Parallel Programming

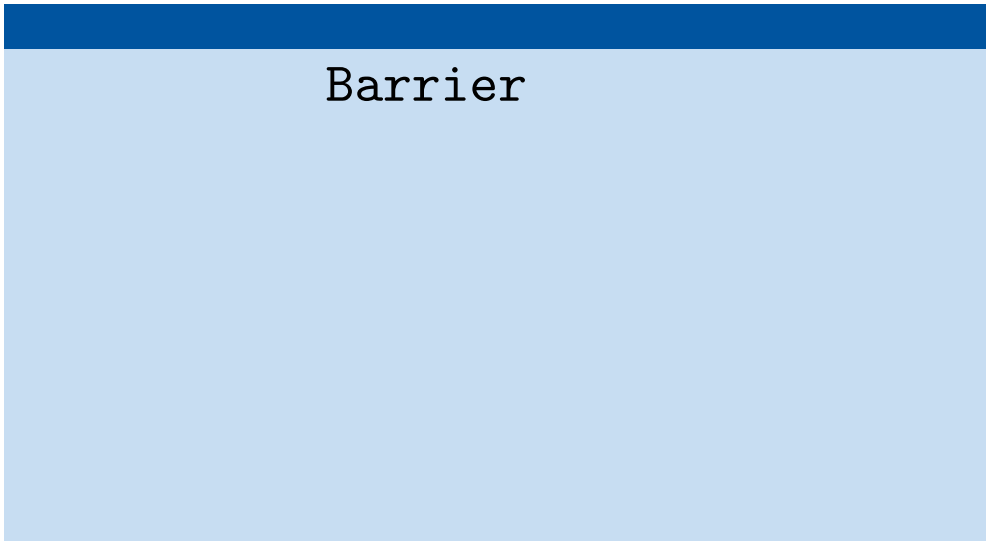
Prof. **Paolo Bientinesi**

pauldj@aices.rwth-aachen.de

WS 18/19

High Performance and
Automatic Computing

Collective Communication



Collective Communication

Barrier

Broadcast $\leftrightarrow$ Reduce

Collective Communication

Barrier

Broadcast $\leftrightarrow$ Reduce

Scatter $\leftrightarrow$ Gather

Collective Communication

	Barrier	
Broadcast	$\leftrightarrow$	Reduce
Scatter	$\leftrightarrow$	Gather
Allgather	$\leftrightarrow$	Reduce-scatter

Collective Communication

Barrier

Broadcast $\leftrightarrow$ Reduce

Scatter $\leftrightarrow$ Gather

Allgather $\leftrightarrow$ Reduce-scatter

Allreduce

Collective Communication

Barrier

Broadcast $\leftrightarrow$ Reduce

Scatter $\leftrightarrow$ Gather

Allgather $\leftrightarrow$ Reduce-scatter

Allreduce

Alltoall

$\vdots$

Collective Communication

```
          Barrier
Broadcast  ↔  Reduce
  Scatter  ↔  Gather
Allgather  ↔  Reduce-scatter
       Allreduce
        Alltoall
           ⋮
```

References

- Collective Communications in MPI

  `http://www.mcs.anl.gov/research/projects/mpi/tutorial/gropp/node72.html`

- "Collective Communication: Theory, Practice, and Experience", Chan, Heimlich, Purkayastha, van de Geijn. (FLAME working note #22)

Collective Communication

- Synchronization
  `Barrier` $\leftarrow$ **Almost never needed!**

Collective Communication

- Synchronization
  `Barrier` ← **Almost never needed!**
- Data Movement
  `Broadcast, Scatter, Gather, Allgather, Alltoall`

Collective Communication

- Synchronization
  `Barrier` ← **Almost never needed!**
- Data Movement
  `Broadcast, Scatter, Gather, Allgather, Alltoall`
- Reductions
  `Reduce, Reduce-scatter, Allreduce, Scan, ...`

Collective Communication

- Synchronization
  `Barrier` ← **Almost never needed!**
- Data Movement
  `Broadcast, Scatter, Gather, Allgather, Alltoall`
- Reductions
  `Reduce, Reduce-scatter, Allreduce, Scan, ...`

Collectives involve **all** processes in the communicator. *—SPMD paradigm—*
All processes invoke the same operation with the same arguments. No tags.

