Основные понятия

Наиболее распространенной технологией программирования для параллельных компьютеров с распределенной памятью в настоящее время является MPI. Основным способом взаимодействия параллельных процессов в таких системах является передача сообщений друг другу. Это и отражено в названии данной технологии — *Message Passing Interface (интерфейс передачи сообщений)*. Стандарт MPI фиксирует интерфейс, который должен соблюдаться как системой программирования на каждой вычислительной платформе, так и пользователем при создании своих программ. Современные реализации, чаще всего, соответствуют стандарту MPI версии 1.1. В 1997—1998 годах появился стандарт MPI-2.0, значительно расширивший функциональность предыдущей версии. Однако до сих пор этот вариант MPI не получил широкого распространения и в полном объеме не реализован ни на одной системе. Везде далее, если иного не оговорено, мы будем иметь дело со стандартом MPI-1.1.

MPI поддерживает работу с языками Фортран и Си. В данном пособии примеры и описания всех процедур будут даны с использованием языка Фортран. Однако это совершенно не является принципиальным, поскольку основные идеи MPI и правила оформления отдельных конструкций для этих языков во многом схожи. Полная версия интерфейса содержит описание более 125 процедур и функций. Наша задача — объяснить идею технологии и помочь освоить необходимые на практике компоненты. Дополнительную информацию об интерфейсе MPI можно найти на тематической странице Информационно-аналитического центра по параллельным вычислениям в сети Интернет http://parallel.ru/tech/tech_dev/mpi.html.

Интерфейс MPI поддерживает создание параллельных программ в стиле MIMD (Multiple Instruction Multiple Data), что подразумевает объединение процессов с различными исходными текстами. Однако писать и отлаживать такие программы очень сложно, поэтому на практике программисты гораздо чаще используют *SPMD-модель (Single Program Multiple Data)* параллельного программирования, в рамках которой для всех параллельных процессов используется один и тот же код. В настоящее время все больше и больше реализаций MPI поддерживают работу с нитями.

Поскольку MPI является библиотекой, то при компиляции программы необходимо прилинковать соответствующие библиотечные модули. Это можно сделать в командной строке или воспользоваться предусмотренными в большинстве систем командами или скриптами **`mpicc`** (для программ на языке Си), **`mpiCC`** (для программ на языке Си++), и **`mpif77/mpif90`** (для программ на языках Фортран 77/90). Опция компилятора "**`-o name`**" позволяет задать имя

name для получаемого выполнимого файла, по умолчанию выполнимый файл называется **a.out**, например:

```
mpif77 -o program program.f
```

После получения выполнимого файла необходимо запустить его на требуемом количестве процессоров. Для этого обычно предоставляется команда запуска MPI-приложений **mpirun**, например:

```
mpirun -np N <программа с аргументами>,
```

где **N** - число процессов, которое должно быть не более разрешенного в данной системе числа процессов для одной задачи. После запуска одна и та же программа будет выполняться всеми запущенными процессами, результат выполнения в зависимости от системы будет выдаваться на терминал или записываться в файл с предопределенным именем.

Все дополнительные объекты: имена процедур, константы, предопределенные типы данных и т.п., используемые в MPI, имеют префикс **MPI_**. Если пользователь не будет использовать в программе имен с таким префиксом, то конфликтов с объектами MPI заведомо не будет. В языке Си, кроме того, является существенным регистр символов в названиях функций. Обычно в названиях функций MPI первая буква после префикса **MPI_** пишется в верхнем регистре, последующие буквы – в нижнем регистре, а названия констант MPI записываются целиком в верхнем регистре. Все описания интерфейса MPI собраны в файле **mpif.h** (**mpi.h**), поэтому в начале MPI-программы должна стоять директива **include 'mpif.h'** (**#include "mpi.h"** для программ на языке Си).

MPI-программа — это множество параллельных взаимодействующих процессов. Все процессы порождаются один раз, образуя параллельную часть программы. В ходе выполнения MPI-программы порождение дополнительных процессов или уничтожение существующих не допускается (в MPI-2.0 такая возможность появилась). Каждый процесс работает в своем адресном пространстве, никаких общих переменных или данных в MPI нет. Основным способом взаимодействия между процессами является явная посылка сообщений.

Для локализации взаимодействия параллельных процессов программы можно создавать *группы процессов*, предоставляя им отдельную среду для общения — *коммуникатор*. Состав образуемых групп произволен. Группы могут полностью совпадать, входить одна в другую, не пересекаться или пересекаться частично. Процессы могут взаимодействовать только внутри некоторого коммуникатора, сообщения, отправленные в разных коммуникаторах, не пересекаются и не мешают друг другу. Коммуникаторы имеют в языке Фортран тип **INTEGER** (в языке Си – предопределенный тип **MPI_Comm**).

При старте программы всегда считается, что все порожденные процессы работают в рамках всеобъемлющего коммуникатора, имеющего предопределенное имя **MPI_COMM_WORLD**. Этот коммуникатор существует всегда и служит для взаимодействия всех запущенных процессов MPI-программы. Кроме него при старте программы имеется коммуникатор **MPI_COMM_SELF**, содержащий только один текущий процесс, а также коммуникатор **MPI_COMM_NULL**, не содержащий ни одного процесса. Все взаимодействия процессов протекают в рамках определенного коммуникатора, сообщения, переданные в разных коммуникаторах, никак не мешают друг другу.

Каждый процесс MPI-программы имеет в каждой группе, в которую он входит, уникальный атрибут *номер процесса*, который является целым неотрицательным числом. С помощью этого атрибута происходит значительная часть взаимодействия процессов между собой. Ясно, что в одном и том же коммуникаторе все процессы имеют различные номера. Но поскольку процесс может одновременно входить в разные коммуникаторы, то его номер в одном коммуникаторе может отличаться от его номера в другом. Отсюда становятся понятными *два основных атрибута процесса: коммуникатор и номер в коммуникаторе*. Если группа содержит n процессов, то номер любого процесса в данной группе лежит в пределах от 0 до $n - 1$.

Основным способом общения процессов между собой является явная посылка сообщений. *Сообщение* — это набор данных некоторого типа. Каждое сообщение имеет несколько *атрибутов*, в частности, номер процесса-отправителя, номер процесса-получателя, идентификатор сообщения и другие. Одним из важных атрибутов сообщения является его идентификатор или тэг. По идентификатору процесс, принимающий сообщение, например, может различить два сообщения, пришедшие к нему от одного и того же процесса. Сам идентификатор сообщения является целым неотрицательным числом, лежащим в диапазоне от **0** до **MPI_TAG_UP**, причем гарантируется, что **MPI_TAG_UP** не меньше **32767**. Для работы с атрибутами сообщений введен массив (в языке Си – структура), элементы которого дают доступ к их значениям.

В последнем аргументе (в языке Си – в возвращаемом значении функции) большинство процедур MPI возвращают информацию об успешности завершения. В случае успешного выполнения возвращается значение **MPI_SUCCESS**, иначе – код ошибки. Вид ошибки, которая произошла при выполнении процедуры, можно будет определить из ее описания. Предопределенные значения, соответствующие различным ошибочным ситуациям, перечислены в файле **mpif.h**.