Технологии параллельного программирования на C++

Семинар 11
MPI
Группы и коммуникаторы

Группы и коммуникаторы

- Группа - это упорядоченный набор процессов. Каждый процесс в группе связан с уникальным целочисленным *рангом*. Ранг принимает значения от нуля до N-1, где N - число процессов в группе
- В группу могут входить все процессы параллельной программы или в группе может находиться только часть имеющихся процессов. Один и тот же процесс может принадлежать нескольким группам
- Коммуникатор - специально создаваемый служебный объект, объединяющий в своем составе группу процессов и ряд дополнительных параметров (контекст), используемых при выполнении операций передачи данных

Группы процессов. Конструкторы

- Группы процессов могут быть созданы только из уже существующих групп
- **MPI_GROUP_EMPTY** – пустая группа
- **MPI_GROUP_NULL** – значение, используемое для ошибочной группы

- Получения группы, связанной с существующим коммуникатором:
  int MPI_Comm_group (MPI_Comm comm, MPI_Group *group)

Группы процессов. Конструкторы

- Создание новой группы **newgroup** из существующей группы **oldgroup**, которая будет включать в себя **n** процессов, ранги которых перечисляются в массиве **ranks**:

  int MPI_Group_incl(MPI_Group oldgroup, int n, int *ranks, MPI_Group *newgroup)

- Создание новой группы **newgroup** из существующей группы **oldgroup**, которая будет включать в себя **n** процессов, ранги которых не совпадают с рангами, перечисленными в массиве **ranks**:

  int MPI_Group_excl(MPI_Group oldgroup, int n, int *ranks, MPI_Group *newgroup)

Группы процессов. Конструкторы

- Создание новой группы **newgroup** как объединения групп **group1** и **group2**:

  int MPI_Group_union(MPI_Group group1, MPI_Group group2, MPI_Group *newgroup);

- Создание новой группы **newgroup** как пересечения групп **group1** и **group2**:

  int MPI_Group_intersection (MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)

- Создание новой группы **newgroup** как разности групп **group1** и **group2**:

  int MPI_Group_difference (MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)

Средства доступа группы

- Получение количества процессов в группе:

  int MPI_Group_size (MPI_Group group, int *size)

- Получение ранга текущего процесса в группе:

  int MPI_Group_rank (MPI_Group group, int *rank)

- Процесс может входить в несколько групп. MPI_Group_translate_ranks преобразует ранг ranks1[i] процесса i в одной группе **group1** в его ранг ranks2[i] в другой группе **group2** :
- **int MPI_Group_translate_ranks(MPI_Group group1, int n, int *ranks1, MPI_Group group2, int *ranks2)**
- Сравнения групп group1 и group2:

  int MPI_Group_compare(MPI_Group group1, MPI_Group group2, int *result)

  - Если группы полностью совпадают, возвращается значение **MPI_IDENT**.
  - Если члены обеих групп одинаковы, но их ранги отличаются, результатом будет значение **MPI_SIMILAR**.
  - Если группы различны, результатом будет **MPI_UNEQUAL**.

Группа процессов. Деструктор

- После завершения использования группа должна быть удалена:

  int MPI_Group_free (MPI_Group *group)

- Выполнение данной операции не затрагивает коммуникаторы, в которых используется удаляемая группа

Пример разбиения на четные и нечетные

```
int num, p;
int Neven, Nodd, *members, even_rank, odd_rank;
MPI_Group group_world, even_group, odd_group;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &num);
MPI_Comm_size(MPI_COMM_WORLD, &p);
Neven = (p + 1)/2;
Nodd = p - Neven;
members = new int[Neven];
for (int i =0; i<Neven;i++){
                members[i] = i*2;
}
MPI_Comm_group(MPI_COMM_WORLD, &group_world);
MPI_Group_incl(group_world, Neven, members, &even_group);
MPI_Group_excl(group_world, Neven, members,  &odd_group);

MPI_Group_rank(even_group, &even_rank);
MPI_Group_rank( odd_group,  &odd_rank);
  …
```

num	even	odd
9	-32766	4
10	5	-32766
11	-32766	5
0	0	-32766
1	-32766	0
2	1	-32766
3	-32766	1
4	2	-32766
6	3	-32766
7	-32766	3
8	4	-32766
5	-32766	2

