Kommunikations- und Optimierungsaspekte paralleler Programmiermodelle auf hybriden HPC-Plattformen

Rolf Rabenseifner
rabenseifner@hlrs.de

Universität Stuttgart,
Höchstleistungsrechenzentrum
Stuttgart (HLRS)
www.hlrs.de

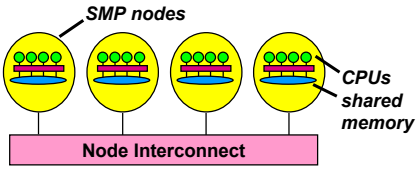
Gerhard Wellein
gerhard.wellein@rrze.uni-erlangen.de

Friedrich-Alexander-Universität
Erlangen-Nürnberg,
Regionales Rechenzentrum Erlangen
(RRZE)

ZKI, AK Supercomputing
RWTH Aachen, 3. April 2003.

Motivation

- HPC systems
  - often clusters of SMP nodes
  - i.e., hybrid architectures



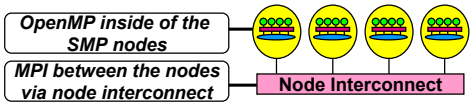
- Using the communication bandwidth of the hardware
- Minimizing synchronization = idle time

} **optimal usage of the hardware**

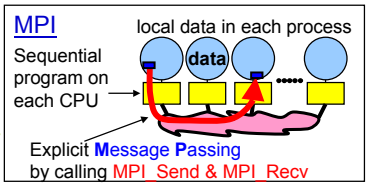
- Appropriate parallel programming models / Pros & Cons

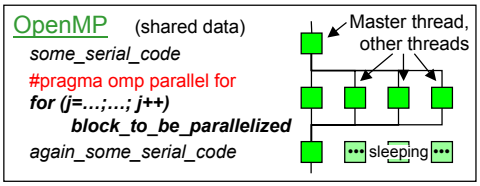
Major Programming models on hybrid systems

- Pure MPI (one MPI process on each CPU)
- Hybrid MPI+OpenMP
  - shared memory OpenMP
  - distributed memory MPI



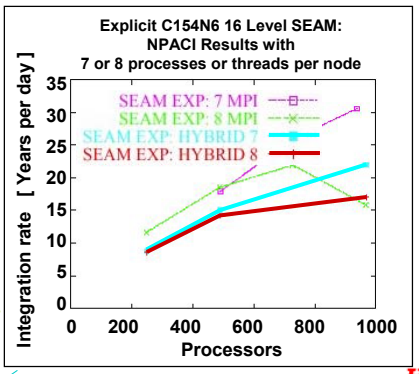
- Other: Virtual shared memory systems, HPF, …
- Often **hybrid programming (MPI+OpenMP)** slower than **pure MPI**
  - why?

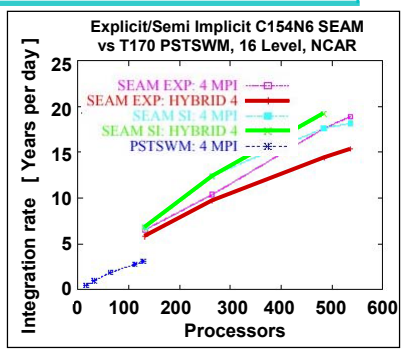




Example from SC 2001

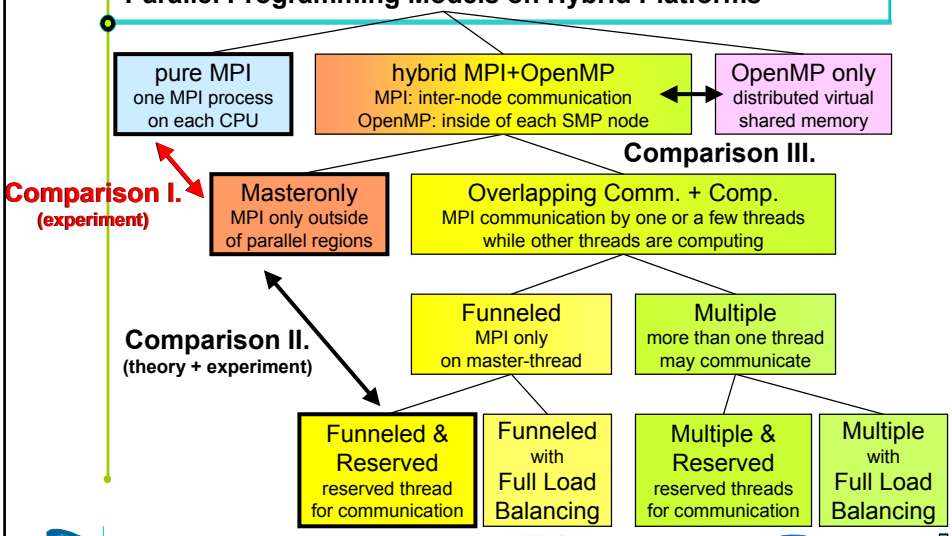
- Pure MPI versus Hybrid MPI+OpenMP (Masteronly)
- What's better?
  → it depends on?





