MPI: Numerical Integration, P2P and Collective Communication

Kameswararao Anupindi

Department of Mechanical Engineering
Indian Institute of Technology Madras (IITM)

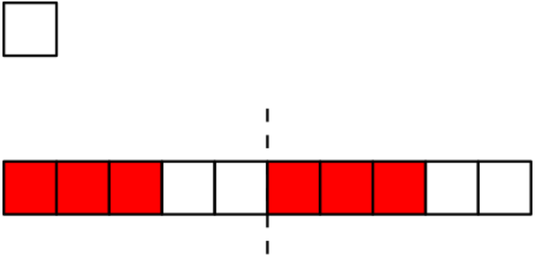
March, 2024

Other Derived Data Types: Contiguous data

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

MPI_TYPE_CONTIGUOUS(count, oldtype, newtype, ierror)
```

Other Derived Data Types: Vector data



- blocklength =
- count =
- stride =

Where would such a pattern of values that has blocks and gaps is needed?

Other Derived Data Types: Vector data

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

MPI_TYPE_VECTOR(count, blocklength, stride, oldtype, newtype, ierror)
```

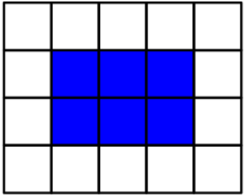
Representing 2D arrays in C

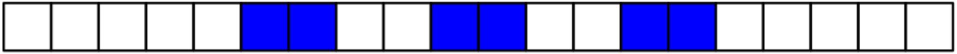
$$A = \begin{bmatrix} a[0][3] & \cdots & \cdots & a[3][3] \\ a[0][2] & \cdots & \cdots & \cdots \\ a[0][1] & a[1][1] & \cdots & \cdots \\ a[0][0] & a[1][0] & a[2][0] & a[3][0] \end{bmatrix}$$

$$A = \begin{bmatrix} 4 & \cdots & \cdots & 16 \\ 3 & \cdots & \cdots & \cdots \\ 2 & 6 & \cdots & \cdots \\ 1 & 5 & 9 & 13 \end{bmatrix}$$

1	2	3	4	5											16

Extracting a sub-array in C





- blocklength =
- count =
- stride =

Sending a sub-array

```
MPI_Type_vector(count, blocklength, stride, oldtype, &newtype)

MPI_Send(&x[1][1], 1, &newtype, ...)
```

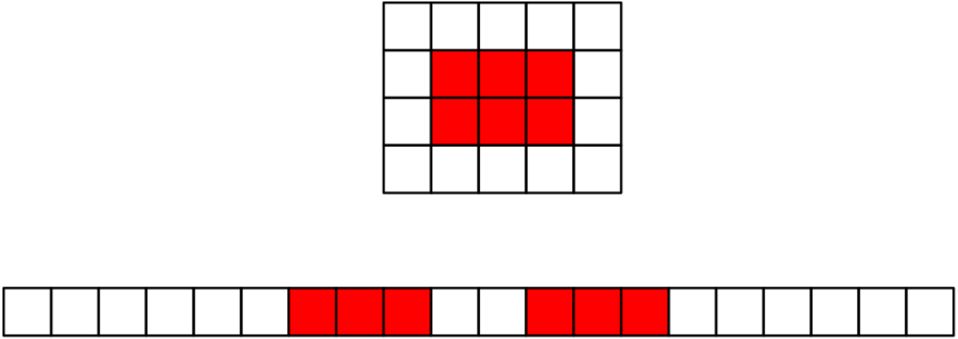
Representing 2D arrays in FORTRAN

$$A = \begin{bmatrix} a[1][4] & \cdots & \cdots & a[4][4] \\ a[1][3] & \cdots & \cdots & \cdots \\ a[1][2] & a[2][2] & \cdots & \cdots \\ a[1][1] & a[2][1] & a[3][1] & a[4][1] \end{bmatrix}$$

$$A = \begin{bmatrix} 13 & \cdots & \cdots & 16 \\ 9 & \cdots & \cdots & \cdots \\ 5 & 6 & \cdots & \cdots \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

1	2	3	4	5											16

Extracting a sub-array in FORTRAN



- blocklength =
- count =
- stride =

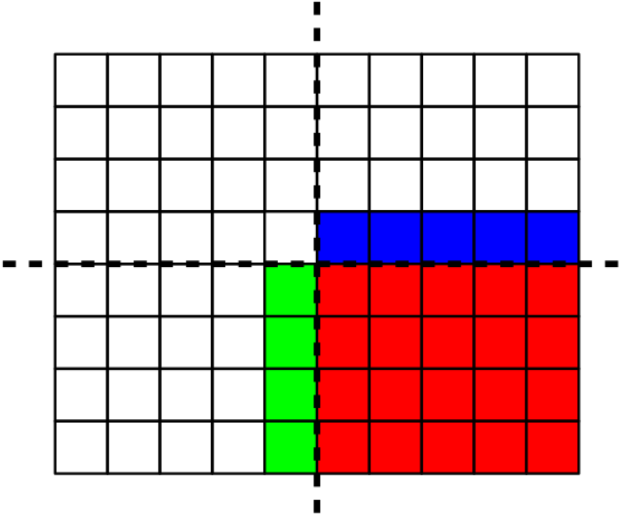
Sending a sub-array

```
MPI_Type_vector(count, blocklength, stride, oldtype, &newtype)

MPI_Send(&x[2][2], 1, &newtype, ...)
```

Remember to commit the new type!

