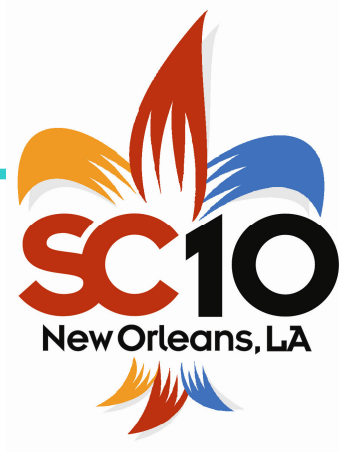


Hybrid MPI & OpenMP Parallel Programming

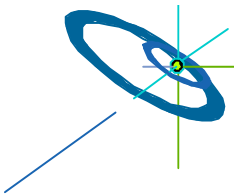
MPI + OpenMP and other models on clusters of SMP nodes

Rolf Rabenseifner[1)] Rabenseifner@hlrs.de
Georg Hager[2)] Georg.Hager@rrze.uni-erlangen.de
Gabriele Jost[3)] gjost@tacc.utexas.edu

[1)] High Performance Computing Center (HLRS), University of Stuttgart, Germany
[2)] Regional Computing Center (RRZE), University of Erlangen, Germany
[3)] Texas Advanced Computing Center, The University of Texas at Austin, USA

HPC-Europa2 Virtual Surgery on "Hybrid MPI/OpenMP"
Dec. 3, 2010

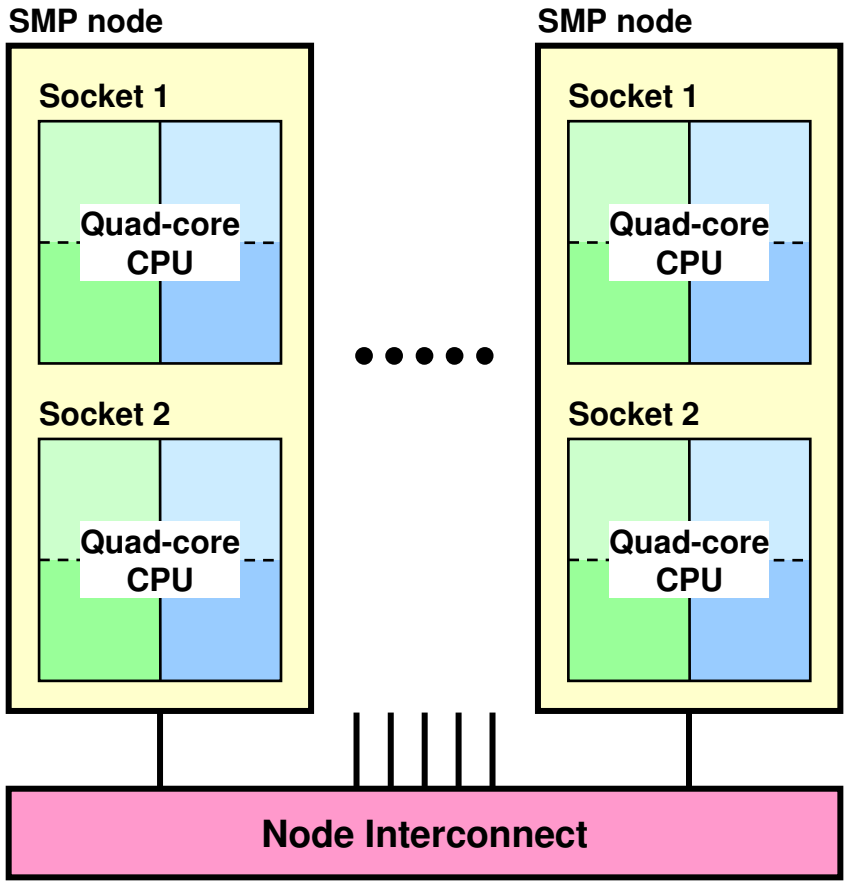
(Summary from "Tutorial M02 at SC10, November 15, 2010, New Orleans, LA, USA")



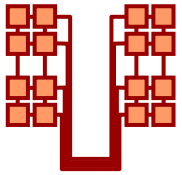
H L R S

TACC

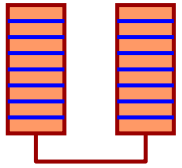
Motivation



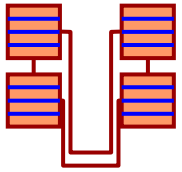
- Which programming model is fastest?
- MPI everywhere?



- Fully hybrid MPI & OpenMP?



- Something between? (Mixed model)



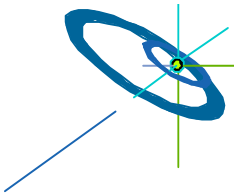
- Often hybrid programming **slower** than pure MPI
  - Examples, Reasons, …

Outline

- **Programming models on clusters of SMP nodes**
- Case Studies / pure MPI vs hybrid MPI+OpenMP
- Mismatch Problems
- Opportunities:
  Application categories that can benefit from hybrid parallelization
- Conclusion

Slides are available from

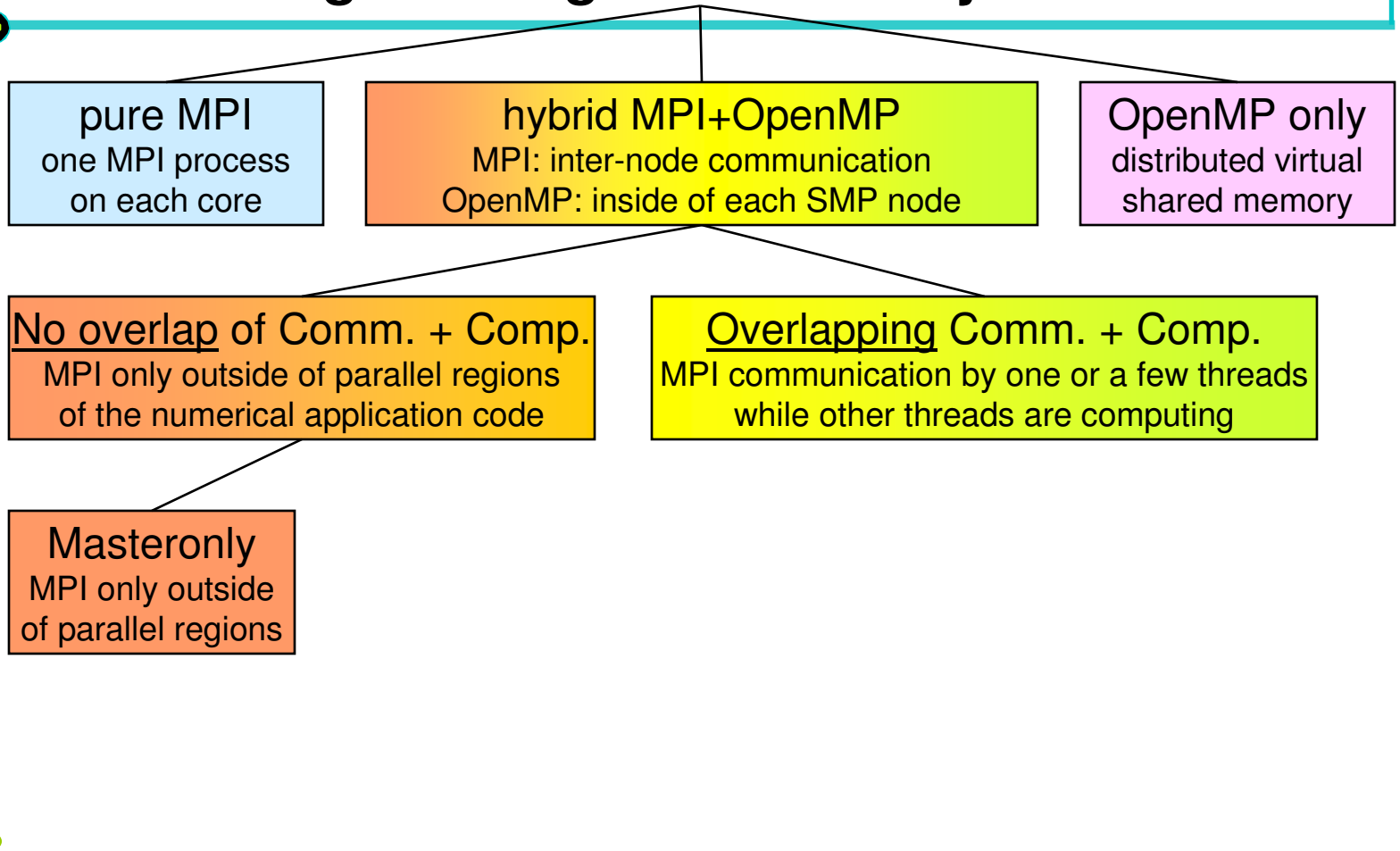
www.hlrs.de/people/rabenseifner
→ List of publications → International teaching



