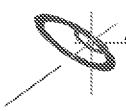
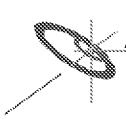
An introduction to advanced MPI programming

Michael M. Resch
HLRS
resch@hlrs.de



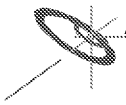
Introduction

- Deeper discussion of MPI
- Better understanding of the standard and what it means to implementors and programmers
- Preview of MPI-2



Contents

- **MPI basics and insight** **(8:30-9:30)**
- Coffee Break (9:30-10:00)
- **Performance** **(10:00-11:15)**
- Break (11:15-11:30)
- **MPI-2** **(11:30-12:30)**

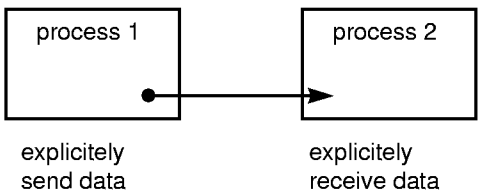


MPI basics and insight

- Understand how MPI implementations work
- Understand the effect of different MPI functions that accomplish the same communication

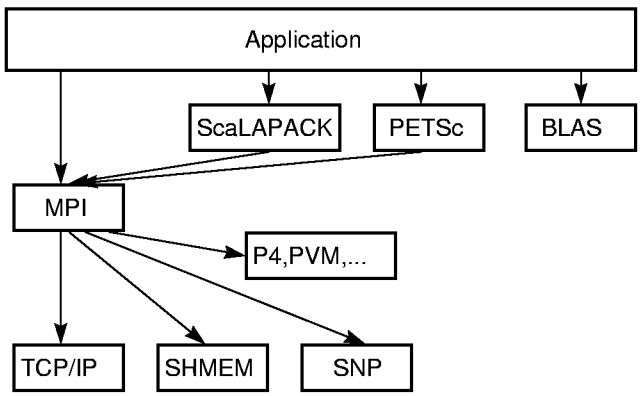
Message-Passing

- explicit exchange of data between independent processes

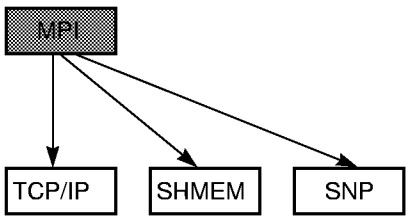


some copying and transfer of data necessary

MPI in a parallel programming environment



Layers of communication



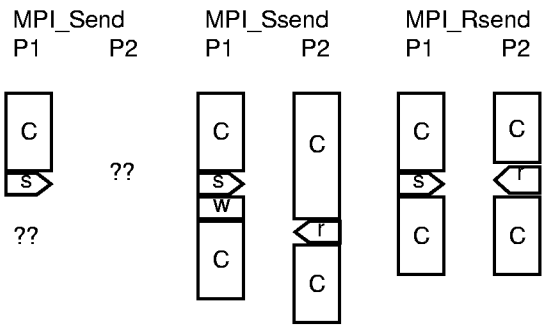
Message concept of MPI

- envelope
  - communicator
  - source information
  - tag
  - length of data part
- data

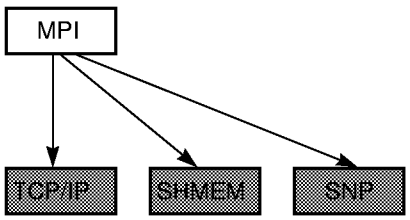
Basic send/recv modes

- MPI_Send: Sends data. May or may not need a matching recv.
- MPI_Recv: Receives data.
- MPI_Ssend: Waits for completion until a matching recv has been issued.
- MPI_Rsend: Expects matching recv to have already been issued.

Basic send/recv modes



Layers of communication



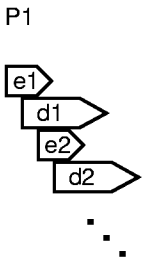
Message protocols

- Eager: Envelope and data sent immediately assuming destination can store everything that arrives.
- Rendezvous: Envelope sent immediately but data only sent after receiver gives ok.

- Short: Same as eager but envelope and data are sent in one piece.