Figures: Richard D. Loft, Stephen J. Thomas, John M. Dennis:
Terascale Spectral Element Dynamical Core for Atmospheric General Circulation Models.
Proceedings of SC2001, Denver, USA, Nov. 2001.
http://www.sc2001.org/papers/pap.pap189.pdf
Fig. 9 and 10.

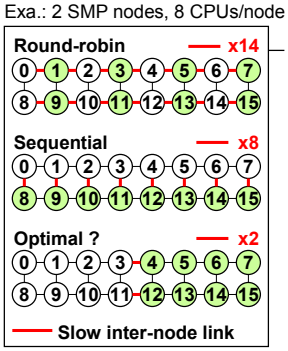
Parallel Programming Models on Hybrid Platforms



Programming Models on Hybrid Platforms: Pure MPI

pure MPI
one MPI process on each CPU

Exa.: 2 SMP nodes, 8 CPUs/node



Advantages
- No modifications on existing MPI codes
- MPI library need not to support multiple threads

Problems
- To fit application topology on hardware topology
- E.g. choosing ranks in MPI_COMM_WORLD
  - **round robin (rank 0 on node 0, rank 1 on node 1, ...)**
  - **Sequential (ranks 0-7 on 1st node, ranks 8-15 on 2nd …)**

Disadvantages
- Message passing overhead inside of SMP nodes (instead of simply accessing the data via the shared memory)! → **Reason for using hybrid programming models**

Programming Models on Hybrid Platforms: Hybrid Masteronly

Masteronly
MPI only outside of parallel regions

```
for (iteration ….)
{
  #pragma omp parallel
    numerical code
  /*end omp parallel */

  /* on master thread only */
    MPI_Send (original data
      to halo areas
      in other SMP nodes)
    MPI_Recv (halo data
      from the neighbors)
} /*end for loop
```

Advantages
- No message passing inside of the SMP nodes
- No topology problem

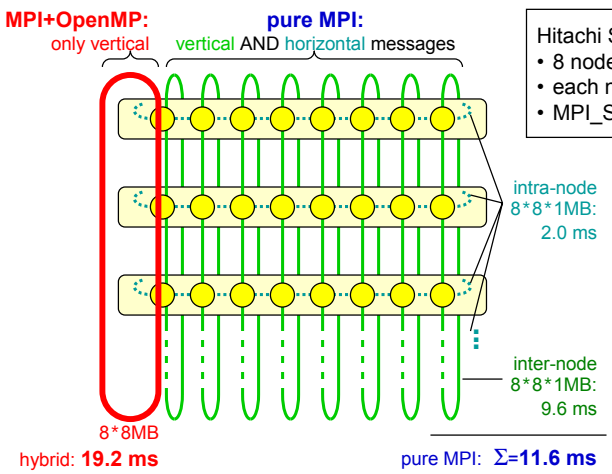
Problems
- MPI-lib must support MPI_THREAD_FUNNELED

Disadvantages
- all other threads are sleeping while master thread communicates

→ **Reason for implementing overlapping of communication & computation**

Experiment: Orthogonal parallel communication

pure MPI
Masteronly

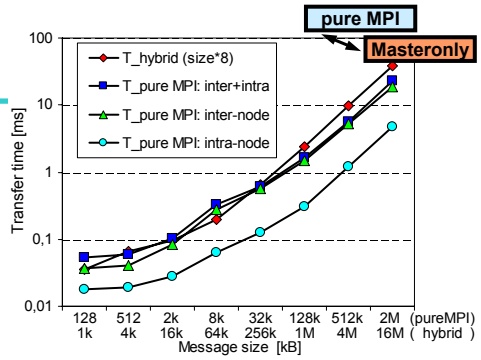


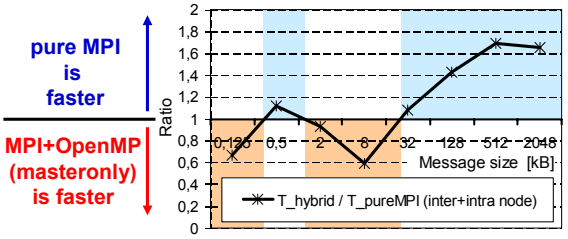
Hitachi SR8000
- 8 nodes
- each node with 8 CPUs
- MPI_Sendrecv