H L R S

TACC

Parallel Programming Models on Hybrid Platforms



Slides: www.hlrs.de/people/rabenseifner → List of publications → International teaching

Pure MPI

pure MPI
one MPI process
on each core

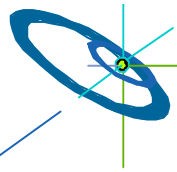
Advantages

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

Major problems

- Does MPI library uses internally different protocols?
  - **Shared memory inside of the SMP nodes**
  - **Network communication between the nodes**
- Does application topology fit on hardware topology?
- Unnecessary MPI-communication inside of SMP nodes!

Discussed in detail later on in the section **Mismatch Problems**



H L R S

TACC

Slides: www.hlrs.de/people/rabenseifner → List of publications → International teaching

Hybrid Masteronly

Masteronly
MPI only outside of parallel regions

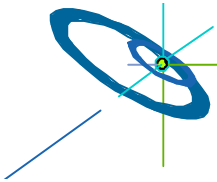
```
for (iteration ….)
{
  #pragma omp parallel
    numerical code
  /*end omp parallel */

  /* 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

Major Problems

- All other threads are sleeping while master thread communicates!
- Which inter-node bandwidth?
- MPI-lib must support at least MPI_THREAD_FUNNELED



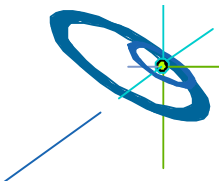
H L R S

TACC

Slides: www.hlrs.de/people/rabenseifner → List of publications → International teaching

Outline

- Programming models on clusters of SMP nodes
- **Case Studies / pure MPI vs hybrid MPI+OpenMP**
- Mismatch Problems
- Opportunities:
  Application categories that can benefit from hybrid parallelization
- Conclusion

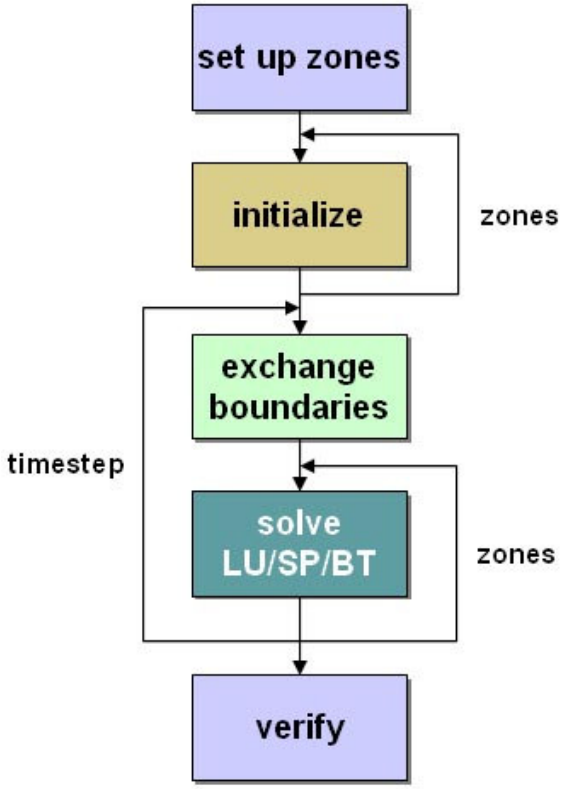


H L R S

TACC

Slides: www.hlrs.de/people/rabenseifner → List of publications → International teaching

The Multi-Zone NAS Parallel Benchmarks



	MPI/OpenMP	MLP	Nested OpenMP
Time step	sequential	sequential	sequential
inter-zones	MPI Processes	MLP Processes	OpenMP
exchange boundaries	Call MPI	data copy+ sync.	OpenMP
intra-zones	OpenMP	OpenMP	OpenMP

- Multi-zone versions of the NAS Parallel Benchmarks LU,SP, and BT
