User Defined MPI Datatypes

ICHEC
Irish Centre for High-End Computing

An Roinn Post, Fiontar agus Nuálaíochta
Department of Jobs, Enterprise and Innovation
sfi science foundation ireland
AN ROINN OIDEACHAIS AGUS SCILEANNA
DEPARTMENT OF EDUCATION AND SKILLS
HEA Higher Education Authority
an tÚdarás um Ard-Oideachas

ICHEC
Irish Centre for High-End Computing

Data Layout and the Describing Datatype Handle

```
struct buff_layout
  { int
  i_val[3];
  double  d_val[5];
  }  buffer;
```

Compiler

```
array_of_types[0] = MPI_INT;
array_of_blocklengths[0] = 3;
array_of_displacements[0] = 0;
array_of_types[1] = MPI_DOUBLE;
array_of_blocklengths[1] = 5;
array_of_displacements[1] = …;

MPI_Type_struct(2,
   array_of_blocklengths,
   array_of_displacements,
   array_of_types, &buff_datatype);

MPI_Type_commit(&buff_datatype);
```

```
MPI_Send(&buffer, 1, buff_datatype, …)
```

&buffer = the start address of the data

the datatype handle describes the data layout

ICHEC
Irish Centre for High-End Computing

Derived Datatypes — Type Maps

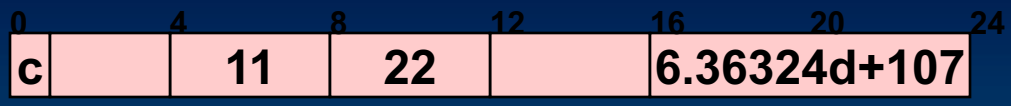
- A derived datatype is logically a pointer to a list of entries:
  - *basic datatype* at *displacement*

basic datatype 0	displacement of datatype 0
basic datatype 1	displacement of datatype 1
...	...
basic datatype n-1	displacement of datatype n-1

ICHEC
Irish Centre for High-End Computing

Derived Datatypes — Type Maps

Example:



derived datatype handle

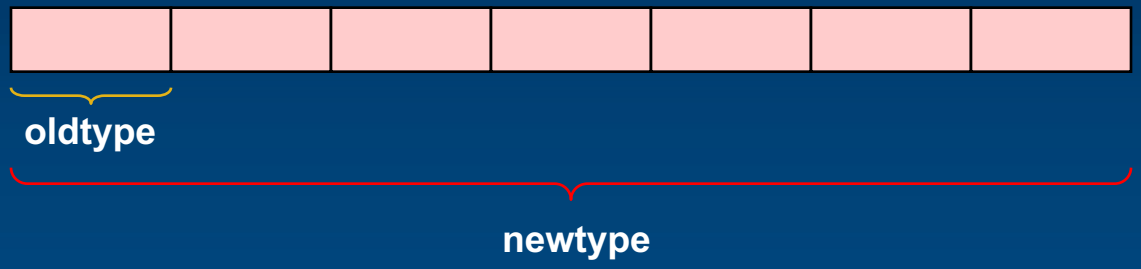
basic datatype	displacement
MPI_CHAR	0
MPI_INT	4
MPI_INT	8
MPI_DOUBLE	16

A derived datatype describes the memory layout of, e.g.,
structures,
common blocks,
subarrays,
some variables in the memory

ICHEC
Irish Centre for High-End Computing

Contiguous Data

- The simplest derived datatype
- Consists of a number of contiguous items of the same datatype



- C:

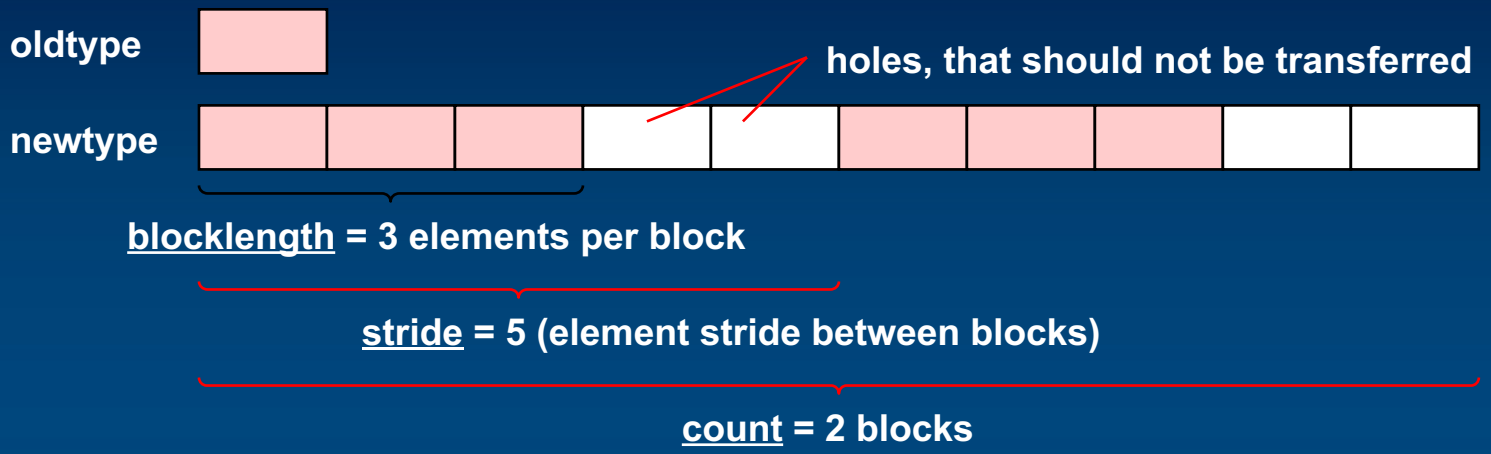
```
int MPI_Type_contiguous(int count,  MPI_Datatype oldtype,
                        MPI_Datatype  *newtype)
```

- Fortran:

```
MPI_Type_contiguous(count, oldtype, newtype, ierror)
            integer :: count, oldtype
            integer :: newtype, ierror
```

ICHEC
Irish Centre for High-End Computing

Vector Datatype

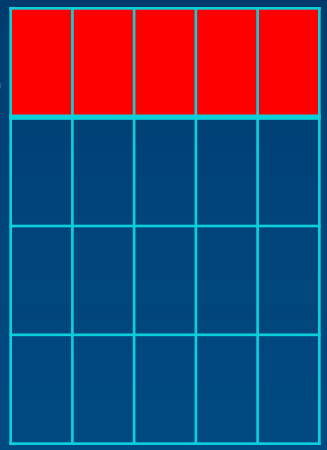


- C:
```
int MPI_Type_vector(int count,  int blocklength, int stride,
    MPI_Datatype oldtype, MPI_Datatype  *newtype)
```
- Fortran:
```
MPI_Type_vector(count, blocklength, stride, oldtype,
        newtype, ierror)
        integer :: count, blocklength, stride
        integer :: oldtype, newtype, ierror
```

ICHEC
Irish Centre for High-End Computing

Sending a row using MPI_TYPE_vector

- C
  - `MPI_Type_vector(1, 5, 1, MPI_INT, ARR_ROW)`
- Fortran
  - `MPI_Type_vector(5, 1, 4, MPI_INT, ARR_ROW)`

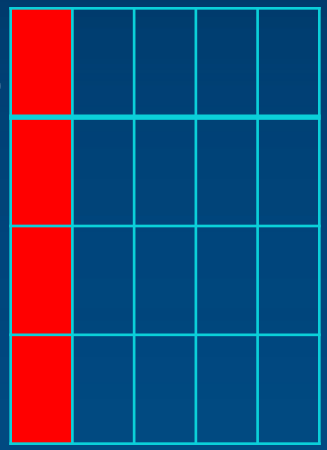


```
MPI_Type_Commit(ARR_ROW)
MPI_Send(&buf …, ARR_ROW…)
MPI_Recv(&buf …, ARR_ROW…)
```

ICHEC
Irish Centre for High-End Computing

Sending a column using MPI_TYPE_vector

- C
  - `MPI_Type_vector(4, 1, 5, MPI_INT, ARR_COL)`
- Fortran
  - `MPI_Type_vector(1, 4, 1, MPI_INT, ARR_COL)`

