



Параллельное программирование с использованием технологии

MPI

Лекция 7

Томский политехнический
университет

Аксёнов Сергей Владимирович

к.т.н., доцент каф.ОСУ ТПУ

Блокирующая проверка завершения всех обменов

```
int MPI_Waitall(int count, MPI_Request *requests, MPI_Status *statuses)
```

Вход count: Размер массивов requests и statuses

Вход/Выход requests: Массив запросов

Выход statuses: Массив статусов

Вызов функции блокирует выполнение процесса до тех пор, пока **все** операции обмена, связанные с активными запросами в массиве requests, не будут выполнены. Возвращается статус этих операций. Статус обменов содержится в массиве statuses.

Неблокирующая проверка завершения всех обменов

```
int MPI_Testall(int count, MPI_Request *requests, int * flag, MPI_Status *statuses)
```

Вход count: Размер массивов requests и statuses

Вход/Выход requests: Массив запросов

Выход: flag: 1 – если все обмены завершены

Выход statuses: Массив статусов

Функция выполняет неблокирующую проверку завершения приема или передачи всех сообщений.

При вызове возвращается значение флага (flag) «истина», если все обмены, связанные с активными запросами в массиве requests, выполнены. Если завершены не все обмены, флагу присваивается значение «ложь», а массив statuses не определен.

Пример

```
MPI_Request *request;
```

```
MPI_Status *status;
```

```
MPI_Comm_rank( MPI_COMM_WORLD, &rank );
```

```
if (rank == 0)
```

```
{
```

```
MPI_Isend(&buf1,20,MPI_INT,1,25,MPI_COMM_WORLD, &requests[0]);
```

```
MPI_Isend(&buf2,20,MPI_INT,1,26,MPI_COMM_WORLD, &requests[1]);
```

```
MPI_Waitall(2,requests, statuses)
```

```
}
```

```
if (rank==1)
```

```
{
```

```
MPI_Irecv(&source1, 20, MPI_INT, 0, 25, MPI_COMM_WORLD, &requests[0]);
```

```
MPI_Irecv(&source2, 20, MPI_INT, 0, 26, MPI_COMM_WORLD, &request[1]);
```

```
MPI_Waitall(2,requests, statuses);
```

```
}
```

Блокирующая проверка завершения одного обмена из набора обменов

```
int MPI_Waitany(int count, MPI_Request *requests, int *index,  
MPI_Status *status)
```

Вход count: Размер массива requests

Вход/Выход requests: Массив запросов

Выход: index: Индекс запроса для завершенной операции

Выход status: Статус

Операция блокирует работу до тех пор, пока не завершится одна из операций из массива активных запросов. Если более чем одна операция задействована и может закончиться, выполняется произвольный выбор. Операция возвращает в `index` индекс этого запроса в массиве и возвращает в `status` статус завершаемого обмена.

Неблокирующая проверка завершения одного обмена из набора обменов

```
int MPI_Testany(int count, MPI_Request *requests, int *index, int *flag, MPI_Status *status)
```

Вход count: Размер массива requests

Вход/Выход requests: Массив запросов

Выход index: индекс запроса для завершенной операции

Выход flag: 1 – если хотя бы один обмен завершен

Выход status: Статус

Смысл и назначение параметров этой функции те же, что и для подпрограммы MPI_Waitany. Дополнительный аргумент flag, принимает значение «истина», если одна из операций завершенна.

Дополнительная функция блокирующей проверки завершения обмена

```
int MPI_Waitsome (int incount, MPI_Request *requests, int *outcount, int *indices, MPI_Status *statuses)
```

Вход	count: Размер массива requests
Вход/Выход	requests: Массив запросов
Выход	outcount: Число завершённых запросов
Выход	indices: Массив индексов завершённых операций
Выход	status: Массив статусов

Функция ожидает, пока, по крайней мере, одна операция, связанная с активным дескриптором в списке, не завершится. Возвращает в outcount число запросов из списка array_of_indices, которые завершены.