- Two hybrid sample implementations
- Load balance heuristics part of sample codes
- www.nas.nasa.gov/Resources/Software/software.html

Courtesy of Gabriele Jost, TACC

The Multi-Zone NAS Parallel Benchmarks

- Aggregate sizes:
  - Class D: 1632 x 1216 x 34 grid points
  - Class E: 4224 x 3456 x 92 grid points
- **BT-MZ: (Block tridiagonal simulated CFD application)**
  - Alternative Directions Implicit (ADI) method
  - #Zones: 1024 (D), 4096 (E)
  - Size of the zones varies widely:
    - large/small about 20
    - requires multi-level parallelism to achieve a good load-balance
- **SP-MZ: (Scalar Pentadiagonal simulated CFD application)**
  - #Zones: 1024 (D), 4096 (E)
  - Size of zones identical
    - no load-balancing required

Expectations:

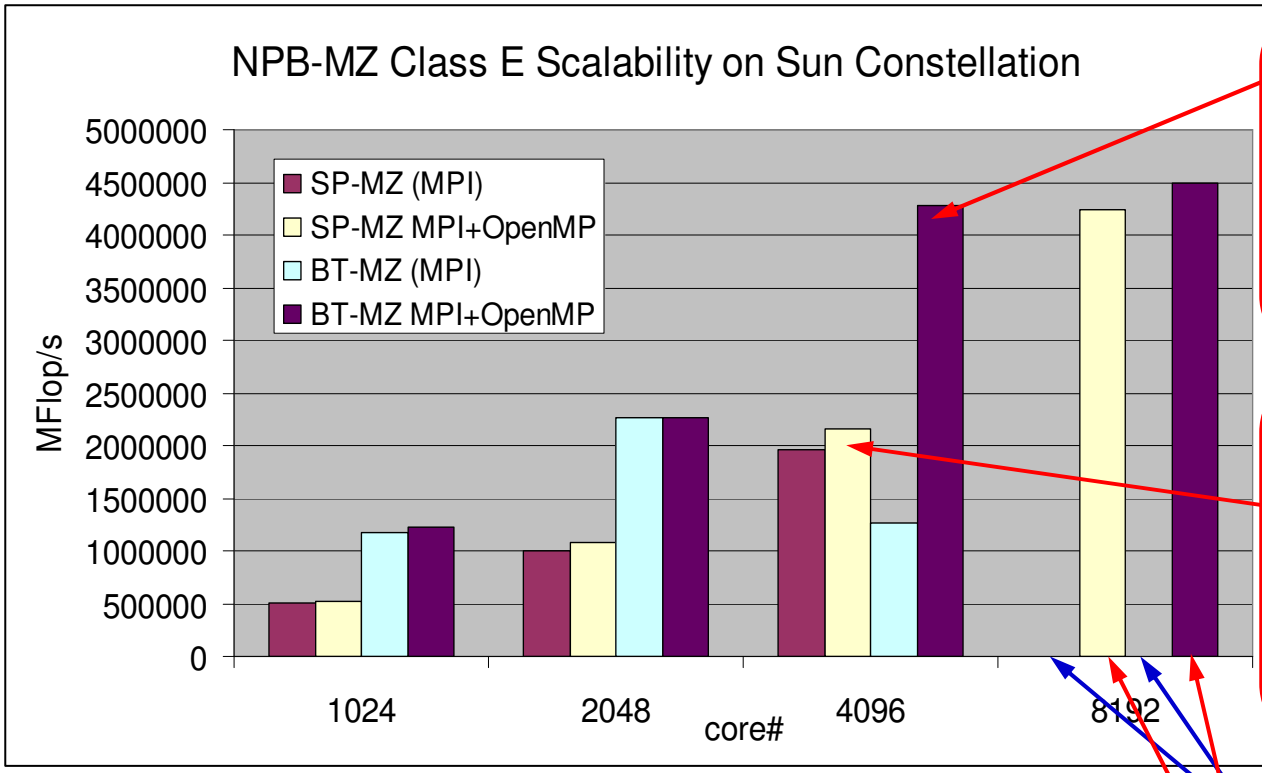
Pure MPI: Load-balancing problems!

Good candidate for MPI+OpenMP

Load-balanced on MPI level: Pure MPI should perform best

Courtesy of Gabriele Jost, TACC

SUN: NPB-MZ Class E Scalability on Ranger



BT

Significant improvement (235%):

Load-balancing issues solved with MPI+OpenMP

SP

Pure MPI is already load-balanced.

But hybrid 9.6% faster, due to smaller message rate at NIC

Cannot be build for 8192 processes!

Hybrid:

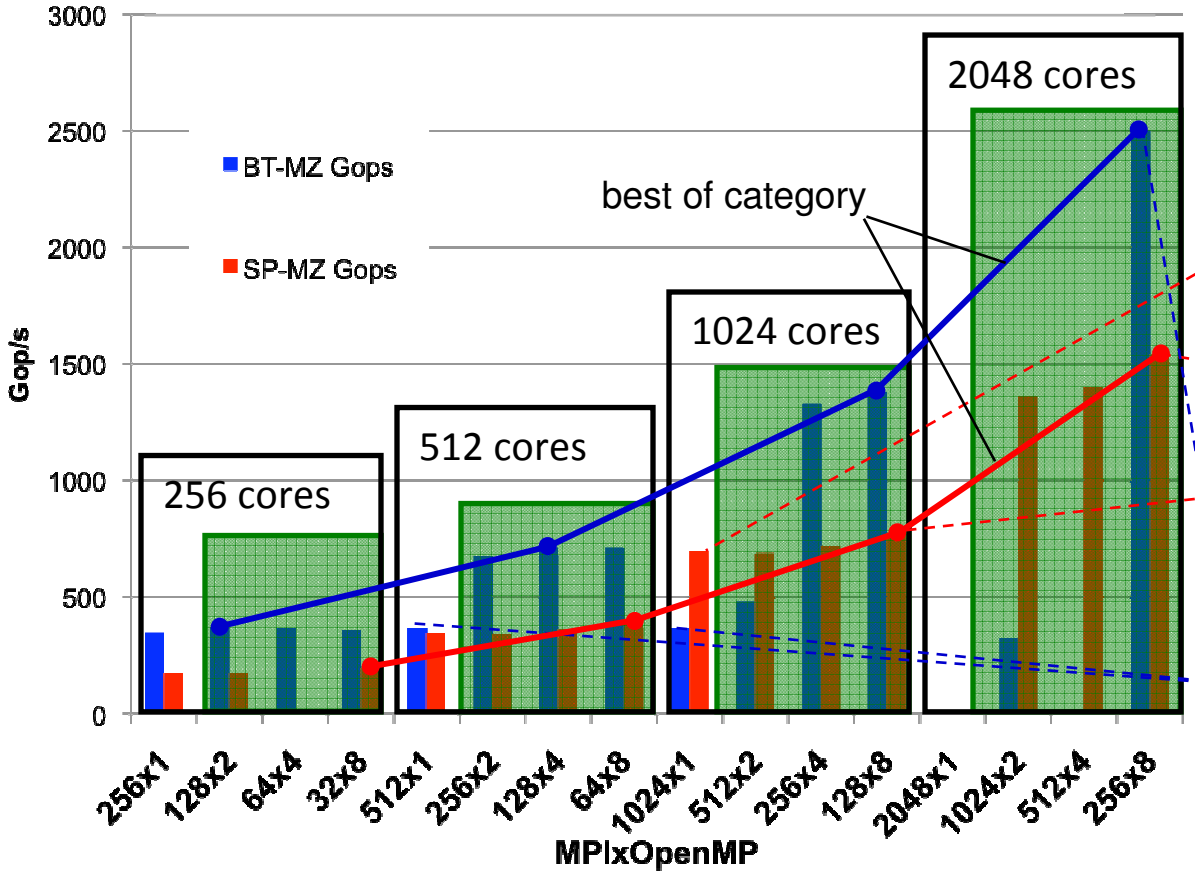
SP: still scales

BT: does not scale

- Scalability in Mflops
- MPI/OpenMP outperforms pure MPI
- Use of numactl essential to achieve scalability

Courtesy of Gabriele Jost, TACC

Cray XT5: NPB-MZ Class D Scalability



Results reported for Class D on 256-2048 cores

Expected: #MPI processes limited

- SP-MZ pure MPI scales up to 1024 cores
- SP-MZ MPI/OpenMP scales to 2048 cores
- SP-MZ MPI/OpenMP outperforms pure MPI for 1024 cores

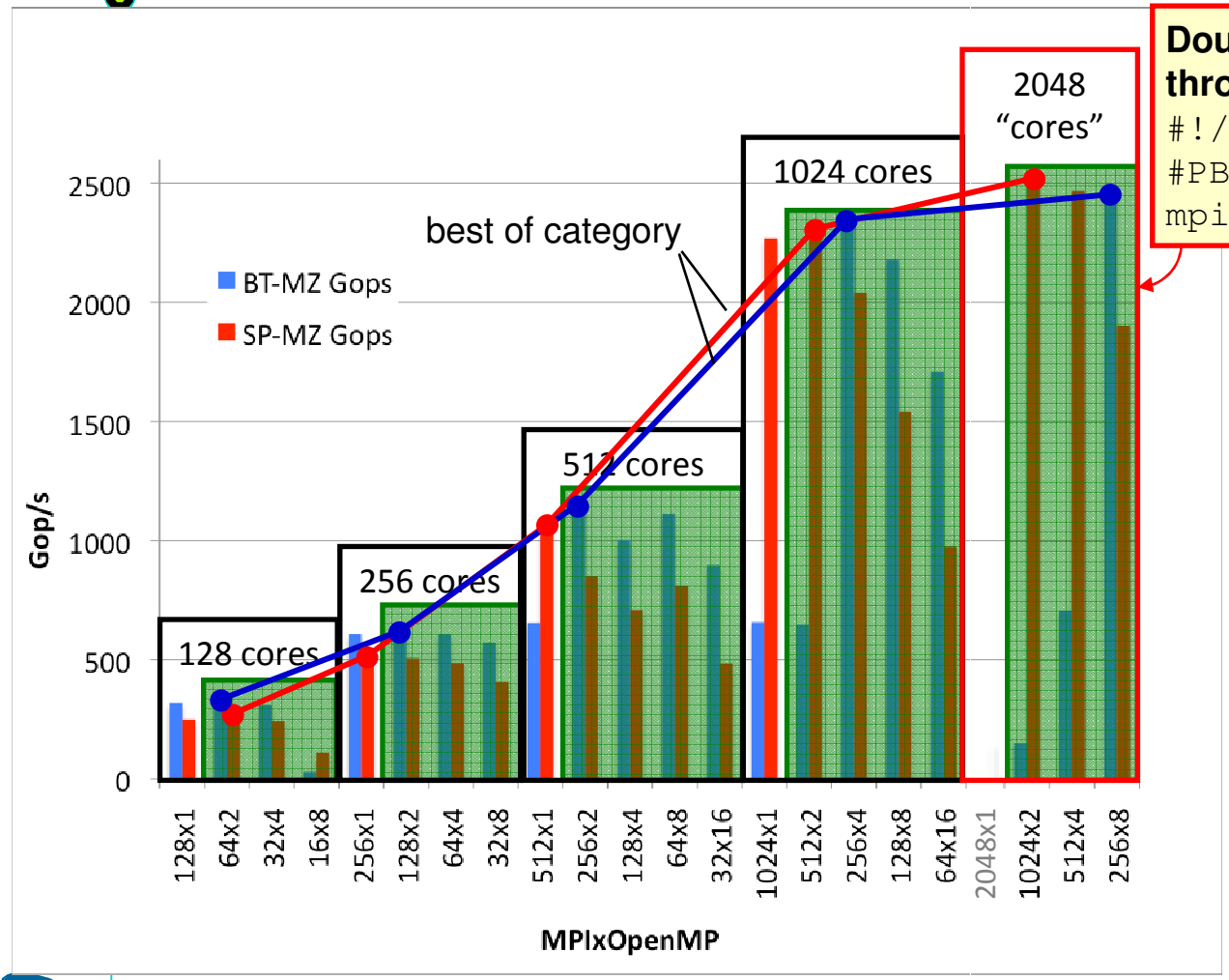
Unexpected!

- BT-MZ MPI does not scale
- BT-MZ MPI/OpenMP scales to 2048 cores, outperforms pure MPI

Expected: Load-Imbalance for pure MPI

Courtesy of Gabriele Jost, TACC

NPB-MZ Class D on IBM Power 6: Exploiting SMT for 2048 Core Results



Doubling the number of threads through hyperthreading (SMT):

