Parallel Programming

Prof. **Paolo Bientinesi**

pauldj@aices.rwth-aachen.de

WS 18/19

Datatypes

- So far: { `memory address, count, datatype` }

  ⇒ only contiguous entries

  only entries of the same MPI type

- What if . . .
  non contiguous data and/or **non elementary datatypes**?

  Examples: vector from matrix, submatrix, descriptor+data, ...

- Entirely wrong idea: ~~many small messages~~

- MPI **derived datatypes**: "Create, commit, use, free"

```
MPI_Datatype newtype;
MPI_Type_*( ..., &newtype);
MPI_Type_commit( &newtype );

// code

MPI_Type_free( &newtype );
```

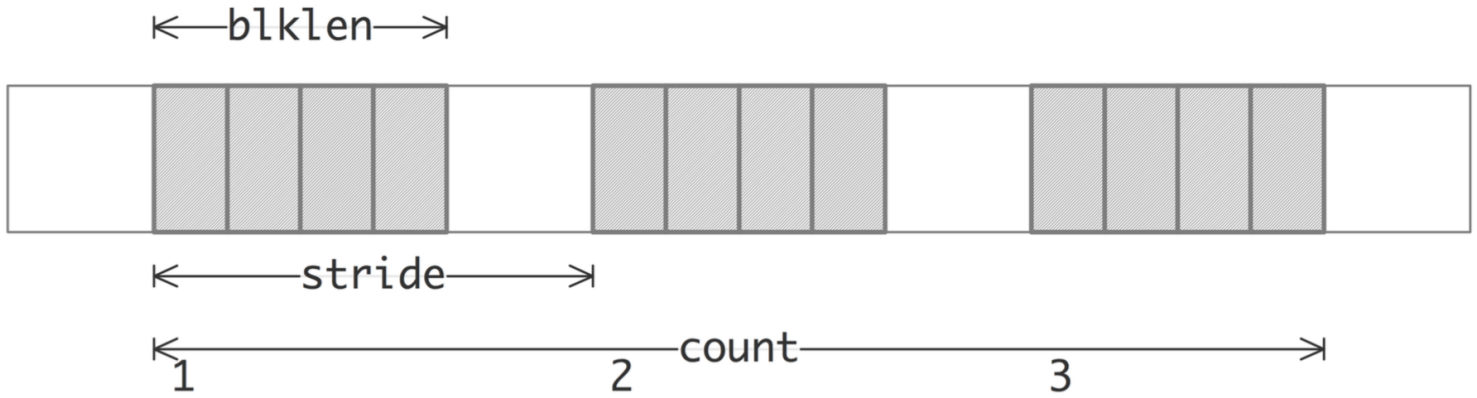
- `int MPI_Type_contiguous(`
  `int count, MPI_Datatype old_type, MPI_Datatype *new_type )`

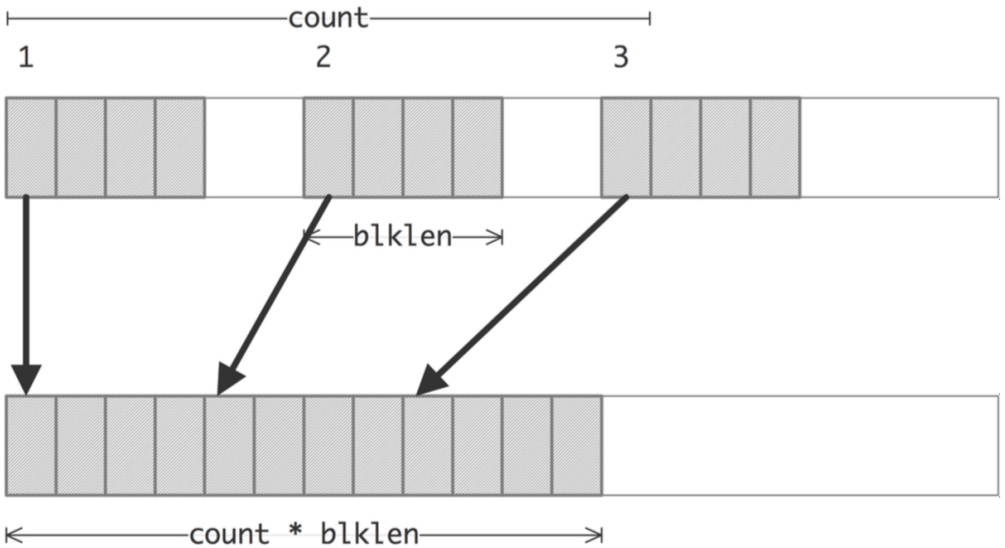
Same as sending `count` entries of `old_type`

Reference

- "Parallel Programming in MPI and OpenMP"
  Victor Eijkhout, Texas dvanced Computing Center
  available online:
  `http://pages.tacc.utexas.edu/~eijkhout/pcse/html/index.html`