Eager protocol

- Sending without checking whether destination process is ready to receive



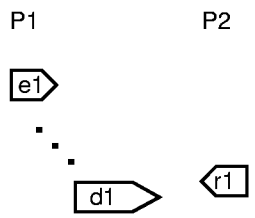
Eager protocol

- Advantages:
  - Reduces synchronization delays
  - Simplifies programming (just transfer information)
- Disadvantages:
  - Requires significant buffering for incoming messages
  - May introduce additional copy operation from system buffer to final destination
- Estimated costs:

communication time = *time to initiate message +*
time to transfer envelope +
time to transfer data to buffer +
time to copy data from buffer

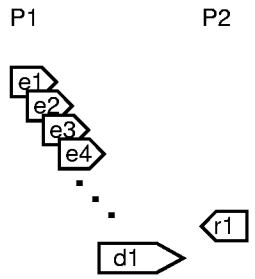
Rendezvous protocol

- Send envelope and wait for recv until delivering data



Mixture of protocols

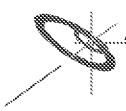
- Send envelopes and wait for recv until delivering data



Rendezvous protocol

- Advantages:
  - Rendezvous is robust and safe
  - May remove one copy operation
- Disadvantages:
  - More complex in programming since tests/waits are necessary
  - Introduces synchronization delay
- Estimated costs:

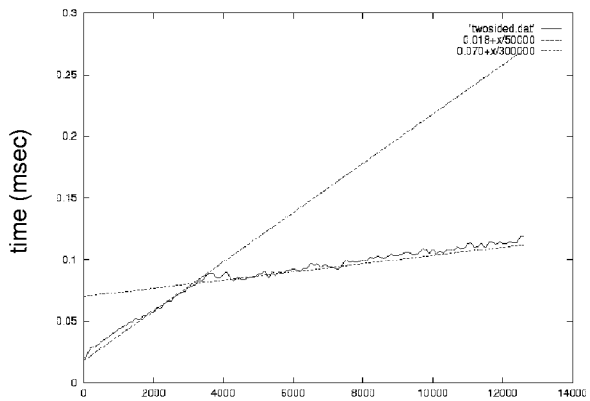
communication time = *time to initiate message +*
time to transfer envelope +
time to wait/test for reply +
time to transfer data



Modes vs. Protocols

	MPI_Send	MPI_Ssend	MPI_Rsend
Eager	ok	ok	no
Rendezvous	???	no	ok

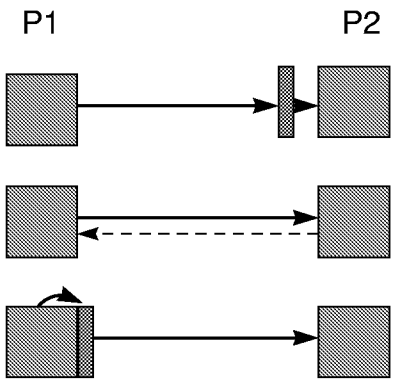
Eager vs. Rendezvous on a T3E



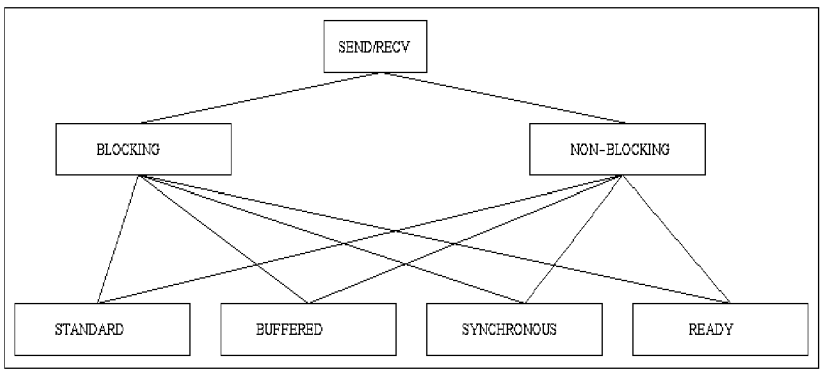
Different ways to pass a message

- Via buffers:
  - Data are copied from the user memory space to the network
  - The network hands them over to the receivers buffer
  - There data is copied into the users memory space
- Without buffers:
  - Data are transferred directly from one user memory space to another

Different ways to pass a message

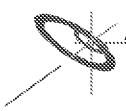


Different ways to pass a message in MPI



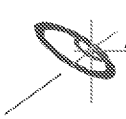
Summary I

- Be aware of the definition of the standard
- Be aware of the implementation details of your implementation
- Be aware of the underlying hardware



Performance issues

- Performance metrics
- non-blocking send/recv
- Optimized communication
- Optimized algorithms



Performance metrics

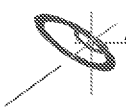
- Latency
  Basically the time it takes to send a zero-size message
- Bandwidth
  Basically the amount of data sent per time unit

