Проект комиссии Президента
по модернизации и технологическому развитию экономики России
«Создание системы подготовки высококвалифицированных кадров
в области суперкомпьютерных технологий и
специализированного программного обеспечения»

УТВЕРЖДАЮ
Председатель экспертного совета
системы НОЦ СКТ, член-корр. РАН
В.В. Воеводин
_____
"_____"_____201__г.

Конспект лекций дисциплины

«Параллельное программирование для многопроцессорных систем
с общей и распределенной памятью»

«**010400.62 –Прикладная математика и информатика**»

Разработчики: Лаева В.И. , Трунов А.А.
Рецензент: проф. Старченко А.В.

Москва

ВВЕДЕНИЕ В ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТАНДАРТА MPI

MPI (**Message Passing Interface –**интерфейс передачи сообщений) – наиболее распространенная технология параллельного программирования для многопроцессорных вычислительных систем с распределенной памятью. Процессы используют обращение к библиотеке MPI для того, чтобы обмениваться данными (информацией) друг с другом. Механизм передачи сообщений позволяет процессам, запущенным на параллельном компьютере, совместно решать большие задачи.

MPI – стандарт передачи сообщений разрабатывался группой *MPI FORUM*, в состав которой входили представители из 40 организаций США и Европы. Первый вариант стандарта **MPI 1.0** был опубликован в 1994 г. Большинство современных реализаций соответствует стандарту **MPI 1.1** (опубликован в 1995г.). MPI пригоден для различных платформ, начиная с массивно - параллельных систем (например, IBM SP2, Cray N3D,Intel Paragon) и заканчивая сетями рабочих станций (Sun4, Dec Alpha). Основной целью, преследуемой при создании MPI, было разработать практичный, переносимый, эффективный и удобный стандарт для передачи сообщений между процессами.

Интерфейс **MPI** поддерживает создание параллельных программ в стиле **MIMD** (**Multiple Instruction – Multiple Data stream –** несколько потоков команд – несколько потоков данных), что подразумевает объединение процессов с различными исходными текстами. Однако на практике чаще используют **SPMD** модель (**Single Program – Multiple Data – одна программа- несколько потоков данных**). На всех процессах выполняется одна и та же программа, обрабатывающая различные фрагменты набора данных.

Полная версия MPI содержит описание более 120 функций для:

- инициализации и закрытия параллельной части приложения;
- приема и передачи сообщений между отдельными процессами;
- осуществления коллективного взаимодействия процессов;
- работы с группами процессов и коммуникаторами;
- определения производных (составных) типов данных;
- создания виртуальных топологий для обеспечения более простого взаимодействия процессов.

Каждая функция MPI характеризуется способом выполнения.

1) Локальная функция выполняется внутри вызывающего процесса, ее завершение не требует коммуникаций.

2) Нелокальная функция – для ее завершения требуется выполнение MPI функции другим процессом

3) Глобальная функция – ее должны выполнить все процессы группы. Несоблюдение этого условия может приводить к зависанию задачи

4) Блокирующая функция – возврат управления из функции гарантирует возможность повторного использования параметров, участвующих в вызове. Никаких изменений в состоянии процесса, вызвавшего блокирующий запрос, до выхода из функции не может происходить

5) Неблокирующая функция – возврат из функции происходит немедленно, без ожидания окончания операции и до того, как будет разрешено повторное использование параметров, участвующих в запросе. Завершение неблокирующих операций осуществляется специальными функциями.

Основные понятия MPI

К базовым понятиям MPI относятся ***процесс, группа процессов и коммуникатор***.

Процесс – это исполнение программы одним процессорным элементом, на котором загружен MPI.

Процессы объединяются в ***группы*** с единой областью связи, внутри которой каждый процесс имеет свой уникальный номер (ранг) в диапазоне от 0 до N-1, где N – количество процессов в группе.

Для взаимодействия процессов в группе используется специальная информационная структура – ***коммуникатор***, который реализует передачу сообщений (обмен данными) между процессами и их синхронизацию. ***Коммуникатор*** осуществляет обмены только внутри области связи группы, с которой он ассоциируется.

Обычно MPI-программа для многопроцессорных вычислительных систем пишется на языке ***C***, ***C***++ или FORTRAN с использованием коммуникационных функций библиотеки MPI. Все описания интерфейса MPI собраны в файлах ***mpi.h*** и ***mpif.h***, поэтому начале MPI – программы должна стоять директива:

#include<mpi.h> - для ***C*** – программ,

include 'mpif.h' - для ***FORTRAN*** – программ.

