Message-Passing and Hybrid Parallelization on Clusters of Multi-Core SMP Nodes

Rolf Rabenseifner

Rabenseifner@hlrs.de

High Performance Computing Center (HLRS),
University of Stuttgart, Germany

www.hlrs.de

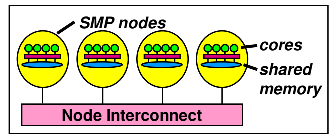
Invited Talk at Potsdam University, Computer Science Institute
Potsdam, Jan. 20, 2009

Aspects & Outline

- **Future High Performance Computing (HPC)**
  - → always hierarchical hardware design
- **Mismatches and opportunities with current MPI based programming models**
  - → Some new features are needed
  - → Some optimizations can be done best by the application itself
- **Optimization always requires knowledge on the hardware:**
  - → Qualitative and quantitative information is needed
  - → through a standardized interface
- **The MPI-3 Forum tries to address those aspects**
  - → MPI-2.1 is only a starting point:
    combination of MPI-1.1 and 2.0 in one book

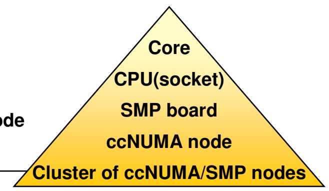
Future High Performance Computing (HPC) → always hierarchical hardware design

- Efficient programming of clusters of SMP nodes
  SMP nodes:
  - **Dual/multi core CPUs**
  - **Multi CPU shared memory**
  - **Multi CPU ccNUMA**
  - **Any mixture with shared memory programming model**



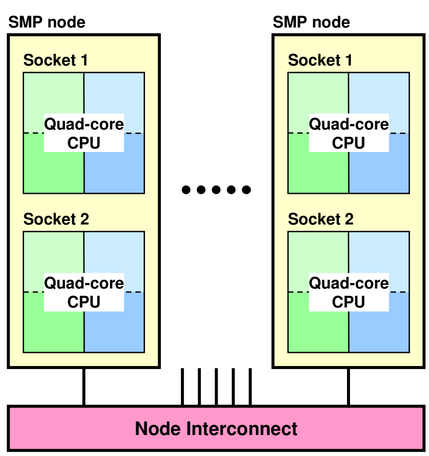
- Hardware range
  - **mini-cluster with dual-core CPUs**
  - **...**
  - **large constellations with large SMP nodes**
    ... with several sockets (CPUs) per SMP node
    ... with several cores per socket

  → Hierarchical system layout

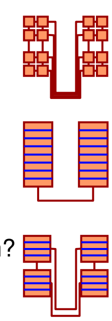


- **Hybrid MPI/OpenMP programming seems natural**
  - **MPI between the nodes**
  - **OpenMP inside of each SMP node**

Which is best programming model?



- Which programming model is fastest?
- MPI everywhere?
- Fully hybrid MPI & OpenMP?
- Something between? (Mixed model)

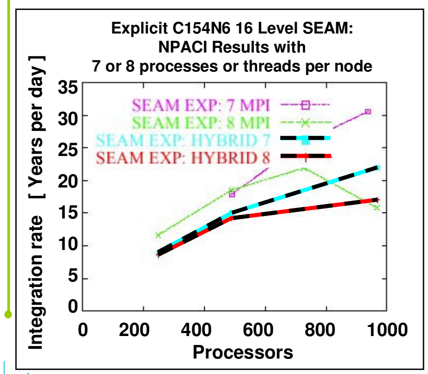


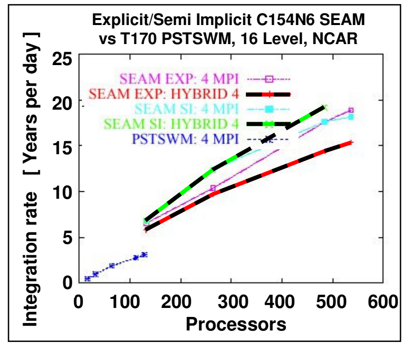
- Often hybrid programming **slower** than pure MPI
  - Examples, Reasons, …

?

Example from SC

- 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.

