MPI lecture and labs 4

Victor Eijkhout

2016

Complicated data

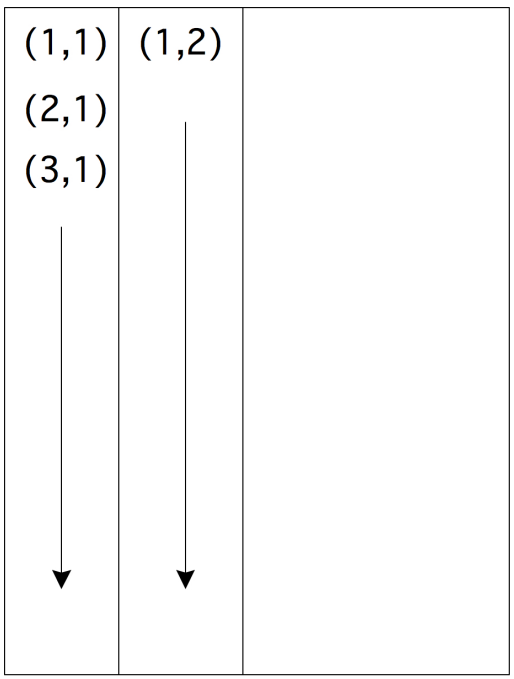
Table of Contents

Non-contiguous data

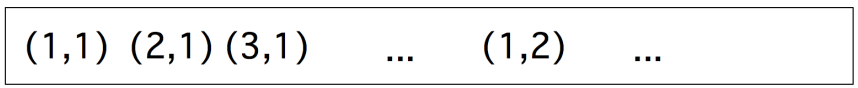
Matrix in column storage:

- Columns are contiguous
- Rows are not contiguous

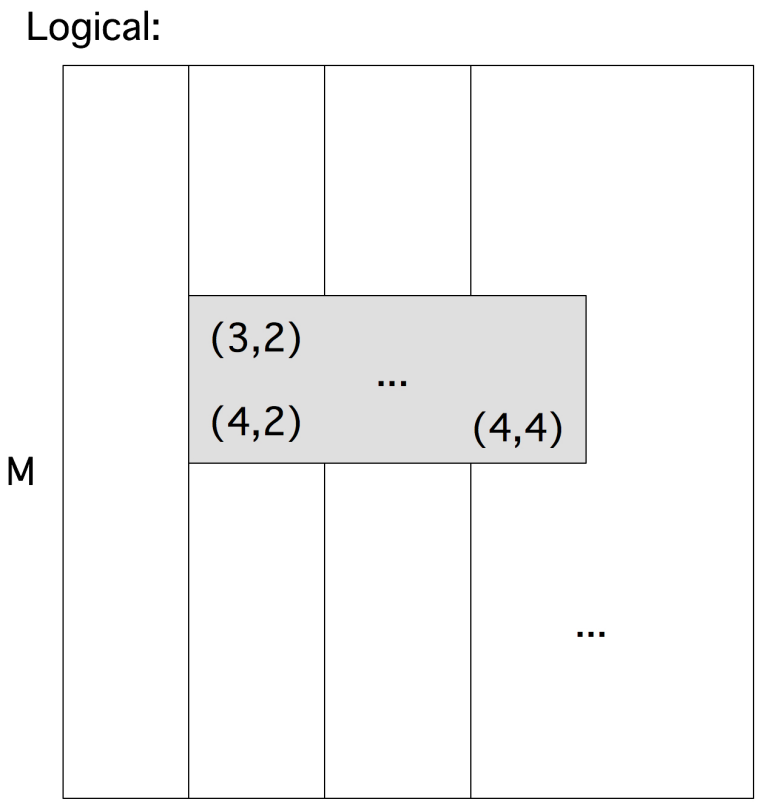
Logical:

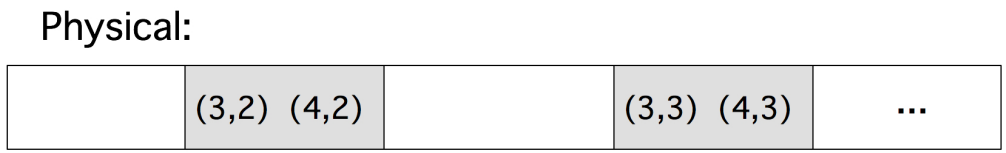


Physical:



Submatrix storage

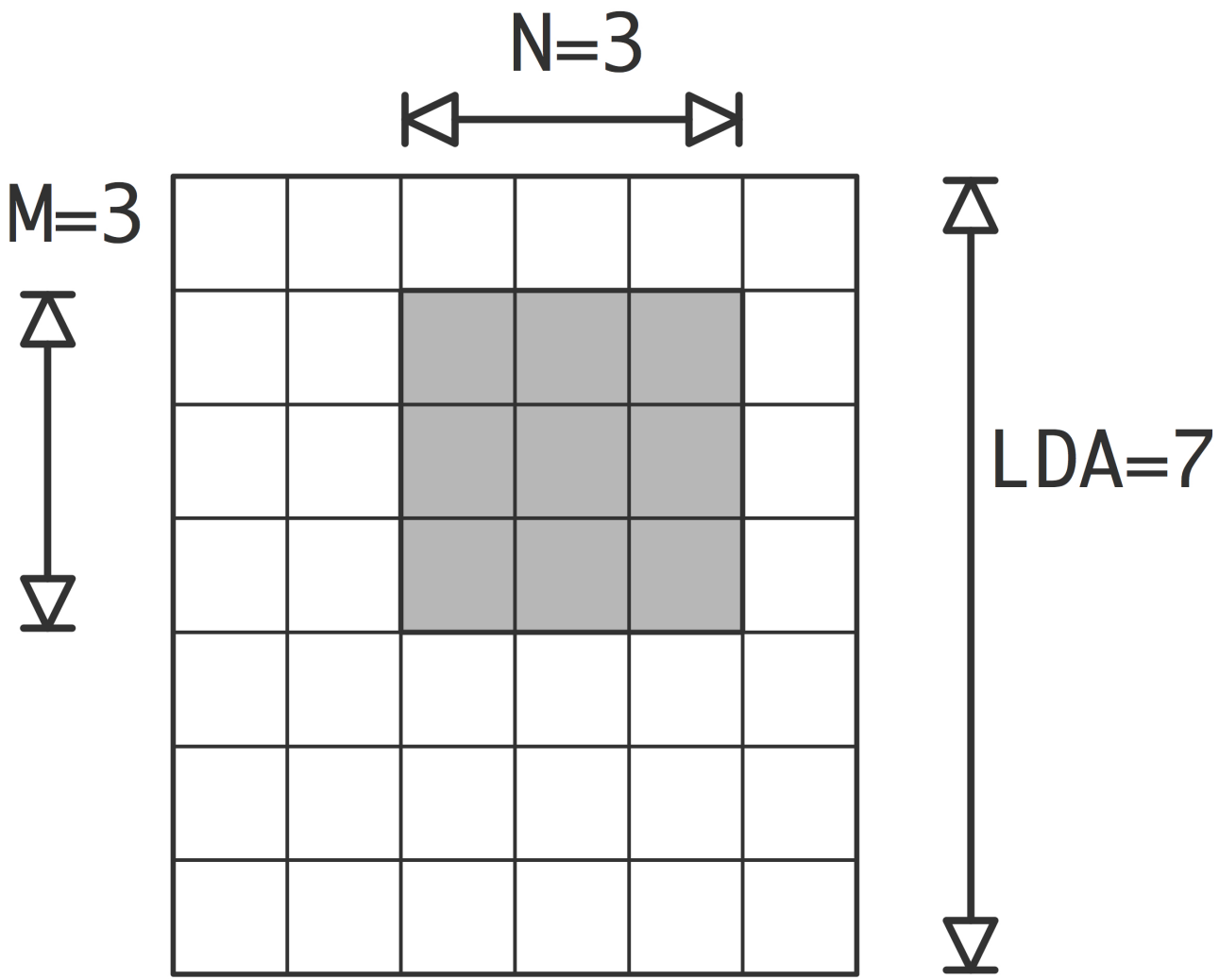




- Location of first element
- Stride, blocksize

BLAS/Lapack storage

Three parameter description:



Exercise 1

How would you describe the memory layout of a submatrix, if the whole matrix has size $M \times N$ and the submatrix $m \times n$?