Collective Communication

- Synchronization
  `Barrier` ← **Almost never needed!**
- Data Movement
  `Broadcast, Scatter, Gather, Allgather, Alltoall`
- Reductions
  `Reduce, Reduce-scatter, Allreduce, Scan, ...`

Collectives involve **all** processes in the communicator. —*SPMD paradigm*—
All processes invoke the same operation with the same arguments. No tags.

Collectives are **blocking** —the routine returns as soon as its participation in the communication is complete— but **not synchronous** —no guarantee is given on the status of the receiving processes—.

`int MPI_BCast(...)`

Before:

Node_0	Node_1	Node_2	Node_3
		α	

`int MPI_BCast(...)`

Before:

Node_0	Node_1	Node_2	Node_3
		α	

After:

Node_0	Node_1	Node_2	Node_3
α	α	α	α

`int MPI_BCast(...)`

Before:

Node_0	Node_1	Node_2	Node_3
		α	

After:

Node_0	Node_1	Node_2	Node_3
α	α	α	α

- How would you implement the broadcast in terms of individual sends and receives?

`int MPI_BCast(...)`

Before:

Node_0	Node_1	Node_2	Node_3
		α	

After:

Node_0	Node_1	Node_2	Node_3
α	α	α	α

- How would you implement the broadcast in terms of individual sends and receives?
- How many steps does it take to broadcast to np processes?

int MPI_BCast(...)

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
		α	

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
α	α	α	α

```
MPI_Bcast( Buffer, count, type, root, communicator );
```

```
MPI_Datatype: MPI_CHAR,  MPI_INT, MPI_UNSIGNED,
              MPI_FLOAT, MPI_DOUBLE, ...
```

`int MPI_Reduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

`int MPI_Reduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

After:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3
		$\mathop{\text{Op}}_{i=0}^{p-1} \delta_i$	

`int MPI_Reduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

After:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3
		$\bigoplus_{i=0}^{p-1} \delta_i$	

`MPI_Op:`
`MPI_MAX, MPI_MIN, MPI_SUM, MPI_PROD, MPI_LAND, MPI_BAND, ...,`
`MPI_MINLOC, MPI_MAXLOC`

`int MPI_Reduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

After:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3
		$\mathop{\text{Op}}_{i=0}^{p-1} \delta_i$	

`MPI_Op`:
`MPI_MAX, MPI_MIN, MPI_SUM, MPI_PROD, MPI_LAND, MPI_BAND, ..., MPI_MINLOC, MPI_MAXLOC`

`Op`: Associative. Why is this needed?

`int MPI_Reduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

After:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3
		$\text{Op}_{i=0}^{p-1} \delta_i$	

```
MPI_Reduce( sendBuffer, recvBuffer, count, type,
            operation, root, communicator );
```

`int MPI_Reduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

After:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3
		$\overset{p-1}{\underset{i=0}{\text{Op}}}\ \delta_i$	

```
MPI_Reduce( sendBuffer, recvBuffer, count, type,
            operation, root, communicator );
```

`MPI_Op_create`

`int MPI_Scatter(...)`

	$Node_0$	$Node_1$	$Node_2$	$Node_3$
Before:			v[0] v[1] v[2] v[3] v[4] v[5] v[6] v[7]	

`int MPI_Scatter(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
		v[0]	
		v[1]	
		v[2]	
		v[3]	
		v[4]	
		v[5]	
		v[6]	
		v[7]	

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]		v[0]	
v[1]		v[1]	
	v[2]	v[2]	
	v[3]	v[3]	
		v[4]	
		v[5]	
		v[6]	v[6]
		v[7]	v[7]

int MPI_Scatter(...)

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
		v[0] v[1] v[2] v[3] v[4] v[5] v[6] v[7]	

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0] v[1]		v[0] v[1] v[2] v[3] v[4] v[5] v[6] v[7]	
	v[2] v[3]		
			v[6] v[7]

```
MPI_Scatter( sendBuffer, sendCount, sendType,
             recvBuffer, recvCount, recvType,
             root, communicator );
```

Note: `sendCount` is the amount of data sent to each process.