Использование библиотеки MPI имеет некоторые отличия в языках ***C*** и FORTRAN. В языке ***C*** все подпрограммы являются функциями, и большинство из них возвращают код ошибки. В языке FORTRAN большинство подпрограмм являются процедурами (subroutine) и вызываются с помощью оператора CALL, а код ошибки возвращают через дополнительный последний параметр (error). Однако в дальнейшем для единообразия и те, и другие будем называть *функциями*.

Все объекты MPI: имена функций, константы, предопределенные типы данных имеют префикс ***MPI_***. В языке ***C*** существенным является регистр символов: идентификаторы предопределенных констант записываются в верхнем регистре, при вызове функции ***MPI_*** и первая буква имени записываются в верхнем регистре, остальные буквы – в нижнем регистре. Для языка FORTRAN буквы, набранные в верхнем и нижнем регистре, в идентификаторах не различаются.

В MPI принята система обозначений базовых типов данных, которая соответствует типам в языках ***C*** и FORTRAN.

Соответствие между типами данных языка *C* и *MPI*:

Тип *C*	**тип *MPI***
signed char	***MPI_CHAR***
signed short int	***MPI_SHORT***
signed int	***MPI_INT***
signed long int	***MPI_LONG***
unsigned char	***MPI_UNSIGNED_CHAR***
unsigned short int	***MPI_UNSIGNED_SHORT***
unsigned long int	***MPI_UNSIGNED_LONG***
float	***MPI_FLOAT***
double	***MPI_DOUBLE***
long double	***MPI_LONG_DOUBLE***
	MPI_BYTE
	MPI_PACKED

Соответствие между типами данных языка FORTRAN **и *MPI:***

тип *FORTRAN*	**тип *MPI***
INTEGER	***MPI_INTEGER***
REAL	***MPI_REAL***
DOUBLE PRECISION	***MPI_DOUBLE_PRECISION***
COMPLEX	***MPI_COMPLEX***
LOGICAL	***MPI_LOGICAL***
CHARACTER	***MPI_CHARACTER***
	MPI_BYTE
	MPI_PACKED

Типы ***MPI_BYTE***, ***MPI_PACKED*** используются для передачи двоичной информации без какого – либо преобразования.

Запуск на выполнение MPI-программы представляет собой одновременный запуск совокупности параллельных процессов, количество которых определяет пользователь при запуске. В стандарте MPI 1.1 не допускается в ходе выполнения MPI-программы порождение дополнительных процессов или уничтожение уже существующих. Параллельная часть программы начинается с инициализации MPI. При этом все активированные процессы объединяются в группу с коммуникатором, который имеет имя ***MPI_COMM_WORLD***. Этот коммуникатор существует всегда и служит для взаимодействия всех запущенных процессов MPI – программы. Кроме него при старте программы имеются также: ***MPI_COMM_SELF*** – коммуникатор, содержащий только вызывающий процесс; ***MPI_COMM_NULL*** – пустой коммуникатор. Коммуникаторы являются объектами типа MPI_Comm в программах на языке ***C*** и типа INTEGER в языке FORTRAN.

Далее средствами MPI в программу передается число активированных процессов, каждому из них присваивается свой номер (ранг). Процессы, как правило, отличаются тем, что каждый отвечает за исполнение своей ветви параллельной MPI-программы.

Основным способом общения процессов между собой является посылка сообщений. *Сообщение* – это набор данных некоторого типа. Каждое сообщение имеет несколько атрибутов: номер процесса – отправителя, номер процесса – получателя, идентификатор сообщения (тег) и др. Идентификатор (тег) сообщения целое неотрицательное число из диапазона от 0 до 32767. По идентификатору процесс, который принимает сообщение, может различить сообщения, пришедшие от одного и того же процесса.

Для атрибутов сообщения в ***C***-программах введена структура ***MPI_STATUS***, поля которой дают доступ к значениям атрибутов. В FORTRAN – программе ***STATUS*** - массив целого типа размерности ***MPI_STATUS_SIZE***, компоненты которого содержат все атрибуты. Индексы массива ***STATUS*** и поля структуры ***STATUS*** задаются именованными константами: ***MPI_SOURCE, MPI_TAG, MPI_ERROR***.

