Технология программирования MPI (6)

Антонов Александр Сергеевич,

к.ф.-м.н., вед.н.с. лаборатории Параллельных информационных технологий НИВЦ МГУ

Летняя суперкомпьютерная академия
Москва, 2017

MPI

Виртуальные топологии

MPI

Топология – механизм сопоставления процессам альтернативной схемы адресации. В MPI топологии виртуальны, не связаны с физической топологией сети.

Топология используется для более удобного обозначения процессов, и, таким образом, приближения параллельной программы к структуре математического алгоритма.

MPI

В некоторых случаях топология может использоваться системой для оптимизации распределения процессов по физическим процессорам при помощи изменения порядка нумерации процессов внутри коммуникатора.

Два типа топологий:

- *декартова* (прямоугольная решётка произвольной размерности)
- *топология графа*.

MPI

Декартова топология

MPI

```
int MPI_Cart_create(MPI_Comm comm, int ndims, int *dims, int *periods, int reorder, MPI_Comm *comm_cart)
```

Создание коммуникатора **comm_cart** с декартовой топологией. **ndims** – размерность декартовой решётки, **dims** – число элементов в каждом измерении.

MPI

`periods` – массив из **`ndims`** элементов, определяющий, является ли решётка периодической вдоль каждого измерения.

`reorder` – при значении **`1`** системе разрешено менять порядок нумерации процессов.

Процедура является коллективной, а значит, должна быть вызвана всеми процессами коммуникатора **`comm`**.

MPI

Если количество процессов в задаваемой топологии **comm_cart** меньше числа процессов в коммуникаторе **comm**, то некоторым процессам может вернуться значение **MPI_COMM_NULL**, и они не будут принимать участия в создаваемой топологии. Если количество процессов в задаваемой топологии больше числа процессов в исходном коммуникаторе, то вызов ошибочен. Если **ndims** равно **0**, то создается нульмерная декартова топология.

MPI

```
dims[0] = 4;
dims[1] = 3;
dims[2] = 2;
periods[0] = periods[1] = periods[2] = 1;
MPI_Cart_create(MPI_COMM_WORLD, 3, dims,
periods, 1, comm_cart);
```

MPI

```
int MPI_Dims_create(int nnodes,
int ndims, int *dims)
```

Определение размеров **dims** для каждой из **ndims** размерностей при создании декартовой топологии для **nnodes** процессов. **dims[i]** рассчитывается, если перед вызовом функции оно равно **0**, иначе оставляется без изменений.

MPI

Размеры по разным размерностям устанавливаются так, чтобы быть возможно близкими друг к другу. Перед вызовом функции значение **`nnodes`** должно быть кратно произведению ненулевых значений массива **`dims`**. Выходные значения массива **`dims`**, переопределённые процедурой, будут упорядочены в порядке убывания. Процедура является локальной и не требует межпроцессного взаимодействия.

MPI

dims перед вызовом	вызов процедуры	dims после вызова
(0, 0, 0)	MPI_Dims_create(6, 3, dims)	(3, 2, 1)
(0, 0, 0)	MPI_Dims_create(7, 3, dims)	(7, 1, 1)
(0, 3, 0)	MPI_Dims_create(6, 3, dims)	(2, 3, 1)
(0, 3, 0)	MPI_Dims_create(7, 3, dims)	ошибка

MPI

```
int MPI_Cart_coords(MPI_Comm comm, int rank, int maxdims, int *coords)
```

Определение декартовых координат процесса по его рангу **`rank`**. Координаты возвращаются в массиве **`coords`**. Отсчёт координат по каждому измерению начинается с **`0`**.

MPI

```
int MPI_Cart_rank(MPI_Comm comm,
int *coords, int *rank)
```

Определение ранга **`rank`** процесса по его декартовым координатам **`coords`**. Для периодических решёток координаты вне допустимых интервалов пересчитываются, для непериодических – ошибочны.