Дополнительная функция неблокирующей проверки завершения обмена

```
int MPI_Testsome (int incount, MPI_Request *requests, int *outcount, int *indices, MPI_Status *statuses)
```

Вход	count: Размер массива requests
Вход/Выход	requests: Массив запросов
Выход	outcount: Число завершённых запросов
Выход	indices: Массив индексов завершённых операций
Выход	status: Массив статусов

Функция `MPI_TESTSOME` ведёт себя подобно `MPI_WAITSSOME` за исключением того, что заканчивается немедленно. Если ни одной операции не завершено, она возвращает `outcount = 0`.

Неблокирующая проба

```
int MPI_Iprobe(int source, int tag, MPI_Comm comm, int *flag, MPI_Status *status)
```

Вход	source:	Номер источника
Вход	tag:	Значение тега
Вход	comm:	Коммуникатор
Выход	flag:	1 – существует входящее сообщение
Выход	status:	Статус

Операции `MPI_PROBE` и `MPI_Iprobe` позволяют проверить входные сообщения без реального их приема. Пользователь затем может решить, как ему принимать эти сообщения, основываясь на информации, возвращенной при пробе.

Блокирующая проба

```
int MPI_Probe(int source, int tag, MPI_Comm comm, MPI_Status *status)
```

Вход	source:	Номер источника
Вход	tag:	Значение тега
Вход	comm:	Коммуникатор
Выход	status:	Статус

MPI_PROBE ведет себя подобно MPI_Iprobe, исключая то, что функция MPI_PROBE является блокирующей и заканчивается после того, как соответствующее сообщение было найдено.

Сообщение не обязательно должно быть получено сразу после опробования, оно может опробоваться несколько раз перед его получением.

Пример

```
int myid, numprocs, **buf, source, i;
int message[3] = {0, 1, 2};
int myrank, count, TAG = 0;
MPI_Status status;
if (myrank == 0) {
    MPI_Send(&message, 3, MPI_INT, 2, TAG, MPI_COMM_WORLD);
}
else
{
    MPI_Probe(MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &status);
    source = status.MPI_SOURCE;
    MPI_Get_count(&status, MPI_INT, &count);
    buf = malloc(count*sizeof(int));
    MPI_Recv(buf, count, MPI_INT, source, TAG, MPI_COMM_WORLD, &status);
    for (i = 0; i < count; i++){
        printf("received: %d\n", buf[i]);
    }
}
```

Отмена ждущих неблокирующих операций обмена

```
int MPI_Cancel(MPI_Request *request)
```

Вход request: Коммуникационный запрос

Обращение к `MPI_CANCEL` маркирует для отмены ждущие неблокирующие операции обмена (передача или прием). Вызов `cancel` является локальным. Он заканчивается немедленно, возможно перед действительной отменой обмена. После маркировки необходимо завершить эту операцию обмена, используя вызов `MPI_WAIT` или `MPI_TEST` (или любые производные операции).

Проверка отмены передачи

```
int MPI_Test_cancelled(MPI_Status *status, int *flag)
```

Вход status: Статус

Выход flag: 1 – если была успешно произведена отмена

обмена

Функция `MPI_TEST_CANCELLED` возвращает `flag = true`, если обмен, связанный со статусным объектом, был отменен успешно. В таком случае все другие поля статуса (такие как `count` или `tag`) не определены. В противном случае возвращается `flag = false`.

Пример

```
int myid, flag;
int message[3] = {0, 1, 2};
int myrank, data = 2222, TAG = 0;
MPI_Status status;
MPI_Request request;
if (myrank == 0)
{
    MPI_BSend(&data, 1, MPI_INT, 2, TAG, MPI_COMM_WORLD);
}
else if (myrank == 1) {
    MPI_BSend(&message, 3, MPI_INT, 2, TAG, MPI_COMM_WORLD);
}
else
{
    MPI_Irecv(&message, 3, MPI_INT, 1, TAG, MPI_COMM_WORLD, &request);
    MPI_Cancel(&request);
    MPI_Wait(&request, &status);
    MPI_Test_cancelled(&status, &flag);
    If (flag==1) printf("Приём был отменён");
}
```