Datatypes in MPI

We need data structures with gaps, or heterogeneous types.
MPI allows for recursive construction of data types.

- Elementary types
- Derived types
- Packed data

Table of Contents

Elementary datatypes

C/C++	Fortran
MPI_CHAR	MPI_CHARACTER
MPI_UNSIGNED_CHAR	
MPI_SIGNED_CHAR	
	MPI_LOGICAL
MPI_SHORT	
MPI_UNSIGNED_SHORT	
MPI_INT	MPI_INTEGER
MPI_UNSIGNED	
MPI_LONG	
MPI_UNSIGNED_LONG	
MPI_FLOAT	MPI_REAL
MPI_DOUBLE	MPI_DOUBLE_PRECISION
MPI_LONG_DOUBLE	
	MPI_COMPLEX
	MPI_DOUBLE_COMPLEX

How to use derived types

Create, commit, use, free:

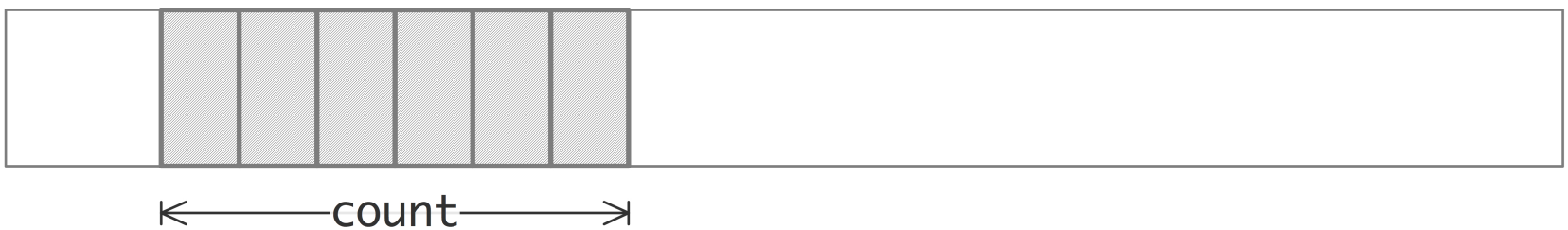
```
MPI_datatype newtype;
MPI_Type_xxx( .... &newtype);
MPI_Type_commit ( &newtype );

  // code using the new type

MPI_Type_free ( &newtype );
```

Contiguous type

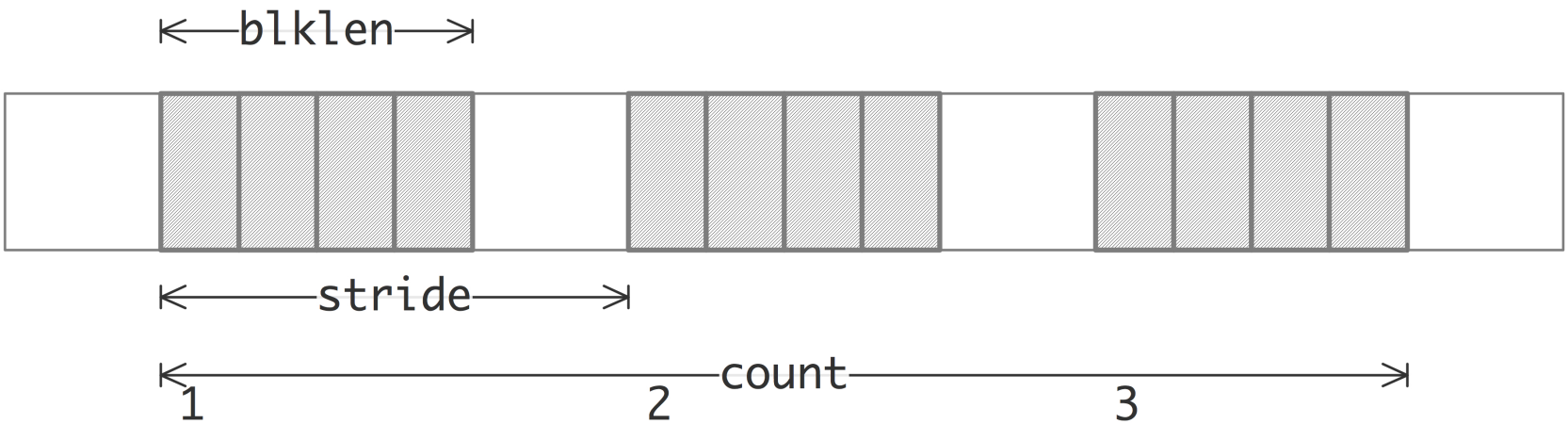
```
int MPI_Type_contiguous(
  int count, MPI_Datatype old_type, MPI_Datatype *new_type_p)
```