→ **1.6x slower** than with pure MPI, **although**
- only half of the transferred bytes
- and less latencies due to 8x longer messages

Results of the experiment

- pure MPI is better for message size > 32 kB
- long messages: $T_{hybrid} / T_{pureMPI} > 1.6$
- OpenMP master thread cannot saturate the inter-node network bandwidth

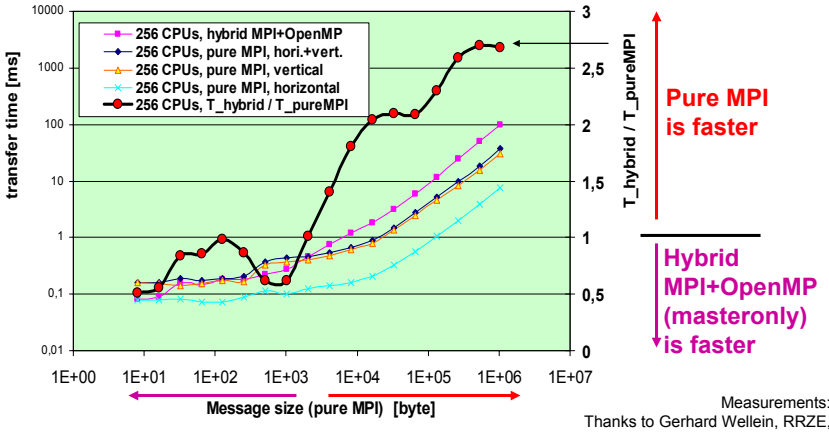




pure MPI
Masteronly

Ratio T_hybrid MPI+OpenMP / T_pure MPI – 16*16 CPUs

IBM RS/6000 SP, 208 SMP nodes, each SMP node with 16 POWER3+ CPUs, at NERSC



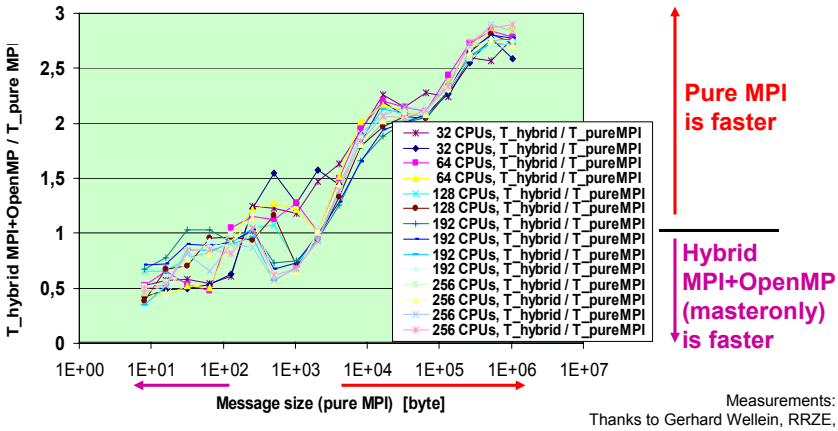
Measurements:
Thanks to Gerhard Wellein, RRZE, and Horst Simon, NERSC.

Additional slides (measured at NERSC)

pure MPI
Masteronly

Ratio T_hybrid MPI+OpenMP / T_pure MPI

IBM RS/6000 SP, 208 SMP nodes, each SMP node with 16 POWER3+ CPUs, at NERSC



Measurements:
Thanks to Gerhard Wellein, RRZE, and Horst Simon, NERSC.