MPI

```
int MPI_Cart_sub(MPI_Comm comm,
int *dims, MPI_Comm *newcomm)
```

Расщепление коммуникатора **comm** на подгруппы, соответствующие декартовым подрешёткам меньшей размерности. **i**–ый элемент массива **dims** равен **1**, если **i**–ое измерение должно остаться в подрёшетке.

MPI

```
dims[0] = 1;
dims[1] = 0;
dims[2] = 1;
MPI_Cart_sub(comm, dims, &newcomm);
```

MPI

```
int MPI_Cartdim_get(MPI_Comm comm, int *ndims)
```

Определение размерности **ndims** декартовой топологии коммуникатора **comm**.

MPI

```
int MPI_Cart_get(MPI_Comm comm,
int maxdims, int *dims, int
*periods, int *coords)
```

Получение информации о декартовой топологии коммуникатора **comm** и координатах в ней вызвавшего процесса **coords**.

MPI

```
int MPI_Cart_shift(MPI_Comm
comm, int direction, int disp,
int *source, int *dest)
```

Получение номеров посылающего (**source**) и принимающего (**dest**) процессов в декартовой топологии коммуникатора **comm** для осуществления сдвига вдоль измерения **direction** на величину **disp**.

MPI

Для периодических измерений осуществляется циклический сдвиг, для непериодических – линейный сдвиг. В случае линейного сдвига на некоторых процессах в качестве номеров посылающего или принимающего процессов может быть получено значение **`MPI_PROC_NULL`**, означающее выход за границы диапазона.

MPI

Для **n**-мерной декартовой решётки значение **direction** должно быть в пределах от **0** до **n-1**.

Значения **source** и **dest** можно далее использовать, например, для обмена функцией **MPI_Sendrecv**.

MPI

```
periods[0]=1;
periods[1]=1;
MPI_Cart_create(MPI_COMM_WORLD, 2, dims,
periods, 1, &comm);
MPI_Comm_rank(comm, &rank);
MPI_Cart_coords(comm, rank, 2, coords);
shift = 2;
MPI_Cart_shift(comm, 0, shift, &source,
&dest);
MPI_Sendrecv_replace(a, 1, MPI_FLOAT, dest, 0,
source, 0, comm, &status);
```

MPI

```
int MPI_Cart_map(MPI_Comm comm,
int ndims, int *dims, int
*periods, int *newrank)
```

Вычисление «оптимального» относительно данной декартовой топологии расположения процессов на физических процессорах (если поддерживается реализацией). В **`newrank`** возвращается новый ранг процесса. Если вызывающий процесс не включен в топологию, то вернется **`MPI_UNDEFINED`**.

MPI

Топология графа

MPI

```
int MPI_Graph_create(MPI_Comm
comm, int nnodes, int *index,
int *edges, int reorder,
MPI_Comm *comm_graph)
```

Создание топологии графа **comm_graph**.

nnodes – число вершин графа,

index[i-1] содержит суммарное количество соседей для первых **i** вершин.

MPI

`edges` содержит упорядоченный список номеров процессов-соседей.

`reorder` – при значении **`1`** системе разрешено менять порядок нумерации процессов.

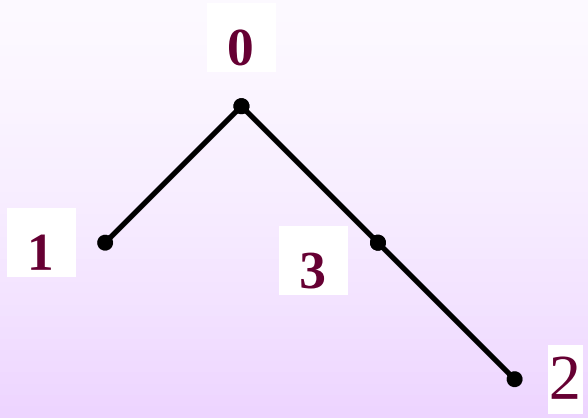
Данная процедура является коллективной, а значит, должна быть вызвана всеми процессами исходного коммуникатора **`comm`**.

