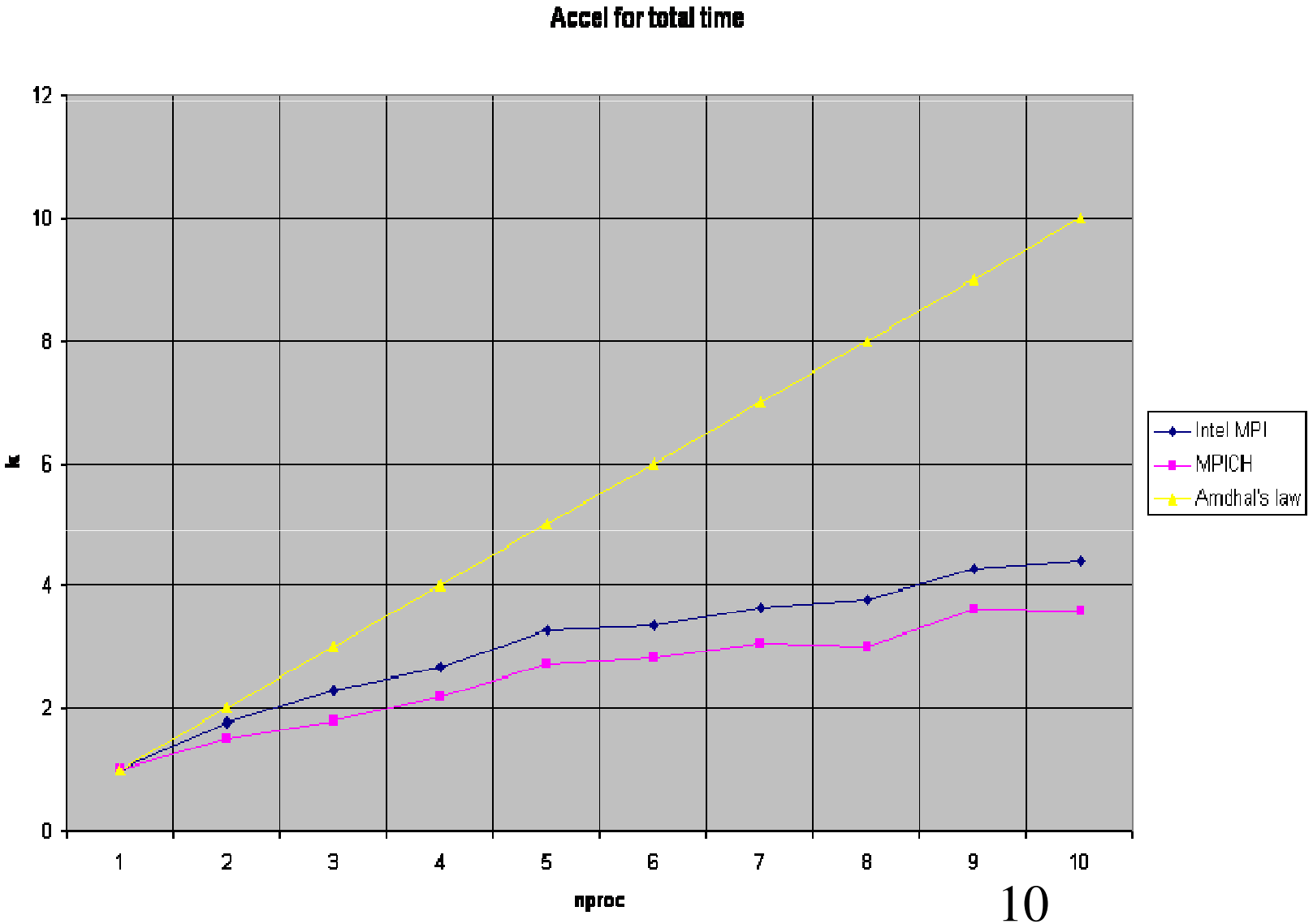
Спецификация MPI-2

Является дальнейшим развитием MPI. Новое в MPI-2:

- возможность создания новых процессов во время выполнения MPI-программы (в MPI-1 количество процессов фиксировано, крах одного приводит к краху всей программы);
- новые разновидности двухточечных обменов (односторонние обмены);
- новые возможности коллективных обменов;
- поддержка внешних интерфейсов;
- операции параллельного ввода-вывода с файлами.