```
#!/bin/csh
#PBS -l select=32:ncpus=64:
mpiprocs=NP:ompthreads=NT
```

- Results for 128-2048 cores
- Only 1024 cores were available for the experiments
- BT-MZ and SP-MZ show benefit from **Simultaneous Multithreading** (**SMT**): 2048 threads on 1024 cores

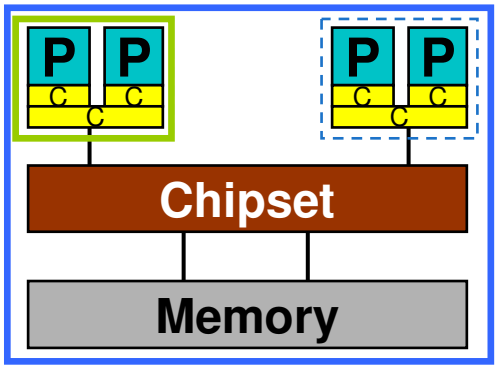
HLRS TACC

Courtesy of Gabriele Jost, TACC

Intra-node MPI characteristics: IMB Ping-Pong benchmark

- Code (to be run on 2 processors):

```
wc = MPI_WTIME()

do i=1,NREPEAT

  if(rank.eq.0) then
   MPI_SEND(buffer,N,MPI_BYTE,1,0,MPI_COMM_WORLD,ierr)
   MPI_RECV(buffer,N,MPI_BYTE,1,0,MPI_COMM_WORLD, &
                 status,ierr)
  else
   MPI_RECV(…)
   MPI_SEND(…)
  endif

