Часть 2.
СРЕДА ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ MPI

Коммуникационная библиотека MPI стала общепризнанным стандартом в параллельном программировании с использованием механизма передачи сообщений. Полное и строгое описание среды программирования MPI можно найти в авторском описании разработчиков [15, 16]. К сожалению, до сих пор нет перевода этого документа на русский язык. Предлагаемое вниманию читателя описание MPI не является полным, однако содержит значительно больше материала, чем принято представлять во введениях в MPI. Цель данного пособия состоит в том, чтобы, во-первых, ознакомить читателя с функциональными возможностями этой коммуникационной библиотеки и, во-вторых, рассмотреть набор подпрограмм, достаточный для программирования любых алгоритмов. Примеры параллельных программ с использованием коммуникационной библиотеки MPI, приведенные в конце данной части, протестированы на различных многопроцессорных системах (nCUBE2, Linux-кластере, 2-х процессорной системе Alpha DS20E).

Глава 6.
ОБЩАЯ ОРГАНИЗАЦИЯ MPI

MPI-программа представляет собой набор независимых процессов, каждый из которых выполняет свою собственную программу (не обязательно одну и ту же), написанную на языке C или FORTRAN. Появились реализации MPI для C++, однако разработчики стандарта MPI за них ответственности не несут. Процессы MPI-программы взаимодействуют друг с другом посредством вызова коммуникационных процедур. Как правило, каждый процесс выполняется в своем собственном адресном пространстве, однако допускается и режим разделения памяти.

MPI не специфицирует модель выполнения процесса – это может быть как последовательный процесс, так и многопотоковый. MPI не предоставляет никаких средств для распределения процессов по вычислительным узлам и для запуска их на исполнение. Эти функции возлагаются либо на операционную систему, либо на программиста. В частности, на nCUBE2 используется стандартная команда xnc, а на кластерах – специальный командный файл (*скрипт*) mpirun, который предполагает, что исполнимые модули уже каким-то образом распределены по компьютерам кластера. Описываемый в данной книге стандарт MPI 1.1 не содержит механизмов динамического создания и уничтожения процессов во время выполнения программы. MPI не накладывает каких-либо ограничений на то, как процессы будут распределены по процессорам, в частности, возможен запуск MPI-программы с несколькими процессами на обычной однопроцессорной системе.

Для идентификации наборов процессов вводится понятие *группы*, объединяющей все или какую-то часть процессов. Каждая группа образует *область связи*, с которой связывается специальный объект – *коммуникатор* области связи. Процессы внутри группы нумеруются целым числом в диапазоне 0..groupsize-1. Все коммуникационные операции с некоторым коммуникатором будут выполняться только внутри области связи, описываемой этим коммуникатором. При инициализации MPI создается предопределенная область связи, содержащая все процессы MPI-программы, с которой связывается предопределенный коммуникатор MPI_COMM_WORLD. В большинстве случаев на каждом процессоре запускается один отдельный процесс, и тогда термины процесс и процессор становятся синонимами, а величина groupsize становится равной NPROCS – числу процессоров, выделенных задаче. В дальнейшем обсуждении мы будем понимать именно такую ситуацию и не будем очень уж строго следить за терминологией.

Итак, если сформулировать коротко, MPI – это библиотека функций, обеспечивающая взаимодействие параллельных процессов с помощью механизма передачи сообщений. Это достаточно объемная и сложная библиотека, состоящая примерно из 130 функций, в число которых входят:

- функции инициализации и закрытия MPI-процессов;
- функции, реализующие коммуникационные операции типа точка-точка;
- функции, реализующие коллективные операции;
- функции для работы с группами процессов и коммуникаторами;
- функции для работы со структурами данных;
- функции формирования топологии процессов.

Набор функций библиотеки MPI далеко выходит за рамки набора функций, минимально необходимого для поддержки механизма передачи сообщений, описанного в первой части. Однако сложность этой библиотеки не должна пугать пользователей, поскольку, в конечном итоге, все это множество функций предназначено для облегчения разработки эффективных параллельных программ. В конце концов, пользователю принадлежит право самому решать, какие средства из предоставляемого арсенала использовать, а какие нет. В принципе, любая параллельная программа может быть написана с использованием всего 6 MPI-функций, а достаточно полную и удобную среду программирования составляет набор из 24 функций [11].

Каждая из MPI функций характеризуется способом выполнения:

1. *Локальная функция* – выполняется внутри вызывающего процесса. Ее завершение не требует коммуникаций.
2. *Нелокальная функция* – для ее завершения требуется выполнение MPI-процедуры другим процессом.
3. *Глобальная функция* – процедуру должны выполнять все процессы группы. Несоблюдение этого условия может приводить к зависанию задачи.

4. *Блокирующая функция* – возврат управления из процедуры гарантирует возможность повторного использования параметров, участвующих в вызове. Никаких изменений в состоянии процесса, вызвавшего блокирующий запрос, до выхода из процедуры не может происходить.
5. *Неблокирующая функция* – возврат из процедуры происходит немедленно, без ожидания окончания операции и до того, как будет разрешено повторное использование параметров, участвующих в запросе. Завершение неблокирующих операций осуществляется специальными функциями.

Использование библиотеки MPI имеет некоторые отличия в языках C и FORTRAN.

В языке C все процедуры являются функциями, и большинство из них возвращает код ошибки. При использовании имен подпрограмм и именованных констант необходимо строго соблюдать регистр символов. Массивы индексируются с 0. Логические переменные представляются типом int (true соответствует 1, a false – 0). Определение всех именованных констант, прототипов функций и определение типов выполняется подключением файла mpi.h. Введение собственных типов в MPI было продиктовано тем обстоятельством, что стандартные типы языков на разных платформах имеют различное представление. MPI допускает возможность запуска процессов параллельной программы на компьютерах различных платформ, обеспечивая при этом автоматическое преобразование данных при пересылках. В таблице 6.1 приведено соответствие предопределенных в MPI типов стандартным типам языка C.