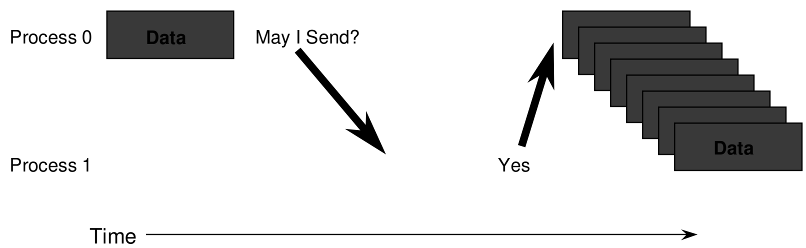
What is message passing?

- Data transfer plus synchronization



- Requires cooperation of sender and receiver
- Cooperation not always apparent in code

1

Quick review of MPI Message passing

- Basic terms
  » nonblocking - Operation does not wait for completion
  » synchronous - Completion of send *requires* initiation (but not completion) of receive
  » ready - *Correct* send requires a matching receive
  » asynchronous - communication and computation take place simultaneously, ***not*** an MPI concept (implementations *may* use asynchronous methods)

2

Message protocols

- Message consists of “envelope” and data
  - » Envelope contains tag, communicator, length, source information, plus impl. private data
- Short
  - » Message data (message for short) sent with envelope
- Eager
  - » Message sent assuming destination can store
- Rendezvous
  - » Message not sent until destination oks

3

Special Protocols for DSM

- Message passing is a good way to use distributed shared memory (DSM) machines because it provides a way to express memory locality.
- Put
  - » Sender puts to destination memory (user or MPI buffer). Like Eager.
- Get
  - » Receiver gets data from sender or MPI buffer. Like Rendezvous.
- Short, long, rendezvous versions of these

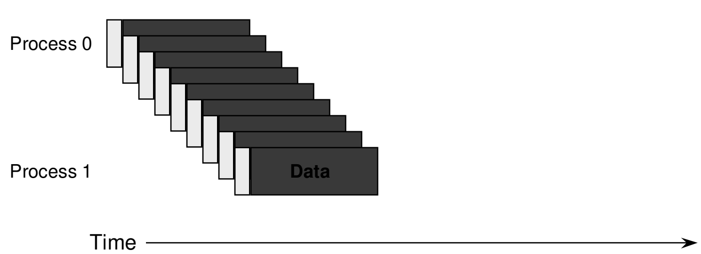
4

Message Protocol Details

- Eager not Rsend, rendezvous not Ssend resp., but related
- User versus system buffer space
- Packetization
- Collective operations
- Datatypes, particularly non-contiguous
  - » Handling of important special cases
    - – Constant stride
    - – Contiguous structures

5

Eager Protocol



- Data delivered to process 1
  - » No matching receive may exist; process 1 must then buffer and copy.

6

3

Eager Features

- Reduces synchronization delays
- Simplifies programming (just MPI_Send)
- Requires significant buffering
- May require active involvement of CPU to drain network at receiver's end
- May introduce additional copy (buffer to final destination)

7

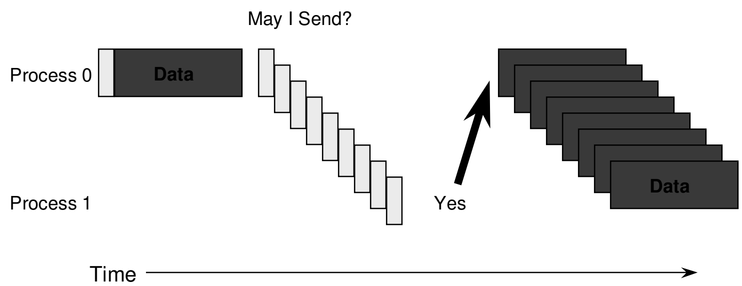
How Scaleable is Eager Delivery?

- Buffering must be reserved for arbitrary senders
- User-model mismatch (often expect buffering allocated entirely to "used" connections).
- Common approach in implementations is to provide same buffering for all members of MPI_COMM_WORLD; this is optimizing for non-scaleable computations
- Scaleable implementations that exploit message patterns are possible

8

4

Rendezvous Protocol



- Envelope delivered first
- Data delivered when user-buffer available
  - » Only buffering of envelopes required

9

Rendezvous Features

- Robust and safe
  - » (except for limit on the number of envelopes…)
- May remove copy (user to user direct)
- More complex programming (waits/tests)
- May introduce synchronization delays (waiting for receiver to ok send)

10

Short Protocol

- Data is part of the envelope
- Otherwise like eager protocol
- May be performance optimization in interconnection system for short messages, particularly for networks that send fixed-length packets (or cache lines)

User and System Buffering

- Where is data stored (or staged) while being sent?
  - » User’s memory
    - – Allocated on the fly
    - – Preallocated
  - » System memory
    - – May be limited
    - – Special memory may be faster

Implementing MPI_Isend

- Simplest implementation is to always use rendezvous protocol:
  - » MPI_Isend delivers a request-to-send control message to receiver
  - » Receiving process responds with an ok-to-send
    - – May or may not have matching MPI receive; only needs buffer space to store incoming message
  - » Sending process transfers data
- Wait for MPI_Isend request
  - » wait for ok-to-send message from receiver
  - » wait for data transfer to be complete on sending side

13

Alternatives for MPI_Isend

- Use a short protocol for small messages
  - » No need to exchange control messages
  - » Need guaranteed (but small) buffer space on destination for short message envelope
  - » Wait becomes a no-op
- Use eager protocol for modest sized messages
  - » Still need guaranteed buffer space for both message envelope and eager data on destination
  - » Avoids exchange of control messages

14

Implementing MPI_Send

- Can't use eager always because this could overwhelm the receiving process

```
if (rank != 0) MPI_Send( 100 MB of data )
else receive 100 MB from each process
```

- Would like to exploit the blocking nature (can wait for receive)
- Would like to be fast
- Select protocol based on message size (and perhaps available buffer space at destination)
  - » Short and/or eager for small messages
  - » Rendezvous for longer messages

Implementing MPI_Rsend

- Just use MPI_Send; no advantage for users
- Use eager always (or short if small)
  - » even for long messages

Why Bsend?

- Buffer space not infinite
- Careful management of buffering required for *correctness* and *performance*
- MPI provides user control over buffering modes
- Allows implementations freedom to pick specific protocols
  - Users give up some control for more convenient general-purpose send (MPI_Send)
- Implementations *could* make Bsend fast, at least when buffering is available at the destination

17