`int MPI_Gather(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0] v[1]	v[2] v[3]	v[4] v[5]	v[6] v[7]

`int MPI_Gather(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0] v[1]	v[2] v[3]	v[4] v[5]	v[6] v[7]

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0] v[1]	v[2] v[3]	v[0] v[1] v[2] v[3] v[4] v[5] v[6] v[7]	v[6] v[7]

int MPI_Gather(...)

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0] v[1]	v[2] v[3]	v[4] v[5]	v[6] v[7]

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0] v[1]	v[2] v[3]	v[0] v[1] v[2] v[3] v[4] v[5] v[6] v[7]	v[6] v[7]

```
MPI_Gather( sendBuffer, sendCount, sendType,
            recvBuffer, recvCount, recvType,
            root, communicator );
```

Note: `recvCount` is the number of elements for any single receive.

`int MPI_Allgather(...)`

	Node_0	Node_1	Node_2	Node_3
Before:	v[0]	v[1]	v[2]	v[3]

`int MPI_Allgather(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]	v[1]	v[2]	v[3]

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]	v[0]	v[0]	v[0]
v[1]	v[1]	v[1]	v[1]
v[2]	v[2]	v[2]	v[2]
v[3]	v[3]	v[3]	v[3]

`int MPI_Allgather(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]	v[1]	v[2]	v[3]

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]	v[0]	v[0]	v[0]
v[1]	v[1]	v[1]	v[1]
v[2]	v[2]	v[2]	v[2]
v[3]	v[3]	v[3]	v[3]

```
MPI_Allgather( sendBuffer, sendCount, sendType,
               recvBuffer, recvCount, recvType,
               communicator );
```

`int MPI_Reduce_scatter(...)`

	$Node_0$	$Node_1$	$Node_2$	$Node_3$
Before:	$v_0[0]$	$v_1[0]$	$v_2[0]$	$v_3[0]$
	$v_0[1]$	$v_1[1]$	$v_2[1]$	$v_3[1]$
	$v_0[2]$	$v_1[2]$	$v_2[2]$	$v_3[2]$
	$v_0[3]$	$v_1[3]$	$v_2[3]$	$v_3[3]$

`int MPI_Reduce_scatter(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$v_0[0]$	$v_1[0]$	$v_2[0]$	$v_3[0]$
$v_0[1]$	$v_1[1]$	$v_2[1]$	$v_3[1]$
$v_0[2]$	$v_1[2]$	$v_2[2]$	$v_3[2]$
$v_0[3]$	$v_1[3]$	$v_2[3]$	$v_3[3]$

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$\underset{i}{\text{Op}}\ v_i[0]$	$\underset{i}{\text{Op}}\ v_i[1]$	$\underset{i}{\text{Op}}\ v_i[2]$	$\underset{i}{\text{Op}}\ v_i[3]$

int MPI_Reduce_scatter(...)

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$v_0[0]$	$v_1[0]$	$v_2[0]$	$v_3[0]$
$v_0[1]$	$v_1[1]$	$v_2[1]$	$v_3[1]$
$v_0[2]$	$v_1[2]$	$v_2[2]$	$v_3[2]$
$v_0[3]$	$v_1[3]$	$v_2[3]$	$v_3[3]$

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$\underset{i}{\text{Op}}\ v_i[0]$	$\underset{i}{\text{Op}}\ v_i[1]$	$\underset{i}{\text{Op}}\ v_i[2]$	$\underset{i}{\text{Op}}\ v_i[3]$

```
MPI_Reduce_scatter( sendBuffer,   recvBuffer,
                    recvCounts[], type, operation,
                    communicator );
```

`int MPI_Allreduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

`int MPI_Allreduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

After:

Node_0	Node_1	Node_2	Node_3
$\bigoplus_{i=0}^{p-1} \delta_i$	$\bigoplus_{i=0}^{p-1} \delta_i$	$\bigoplus_{i=0}^{p-1} \delta_i$	$\bigoplus_{i=0}^{p-1} \delta_i$

`int MPI_Allreduce(...)`

Before:

Node_0	Node_1	Node_2	Node_3
δ_0	δ_1	δ_2	δ_3

After:

Node_0	Node_1	Node_2	Node_3
$\text{Op}_{i=0}^{p-1}\ \delta_i$	$\text{Op}_{i=0}^{p-1}\ \delta_i$	$\text{Op}_{i=0}^{p-1}\ \delta_i$	$\text{Op}_{i=0}^{p-1}\ \delta_i$

```
MPI_Allreduce( sendBuffer,  recvBuffer,
               count, type, operation,
               communicator );
```

`int MPI_Alltoall(...)`

	$Node_0$	$Node_1$	$Node_2$	$Node_3$
Before:	$v_0[0]$	$v_1[0]$	$v_2[0]$	$v_3[0]$
	$v_0[1]$	$v_1[1]$	$v_2[1]$	$v_3[1]$
	$v_0[2]$	$v_1[2]$	$v_2[2]$	$v_3[2]$
	$v_0[3]$	$v_1[3]$	$v_2[3]$	$v_3[3]$

`int MPI_Alltoall(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$v_0[0]$	$v_1[0]$	$v_2[0]$	$v_3[0]$
$v_0[1]$	$v_1[1]$	$v_2[1]$	$v_3[1]$
$v_0[2]$	$v_1[2]$	$v_2[2]$	$v_3[2]$
$v_0[3]$	$v_1[3]$	$v_2[3]$	$v_3[3]$

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$v_0[0]$	$v_0[1]$	$v_0[2]$	$v_0[3]$
$v_1[0]$	$v_1[1]$	$v_1[2]$	$v_1[3]$
$v_2[0]$	$v_2[1]$	$v_2[2]$	$v_2[3]$
$v_3[0]$	$v_3[1]$	$v_3[2]$	$v_3[3]$

`int MPI_Alltoall(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$v_0[0]$	$v_1[0]$	$v_2[0]$	$v_3[0]$
$v_0[1]$	$v_1[1]$	$v_2[1]$	$v_3[1]$
$v_0[2]$	$v_1[2]$	$v_2[2]$	$v_3[2]$
$v_0[3]$	$v_1[3]$	$v_2[3]$	$v_3[3]$

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$v_0[0]$	$v_0[1]$	$v_0[2]$	$v_0[3]$
$v_1[0]$	$v_1[1]$	$v_1[2]$	$v_1[3]$
$v_2[0]$	$v_2[1]$	$v_2[2]$	$v_2[3]$
$v_3[0]$	$v_3[1]$	$v_3[2]$	$v_3[3]$

```
MPI_Alltoall( sendBuffer, sendCount, sendType,
              recvBuffer, recvCount, recvType,
              communicator );
```

`int MPI_Alltoall(...)`

	$Node_0$	$Node_1$	$Node_2$	$Node_3$
Before:	$v_0[0]$	$v_1[0]$	$v_2[0]$	$v_3[0]$
	$v_0[1]$	$v_1[1]$	$v_2[1]$	$v_3[1]$
	$v_0[2]$	$v_1[2]$	$v_2[2]$	$v_3[2]$
	$v_0[3]$	$v_1[3]$	$v_2[3]$	$v_3[3]$
	$v_0[4]$	$v_1[4]$	$v_2[4]$	$v_3[4]$
	$v_0[5]$	$v_1[5]$	$v_2[5]$	$v_3[5]$
	$v_0[6]$	$v_1[6]$	$v_2[6]$	$v_3[6]$
	$v_0[7]$	$v_1[7]$	$v_2[7]$	$v_3[7]$

`int MPI_Alltoall(...)`

Before:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$v_0[0]$	$v_1[0]$	$v_2[0]$	$v_3[0]$
$v_0[1]$	$v_1[1]$	$v_2[1]$	$v_3[1]$
$v_0[2]$	$v_1[2]$	$v_2[2]$	$v_3[2]$
$v_0[3]$	$v_1[3]$	$v_2[3]$	$v_3[3]$
$v_0[4]$	$v_1[4]$	$v_2[4]$	$v_3[4]$
$v_0[5]$	$v_1[5]$	$v_2[5]$	$v_3[5]$
$v_0[6]$	$v_1[6]$	$v_2[6]$	$v_3[6]$
$v_0[7]$	$v_1[7]$	$v_2[7]$	$v_3[7]$