2D or 3D Jacobi/GS



Bubble Sort

```
void Bubble_sort(int a[], int n,)
{
int list_length, i, temp;

for (list_length = n; list_length >= 2; list_length--)
  for (i = 0; i < list_length-1; i++)
    if (a[i] > a[i+1])
      {
        temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
      }
}
```

Odd-Even Transposition Sort

```
for (pass = 0; pass < n; pass++)
  if (pass%2 == 0) {
   for (i = 1; i < n; i += 2)
     if (a[i-1] > a[i]) {
       temp = a[i];
       a[i] = a[i-1];
       a[i-1] = temp;
       }
  }else {
   for (i = 1; i < n-1; i += 2)
     if (a[i] > a[i+1]) {
       temp = a[i];
       a[i] = a[i+1];
       a[i+1] = temp;
       }
  }
```

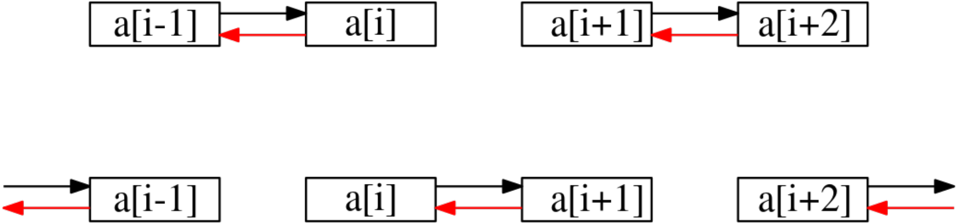
Odd-Even Transposition Sort contd...

```
Even-pass: (a[0], a[1]), (a[2], a[3]), (a[4], a[5]), ...

Odd-pass:         (a[1], a[2]), (a[3], a[4]), (a[5], a[6]), ...
```

```
Given list:  5, 9, 4, 3
Even-pass:  (5, 9),  (4, 3) -> 5, 9, 3, 4
Odd-pass:   5, (9, 3), 4    -> 5, 3, 9, 4
Even-pass:  (5, 3), (9, 4)  -> 3, 5, 4, 9
Odd-pass:   3, (5, 4), 9    -> 3, 4, 5, 9
```

Parallel Odd-Even Transposition sort for $n = p$



Parallel Odd-Even Transposition sort for $n >> p$

	Process 0	Process 1	Process 2	Process 3
Given	15, 11, 9, 16	3, 14, 8, 7	4, 6, 12, 10	5, 2, 13, 1
After local sort	9, 11, 15, 16	3, 7, 8, 14	4, 6, 10, 12	1, 2, 5, 13
After phase 0	3, 7, 8, 9	11, 14, 15, 16	1, 2, 4, 5	6, 10, 12, 13
After phase 1	3, 7, 8, 9	1, 2, 4, 5	11, 14, 15, 16	6, 10, 12, 13
After phase 2	1, 2, 3, 4	5, 7, 8, 9	6, 10, 11, 12	13, 14, 15, 16
After phase 3	1, 2, 3, 4	5, 6, 7, 8	9, 10, 11, 12	13, 14, 15, 16

Parallel Odd-Even Transposition Sort – Algorithm

```
Sort local elements;
for (pass = 0; pass < comm_sz; pass++) {
    partner = compute_partner(pass, my_rank);
    if (I am active) {
        Send my elements to partner;
        Receive elements from partner;
        if (my_rank < partner)
          Keep smaller elements;
        else
          Keep larger elements;
   }
}
```

Safety in MPI programs

```
MPI_Send(my_elements, n/p, MPI INT, partner, 0, comm);
MPI_Recv(temp_elements, n/p, MPI INT, partner, 0, comm, &status);
```

- A program that relies on MPI-provided buffering is **unsafe**
- How can we tell if a program is unsafe?
- How to modify the communication to make it safe?

How can we tell if a program is safe?

```
MPI_Send --> MPI_Ssend
```

```
MPI_Ssend (
          void*           message_buffer_p,
          int             message_size,
          MPI_Datatype    message_type,
          int             dest_process,
          int             tag,
          MPI_Comm        communicator);
```

How to modify the communication to make it safe?

```
MPI_Send(msg, size, MPI_INT, (myid+1)%p, 0, comm);
MPI_Recv(new_msg, size, MPI_INT, (myid+p-1)%p, 0, comm, &status);
```

```
if (myid % 2 == 0){
  MPI_Send(msg, size, MPI_INT, (myid+1)%p, 0, comm);
  MPI_Recv(new_msg, size, MPI_INT, (myid+p-1)%p, 0, comm, &status);
}
else{
  MPI_Recv(new_msg, size, MPI_INT, (myid+p-1)%p, 0, comm, &status);
  MPI_Send(msg, size, MPI_INT, (myid+1)%p, 0, comm);
}
```

MPI alternative to manual scheduling

```
int MPI_Sendrecv(
          void*         send_buf_p,
          int           send_buf_size,
          MPI_Datatype  send_buf_type,
          int           dest,
          int           send_tag,
          void*         recv_buf_p,
          int           recv_buf_size,
          MPI_Datatype  recv_buf_type,
          int           source,
          int           recv_tag,
          MPI_Comm      communicator,
          MPI_Status*   status_p);
```

For the same send/recv buffers

```
int MPI_Sendrecv_replace(
          void*         buf_p,
          int           buf_size,
          MPI_Datatype  buf_type,
          int           dest,
          int           send_tag,
          int           source,
          int           recv_tag,
          MPI_Comm      communicator,
          MPI_Status*   status_p);
```