- Communication time

  $time = latency + message\text{-}size / bandwidth$

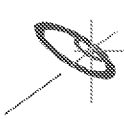
- Results for T3E

  $size < 4096 \quad time = 16+size/50$
  $size > 4096 \quad time = 70+size/300$

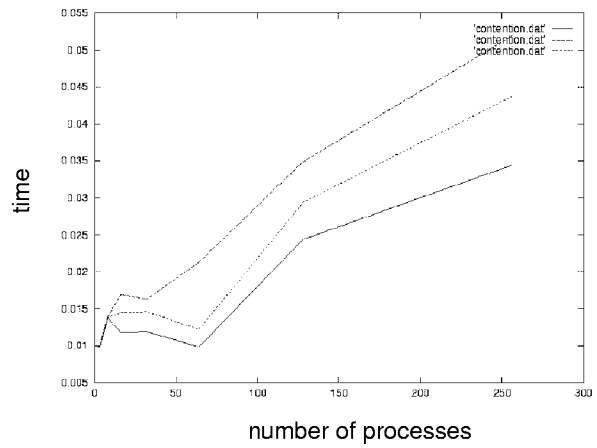


Contention

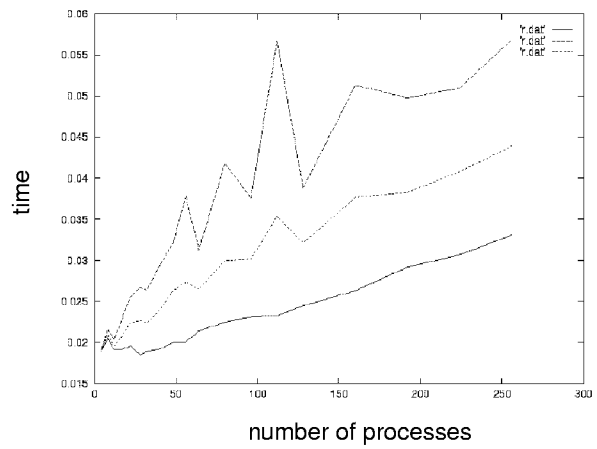
- An increasing number of messages may decrease available bandwidth for single messages
- Typical communication patterns of numerical codes yield to a number of messages sent at the same time



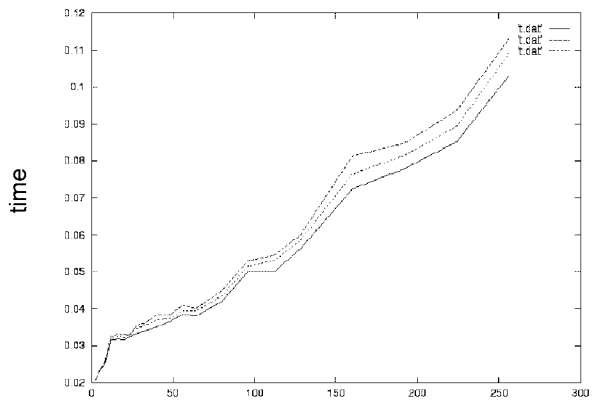
Contention (T3E: even-odd communication)



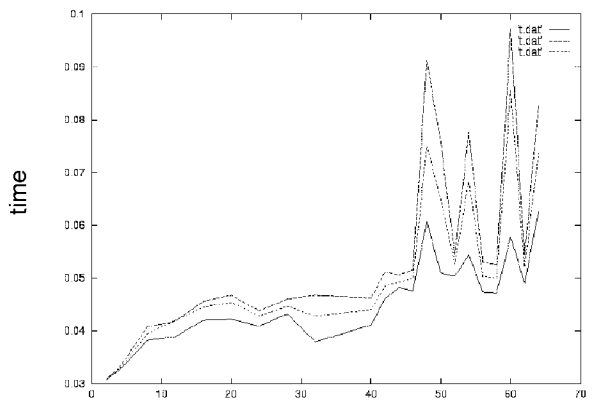
Contention (T3E: random number communication)



Contention (T3E: 2D-topology communication)

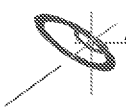


Contention (T3E: 3D-topology communication)



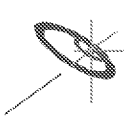
Contention

- Avoid sending all messages at the same time
- Redesign program such that number of processes communicating at the same time is small
- Chose number of nodes according to hardware architecture
- Chose topology according to hardware topology