Possible Reasons

- Hardware:
  - is one CPU able to saturate the inter-node network?
- Software:
  - internal MPI buffering may cause additional memory traffic
    → memory bandwidth may be the real restricting factor?

pure MPI
Masteronly

Optimizing the hybrid masteronly model

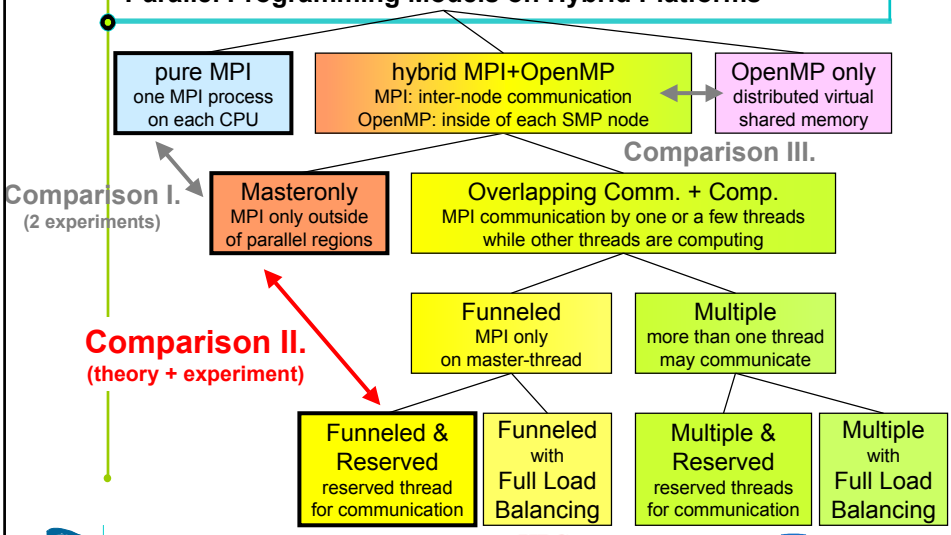
- By the MPI library:
  - Using multiple threads
    - using multiple memory paths (e.g., for strided data)
    - using multiple floating point units (e.g., for reduction)
    - using multiple communication links
      (if one link cannot saturate the hardware inter-node bandwidth)
  - requires knowledge about free CPUs,
    e.g., via new MPI_THREAD_MASTERONLY
- **By the user-application:**
  - **unburden MPI from work,**
    that can be done by the application
    - **e.g., concatenate strided data in parallel regions**
      instead of using MPI derived datatypes

pure MPI
Masteronly

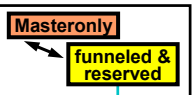
PROs & CONs "Hybrid"

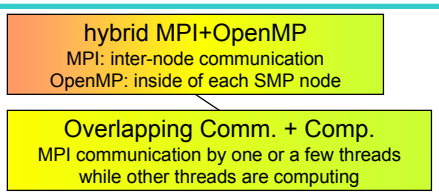
- PROs "hybrid"
  - more easy to fit hybrid hardware topology
  - no communication overhead inside of the SMP nodes
  - reduced #MPI messages & larger message sizes
    - **reduced latency-based overheads**
  - save memory
  - reduced #MPI processes
    - **better speedup (Amdahl's law)**
    - **faster convergence, e.g., if multigrid numeric is computed only on a partial grid**
- CONs "hybrid"
  - You may loose your cache (cache flushes inside of OpenMP)
  - Requires additional level of parallelism in problem/program
    - **Programming complexity increases & additional synchronization overhead**
  - Requires support by MPI library & mature OpenMP compiler
  - Problem to saturate the inter-node network
  - Idling CPUs in the *Masteronly* scheme

Parallel Programming Models on Hybrid Platforms



Overlapping computation & communication





Implications:

- all other threads are computing while master thread communicates
  - **better CPU usage**
- MPI-lib must provide at least MPI_THREAD_FUNNELED thread-safety
- inter-node bandwidth may be still used only partially
- Major drawback: load balancing necessary
  - alternative: reserved thread for communication → next slide

Masteronly ↔ funneled & reserved

Overlapping computation & communication

Alternative:

Funneled
MPI only
on master-thread

Funneled & Reserved
reserved thread
for communication

i.e., reserved tasks on threads:
- master thread: communication
- all other threads: computation