enddo

wc = MPI_WTIME() - wc
```

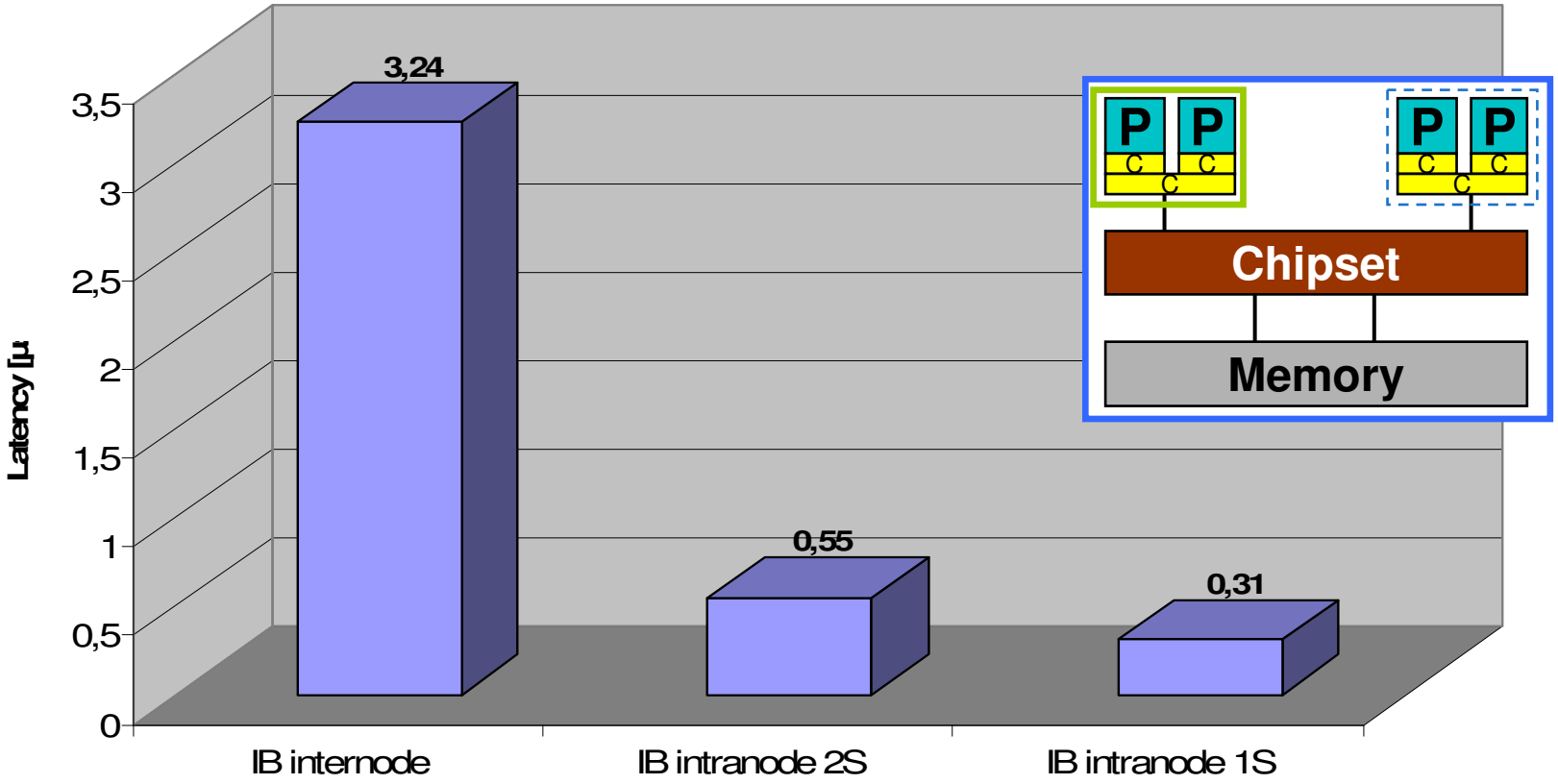


- Intranode (1S): `mpirun -np 2 -pin "1 3" ./a.out`
- Intranode (2S): `mpirun -np 2 -pin "2 3" ./a.out`
- Internode: `mpirun -np 2 -pernode ./a.out`

Courtesy of Georg Hager, RRZE

IMB Ping-Pong: Latency

Intra-node vs. Inter-node on Woodcrest DDR-IB cluster (Intel MPI 3.1)



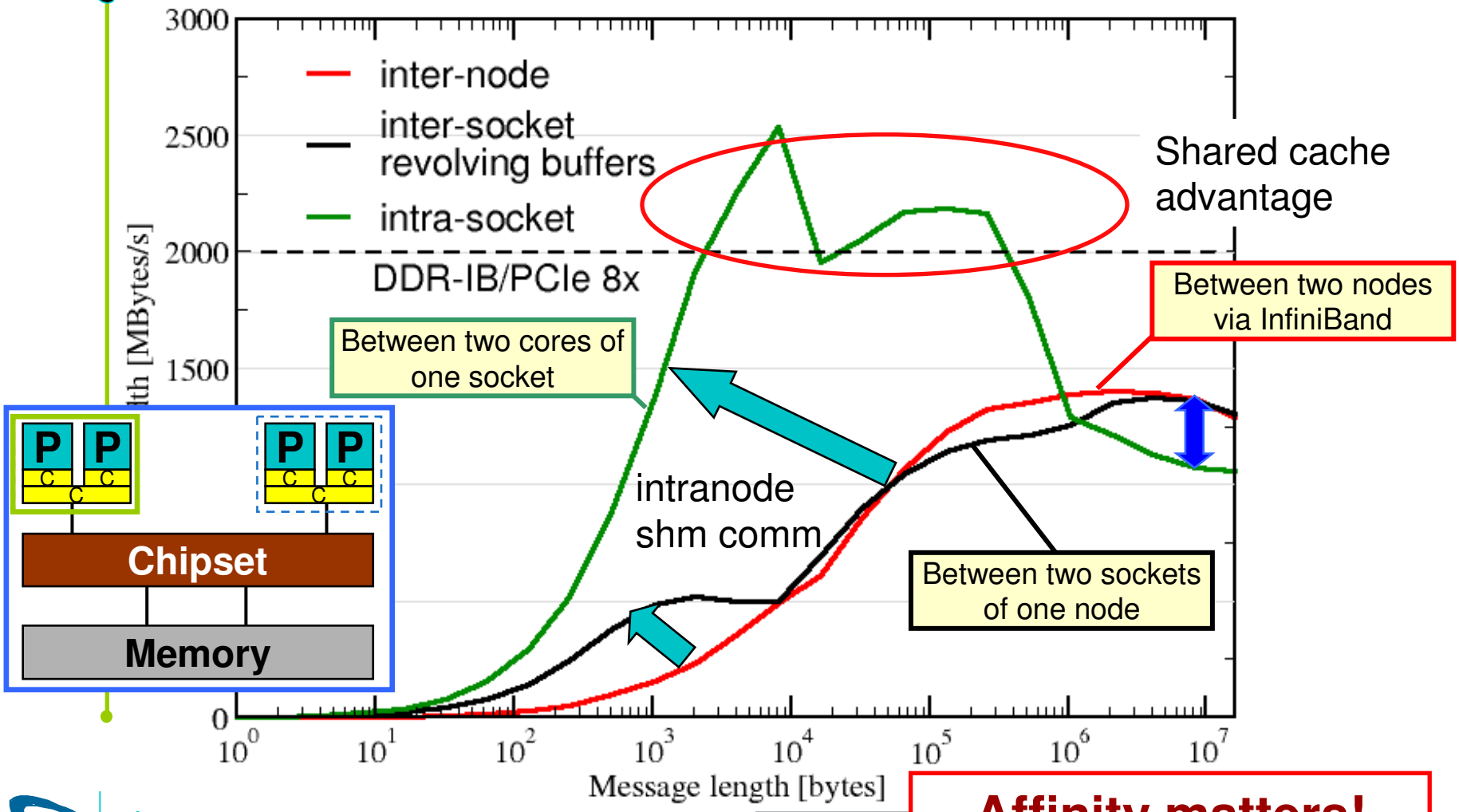
Affinity matters!

HLRS TACC

Courtesy of Georg Hager, RRZE

IMB Ping-Pong: Bandwidth Characteristics

Intra-node vs. Inter-node on Woodcrest DDR-IB cluster (Intel MPI 3.1)



Affinity matters!

Thread/Process Affinity ("Pinning")

- Highly OS-dependent system calls
  - But available on all systems

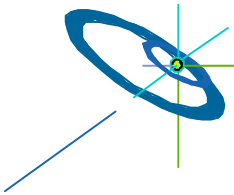
    Linux: `sched_setaffinity()`, PLPA (see below) → hwloc
    Solaris: `processor_bind()`
    Windows: `SetThreadAffinityMask()`
    …
- Support for "semi-automatic" pinning in some compilers/environments
  - Intel compilers > V9.1 (**`KMP_AFFINITY`** environment variable)
  - Pathscale
  - SGI Altix **`dplace`** (works with logical CPU numbers!)
  - Generic Linux: **`taskset`**, **`numactl`**, **`likwid-pin`**
- Affinity awareness in MPI libraries
  - SGI MPT
  - OpenMPI
  - Intel MPI
  - …

Used on SUN Ranger slides

Courtesy of Georg Hager, RRZE

Outline

- Programming models on clusters of SMP nodes
- Case Studies / pure MPI vs hybrid MPI+OpenMP
- **Mismatch Problems**
- Opportunities:
  Application categories that can benefit from hybrid parallelization
- Conclusion

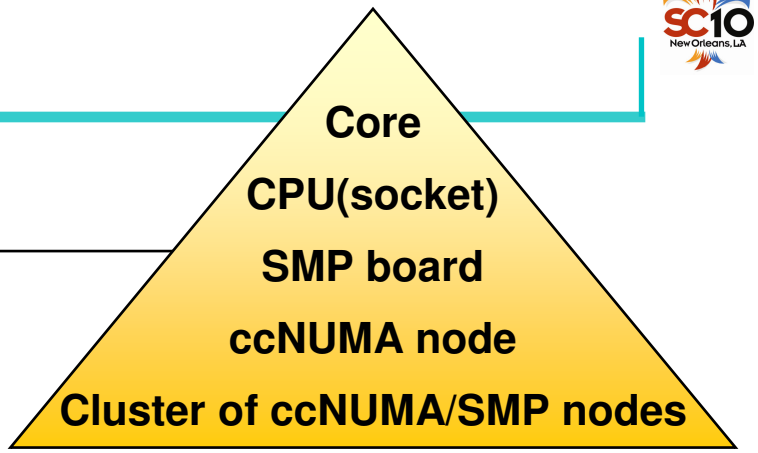


H L R S

TACC

Mismatch Problems

- None of the programming models fits to the hierarchical hardware (cluster of SMP nodes)
- Several mismatch problems
  → following slides
- Benefit through hybrid programming
  → Opportunities, see next section
- Quantitative implications
  → depends on you application



Examples:	No.1	No.2
Benefit through hybrid (see next section)	30%	10%
Loss by mismatch problems	–10%	–25%
Total	+20%	–15%

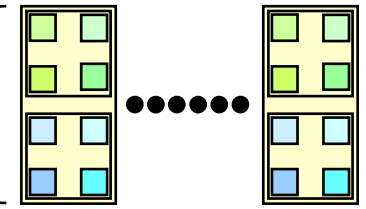
In most cases: **Both categories!**

