Monday, 01/23/12:

Ch. 6 Grouping Data for Communications:

Motivating example:

trapezoidal rule: $a, b \in \mathbb{R}$, $n \in \mathbb{Z}$, want to broadcast them, but in one MPI call, as opposed to 3:

```
MPI_Bcast(&a, 1, MPI_DOUBLE, ...
MPI_Bcast(&b, 1, MPI_DOUBLE, ...
MPI_Bcast(&n, 1, MPI_INT, ...
```

The idea to combine data might be to define a vector, but this would only work for homogeneous datatypes, like a and b together.

How to use this idea to compute min and max in one call:

Have `MPI_Reduce(&l_x, &x_min, 1, MPI_DOUBLE, MPI_MIN, ...)`
`MPI_Reduce(&l_x, &x_max, ..., MPI_MAX, ...)`

replace this by:

```
double v[2], vminmax[2]
v[0] = l_x, v[1] = -l_x        // negative l_x
MPI_Reduce(v, vminmax, 2, MPI_DOUBLE, MPI_MIN, ...)
x_min = vminmax[0]
x_max = -vminmax[1]            // negative sign
```

We need to handle combinations of any datatypes $\Longrightarrow$ Done by

Derived datatypes

Idea of chapter 6: from general to specific!

MPI has intrinsic datatypes like MPI_INT, MPI_DOUBLE, etc.

Based on these, we can construct new types that are then created at runtime, potentially optimized. ($\rightarrow$ Ch. 7).

What is the essence of a datatype?

→ type signature = Sequence of pairs of (type, displacement)

$\{(t_0, d_0), (t_1, d_1), (t_2, d_2), \ldots, (t_{n-1}, d_{n-1})\}$

t_i = type of the ith data (can be derived itself => doubly derived datatypes possible)

d_i = displacement in memory location relative to starting address of 0-th data

Most general MPI command:

```
MPI_Type_struct (
        int          count
        int          *bl        (= block lengths)      } return
        MPI_Aint     *dl        (= displacements)      } of length
        MPI_Datatype *tl                                } count
        MPI_Datatype *new_mpi_t );
```

C convention for new datatype

Ex.: Pacheco pp. 92-94

count = 3

bl[0] = bl[1] = bl[2] = 1 (⟸ a, b, n scalars)

tl[0] = tl[1] = MPI_DOUBLE (for a and b)

tl[2] = MPI_INT (for n)

dl[0] = 0 (displacement of 0th data from itself)

/* displacement of &b from &a : */

MPI_Address (&a, &start_address)

MPI_Address (&b, &address) } MPI_Aint

dl[1] = address − start_address (= displacement in units Aint)

/* displacement of &n from &a : */

MPI_Address (&n, &address)

dl[2] = address − start_address

MPI_Type_struct (count, bl, tl, dl, &new_mpi_t)

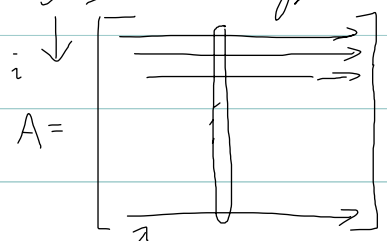
MPI_Type_commit (&new_mpi_t)

↳ necessary to actually create the derived type;
only needed for final one in case of doubly derived types

There are several earlier derived types for arrays of homogenous data

Following Pacheco, consider example of double A[10][10] assumed to be consecutive in memory.
Have A on Process 0 and want to send one column to another process

MPI_Type_vector (
  int count
  int bl
  int stride
  MPI_Datatype el_type → of the original array
  MPI_Datatype & new_mpi_t)



count = 10 = number of rows in A
bl = 1
stride = 10 = number of columns in A

Notice: A[10][10] is in row-oriented storage in C !

MPI_Type_vector(10, 1, 10, MPI_DOUBLE, & col_type)

MPI_Type_commit (& col_type)

Usage in Pacheco:

if (id == 0)
  MPI_Send (&(A[0][j]), 1, col_type, j ← destination, 0, MPI_COMM_WORLD)
  address of first element in jth column of A

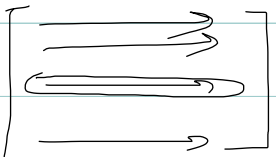
if (id == j)
  MPI_Recv (&(A[0][j]), 1, col_type, 0 ← source, 0, MPI_COMM_WORLD, & status)

Much better example: receive not into A[10][10] on Proc j, but receive exactly the data sent (10 DOUBLE) into a vector

if (id == j)
  double z[10]
  MPI_Recv (z, 10, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &status)

The only criterion for whether a Send and Recv work together is that the type signatures of sent and received data must match.

Some other commands:

Ex.: derived type for row of A $A =$ 

MPI_Type_vector(10, 1, 1, MPI_DOUBLE, & row_type)

or:

MPI_Type_vector(1, 10, 1, MPI_DOUBLE, & row_type)

→ For consecutive data, can also use

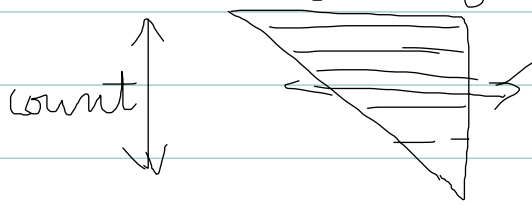
MPI_Type_contiguous(
    int count
    MPI_Datatype old_type
    MPI_Datatype *new_type)

Here: MPI_Type_contiguous(10, MPI_DOUBLE, & row_type)

Generalize int bl into vector

MPI_Type_indexed(
    int count,
    int *bl
    int *dl
    MPI_Datatype old_type
    MPI_Datatype *new_type)

can be used for triangular matrix:



→ see Fabrew

Finish Ch. 6:

Special intrinsic type is MPI_PACKED

Created by MPI_Pack, disassembled by MPI_Unpack

→ Parlero example for a, b, n

Only situation for use:

if size of a variable is part of the message.

Pseudo code:

- Pack size, then pack array of data.
- Send whole package
- Unpack size => allocate space for array of necessary size, then unpack data into this array.