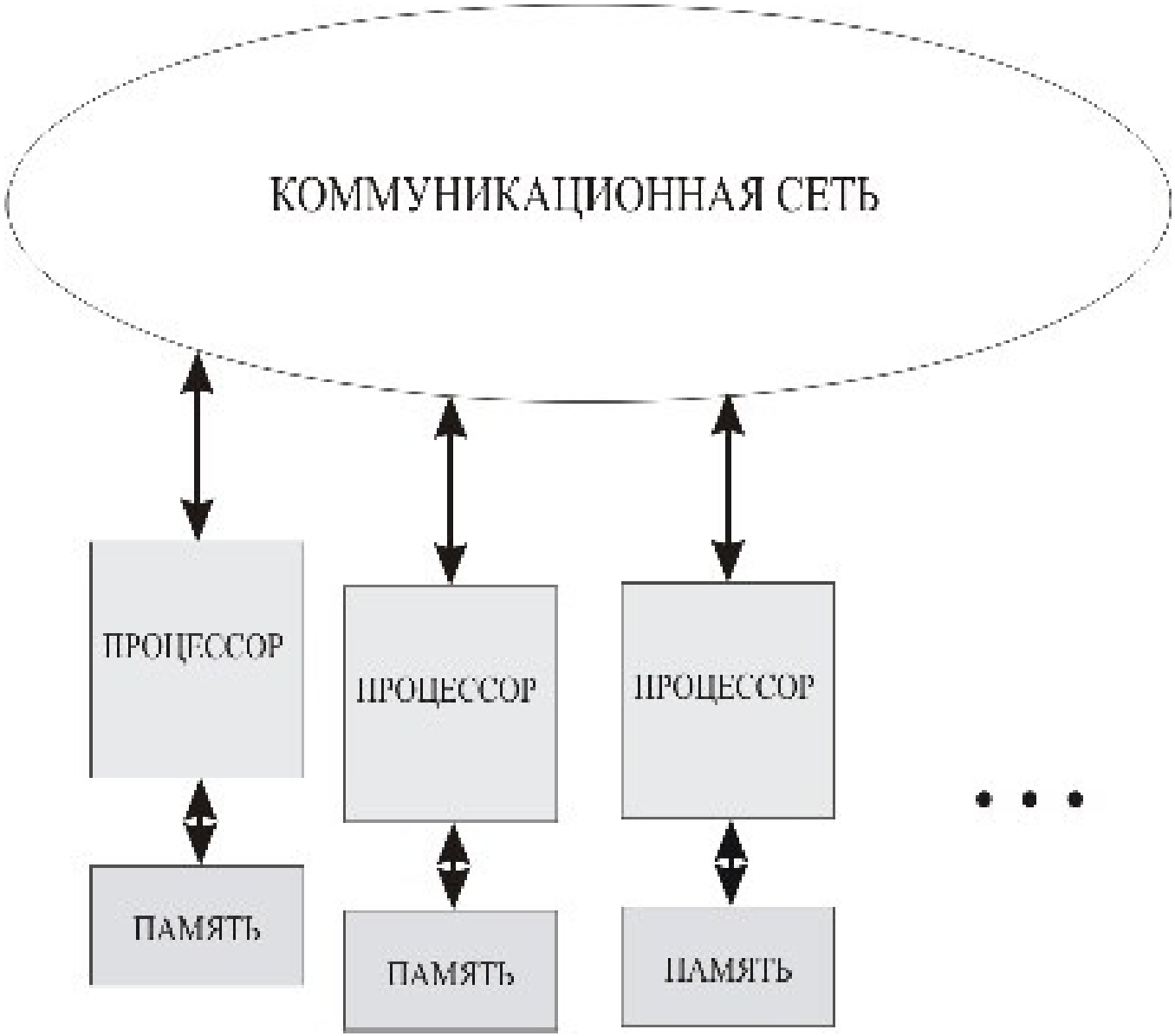
Масштабируемость MPI-программ. Сравнение двух реализаций

Зависимость ускорения параллельной версии кода ACE от количества узлов:



Основные понятия, терминология

Основные понятия, терминология



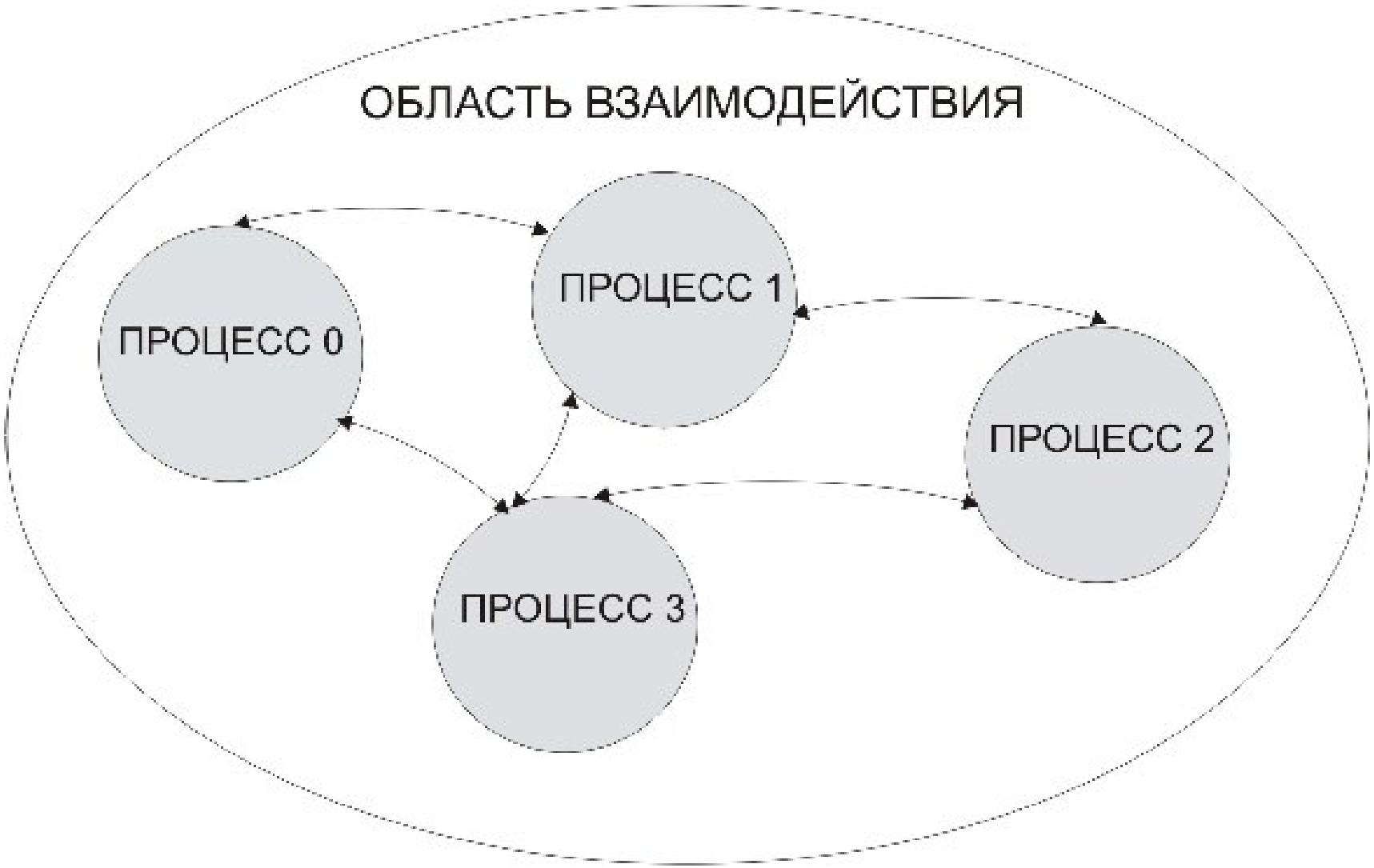
Основные понятия, терминология

Двухточечные обмены используются для организации локальных и неструктурированных коммуникаций. При выполнении глобальных операций используются коллективные обмены.

Асинхронные коммуникации реализуются с помощью запросов о получении сообщений.