H L R I S

Goals

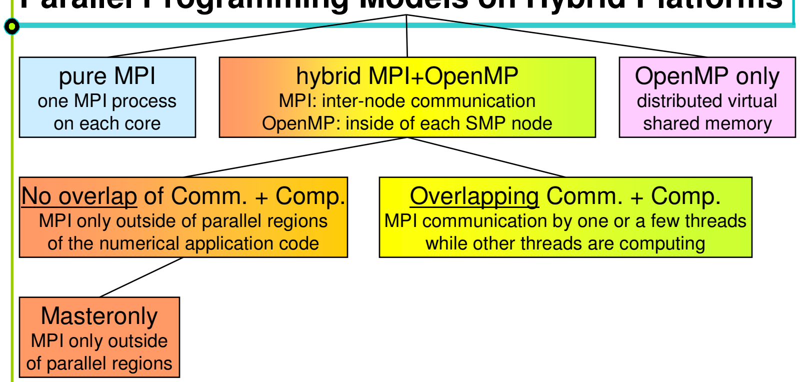
Minimizing

- Communication overhead,
  - **e.g., messages inside of one SMP node**
- Synchronization overhead
  - **e.g., OpenMP fork/join**
- Load imbalance
  - **e.g., using OpenMP *guided* worksharing schedule**
- Memory consumption
  - **e.g., replicated data in MPI parallelization**
- Computation overhead
  - **e.g., duplicated calculations in MPI parallelization**

} Optimal parallel scaling

H L R I S

Parallel Programming Models on Hybrid Platforms



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!

Hybrid Masteronly

Masteronly
MPI only outside of parallel regions

Advantages

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

```
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
```

Major Problems

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

Overlapping Communication and Computation

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

```
if (my_thread_rank < …) {
  MPI_Send/Recv….
    i.e., communicate all halo data
} else {
  Execute those parts of the application
    that do not need halo data
    (on non-communicating threads)
}

Execute those parts of the application
  that need halo data
  (on all threads)
```

Pure OpenMP (on the cluster)

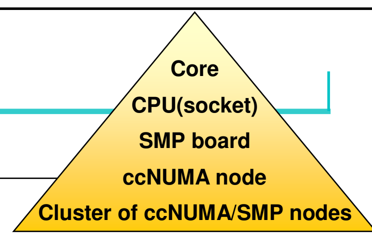
OpenMP only
distributed virtual
shared memory

- Distributed shared virtual memory system needed
- Must support clusters of SMP nodes
- e.g., Intel® Cluster OpenMP
  - Shared memory parallel inside of SMP nodes
  - Communication of modified parts of pages at OpenMP flush (part of each OpenMP barrier)

by rule of thumb:
Communication may be 10 times slower than with MPI
→ Appendix

i.e., the OpenMP memory and parallelization model
is prepared for clusters!

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

- **Less frustration and**
- **More success**

with your parallel program on clusters of SMP nodes

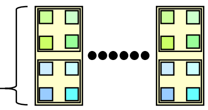
Examples:	No.1	No.2
Benefit through hybrid (see next section)	30%	10%
Loss by mismatch problems	–10%	–25%
Total	+20%	–15%

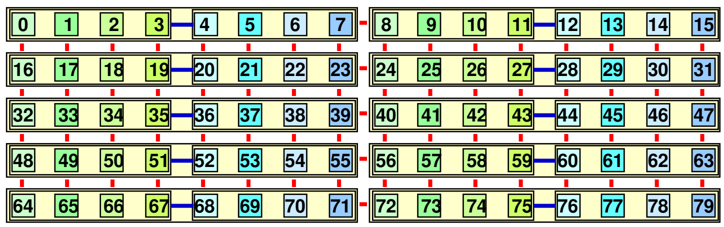
In most cases: **Both categories!**

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

Does it matter?

