Производные типы данных

Параллельное программирование

Ограничение стандартных типов

- Коммуникационные операции позволяют посылать и получать *последовательность элементов одного типа*, занимающих **смежные области памяти**.
- При разработке параллельных программ иногда возникает потребность передавать данные разных типов (структуры) или данные, расположенные в несмежных областях памяти (части массивов, не образующих непрерывную последовательность элементов)

В MPI предоставляются два механизма эффективной пересылки данных:

- создание производных типов;
- пересылка упакованных данных.

Производные типы данных

Производные типы данных MPI *не являются в полном смысле* типами данных

Они **не могут** использоваться ни в каких других операциях, кроме *коммуникационных*

Производный тип MPI представляет собой скрытый объект, который специфицирует две вещи:

- последовательность базовых типов
- последовательность смещений

$$Typemap = \{(type_0, disp_0), \ldots, (type_{n-1}, disp_{n-1})\}$$

Сценарий определения и использования производных типов

1. Производный тип строится из предопределенных типов MPI и ранее определенных типов с помощью специальных функций-конструкторов:

   MPI_Type_contiguous, MPI_Type_vector, MPI_Type_hvector, MPI_Type_indexed, MPI_Type_hindexed, MPI_Type_struct.

1. Новый производный тип регистрируется вызовом функции MPI_Type_commit. Только после регистрации новый тип можно использовать.
2. Когда производный тип становится ненужным, он уничтожается функцией MPI_Type_free.

Характеристика любого типа данных в MPI

- протяженность

Протяженность типа определяет, сколько байт переменная данного типа занимает в памяти. Эта величина может быть вычислена как: *адрес последней ячейки данных – адрес первой ячейки данных + длина последней ячейки данных*.

- размер

Размер типа определяет количество реально передаваемых байт. Эта величина равна *сумме длин всех базовых элементов определяемого типа*.

Функция MPI_Type_extent определяет протяженность элемента некоторого типа

```
int MPI_Type_extent (MPI_Datatype datatype, MPI_Aint *extent)
```

- Входные параметры:

`datatype`	тип данных

- Выходные параметры:

`extent`	протяженность элемента заданного типа

Функция MPI_Type_size определяет «чистый» размер элемента некоторого типа

```
int MPI_Type_size (MPI_Datatype datatype, int *size)
```

- Входные параметры:

`datatype`	тип данных

- Выходные параметры:

`size`	размер элемента заданного типа

Самый простой конструктор типов MPI_Type_contiguous

Создает новый тип, элементы которого состоят из указанного числа элементов базового типа, занимающих смежные области памяти.

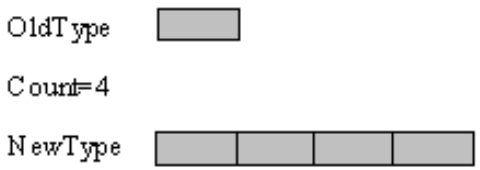
```
int MPI_Type_contiguous (int count, MPI_Datatype
oldtype, MPI_Datatype *newtype)
```

- Входные параметры:

`count`	число элементов базового типа
`oldtype`	базовый тип данных

- Выходные параметры:

`newtype`	новый производный тип данных



Функция MPI_Type_commit регистрирует созданный производный тип

```
int MPI_Type_commit (MPI_Datatype *datatype)
```

- Входные параметры:

`datatype`	тип данных

- Выходные параметры:

`datatype`	тип данных

Только после регистрации новый тип можно использовать в коммуникационных операциях.

Функция MPI_Type_free уничтожает описатель производного типа

```
int MPI_Type_free (MPI_Datatype *datatype)
```

- Входные параметры:

`datatype`	тип данных

- Выходные параметры:

`datatype`	тип данных

Созданный производный тип данных обязательно должен быть уничтожен до конца работы параллельной программы

Функция MPI_Get_elements возвращает число элементов простого типа (базового типа) , содержащихся в сообщении

```
int MPI_Get_elements (MPI_Status *status,
MPI_Datatype datatype, int *count)
```

- Входные параметры:

`status`	статус сообщения
`datatype`	тип элементов сообщения

- Выходные параметры:

`count`	число элементов простого (базового) типа, содержащихся в сообщении

Конструктор типов MPI_Type_vector

Создает новый тип, элементы которого представляют собой несколько равноудаленных друг от друга блоков из одинакового числа смежных элементов базового типа.

```
int MPI_Type_vector (int count, int blocklength, int stride,  MPI_Datatype oldtype, MPI_Datatype *newtype)
```

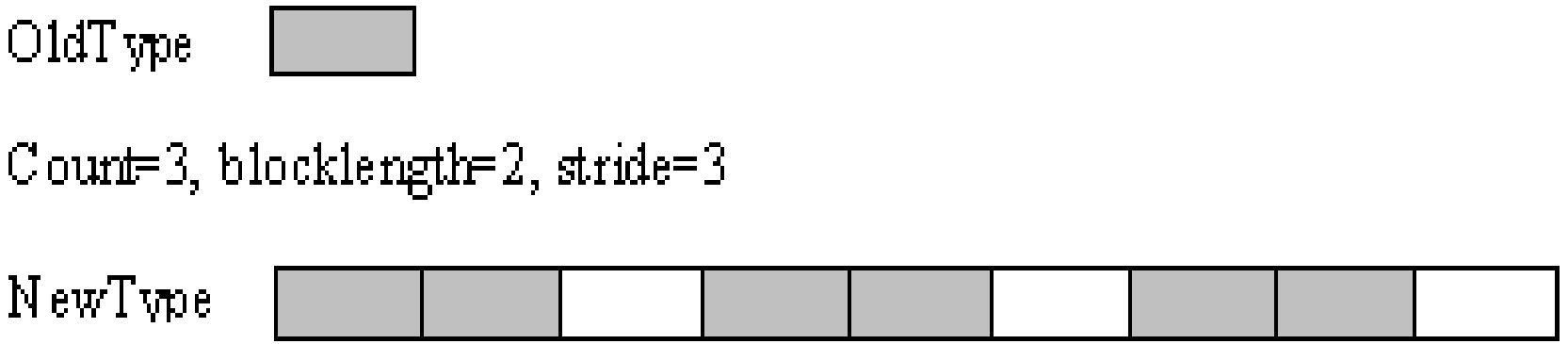
- Входные параметры:

`count`	число блоков
`blocklength`	число элементов базового типа в каждом блоке
`stride`	шаг между началами соседних блоков, измеренный **числом элементов** базового типа
`oldtype`	базовый тип данных

- Выходные параметры:

`newtype`	новый производный тип данных

Конструктор типов MPI_Type_vector



Конструктор типов MPI_Type_hvector

Конструктор типа MPI_Type_hvector расширяет возможности конструктора MPI_Type_vector, позволяя задавать произвольный шаг между началами блоков в **байтах**.

```
int MPI_Type_hvector (int count, int blocklength,
MPI_Aint stride,  MPI_Datatype oldtype, MPI_Datatype
*newtype)
```

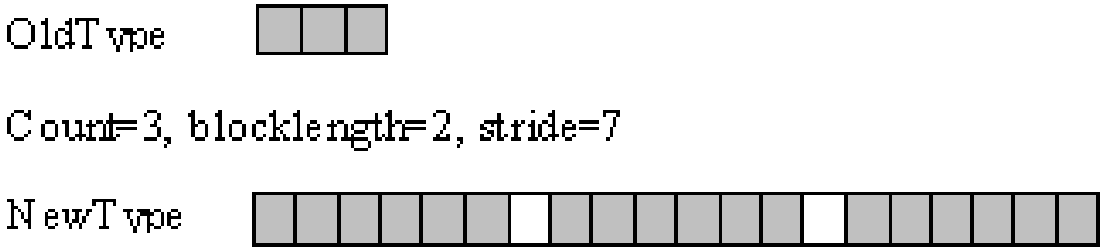
- Входные параметры:

`count`	число блоков
`blocklength`	число элементов базового типа в каждом блоке
`stride`	шаг между началами соседних блоков **в байтах**
`oldtype`	базовый тип данных

- Выходные параметры:

`newtype`	новый производный тип данных

Конструктор типов MPI_Type_hvector



Конструктор типов MPI_Type_indexed

Создает новый тип, элементы которого состоят из произвольных по длине блоков с произвольным смещением блоков от начала размещения элемента.

```
int MPI_Type_indexed (int count, int
*array_of_blocklength, int *array_of_displacements,
MPI_Datatype oldtype, MPI_Datatype *newtype)
```

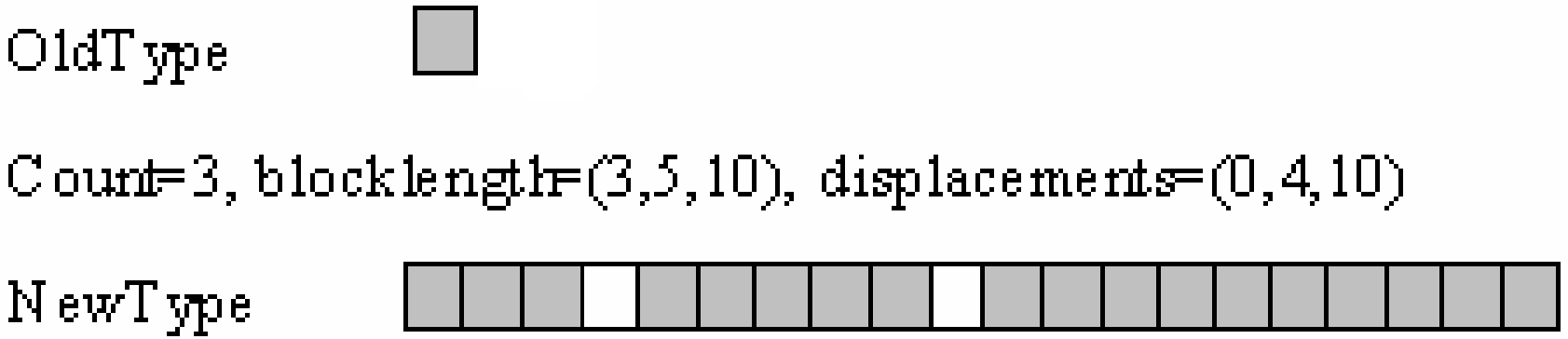
- Входные параметры:

count	число блоков
array_of_b..	массив, содержащий число элементов базового типа в каждом блоке
array_of_di.	массив смещений каждого блока от начала размещения элемента нового типа (элемент)
oldtype	базовый тип данных

- Выходные параметры:

newtype	новый производный тип данных

Конструктор типов MPI_Type_indexed



Конструктор типов MPI_Type_hindexed

Создает новый тип, элементы которого состоят из произвольных по длине блоков с произвольным смещением блоков от начала размещения элемента.

```
int MPI_Type_hindexed (int count, int
*array_of_blocklength, MPI_Aint *array_of_displacements,
MPI_Datatype oldtype, MPI_Datatype *newtype)
```

- Входные параметры:

count	число блоков
array_of_b..	массив, содержащий число элементов базового типа в каждом блоке
array_of_di.	массив смещений каждого блока от начала размещения элемента нового типа (в байтах)
oldtype	базовый тип данных

- Выходные параметры:

newtype	новый производный тип данных

Конструктор типов MPI_Type_hindexed

OldType 

Count=3, blocklength=(2,3,1), displacements=(0,7,18)

NewType 

Конструктор типов MPI_Type_struct

```
int MPI_Type_struct (int count, int
*array_of_blocklength, MPI_Aint
*array_of_displacements,  MPI_Datatype
*array_of_types, MPI_Datatype *newtype)
```

- Входные параметры:

count	число блоков
array_of_b..	массив, содержащий число элементов базового типа в каждом блоке
array_of_di.	массив смещений каждого блока от начала размещения элемента нового типа в байтах
array_of_ty.	массив, содержащий тип элементов в каждом блоке

- Выходные параметры:

newtype	новый производный тип данных

Конструктор типов MPI_Type_struct

oldtypes = (MPI_INT, MPI_SHORT, MPI_CHAR)

count = 3 blocklenght = (1, 6, 4) displacements = (0, 12, 26)

newtype