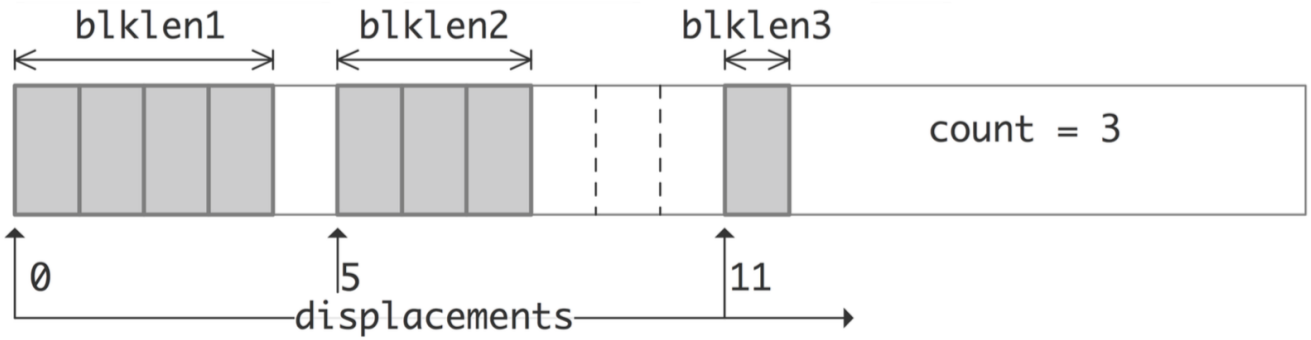
- `int MPI_Type_vector(`
  `int count, int blklen, int stride,`
  `MPI_Datatype old_type, MPI_Datatype *new_type )`



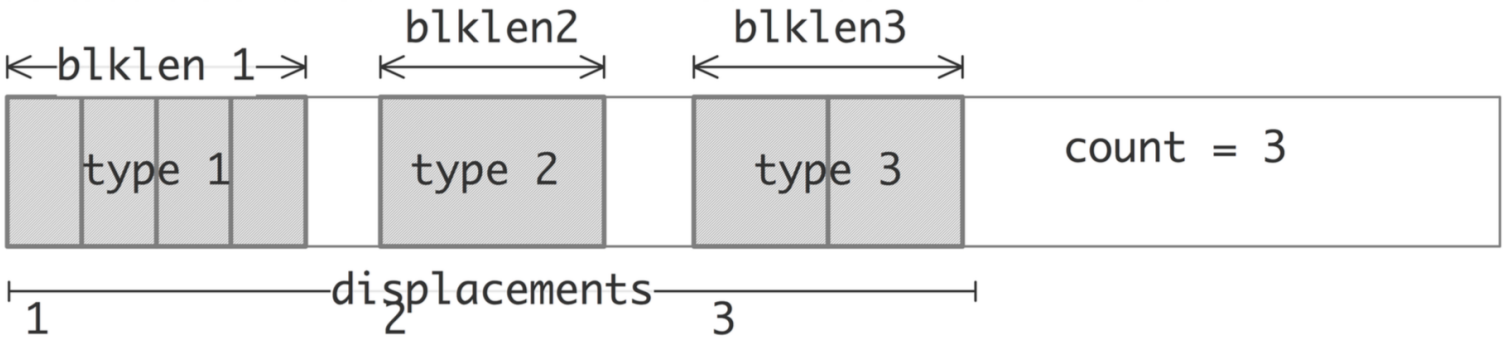


Note (once again): Receive type can be different from Send type

- `int MPI_Type_indexed(`
  `int count, int blklens[], int displacements[],`
  `MPI_Datatype old_type, MPI_Datatype *new_type )`



- `int MPI_Type_create_struct(`
  `int count, int blklen[], MPI_Aint displacements[],`
  `MPI_Datatype types[], MPI_Datatype *new_type )`



Exercise

The `root` process owns an array `v` of length 10*`p`,
where `p` is the number of processes participating in the computation.

The entries at position 0, `p`, `2p`, ..., `9p`, need to be sent to process 0;

the entries at position 1, `p+1`, `2p+1`, ..., `9p+1`, need to be sent to process 1;

⋮

the entries at position `p-1`, `2p-1`, ..., `10p-1`, need to be sent to process `p-1`;

Write a program that performs this distribution using a vector datatype for the send, and a contiguous buffer for the receive.

More

- `MPI_Type_create_subarray`
  Subarray of a regular, multidimensional array

- `MPI_Type_create_darray`
  Distributed array

...and more

- `MPI_Type_extent`
  Memory span by a datatype (extension of `sizeof`)

- `MPI_Pack, MPI_Unpack`
  Pack/unpack memory into contiguous memory

- `MPI_Type_create_resized`
  Adjust strides
  :

Communicators

```
MPI_Comm_split(
    MPI_Comm comm,          <- EVERYBODY involved!
    int color,              <- where do I belong
    int key,                <- for the new rank
    MPI_Comm* newcomm)
```

```
MPI_Comm_dup
MPI_Comm_create
MPI_Comm_create_group
MPI_Group_incl
...
MPI_Group_union, MPI_Group_intersection
```

Topologies: Who are my neighbors?

Cartesian topology

```
MPI_Cart_create(
    old_communicator,
    ndims,                  <- number of dimensions of the grid
    dims[],                 <- size of each dimension
    periods[],              <- periodic boundaries?
    reorder,
    *cart_communicator      <- new communicator
    )
```

Who am I?

```
MPI_Comm_rank(   cart_COMM, &rank );
MPI_Cart_coords( cart_COMM, rank, ndims, coords[] );
```

My neighbors? No neighbor: `MPI_PROC_NULL`

```
MPI_Cart_shift( cart_COMM, direction, displacement, *rank_source, *rank_dest )
```

Other topologies: `MPI_Graph_create`