- cons:
  - bad load balance, if

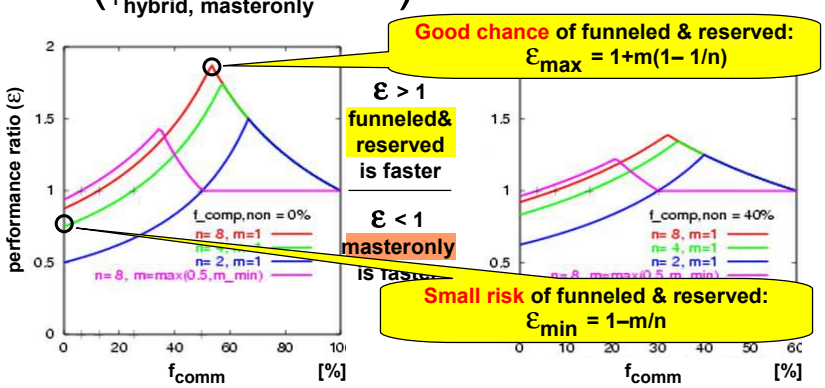
$$\frac{T_{communication}}{T_{computation}} \neq \frac{n_{communication\_threads}}{n_{computation\_threads}}$$

- pros:
  - more easy programming scheme than with full load balancing
  - chance for good performance!

Masteronly ↔ funneled & reserved

Performance ratio (theory)

- $\varepsilon = \left(\frac{T_{hybrid,\ funneled\&reserved}}{T_{hybrid,\ masteronly}}\right)^{-1}$

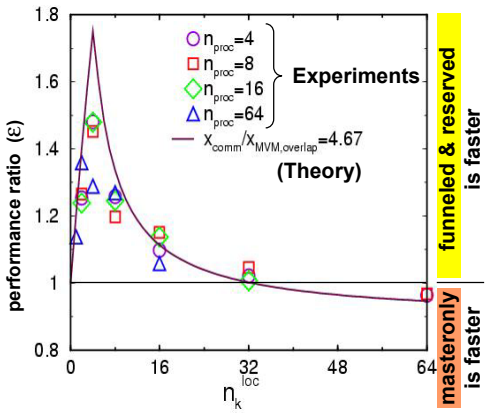


$T_{hybrid,\ masteronly} = (f_{comm} + f_{comp,\ non\text{-}overlap} + f_{comp,\ overlap})\ T_{hybrid,\ masteronly}$

n = # threads per SMP node, m = # reserved threads for MPI communication

Masteronly
funneled & reserved

Experiment: Matrix-vector-multiply (MVM)

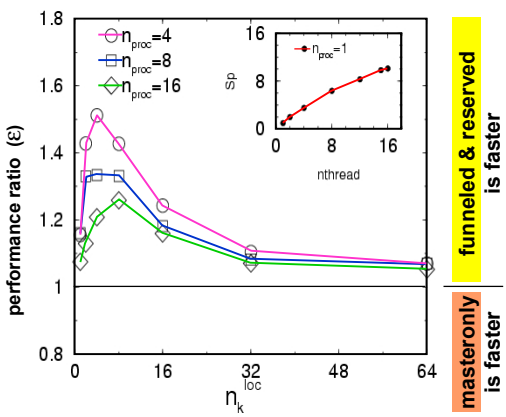


- Jacobi-Davidson-Solver
- Hitachi SR8000
- 8 CPUs / SMP node
- JDS (Jagged Diagonal Storage)
- vectorizing
- n_{proc} = # SMP nodes
- D_{Mat} = $512*512*(n_k^{loc}*n_{proc})$
- Varying n_k^{loc}
  $\Rightarrow$ Varying $1/f_{comm}$
- $\frac{f_{comp,non\text{-}overlap}}{f_{comp,overlap}} = \frac{1}{6}$

Source: R. Rabenseifner, G. Wellein:
Communication and Optimization Aspects of Parallel Programming Models.
EWOMP 2002, Rome, Italy, Sep. 18–20, 2002

Masteronly
funneled & reserved

Experiment: Matrix-vector-multiply (MVM)



- Same experiment on **IBM SP Power3** nodes with **16 CPUs per node**
- funneled&reserved is **always faster** in this experiments
- Reason:
  Memory bandwidth is already saturated by 15 CPUs, see inset
- Inset:
  Speedup on 1 SMP node using different number of threads

Source: R. Rabenseifner, G. Wellein:
Communication and Optimization Aspects of Parallel Programming Models on Hybrid Architectures.
International Journal of High Performance Computing Applications, Vol. 17, No. 1, 2003, Sage Science Press .