After:

$Node_0$	$Node_1$	$Node_2$	$Node_3$
$v_0[0]$	$v_0[2]$	$v_0[4]$	$v_0[6]$
$v_0[1]$	$v_0[3]$	$v_0[5]$	$v_0[7]$
$v_1[0]$	$v_1[2]$	$v_1[4]$	$v_1[6]$
$v_1[1]$	$v_1[3]$	$v_1[5]$	$v_1[7]$
$v_2[0]$	$v_2[2]$	$v_2[4]$	$v_2[6]$
$v_2[1]$	$v_2[3]$	$v_2[5]$	$v_2[7]$
$v_3[0]$	$v_3[2]$	$v_3[4]$	$v_3[6]$
$v_3[1]$	$v_3[3]$	$v_3[5]$	$v_3[7]$

More Collectives

Variable length

- `MPI_Scatterv`
- `MPI_Gatherv`
- `MPI_Allgatherv`
- `MPI_Alltoallv`

More Collectives

Variable length

- `MPI_Scatterv`
- `MPI_Gatherv`
- `MPI_Allgatherv`
- `MPI_Alltoallv`

```
MPI_Scatter(
    sendBuffer, sendCount,
                sendType,
    recvBuffer, recvCount,
                recvType,
    root, communicator );
```

```
MPI_Scatterv(
    sendBuffer, sendCounts[],
    displs[],   sendType,
    recvBuffer, recvCount,
                recvType,
    root, communicator )
```

Even more Collectives

Partial reduction

- `MPI_Scan`

Non-blocking collectives

- `MPI_I*`

`int MPI_Scan(...)`

Before:

Node_0	Node_1	Node_2	...	Node_{p-1}
δ_0	δ_1	δ_2	...	δ_{p-1}

`int MPI_Scan(...)`

Before:

Node_0	Node_1	Node_2	...	Node_{p-1}
δ_0	δ_1	δ_2	...	δ_{p-1}

After:

Node_0	Node_1	Node_2	...	Node_{p-1}
δ_0	$\text{Op}_{i=0}^{1} \delta_i$	$\text{Op}_{i=0}^{2} \delta_i$	...	$\text{Op}_{i=0}^{p-1} \delta_i$

`int MPI_Scan(...)`

Before:

Node_0	Node_1	Node_2	...	Node_{p-1}
δ_0	δ_1	δ_2	...	δ_{p-1}

After:

Node_0	Node_1	Node_2	...	Node_{p-1}
δ_0	$\text{Op}_{i=0}^{1} \delta_i$	$\text{Op}_{i=0}^{2} \delta_i$	...	$\text{Op}_{i=0}^{p-1} \delta_i$

```
MPI_Scan( sendBuffer,  recvBuffer,
          count, type, operation,  communicator );
```

`int MPI_Exscan(...)`

Before:

Node_0	Node_1	Node_2		Node_{p-1}
δ_0	δ_1	δ_2	...	δ_{p-1}

`int MPI_Exscan(...)`

Before:

Node_0	Node_1	Node_2		Node_{p-1}
δ_0	δ_1	δ_2	...	δ_{p-1}

After:

Node_0	Node_1	Node_2		Node_{p-1}
-	δ_0	$\overset{1}{\underset{i=0}{\text{Op}}}\ \delta_i$	...	$\overset{p-2}{\underset{i=0}{\text{Op}}}\ \delta_i$

`int MPI_Exscan(...)`

Before:

Node_0	Node_1	Node_2		Node_{p-1}
δ_0	δ_1	δ_2	...	δ_{p-1}

After:

Node_0	Node_1	Node_2		Node_{p-1}
-	δ_0	$\overset{1}{\underset{i=0}{\text{Op}}}\ \delta_i$	...	$\overset{p-2}{\underset{i=0}{\text{Op}}}\ \delta_i$

```
MPI_Exscan( sendBuffer,  recvBuffer,
            count, type, operation,  communicator );
```