Non-blocking send/recv

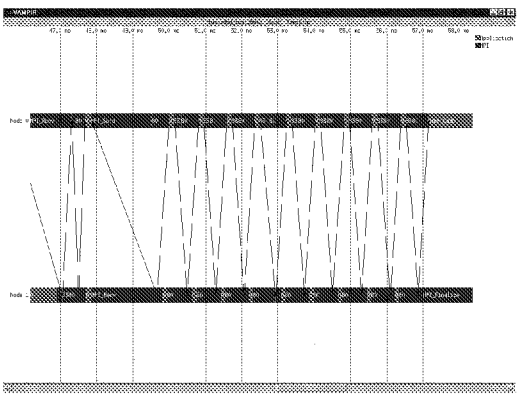
- MPI_Isend(..... request ...)
- MPI_Wait/Test(request....)
- Allows to overlap communication and computation



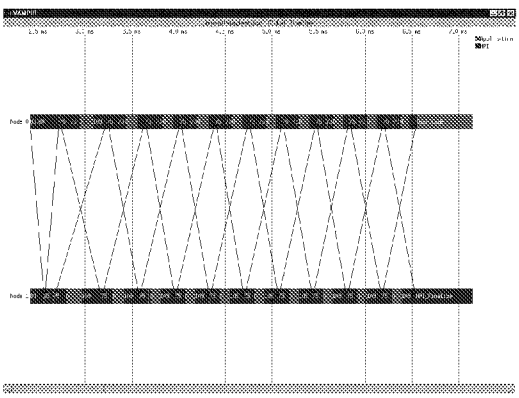
Non-blocking send/recv

- Need not necessarily be faster
- An implementation may
  - Initiate communication by MPI_Isend
  - Do actual communication only when an MPI_Wait/Test is called
- Performance may not be an issue but non-blocking may avoid deadlocks

Non-blocking send/recv



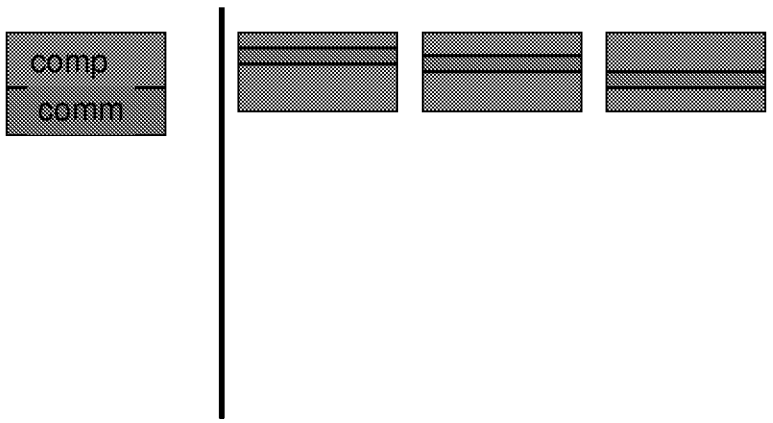
Non-blocking send/recv



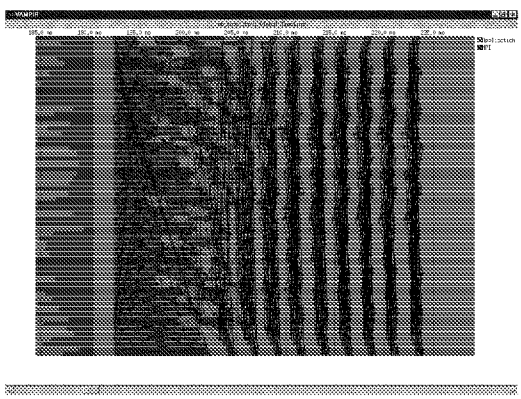
Non-blocking send/recv -> Contention

- Non-blocking send/recv may allow to arrange communication in a way that avoids contention
- Data that can be sent very early are computed first
- MPI_Isend is not issued immediately but in a staggered mode