Область взаимодействия (область связи) определяет группу процессов. Все процессы, принадлежащие одной области взаимодействия, могут обмениваться сообщениями. Описывает это множество процессов специальная информационная структура, которая называется *коммуникатором*. Этот механизм скрывает от программиста внутренние коммуникационные структуры.

Основные понятия, терминология



Основные понятия, терминология

Коммуникаторы, создаваемые по умолчанию:

- **MPI_COMM_WORLD** - содержит все процессы;
- **MPI_COMM_SELF** - коммуникатор, содержащий только вызывающий процесс;
- **MPI_COMM_NULL** - пустой коммуникатор.

Основные понятия, терминология

Сообщение содержит пересылаемые данные и служебную информацию:

- идентификатор процесса-отправителя сообщения (*ранг* процесса);
- адрес, по которому размещаются пересылаемые данные процесса-отправителя;
- тип пересылаемых данных;
- количество данных (размер буфера сообщения для того, чтобы принять сообщение, процесс должен отвести для него достаточный объем оперативной памяти);
- идентификатор процесса, который должен получить сообщение;
- адрес, по которому должны быть размещены данные процессом-получателем;
- идентификатор коммуникатора, описывающего область взаимодействия, внутри которой происходит обмен.

Основные понятия, терминология

Ранг источника дает возможность различать сообщения, приходящие от разных процессов

Тег - это задаваемое пользователем целое число от 0 до 32767, идентификатор сообщения.

Теги могут использоваться для соблюдения определенного порядка приема сообщений.

Основные понятия, терминология

Данные, содержащиеся в сообщении, в общем случае организованы в массив элементов, каждый из которых имеет один и тот же тип.

Кроме пересылки данных система передачи сообщений поддерживает пересылку информации о состоянии процессов коммуникации. Это может быть, например, уведомление о том, что прием данных, отправленных другим процессом, завершен.

Основные понятия, терминология

Перед использованием процедур передачи сообщений программа должна "подключиться" к системе обмена сообщениями.

Подключение выполняется с помощью соответствующего вызова процедуры из библиотеки. В одних реализациях модели допускается только одно подключение, а в других несколько подключений к системе.

Прием сообщения начинается с подготовки буфера достаточного размера. В этот буфер записываются принимаемые данные.

Данные могут быть закодированы, в этом случае при их считывании выполняется декодирование. Кодирование данных (преобразование в некоторый универсальный формат) требуется для того, чтобы обеспечить обмен сообщениями между системами с разным форматом представления данных.

Основные понятия, терминология

Операция отправки или приема сообщения считается завершенной, если программа может вновь использовать такие ресурсы, как буферы сообщений.

Состав пакета MPICH

Состав пакета MPICH

Структура каталога MPICH:

- **/usr/local/mpich/** - содержит файлы и подкаталоги MPICH;
- **/usr/local/mpich/bin/** - исполняемые файлы;
- **/usr/local/mpich/examples/** - примеры программ;
- **/usr/local/mpich/doc/** - документация по установке и работе с MPICH;
- **/usr/local/mpich/include/** - заголовочные файлы (в том числе **mpi.h** и **mpif.h**);
- **/usr/local/mpich/lib/** - библиотечные файлы;
- **/usr/local/mpich/src/** - исходные тексты системы;
- **/usr/local/mpich/man/** - справочные страницы по подпрограммам и утилитам MPICH.

Имеются и другие подкаталоги.

Компиляция и запуск программ в MPI-1

Файлы конфигурации

MPI-1

При запуске MPI-программ с использованием Remote Shell в ОС Linux следует создать файл **.rhosts**.

Пример:

- f01.ptc.spbu.ru
- f02.ptc.spbu.ru
- f03.ptc.spbu.ru
- f04.ptc.spbu.ru

Права доступа:

r w - - - - - - -

MPI-1

Файл **machines**.