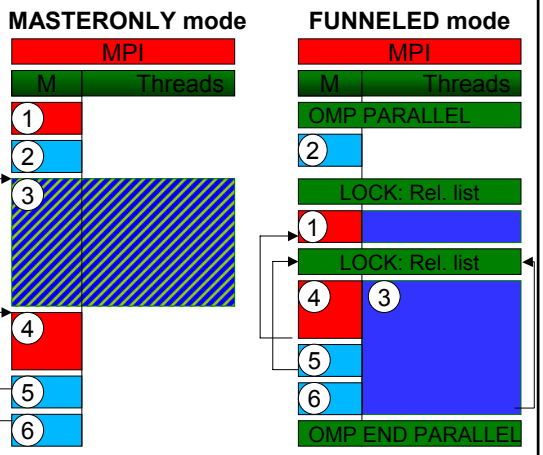
Hybrid MVM implementation

MASTERONLY mode:

- Automatic parallel. of inner i loop (data parallel)
- Single threaded MPI calls

FUNNELED mode:

- Functional parallelism: Simulate asynchronous data transfer ! (OpenMP)
- Release list - LOCK
- Single threaded MPI calls



Parallel Programming Models on Hybrid Platforms

hybrid MPI+OpenMP
MPI: inter-node communication
OpenMP: inside of each SMP node

OpenMP only
distributed virtual shared memory

Comparison III.

Comparison I.
(experiment)

Comparison II.
(theory + experiment)

Comparing other methods

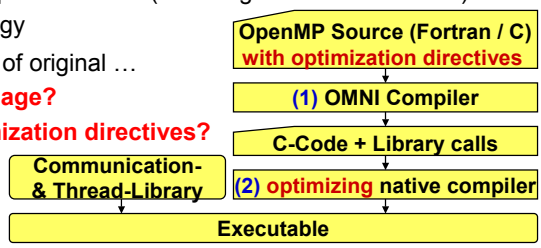
Memory copies from remote memory to local CPU register and vice versa

Access method	Copies	Remarks	bandwidth *b(message size)*
2-sided MPI	2	internal MPI buffer + application receive buf.	b(size) $= b_\infty / (1 + b_\infty T_{latency}/size)$
1-sided MPI	1	application receive buffer	same formula, but probably better b_∞ and $T_{latency}$
Compiler based: OpenMP on DSM (distributed shared memory) or with cluster extensions, UPC, Co-Array Fortran, HPF	1	page based transfer	extremely poor, if only parts are needed
	0	word based access	8 byte / $T_{latency}$, e.g, **8 byte / 0.33µs = 24MB/s**
	0	**latency hiding with pre-fetch**	**b_∞**
	1	latency hiding with buffering	see 1-sided communication

Compilation and Optimization

- Library based communication (e.g., MPI)
  - clearly separated optimization of
    - **(1)** communication → MPI library
    - **(2)** computation → Compiler

    } **essential for success of MPI**
- Compiler based parallelization (including the communication):
  - similar strategy
  - preservation of original …
    - **… language?**
    - **… optimization directives?**



- **Optimization of the computation <u>more important than</u> optimization of the communication**

Comparison I & II – Some conclusions

pure MPI
one MPI process
on
each CPU

Comparison I.
(experiment)

hybrid MPI+OpenMP
Masteronly
MPI only outside of parallel regions

Comparison II.
(theory + experiment)

Overlapping Comm. + Comp.
Funneled & Reserved
reserved thread
for communication

- Hybrid masteronly may not saturate the inter-node bandwidth
- Experiments: 1.5 … 2.7 times faster message transfer with pure MPI (only communication part!)
- hard topology problem with pure MPI

- Performance chance $\varepsilon < 2$ (with overlapping computation & communication and one communication thread per SMP)
- up to 50% total performance benefit with real matrix-vector-multiply
- cons: overlapping is hard & tricky

Conclusions

- ***Pure MPI*** versus ***hybrid masteronly*** model:
  - Topology problem with pure MPI may be hard, e.g., with unstructured grids
  - Communication is bigger with pure MPI, but may be nevertheless faster
  - On the other hand, typically communication is only some percent → relax
- Efficient hybrid programming:
  - one may overlap communication and computation → hard to do!
  - using simple ***hybrid funneled&reserved*** model, you maybe up to 50% faster (compared to *masteronly* model)
- If you want to use ***pure OpenMP*** (based on virtual shared memory)
  - try to use still the full bandwidth of the inter-node network (keep pressure on your compiler/DSM writer)
  - be sure that you do not lose any computational optimization
    - **e.g., best Fortran compiler & optimization directives should be usable**
- → optimal parallel programming model on clusters of SMP nodes, depends on many factors → no general recommendation