Синтаксис основных функций MPI.

Рассмотрим синтаксис функций, не связанных с пересылками данных. Как правило, все эти функции необходимы в каждой параллельной программе.

Параллельная часть программы начинается с инициализации MPI, которая осуществляется при вызове функции ***MPI_Init***:

C: ***int MPI_Init(int *argc, char ***argv)***

FORTRAN:

```
MPI_Init(error)
INTEGER error
```

Вызов этой функции может быть выполнен каждым процессом только один раз. Все другие MPI – функции могут быть вызваны только после вызова ***MPI_Init***. Функция возвращается код ошибки ***error***. Если ***error***=0, то оператор проработал успешно. При этом все активированные процессы объединяются в группу с коммуникатором, который имеет имя ***MPI_COMM_WORLD***.

Определение числа процессов :

C: ***int MPI_Comm_size(MPI_Comm comm, int *size)***

FORTRAN:

```
MPI_Comm_size(comm, size, error)
INTEGER comm, size, error
```

Функция возвращает количество процессов ***size*** в области связи с коммуникатором ***comm***.

Определение номера процесса:

C: ***int MPI_Comm_rank(MPI_Comm comm, int *rank)***

FORTRAN:

MPI_Comm_ rank(comm, rank, error)

INTEGER comm, rank, error

Функция возвращает номер процесса ***rank***, вызвавшего ее, из области связи с коммуникатором ***comm.***

Завершение MPI производится при вызове функции ***MPI_Finalize***:

C: ***int MPI_Finalize(void)***

FORTRAN:

MPI_Finalize(error)

INTEGER error

Все последующие обращения к любым MPI – функциям запрещены. К моменту вызова этой функции каждым процессом все обмены сообщениями должны быть завершены.

Обычно для оценки эффективности и ускорения работы параллельной программы необходимо определить время, затрачиваемое на ее выполнение. Для этой цели в библиотеке MPI предусмотрена функция ***MPI_Wtime( )***.

C: ***double MPI_Wtime(void)***

FORTRAN: ***DOUBLE PRECISION MPI_Wtime( )***

Функция возвращает астрономическое время в секундах, прошедшее с некоторого фиксированного момента в прошлом.

Функция ***MPI_Wtick( )***, имеющая точно такой же синтаксис, возвращает разрешение таймера (минимальное значение кванта времени).

Пример простейшей MPI программы:

```
Program Example1
      Implicit None
      Include 'mpif.h'
      Integer Rank, Size, error, Comm
C Инициализация MPI и определение процессорной
C  конфигурации
      Call MPI_Init(error)
      Comm=MPI_COMM_WORLD
      Call MPI_COMM_SIZE(Comm,Size,error)
      Call MPI_COMM_RANK(Comm, Rank,error)
      Write(6,*) 'Process ',Rank,' of ',Size,' is ready to work.'
C   Завершение работы функций MPI
      Call MPI_FINALIZE(error)
      End
```

Коды завершения

Коды завершения возвращаются в качестве значения функции ***C*** или через последний аргумент (***error***) подпрограммы FORTRAN. Исключение составляют функции ***MPI_Wtime*** и ***MPI_Wtick***, в которых кода ошибки не предусмотрено.

Для кодов используются специальные именованные константы:

MPI_SUCCESS – при успешном завершении вызова;

MPI_ERR_OTHER – при попытке повторного вызова ***MPI_Init;***

MPI_ERR_BUFFER – неправильный указатель на буфер;

MPI_ERR_COMM – неправильный коммуникатор;

MPI_ERR_IN_STATUS – ошибка в случае неуспешного выполнения одной или более операции обмена;

MPI_ERR_RANK - неправильный ранг;

MPI_ERR_OP – неправильная операция;

MPI_ERR_ARG - неправильный аргумент, ошибочное задание которого не попадает ни в один класс ошибок;

MPI_ERR_COUNT – неправильно задано количество пересылаемых значений;

MPI_ERR_UNKNOWN – неизвестная ошибка;

MPI_ERR_TRUNCATE – сообщение обрезано при приеме;

MPI_ERR_INTERN – внутренняя ошибка (обычно, если системе не хватает памяти);

MPI_ERR_TYPE – неправильное значение аргумента, задающего тип данных;

MPI_ERR_TAG – неправильно указан тег сообщения;

MPI_ERR_REQUEST – неправильный запрос на выполнение операции обмена и др.