Пример файла machines

f01.ptc.spbu.ru
f02.ptc.spbu.ru
f03.ptc.spbu.ru
f04.ptc.spbu.ru
fserver.ptc.spbu.ru:2

MPI-1

Трансляция

```
# mpicc -compile_info
cc -DUSE_STDARG -DHAVE_STDLIB_H=1 -DHAVE_STRING_H=1 -
    DHAVE_UNISTD_H=1 -DHAVE_STDARG_H=1 -DUSE_STDARG=1 -
    DMALLOC_RET_VOID=1 -I/usr/local/mpich/include –c

# mpicc -link_info
cc -DUSE_STDARG -DHAVE_STDLIB_H=1 -DHAVE_STRING_H=1 -
    DHAVE_UNISTD_H=1 -DHAVE_STDARG_H=1 -DUSE_STDARG=1 -
    DMALLOC_RET_VOID=1 -L/usr/local/mpich/lib –lmpich

# mpif77 -compile_info
f77 -I/usr/local/mpich/include –c

# mpif77 -link_info
f77 -L/usr/local/mpich/lib -lmpich
```

MPI-1

Выполнение

mpirun –np n [ключи MPI] программа [ключи и аргументы программы]

MPI-1

Основные ключи загрузчика приложений mpirun:

-arch <архитектура>
Архитектура параллельной вычислительной системы, которая должна соответствовать суффиксу в имени файла machines.<архитектура>

-machinefile файл
Использовать список компьютеров из указанного файла

Компиляция и запуск программ в MPI-2

Демон mpd

Файлы конфигурации

MPI-2

Демон mpd

В MPI-2 демон `mpd` играет важную роль. Параллельная программа может выполняться только если предварительно были запущены демоны `mpd`, образующие «кольцо демонов». Он управляет выполнением процессов MPI-программы на данном вычислительном узле. Демоны запускаются от имени конкретных пользователей ОС и не оказывают влияния друг на друга.

- Кольцо демонов создаётся один раз и может быть использовано многократно, разными программами одного пользователя.
- Кольцо демонов прекращает свой существование в результате выполнения команды `mpdallexit`.
- Кольцо демонов одного пользователя не может взаимодействовать с демонами `mpd` других пользователей.
- Благодаря демонам `mpd` запуск MPI-программ выполняется быстрее.
- Локальный экземпляр MPI-процесса завершается по нажатию клавиш Ctrl+C, при этом завершаются и все остальные процессы.
- Исключена возможность «зависания» процессов.

В программных реализациях MPI имеются средства управления работой демонов mpd.
Запуску параллельной программы предшествует запуск демона mpd на всех узлах вычислительной системы. Демоны взаимодействуют друг с другом.

Запуск демонов (в этом примере 5) выполняется командой:

```
mpdboot -n 5
```

Проверка взаимодействия демонов между собой выполняется командой:

```
mpdtrace
```

Если при выполнении этой команды выводятся сообщения об ошибках, это говорит о неправильной настройке MPI или локальной сети.

Завершение работы демонов выполняется командой:

```
mpdallexit
```

MPI-2

Конфигурационный файл .mpd.conf

Пример файла **.mpd.conf**

MPD_SECRETWORD=kalosha
MPD_PORT_RANGE=1003:10003

Файл mpd.hosts

Пример файла **.mpd.conf**

pd00 ifhn=195.168.0.69
pd01 ifhn=192.168.0.74
pd02 ifhn=192.168.0.75
pd03 ifhn=192.168.0.76
pd04 ifhn=192.168.0.77
pd05
pd06
pd07

MPI-2

Запуск демонов

mpdboot –n *число_демонов* [ключи]

Запуск параллельной программы

mpiexec [ключи] **–n *число имя_исполняемого файла***

Завершение работы всех демонов

mpdallexit

Привязки к языкам программирования

Привязка к языку C

При использовании MPI в программах на языке C в именах функций используется префикс MPI_, а первая буква имени набирается в верхнем регистре.

Согласно спецификации имена подпрограмм имеют вид Класс_действие_подмножество или Класс_действие. Аналогичное правило действует и для подпрограмм MPI для языка Forttran. В C++ подпрограмма является методом для определенного класса, имя имеет в этом случае вид MPI::Класс::действие_подмножество.
Для некоторых действий введены стандартные наименования:

- **Create** — создание нового объекта;
- **Get** — получение информации об объекте;
- **Set** — установка параметров объекта;
- **Delete** — удаление информации;
- **Is** — запрос о том, имеет ли объект указанное свойство.

Привязка к языку C

Значения кода завершения имеют целый тип и определяются по значению функции. Имена констант MPI записываются в верхнем регистре. Их описания находятся в заголовочном файле **mpi.h**, который обязательно включается в MPI-программу. Имя этого файла может быть другим в других реализациях MPI. Входные параметры функций передаются по значению, а выходные (и INOUT) — по ссылке.