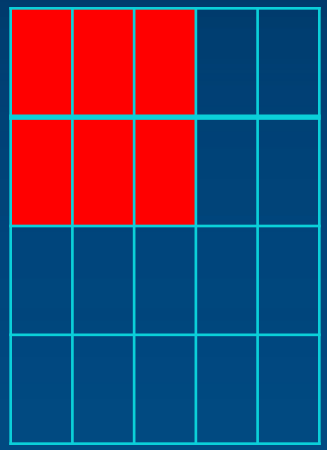


- `MPI_Type_Commit(ARR_COL)`
- `MPI_Send(buf …, ARR_COL…)`
- `MPI_Recv(buf …, ARR_COL…)`

ICHEC
Irish Centre for High-End Computing

Sending a sub-matrix using MPI_TYPE_vector

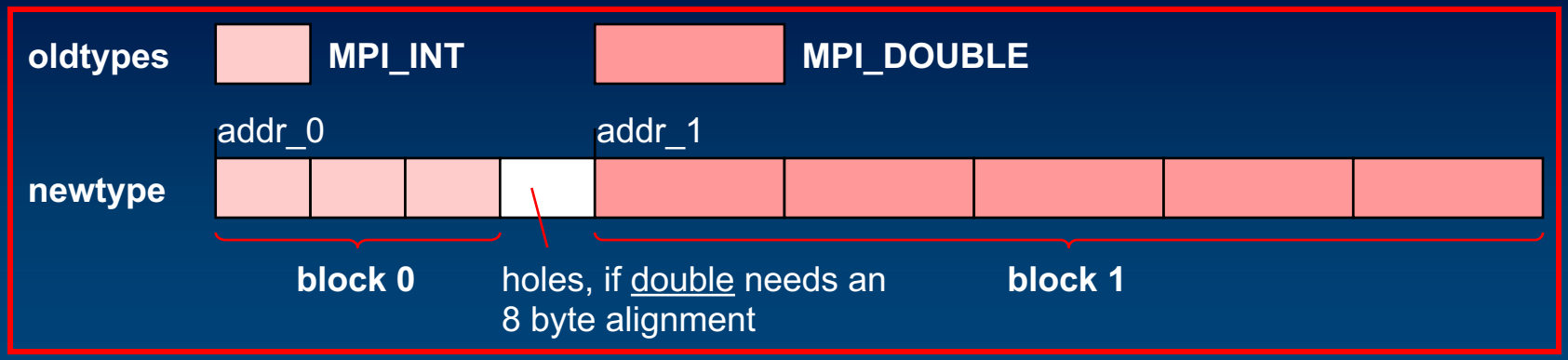
- C
  - `MPI_Type_vector(2, 3, 5, MPI_INT, SUBMAT)`
- Fortran
  - `MPI_Type_vector(3, 2, 4, MPI_INT, SUBMAT)`



- `MPI_Type_Commit(SUBMAT)`
- `MPI_Send(&buf …, SUBMAT…)`
- `MPI_Recv(&buf …, SUBMAT…)`

ICHEC
Irish Centre for High-End Computing

Struct Datatype



- C:
```
int MPI_Type_struct(int count,  int *array_of_blocklengths,
        MPI_Aint *array_of_displacements,
        MPI_Datatype *array_of_types, MPI_Datatype  *newtype)
```

- Fortran:
```
MPI_Type_struct(count, array_of_blocklengths,
      array_of_displacements, array_of_types,newtype,
ierror)
```

count = 2
array_of_blocklengths = (3, 5)
array_of_displacements = (0, addr_1 – addr_0)
array_of_types = (MPI_INT, MPI_DOUBLE)

ICHEC
Irish Centre for High-End Computing

How to compute the displacement

- array_of_displacements[i] := address(block_i) – address(block_0)
- MPI-1
  - C:
```
int MPI_Address(void* location, MPI_Aint *address)
```
  - Fortran:
```
MPI_ADDRESS(location, address, ierror)
    <type>      location(*)
    integer ::  address, ierror
```

ICHEC
Irish Centre for High-End Computing

Committing a Datatype

- Before a datatype handle is used in message passing communication,
  it needs to be committed with `MPI_TYPE_COMMIT`.