Коммуникаторы

- **Коммуникаторы** – контекст обмена группы
- В операциях обмена используются только коммуникаторы
- **MPI_COMM_WORLD** – коммуникатор для всех процессов приложения
- **MPI_COMM_NULL** – значение, используемое для ошибочного коммуникатора
- **MPI_COMM_SELF** – коммуникатор, включающий только вызвавший процесс

Типы:

- **Итракоммуникаторы** – передача сообщений внутри группы процессов
- **Интеркоммуникаторы** – передача сообщений между разными группами процессов

Коммуникаторы. Конструкторы

- Дублирование уже существующего коммуникатора

  int MPI_Comm_dup (MPI_Comm oldcom, MPI_comm *newcomm)

- Создание нового коммуникатора из подмножества процессов существующего коммуникатора:

  int MPI_comm_create (MPI_Comm oldcom, MPI_Group group, MPI_Comm *newcomm)

Коммуникаторы. Конструкторы

- Одновременное создание нескольких коммуникаторов

int MPI_Comm_split (MPI_Comm oldcomm, int split, int key, MPI_Comm *newcomm)

oldcomm – исходный коммуникатор

split – номер коммуникатора, которому должен принадлежать процесс

key – порядок ранга процесса в создаваемом коммуникаторе

newcomm – создаваемый коммуникатор

- Вызов функции **MPI_Comm_split** должен быть выполнен в каждом процессе коммуникатора **oldcomm**
- Процессы разделяются на непересекающиеся группы с одинаковыми значениями параметра **split**.
- На основе сформированных групп создается набор коммуникаторов. При создании коммуникаторов для рангов процессов выбирается такой порядок нумерации, чтобы он соответствовал порядку значений параметров **key** (процесс с большим значением параметра **key** должен иметь больший ранг).
- Процессы, которые не должны войти в новые группы, указывают в качестве split константу **MPI_UNDEFINED**, им в параметре **newcomm** вернется значение **MPI_COMM_NULL**

Коммуникаторы. Деструктор

- После завершения использования коммуникатор должен быть удален:

  int MPI_Comm_free (MPI_Comm *comm)

Пример

- Представим набор процессов в виде двумерной решетки. Пусть s=a*a есть общее количество процессов
- Получим коммуникатор для каждой строки создаваемой топологии:

```
int rank, row;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
row = rank/a;
MPI_Comm_split(MPI_COMM_WORLD, row, rank, &comm);
```

При s = 9 процессы с рангами (0,1,2) образуют первый коммуникатор, процессы с рангами (3,4,5) – второй и т.д.

Задача 1

Создать коммуникатор, в котором нумерация процессов будет вестись в обратном порядке по сравнению с коммуникатором MPI_COMM_WORLD и напечатать ранги процессов в обоих коммуникаторах.

Задача 2

- В каждом процессе, ранг которого делится на 3 (включая главный процесс), даны три целых числа.
- С помощью функции MPI_Comm_split создать новый коммуникатор, включающий процессы, ранг которых делится на 3.
- Используя одну коллективную операцию пересылки данных для созданного коммуникатора, переслать исходные числа в главный процесс и вывести эти числа в порядке возрастания рангов переславших их процессов (включая числа, полученные из главного процесса).

Задача 3

- В каждом процессе дано целое число *N*, которое может принимать два значения: 1 и 2 (имеется хотя бы один процесс с каж-дым из возможных значений). Кроме того, в каждом процессе дано вещественное число *A*. Используя функцию MPI_Comm_split и одну коллективную операцию редукции, найти минимальное значение среди чисел *A*, которые даны в процессах с *N* = 1, и максимальное значение среди чисел *A*, которые даны в процессах с *N* = 2. Найденный минимум вывести в процессах с *N* = 1, а найденный максимум — в процессах с *N* = 2.