# Annex

*[based on questions from ...]*

- Error handling *[Kristin Warnick]*
- Performance of Non-blocking / Sendrecv / Alltoall *[Willem Vermin]*
- Performance of MPI derived datatypes (→slide 108) *[J.-C. Desplat]*
- Optimization with virtual topologies (→slide 112) *[J.-C. Desplat]*
- Communication modes – pros & cons (→slide 66) *[J.-C. Desplat]*
- Overlaying comm.&computation or comm.&comm. *[Willem Vermin]*



## MPI Error Handling

- System/Hardware errors:
  - the communication should be reliable
- Application errors:
  - if the MPI program is erroneous:
    - **by default: abort, if error detected by MPI library otherwise, unpredictable behavior**
    - **Fortran: call MPI_Errhandler_set ( comm, MPI_ERRORS_RETURN, ierr)**
      **C: MPI_Errhandler_set ( comm, MPI_ERRORS_RETURN);**
      **then**
      - **ierror returned by each MPI routine**
      - **undefined state after an erroneous MPI call has occurred (only MPI_ABORT(…) should be still callable)**



## Comparing latencies with "heat" application

| Communication-time | T3E 900 | Hitachi SR2201 | HP-V | |
|---|---|---|---|---|
| (number of PEs) | (16) | (16) | (8) | |
| MPI non-blocking | 3.4 | 5.3 | 2 | [sec] |
| MPI_SENDRECV | 1.9 | 5.8 | 2 | [sec] |
| MPI_ALLTOALLV | 0.8 *) | 15.4 | 2 | [sec] |
| Computation-Time | 0.65 | 1.9 | 2 | [sec] |

MPI targets portable and efficient message-passing programming
but
**efficiency of MPI application-programming** is **not portable**!

*) **up 128 PEs: ALLTOALLV is better than SENDRECV**

(measured April 29, 1999, with heat-mpi1-big.f and stride 179,
CRAY T3E: sn6715 hwwt3e.hww.de 2.0.4.48 unicosmk CRAY T3E mpt.1.3.0.0.6,
Hitachi: HI-UX/MPP hitachi.rus.uni-stuttgart.de 02-03 0 SR2201,
HP: HP-UX hp-v.hww.de B.11.00 A 9000/800 75859)



## Caution 2

Performance

- Some MPI library have a poor performance with derived datatypes
- Always prefer
  - structure of arrays, or
  - independent arrays
- instead of
  - array of structures
- Transfer of non-contiguous data
  - Check which algorithm is faster:
    A) Usage of derived datatypes
    B) - Copy at sender into a local contiguous scratch-buffer
    - Transfer this scratch-buffer into a scratch-buffer at the receiver
    - Copy the receiver's scratch-buffer into its non-contiguous memory locations

## Virtual Topologies

- Convenient process naming.
- Naming scheme to fit the communication pattern.
- Simplifies writing of code.
- Can allow MPI to optimize communications.



## Rules for the communication modes

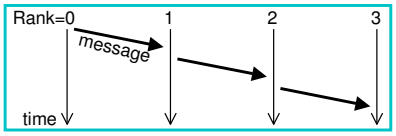
- Standard send (**MPI_SEND**)
  - minimal transfer time
  - may block due to synchronous mode
  - —> risks with synchronous send
- Synchronous send (**MPI_SSEND**)
  - risk of deadlock
  - risk of serialization
  - risk of waiting —> idle time
  - high latency / best bandwidth
- Buffered send (**MPI_BSEND**)
  - low latency / bad bandwidth
- Ready send (**MPI_RSEND**)
  - use **never**, except you have a 200% guarantee that Recv is already called in the current version and all future versions of your code



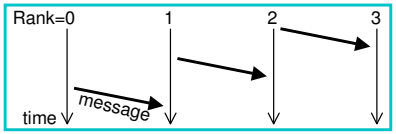
## Synchronization time — **How to avoid serialization**

- Synchronization may cause serialization:

**MPI_Recv(left_neighbor)**
**MPI_Send(right_neighbor)**




**MPI_Send(right_neighbor)**
**MPI_Recv(left_neighbor)**




- Solutions:
  - MPI_I..... (non-blocking routines)
  - MPI_Bsend
  - MPI_Sendrecv



## Non-blocking communication

- Non-blocking
  - latency hiding / overlap of communication and computation,
    - Problem: most MPI implementations communicate only while MPI routines are called
    - Exception: Cluster of SMP nodes / hybrid MPI+OpenMP
    - ==> Do not spent too much effort in such overlap
  - used to avoid deadlocks / serialization
  - used to avoid waiting until sender **and** receiver are ready to communicate, i.e., **to avoid idle time**