Parallel Programming

pauldj@aices.rwth-aachen.de

Anatomy of `MPI_Send` and `MPI_Recv`

```
int MPI_Send(
    *buffer, count, datatype,                    ← "data"
    destination, tag, communicator               ← "envelope"
);

int MPI_Recv(
    *buffer, count, datatype,                    ← "data"
    source, tag, commmunicator,                  ← "envelope"
    *status
);
```

message = data + envelope (+ info)

matching envelopes $\rightarrow$ data transfer

Note: Meanining of `count`: send $\neq$ recv

`count` in send = size of message vs. `count` in receive = size of buffer.

Point-to-point communication

Send

- `MPI_Ssend`
- `MPI_Send`
- `MPI_Isend`
- ⋮
- `MPI_Bsend`

Receive

- `MPI_Recv`
- `MPI_Irecv`

Send+Receive

- `MPI_Sendrecv`
- `MPI_Sendrecv_replace`

Send/Recv Modes

[Send] The stress is on the buffer sent: "When I can I safely overwrite it?"

- `MPI_Ssend`: The program execution is blocked until a matching receive is posted. The buffer is usable as soon as the call completes.
- `MPI_Send`: MPI attempts to copy the outgoing message onto a local (hidden) buffer. If possible, the execution continues and the send buffer is immediately usable, otherwise same as `Ssend`.
- `MPI_Isend`: The execution continues `I`mmediately. The send buffer should not be accessed until the `MPI_request` allows it. To be used in conjunction with `MPI_Wait` or `MPI_Test`*.

[Recv] The stress is on the incoming buffer: "When I can I safely access it?"

- `MPI_Recv`: The program execution is blocked until a matching send is posted. The incoming buffer is usable as soon as the call completes.
- `MPI_Irecv`: The execution continues `I`mmediately. The incoming buffer should not be accessed until the `MPI_request` allows it. To be used in conjunction with `MPI_Wait` or `MPI_Test`*.

*: See also `MPI_Waitany`, `MPI_Waitall`, `MPI_Waitsome`, `MPI_Testany`, `MPI_Testall`, `MPI_Testsome`.

Recap: deadlock

- 2+ processes want to exchange data
- All processes start with a blocking send or a blocking receive
  `Ssend`, `Send` (in the worst case), `Recv`
  $\Rightarrow$ BUG: deadlock
- Solution: BREAK SYMMETRY!
  At the same time, careful not to serialize the code!

  Approach: code, test and debug with `Ssend`; then replace with `Send`
- Other solutions?
  - Non-blocking send (`Isend`)
  - Non-blocking receive (`Irecv`)
  - Simultaneous send-receive (`Sendrecv`)