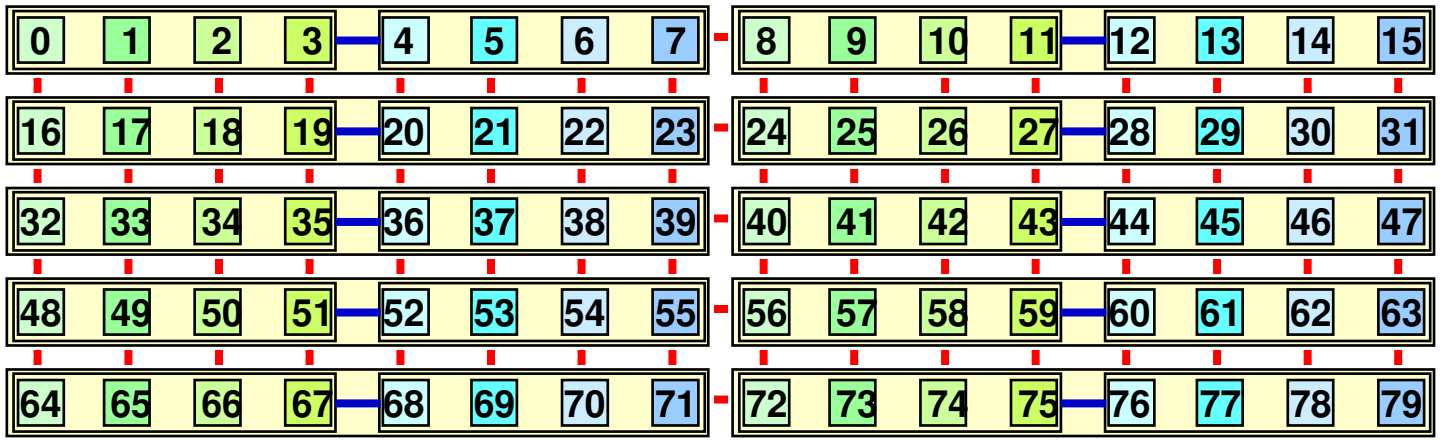
The Topology Problem with **pure MPI**

one MPI process on each core

Application example on 80 cores:

- Cartesian application with 5 x 16 = 80 sub-domains
- On system with 10 x dual socket x quad-core





\+ 17 x inter-node connections per node

− 1 x inter-socket connection per node

Sequential ranking of MPI_COMM_WORLD

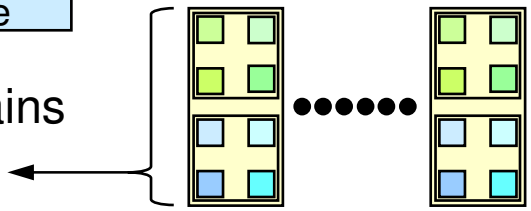
Does it matter?

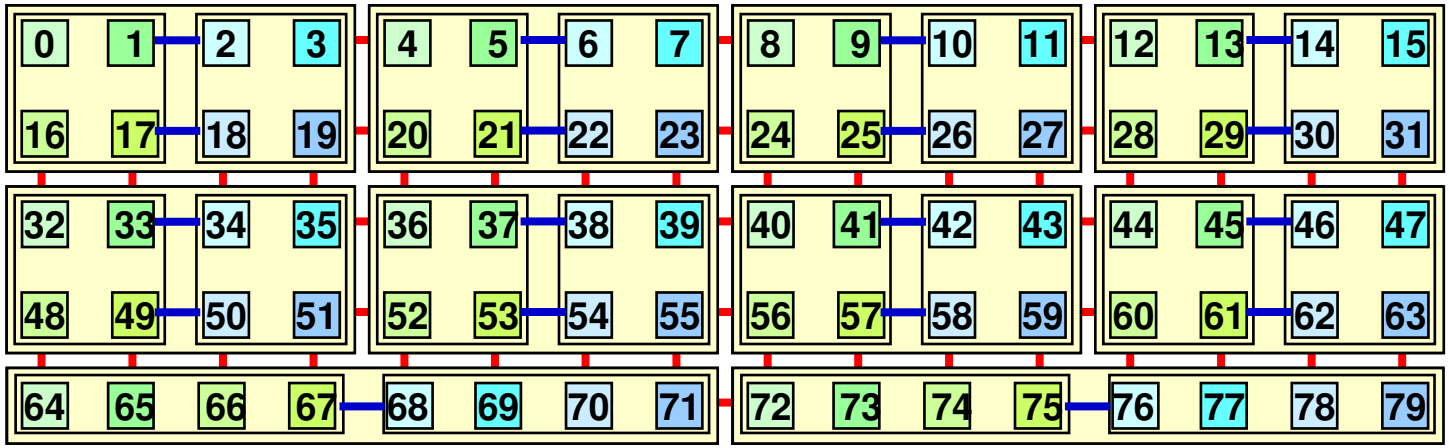
The Topology Problem with **pure MPI**

one MPI process on each core

Application example on 80 cores:

- Cartesian application with 5 x 16 = 80 sub-domains
- On system with 10 x dual socket x quad-core





\+ 12 x inter-node connections per node

\+ 2 x inter-socket connection per node

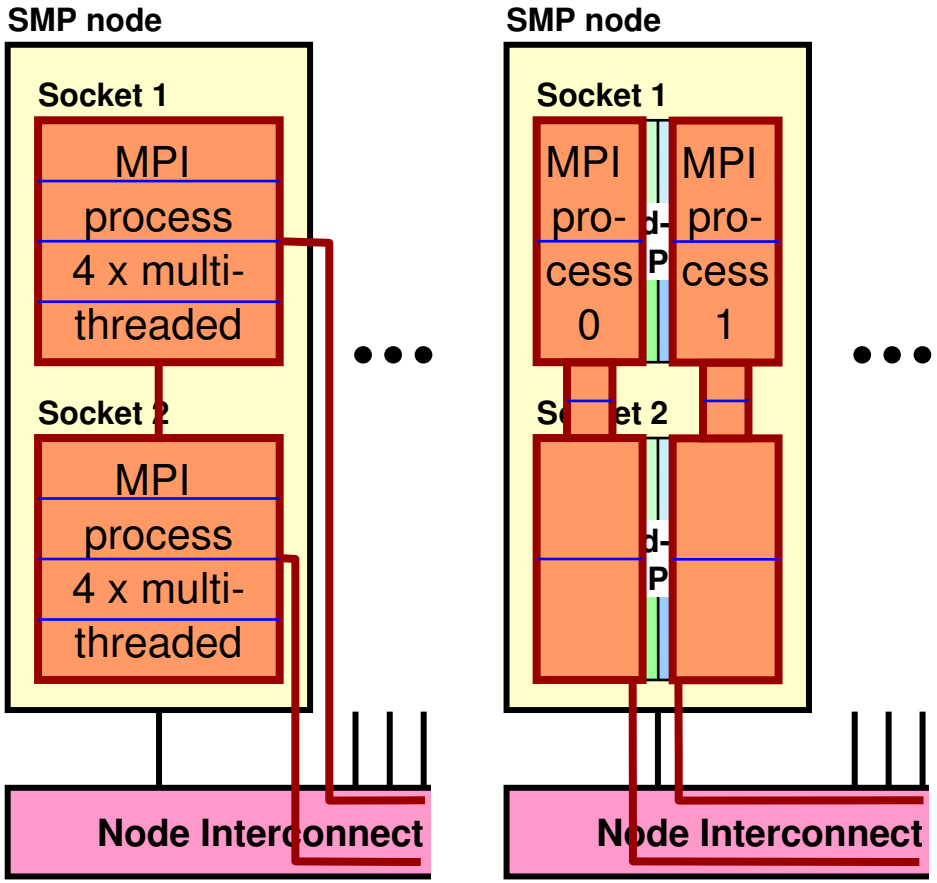
Two levels of domain decomposition

Good affinity of cores to thread ranks

The Mapping Problem with mixed model

pure MPI
&
hybrid MPI+OpenMP

Do we have this? … or that?



Several multi-threaded MPI process per SMP node:

Problem
- Where are your processes and threads really located?

Solutions:
- Depends on your platform,
- e.g., with **numactl**

→ Case study on Sun Constellation Cluster Ranger with BT-MZ and SP-MZ

Further questions:
- Where is the NIC[1)] located?
- Which cores share caches?

HLRS TACC

1) NIC = Network Interface Card

Unnecessary intra-node communication

pure MPI

Mixed model
(several multi-threaded MPI processes per SMP node)

Problem:

- If several MPI process on each SMP node
  → unnecessary intra-node communication

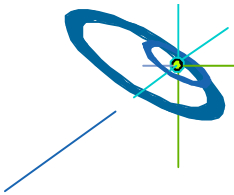
Solution:

- Only one MPI process per SMP node

Remarks:

- MPI library must use appropriate fabrics / protocol for intra-node communication
- Intra-node bandwidth higher than inter-node bandwidth
  → problem may be small
- MPI implementation may cause unnecessary data copying
  → waste of memory bandwidth

Quality aspects of the MPI library



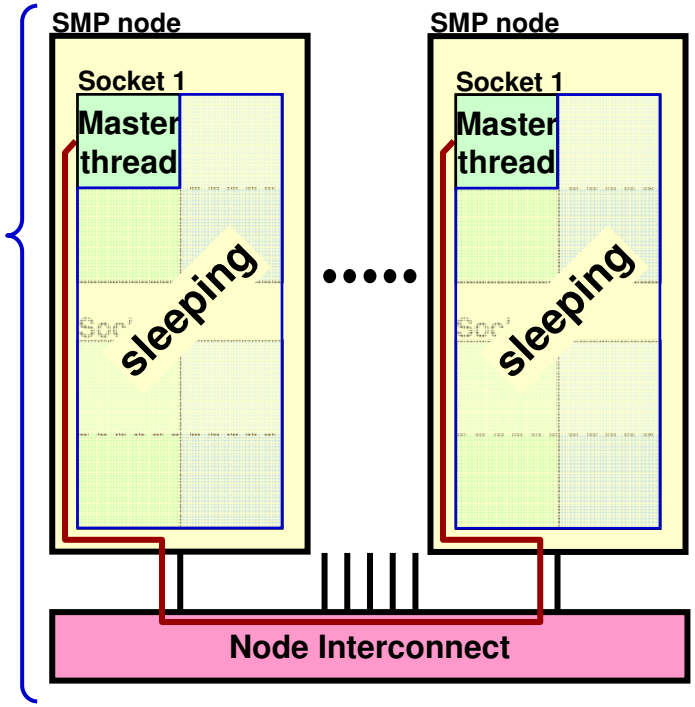
H L R S

TACC

Sleeping threads and network saturation with 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
```