Привязка к языку C

В MPI принята своя система обозначения типов данных, которая соответствует типам данных в языках C и Fortran (соответствие неполное).

Тип данных MPI	Тип данных C
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	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	Нет соответствия
MPI_PACKED	Нет соответствия

Привязка к языку Fortran

Все имена подпрограмм и констант MPI начинаются с MPI_.
Коды завершения передаются через дополнительный параметр целого типа (находится на последнем месте в списке параметров подпрограммы).
Код успешного завершения — MPI_SUCCESS.

Константы и другие объекты MPI описываются в файле **mpi.h**, который обязательно включается в MPI-программу с помощью оператора include. Этот оператор находится в начале программы.

Переменная status является массивом стандартного целого типа. Его размер и индексы задаются именованными константами:

```
integer status(MPI_STATUS_SIZE)
…
if(status(MPI_TAG).EQ.tag1) then
…
```

Привязка к языку Fortran

В программах на языке Fortran такие объекты MPI, как MPI_Datatype или MPI_Comm — целого типа (integer).

В программах на языке C используются библиотечные функции MPI, в программах на языке Fortran — процедуры.

Привязка к языку Fortran

Тип данных MPI	Тип данных FORTRAN
MPI_INTEGER	INTEGER
MPI_REAL	REAL
MPI_DOUBLE_PRECISION	DOUBLE PRECISION
MPI_DOUBLE_COMPLEX	DOUBLE COMPLEX
MPI_COMPLEX	COMPLEX
MPI_LOGICAL	LOGICAL
MPI_CHARACTER	CHARACTER
MPI_BYTE	Нет соответствия
MPI_PACKED	Нет соответствия
Типы, которые имеются не во всех реализациях MPI	
MPI_INTEGER1	INTEGER*1
MPI_INTEGER2	INTEGER*2
MPI_INTEGER4	INTEGER*4
MPI_REAL4	REAL*4
MPI_REAL8	REAL*8

Коды завершения

Коды завершения возвращаются в качестве значения функции C или через последний аргумент процедуры Fortran.
Исключение составляют подпрограммы MPI_Wtime и MPI_Wtick, в которых возвращение кода ошибки не предусмотрено.
Используются стандартные значения MPI_SUCCESS — при успешном завершении вызова и MPI_ERR_OTHER — обычно при попытке повторного вызова процедуры MPI_Init.

Коды завершения

Вместо числовых кодов в программах обычно используют специальные именованные константы. Среди них:

- MPI_ERR_BUFFER — неправильный указатель на буфер;
- MPI_ERR_COMM — неправильный коммуникатор;
- MPI_ERR_RANK — неправильный ранг;
- MPI_ERR_OP — неправильная операция;
- MPI_ERR_ARG — неправильный аргумент;
- MPI_ERR_UNKNOWN — неизвестная ошибка;
- MPI_ERR_INTERN — внутренняя ошибка. Обычно возникает, если системе не хватает памяти.

Структура MPI-программы

Структура MPI-программы

В программе MPI следует соблюдать определенные правила, без которых она окажется неработоспособной.
В начале программы, сразу после ее заголовка, необходимо подключить соответствующий заголовочный файл. В программе на языке C это mpi.h:

```
#include "mpi.h"
```

а в программе на языке FORTRAN — mpif.h:

```
include "mpif.h"
```

В этих файлах содержатся описания констант и переменных библиотеки MPI.

Структура MPI-программы

Первым вызовом библиотечной процедуры MPI в программе должен быть вызов подпрограммы инициализации MPI_Init, перед ним может располагаться только вызов MPI_Initialized, с помощью которого определяют, инициализирована ли система MPI. Вызов процедуры инициализации выполняется только один раз. В языке Fortran у процедуры инициализации единственный аргумент — код ошибки:
integer IERR

```
call mpi_init(ierr)
```

Структура MPI-программы

В C параметры функции инициализации получают адреса аргументов главной программы, задаваемых при ее запуске:

```
MPI_Init(&argc, &argv);
```

Загрузчик mpirun в конец командной строки запуска MPI-программы добавляет служебные параметры, необходимые MPI_Init. В программах на языке Fortran аргументы командной строки не используются.