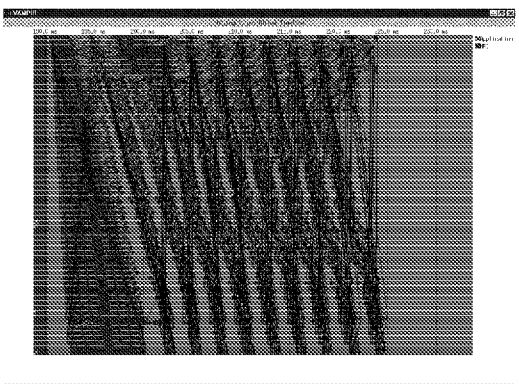
Non-blocking send/recv -> Contention



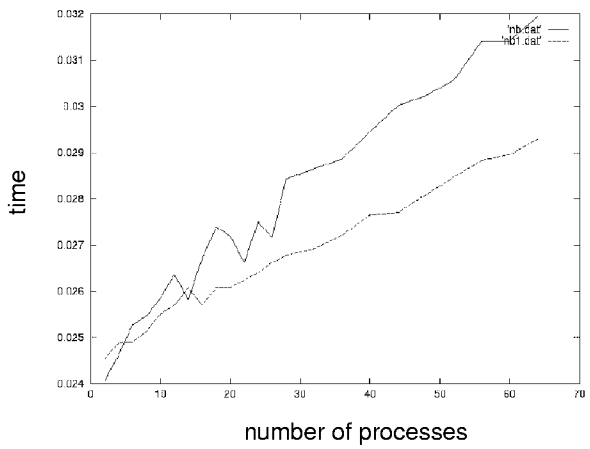
Non-blocking send/recv -> Contention



Non-Blocking send/recv -> Contention

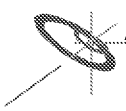


Non-blocking send/recv -> Contention



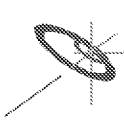
Optimized communication

- Send one big message instead of several small ones
- Send contiguous buffers instead of fragmented ones
- Use collective communication where possible
- Aggregate also collective communication



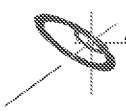
One vs. several messages

- What is the overhead for packing?
- What is the overhead for message initialization?



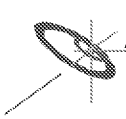
Contiguous buffers

- Fragmented buffers need to be collected before being sent
- If possible try to send contiguous buffers
- Analyse costs for collection of data!

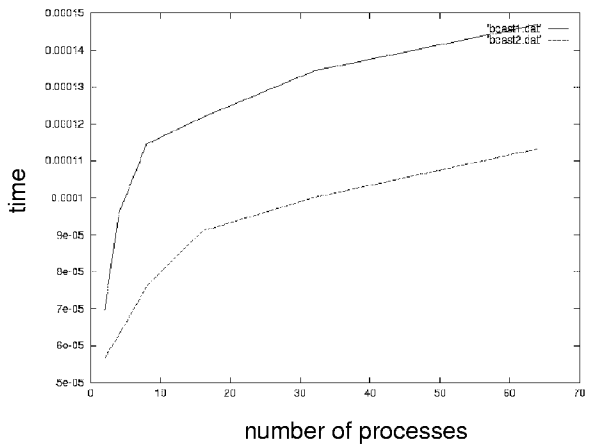


Collective communication

- Collective communication is always faster than the same communication pattern implemented by the user!
- Aggregate also collective communication



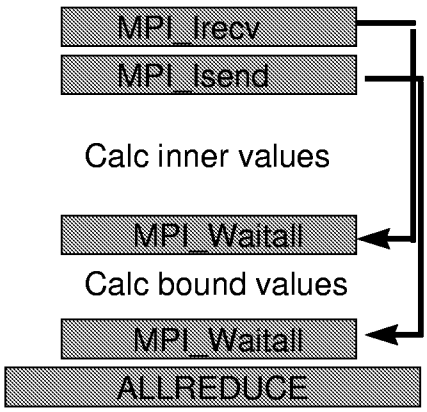
Aggregated collective communication



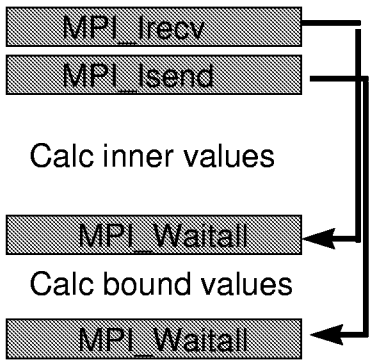
Optimized algorithms

- Avoid global communication
  - Check convergence locally
- Avoid synchronizing communication
  - Use staggered communication mode

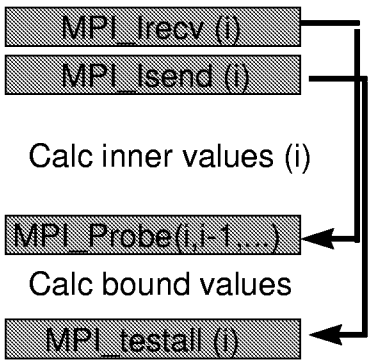
Avoid global communication



Avoid global communication



Staggered Communication mode



Summary II

- There is no general rules!?
- There is at least the rule not to use send/recv if a global communication can do the same job
- Estimation of costs for different methods is always necessary
- Costs may be prescribed by the system
- Costs may be reduced by considering the system features
- Estimation of costs requires performance analysis
  => Friday morning