Problem 1:

- **Can the master thread saturate the network?**

Solution:

- If not, use mixed model
- i.e., several MPI processes per SMP node

Problem 2:

- **Sleeping threads are wasting CPU time**

Solution:

- Overlapping of computation and communication

Problem 1&2 together:

- Producing more idle time through lousy bandwidth of master thread

OpenMP: Additional Overhead & Pitfalls

- Using OpenMP
  - → may prohibit compiler optimization
  - → **may cause significant loss of computational performance**
- Thread fork / join overhead
- On ccNUMA SMP nodes:
  - **Loss of performance due to missing memory page locality or missing first touch strategy**
  - E.g. with the masteronly scheme:
    - One thread produces data
    - Master thread sends the data with MPI

    → data may be internally communicated from one memory to the other one
- Amdahl's law for each level of parallelism
- Using MPI-parallel application libraries? → Are they prepared for hybrid?

See, e.g., the necessary **–O4** flag with mpxlf_r on IBM Power6 systems

Overlapping Communication and Computation

MPI communication by one or a few threads while other threads are computing

Three problems:

- the application problem:
  - one must separate application into:
    - **code that can run before the halo data is received**
    - **code that needs halo data**

  ➔very hard to do !!!

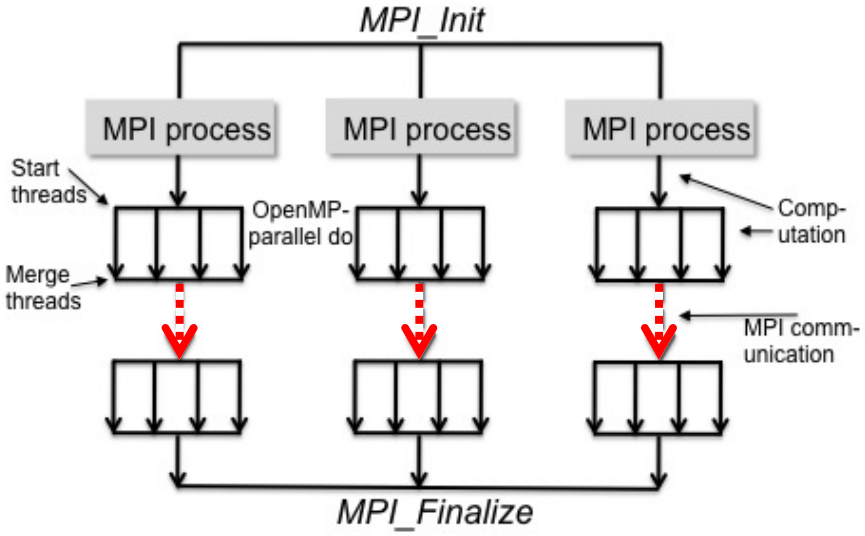
- the thread-rank problem:
  - comm. / comp. via thread-rank
  - cannot use work-sharing directives

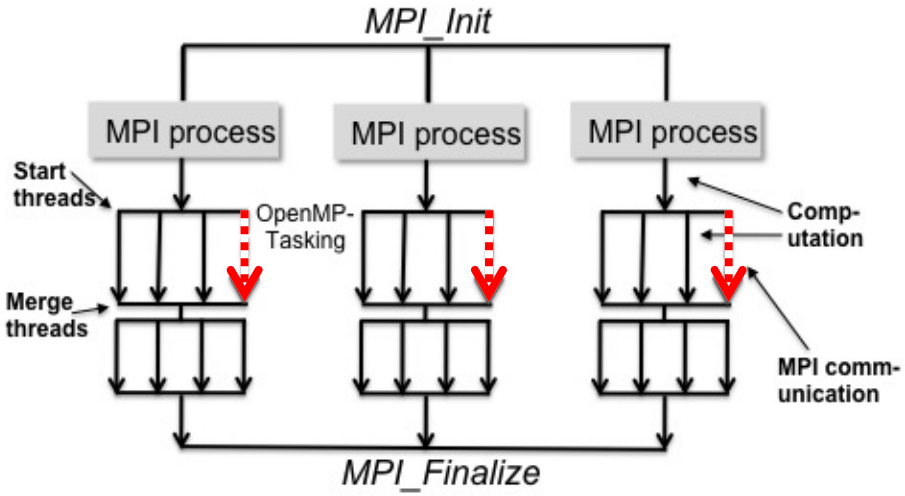
  ➔loss of major OpenMP support
  (see next slide)

- the load balancing problem

```
if (my_thread_rank < 1) {
   MPI_Send/Recv….
} else {
   my_range = (high-low-1) / (num_threads-1) + 1;
   my_low = low + (my_thread_rank+1)*my_range;
   my_high=high+ (my_thread_rank+1+1)*my_range;
   my_high = max(high, my_high)
   for (i=my_low; i<my_high; i++) {
            ….
   }
}
```

Overlapping: Using OpenMP tasks





NEW OpenMP Tasking Model gives a new way to achieve more parallelism form hybrid computation.

Alice Koniges et al.:
Application Acceleration on Current and Future Cray Platforms.
Proceedings, CUG 2010, Edinburgh, GB, May 24-27, 2010.

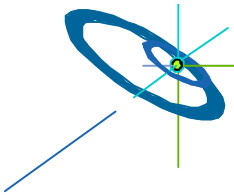
Courtesy of Alice Koniges, NERSC, LBNL

No silver bullet

- The analyzed programming models do **not** fit on hybrid architectures
  - whether drawbacks are minor or major
    - **depends on applications' needs**
  - But there are major opportunities → next section

- In the NPB-MZ case-studies
  - We tried to use optimal parallel environment
    - **for pure MPI**
    - **for hybrid MPI+OpenMP**
  - i.e., the developers of the MZ codes and we tried to minimize the mismatch problems

→ the opportunities in next section dominated the comparisons

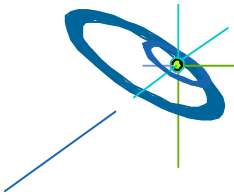


H L R S

TACC

Outline

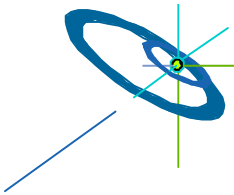
- Programming models on clusters of SMP nodes
- Case Studies / pure MPI vs hybrid MPI+OpenMP
- Mismatch Problems
- **Opportunities:**
  Application categories that can benefit from hybrid parallelization
- Conclusion