Процедура инициализации создает коммуникатор со стандартным именем **MPI_COMM_WORLD**. Это имя указывается во всех последующих вызовах процедур MPI.

Структура MPI-программы

После выполнения всех обменов сообщениями в программе должен располагаться вызов процедуры

MPI_Finalize(ierr)

В результате этого вызова удаляются структуры данных MPI и выполняются другие необходимые действия. Программист должен позаботиться о том, чтобы к моменту вызова процедуры MPI_Finalize были завершены все пересылки данных. После выполнения данного вызова другие вызовы процедур MPI, включая MPI_Init, недопустимы. Исключение составляет подпрограмма MPI_Initialized, которая возвращает значение "истина", если процесс вызывал MPI_Init. Данный вызов может находиться в любом месте программы.

Структура MPI-программы

Организация программы по схеме master-slave

```
program parallel
…
if (процесс = мастер) then
  master
else
  slave
endif
…
end
```

Простейшая MPI-программа

MPI

MPI_Iinit(int *argc, char **argv)

подключение к MPI.
Аргументы **argc** и **argv** требуются только в программах на C, где они задают количество аргументов командной строки запуска программы и вектор этих аргументов. Данный вызов предшествует всем прочим вызовам подпрограмм MPI

MPI

MPI_Finalize()

завершение работы с MPI.
После вызова данной подпрограммы нельзя вызывать подпрограммы MPI. MPI_Finalize должны вызывать все процессы перед завершением своей работы

MPI

MPI_Comm_size(comm, size)

определение размера области взаимодействия.
Здесь **comm** - входной параметр-коммуникатор, выходным является параметр **size** целого типа - количество процессов в области взаимодействия

MPI

MPI_Comm_rank(comm, pid)

определение номера процесса.
Здесь **pid** - идентификатор процесса в указанной области взаимодействия

Простейшая MPI-программа

simple_MPI.cpp

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char *argv[]) {
 int ProcNum, ProcRank, tmp;
 MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
    MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
    printf("Hello world from process %i \n", ProcRank);
    MPI_Finalize();
 return 0;
}
```

Простейшая MPI-программа

simple_MPI.f90

```
program main_mpi
 include 'mpif.h'
 integer myid, numprocs, ierr
 call mpi_init(ierr)
 call mpi_comm_rank(mpi_comm_world, myid, ierr)
 call mpi_comm_size(mpi_comm_world, numprocs, ierr)
 print *, "process ", myid, " of ", numprocs
 call mpi_finalize(ierr)
end
```

Простейшая MPI-программа

```
[nemnugin@pd00 ~]$ mpdboot -n 3
[nemnugin@pd00 ~]$ mpicxx simple_MPI.cpp
[nemnugin@pd00 ~]$ mpiexec -n 3 ./a.out
Hello world from process 0
Hello world from process 1
Hello world from process 2
[nemnugin@pd00 ~]$
```

Заключение

В этой лекции мы рассмотрели:

- основные понятия и особенности реализации MPI;
- привязки к языкам программирования;
- структуру MPI-программы;
- компиляцию и выполнение MPI-программю

Задания для самостоятельной работы

Установить на своем компьютере или в кластере и настроить одну из реализаций MPI.

Если в процессе установки будут возникать вопросы, можно обращаться по электронной почте:

parallel-g112@yandex.ru

Задания для самостоятельной работы

В исходном тексте программы на языке С пропущены вызовы процедур подключения к MPI, определения количества процессов и ранга процесса. Добавить эти вызовы, откомпилировать и запустить программу.

```
#include "mpi.h"
#include <stdio.h>
int main(int argc,char *argv[])
{
int myid, numprocs;
....
fprintf(stdout,"Process %d of %d\n", myid, numprocs);
MPI_Finalize();
return 0;
}
```

Задания для самостоятельной работы

В исходном тексте программы на языке Fortran пропущены вызовы процедур подключения к MPI, определения количества процессов и ранга процесса. Добавить эти вызовы, откомпилировать и запустить программу.

```
program main_mpi
include 'mpif.h'
integer myid, numprocs, ierr
....
print *, "process ", myid, " of ", numprocs
call mpi_finalize(ierr)
stop
end
```