- This must be done only once.

- C:
```
int MPI_Type_commit(MPI_Datatype  *datatype);
```

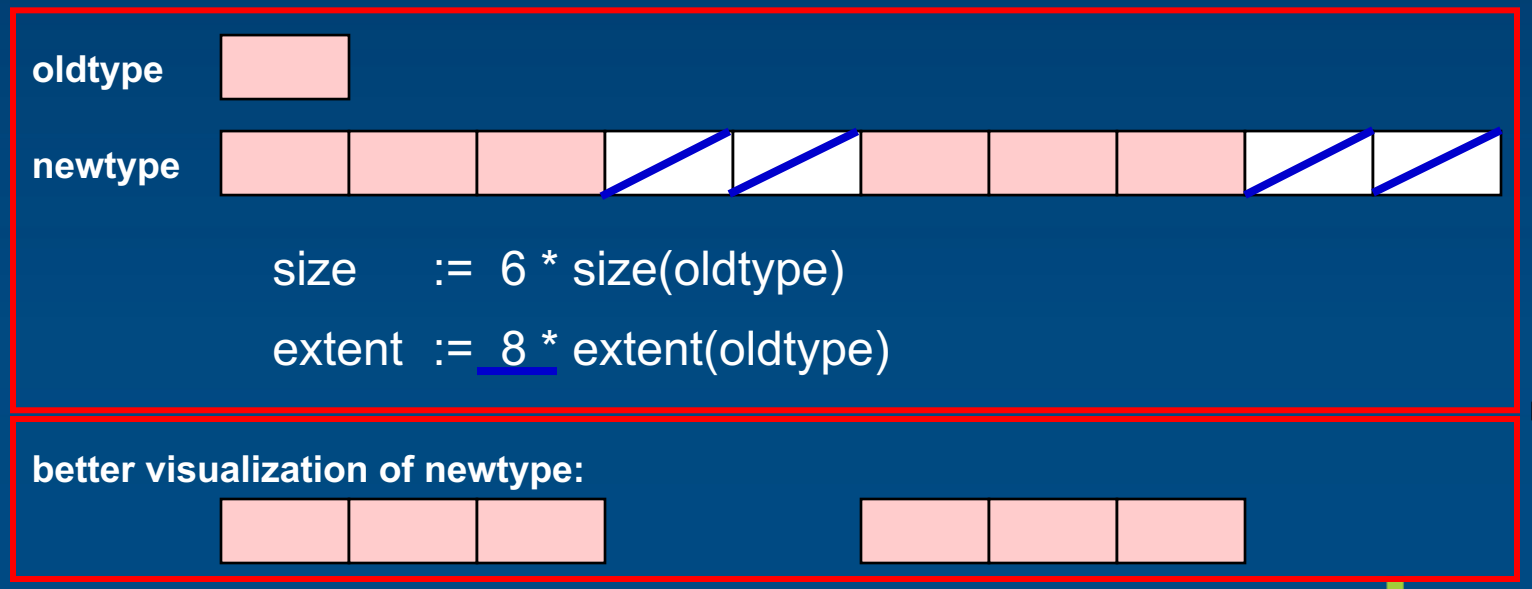
- Fortran:
```
MPI_TYPE_COMMIT(datatype, ierror)
    integer :: datatype, ierror
```

IN-OUT argument

ICHEC
Irish Centre for High-End Computing

Size and Extent of a Datatype

- Size := number of bytes that have to be transferred.
- Extent := spans from first to last byte.
- Basic datatypes: Size = Extent = number of bytes used by the compiler.
- Derived datatypes, an example:



ICHEC
Irish Centre for High-End Computing

Size and Extent of a Datatype

- MPI-1:
  - C: `int MPI_Type_size(MPI_Datatype datatype, int *size)`

    `int MPI_Type_extent(MPI_Datatype datatype, MPI_Aint *extent)`
  - Fortran:

```
MPI_Type_size(datatype, size, ierror)
        integer :: datatype, size, ierror

MPI_Type_extent(datatype, extent, ierror)
        integer :: datatype, extent, ierror
```