Nested Parallelism

- Example NPB: BT-MZ (Block tridiagonal simulated CFD application)
  - Outer loop:
    - **limited number of zones → limited parallelism**
    - **zones with different workload → speedup < $\frac{\text{Sum of workload of all zones}}{\text{Max workload of a zone}}$**
  - Inner loop:
    - **OpenMP parallelized (static schedule)**
    - **Not suitable for distributed memory parallelization**

- Principles:
  - Limited parallelism on outer level
  - Additional inner level of parallelism
  - Inner level not suitable for MPI
  - Inner level may be suitable for static OpenMP worksharing



Load-Balancing (on same or different level of parallelism)

- OpenMP enables
  - Cheap ***dynamic*** and ***guided*** load-balancing
  - Just a parallelization option (clause on omp for / do directive)
  - Without additional software effort
  - Without explicit data movement

```
#pragma omp parallel for schedule(dynamic)
for (i=0; i<n; i++) {
  /* poorly balanced iterations */ …
}
```

- On MPI level
  - **Dynamic load balancing** requires moving of parts of the data structure through the network
  - Significant runtime overhead
  - Complicated software / therefore not implemented

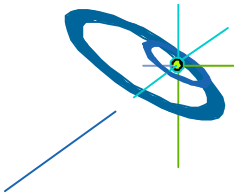
- **MPI & OpenMP**
  - Simple static load-balancing on MPI level, dynamic or guided on OpenMP level

} **medium quality**
cheap implementation

Memory consumption

- Shared nothing
  - Heroic theory
  - In practice: Some data is duplicated

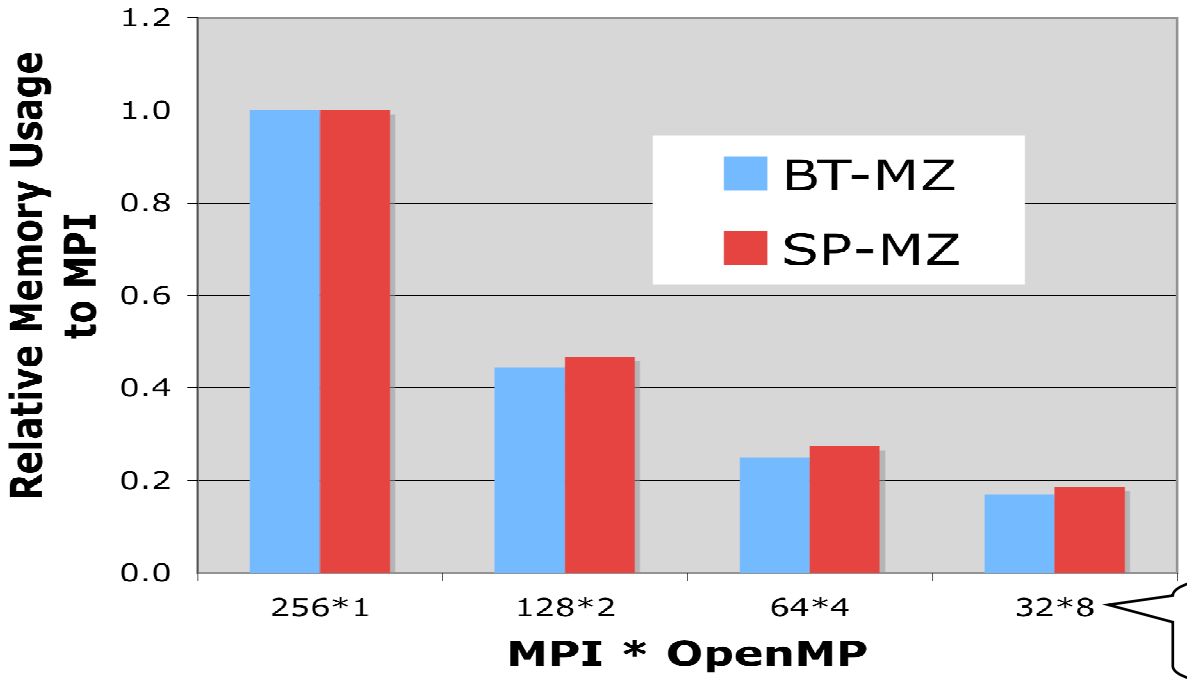
- **MPI & OpenMP**
  With n threads per MPI process:
  - Duplicated data may be reduced by factor n



H L R S

TACC

Case study: MPI+OpenMP memory usage of NPB



Using more OpenMP threads could reduce the memory usage **substantially**, up to **five** times on Hopper Cray XT5 (eight-core nodes).

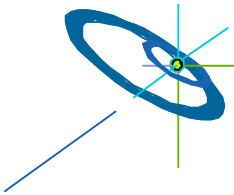
Always same number of cores

Hongzhang Shan, Haoqiang Jin, Karl Fuerlinger, Alice Koniges, Nicholas J. Wright: Analyzing the Effect of Different Programming Models Upon Performance and Memory Usage on Cray XT5 Platorms. Proceedings, CUG 2010, Edinburgh, GB, May 24-27, 2010.

Courtesy of Alice Koniges, NERSC, LBLN

How many threads per MPI process?

- SMP node = with **m sockets** and **n cores/socket**
- How many threads (i.e., cores) per MPI process?
  - Too many threads per MPI process
    → overlapping of MPI and computation may be necessary,
    → some NICs unused?
  - Too few threads
    → too much memory consumption (see previous slides)
- Optimum
  - somewhere between 1 and m x n threads per MPI process,
  - Typically:
    - **Optimum = n, i.e., 1 MPI process per socket**
    - **Sometimes = n/2 i.e., 2 MPI processes per socket**
    - **Seldom = 2n, i.e., each MPI process on 2 sockets**

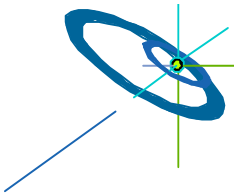


H L R S

TACC

To overcome MPI scaling problems

- Reduced number of MPI messages, reduced aggregated message size } compared to pure MPI
- MPI has a few scaling problems
  - Handling of more than 10,000 MPI processes
  - Irregular Collectives: MPI_....v(), e.g. MPI_Gatherv()
    - **Scaling applications should not use MPI_....v() routines**
  - MPI-2.1 Graph topology (MPI_Graph_create)
    - **MPI-2.2 MPI_Dist_graph_create_adjacent**
  - Creation of sub-communicators with MPI_Comm_create
    - **MPI-2.2 introduces a new scaling meaning of MPI_Comm_create**
  - … see P. Balaji, et al.: **MPI on a Million Processors.** Proceedings EuroPVM/MPI 2009.
- Hybrid programming reduces all these problems (due to a smaller number of processes)

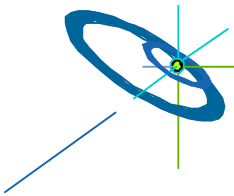


H L R S

TACC

Summary: Opportunities of hybrid parallelization (MPI & OpenMP)

- Nested Parallelism
  → Outer loop with MPI / inner loop with OpenMP
- Load-Balancing
  → Using OpenMP ***dynamic*** and ***guided*** worksharing
- Memory consumption
  → Significantly reduction of replicated data on MPI level
- Opportunities, if MPI speedup is limited due to algorithmic problem
  → Significantly reduced number of MPI processes
- Reduced MPI scaling problems
  → Significantly reduced number of MPI processes



H L R S

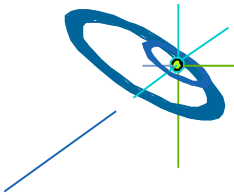
TACC

Conclusions

- Future hardware will be more complicated
  - Heterogeneous → GPU, FPGA, …
  - ccNUMA quality may be lost on cluster nodes
  - ….
- High-end programming → more complex
- Medium number of cores → programming will be more simple (provided that **#cores / SMP-node** will not shrink)
- **MPI+OpenMP → work horse on large systems**
- Pure MPI → still on smaller cluster
- OpenMP → on large ccNUMA nodes
  (**but not with shared virtual memory systems (e.g., ClusterOpenMP)**)

Thank you for your interest

→ Next talk by Rainer Keller



H L R S

TACC