This one is indistinguishable from just sending `count` instances of the `old_type`.

Vector type

```
int MPI_Type_vector(
  int count, int blocklength, int stride,
  MPI_Datatype old_type, MPI_Datatype *newtype_p
);
```



Used to pick a regular subset of elements from an array.

Note: send and receive type not the same!

Exercise 2

Let processor 0 have an array x of length $10P$, where P is the number of processors. Elements $0, P, 2P, \ldots, 9P$ should go to processor zero, $1, P+1, 2P+1, \ldots$ to processor 1, et cetera. Code this as a sequence of send/recv calls, using a vector datatype for the send, and a contiguous buffer for the receive.

For simplicity, skip the send to/from zero. What is the most elegant solution if you want to include that case?

For testing, define the array as $x[i] = i$.

Exercise 3

Allocate a matrix on processor zero, using Fortran column-major storage. Using P sendrecv calls, distribute the rows of this matrix among the processors.

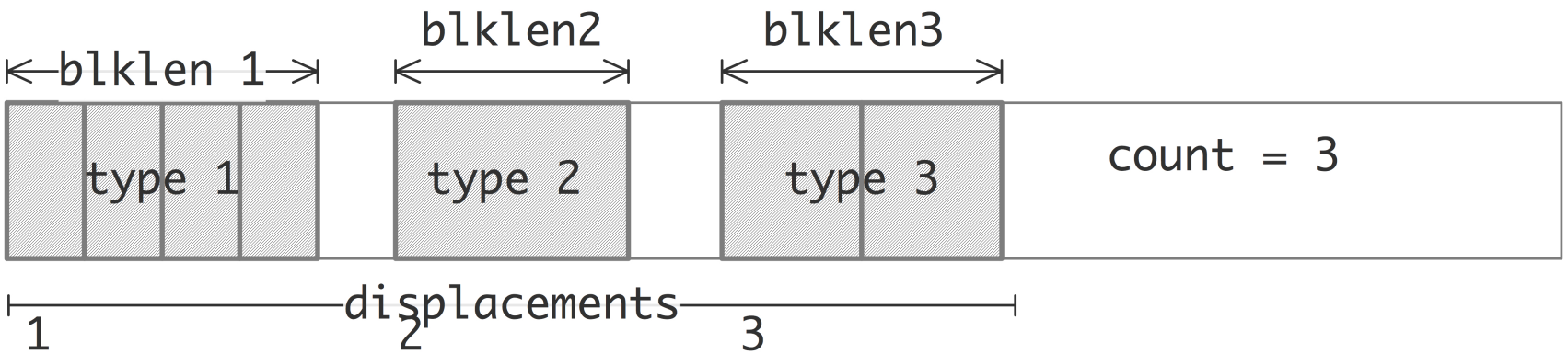
Indexed type

```
int MPI_Type_indexed(
  int count, int blocklens[], int indices[],
  MPI_Datatype old_type, MPI_Datatype *newtype);
```

Also `hindexed` with byte offsets.

Heterogeneous: Structure type

```
int MPI_Type_create_struct(
  int count, int blocklengths[], MPI_Aint displacements[],
  MPI_Datatype types[], MPI_Datatype *newtype);
```



This gets very tedious. . .

Table of Contents

Packing into buffer

```
int MPI_Pack(
  void *inbuf, int incount, MPI_Datatype datatype,
  void *outbuf, int outcount, int *position,
  MPI_Comm comm);

int MPI_Unpack(
  void *inbuf, int insize, int *position,
  void *outbuf, int outcount, MPI_Datatype datatype,
  MPI_Comm comm);
```

Example