MPI

Если **nnodes** меньше числа процессов коммуникатора **comm**, то некоторым процессам вернется значение **MPI_COMM_NULL**, а значит, они не будут принимать участия в создаваемой топологии. Если **nnodes** больше числа процессов коммуникатора **comm**, то вызов процедуры является ошибочным.

MPI

Процесс	Соседи
0	1, 3
1	0
2	3
3	0, 2




```
nnodes=4
index=2, 3, 4, 6
edges=1, 3, 0, 3, 0, 2
```

```
MPI_Graph_create(MPI_COMM_WORLD, nnodes, index,
edges, 1, &comm_graph);
```

MPI

```
int MPI_Graph_neighbors_count
(MPI_Comm comm, int rank, int
*nneighbors)
```

Определение количества **nneighbors** непосредственных соседей процесса с рангом **rank** в графовой топологии, связанной с коммуникатором **comm**.

MPI

```
int MPI_Graph_neighbors(MPI_Comm comm, int rank, int max, int *neighbors)
```

Определяет ранги непосредственных соседей **neighbors** процесса с рангом **rank** в коммуникаторе **comm** в графовой топологии, связанной с данным коммуникатором. Параметр **max** задает ограничение сверху на количество соседей (может быть получено, вызовом **MPI_Graph_neighbors_count**).

MPI

```
int MPI_Graphdims_get(MPI_Comm comm, int *nnodes, int *nedges)
```

Определение числа вершин **nnodes** и числа дуг **nedges** в графовой топологии, связанной с коммуникатором **comm**.

MPI

```
int MPI_Graph_get(MPI_Comm comm,
int maxindex, int maxedges, int
*index, int *edges)
```

Определяет информацию о топологии графа, связанной с коммуникатором **comm**. В массивах **index** и **edges** возвращается описание графовой топологии в том виде, как она задается при создании топологии с помощью процедуры **MPI_Graph_create**.

MPI

Параметры **`maxindex`** и **`maxedges`** задают ограничения на размеры соответствующих массивов (могут быть получены, например, при помощи вызова процедуры **`MPI_Graphdims_get`**).

MPI

```
int MPI_Graph_map(MPI_Comm comm,
int nnodes, int *index, int
*edges, int *newrank)
```

Процедура вычисляет «оптимальное» относительно данной графовой топологии расположение процессов на процессорах (если поддерживается реализацией). В параметре **`newrank`** возвращается новый ранг процесса. Если вызывающий процесс не включен в графовую топологию, то возвращается значение **`MPI_UNDEFINED`**.

MPI

```
#include <stdio.h>
#include "mpi.h"
#define MAXPROC 128
#define MAXEDGES 512
int main(int argc, char **argv)
{
   int rank, rank1, i, size;
   int a, b;
   MPI_Status status;
   MPI_Comm comm_graph;
   int index[MAXPROC], edges[MAXEDGES];
   int num, neighbors[MAXPROC];
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   for(i = 0; i<size; i++) index[i] = size+i-1;
```

MPI

```
    for(i = 0; i<size-1; i++){
        edges[i] = i+1;
        edges[size+i-1] = 0;
    }
    MPI_Graph_create(MPI_COMM_WORLD, size, index, edges, 1,
     &comm_graph);
    MPI_Graph_neighbors_count(comm_graph, rank, &num);
    MPI_Graph_neighbors(comm_graph, rank, num, neighbors);
    for(i = 0; i<num; i++){
        MPI_Sendrecv(&rank, 1, MPI_INT, neighbors[i], 1,
     &rank1, 1, MPI_INT, neighbors[i], 1, comm_graph,
     &status);
        printf("process %d communicate with process %d\n",
     rank, rank1);
    }
    MPI_Finalize();
}
```

MPI

```
int
MPI_Dist_graph_create_adjacent(
MPI_Comm comm, int indegree, int
sources[], int sourceweights[],
int outdegree, int
destinations[], int
destweights[], MPI_Info info,
int reorder, MPI_Comm
*comm_dist_graph)
```

Создание коммуникатора **comm_dist_graph** с топологией распределённого графа.

MPI

Вызывающий процесс не указывает структуру всего графа, а только своих непосредственных соседей (те процессы, с которыми он будет обмениваться данными). Параметр **`indegree`** задаёт число процессов, от которых вызывающий процесс будет получать данные, в массиве **`sources`** задаются номера таких процессов. Параметр **`outdegree`** задаёт число процессов, которым вызывающий процесс будет посылать данные, в массиве **`destinations`** задаются номера таких процессов.

MPI

В массивах **`sourceweights`** и **`destweights`** задаются «веса» (неотрицательные целые числа) соответствующих дуг графа, которые могут использоваться для лучшей маршрутизации сообщений (использование зависит от реализации). Если один и тот же процесс входит в оба списка, то соответствующие веса должны совпадать. Вместо массива весов можно указать предопределённое значение **`MPI_UNWEIGHTED`**, означающее, что все дуги будут иметь одинаковые веса.

MPI

Параметр **`info`** может использоваться для указания способа интерпретации заданных весов. Значения зависят от реализации, например: минимизировать количество дуг или минимизировать сумму весов. Если не требуется, можно использовать значение **`MPI_INFO_NULL`**. Параметр **`reorder`** при значении **`1`** означает, что разрешено менять порядок нумерации процессов для оптимизации их распределения по процессорам; система может использовать значения заданных весов дуг.

MPI

```
int
MPI_Dist_graph_create(MPI_Comm
comm, int n, int sources[], int
degrees[], int destinations[],
int weights[], MPI_Info info,
int reorder, MPI_Comm
*comm_dist_graph)
```

Создание коммуникатора **comm_dist_graph** с топологией распределённого графа. Вызывающий процесс задаёт произвольную часть графа.

MPI

`n` задаёт число описываемых данным процессом начальных вершин дуг, сам список начальных вершин – в массиве **`sources`**. Для каждой начальной вершины в массиве **`degrees`** указывается количество задаваемых дуг. В массиве **`destinations`** в соответствующем порядке задаются конечные вершины дуг. Сначала все конечные вершины для первой начальной вершины, потом – для второй и т.д. В массиве **`weights`** - веса для всех дуг.

MPI

```
int
MPI_Dist_graph_neighbors_count(
MPI_Comm comm, int *indegree,
int *outdegree, int *weighted)
```

Определяется количество **indegree** входящих и **outdegree** исходящих дуг в топологии распределённого графа. В параметре **weighted** возвращается значение **0**, если при создании топологии было задано значение **MPI_UNWEIGHTED**, иначе – значение **1**.

MPI

```
int MPI_Dist_graph_neighbors(
MPI_Comm comm, int maxindegree,
int sources[], int
sourceweights[], int
maxoutdegree, int
destinations[], int
destweights[])
```

В параметре **sources** определяются начальные вершины дуг, входящих в данную, в **destinations** – конечные вершины дуг, исходящих из данной.

MPI

В массивах **`sourceweights`** и **`destweights`** определяются соответствующие веса дуг, если они были заданы при создании топологии. Параметры **`maxindegree`** и **`maxoutdegree`** задают размеры определяемых массивов.

MPI

```
int MPI_Topo_test(MPI_Comm comm,
   int *type)
```

Определение типа топологии, связанной с коммуникатором **comm**.

- **MPI_GRAPH** для графа;
- **MPI_DISTGRAPH** для распределённого графа;
- **MPI_CART** для декартовой топологии;
- **MPI_UNDEFINED** – нет связанной топологии.

MPI

Задание 9: Модифицируйте программу из задания 8 так, чтобы в первой группе осуществляется обмен по кольцевой топологии при помощи сдвига в одномерной декартовой топологии, а в другой – коммуникации по схеме «мастер – рабочие», реализованной при помощи топологии графа.