Sequential ranking of MPI_COMM_WORLD

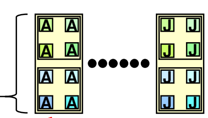
Hybrid Parallelization on Clusters of Multi-Core
Slide 13 / 50 Rolf Rabenseifner

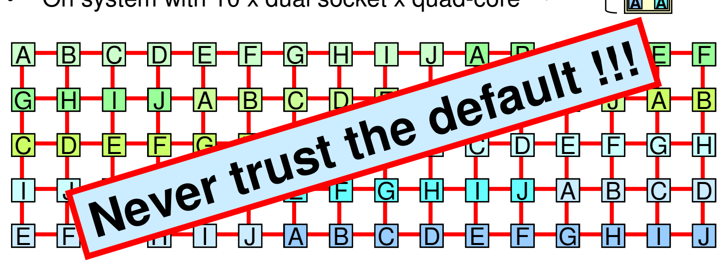
H L R I S

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





+ 32 x inter-node connections per node
– 0 x inter-socket connection per node

Round robin ranking of MPI_COMM_WORLD

Hybrid Parallelization on Clusters of Multi-Core
Slide 14 / 50 Rolf Rabenseifner

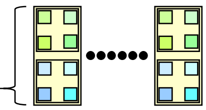
H L R I S

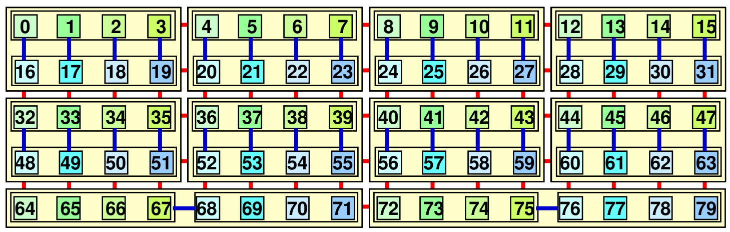
The Topology Problem with **pure MPI**

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





+ 10 x inter-node connections per node

+ 4 x inter-socket connection per node

Two levels of domain decomposition

Bad affinity of cores to thread ranks

Hybrid Parallelization on Clusters of Multi-Core
Slide 15 / 50 Rolf Rabenseifner

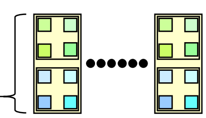
HLRS

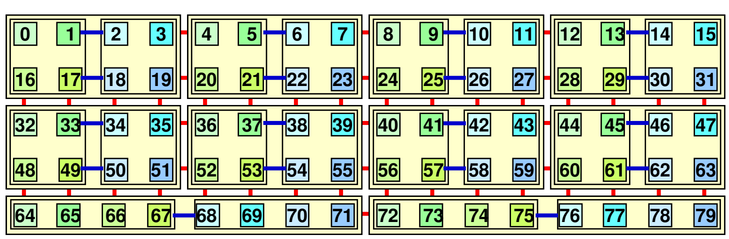
The Topology Problem with **pure MPI**

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





+ 10 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

Hybrid Parallelization on Clusters of Multi-Core
Slide 16 / 50 Rolf Rabenseifner

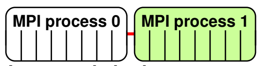
HLRS

The Topology Problem with hybrid MPI+OpenMP

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

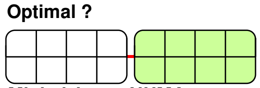
Exa.: 2 SMP nodes, 8 cores/node

Optimal ?



Loop-worksharing on 8 threads

Optimal ?



Minimizing ccNUMA data traffic through domain decomposition inside of each MPI process

Problem
- Does application topology inside of SMP parallelization fit on inner hardware topology of each SMP node?

Solutions:
- Domain decomposition inside of each thread-parallel MPI process, and
- first touch strategy with OpenMP

Successful examples:
- Multi-Zone NAS Parallel Benchmarks (MZ-NPB)

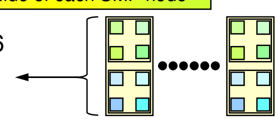
Hybrid Parallelization on Clusters of Multi-Core
Slide 17 / 50 Rolf Rabenseifner

H L R I S

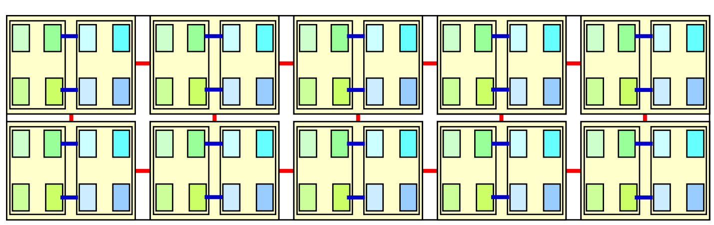
The Topology Problem with hybrid MPI+OpenMP

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

Application example:
- Same Cartesian application aspect ratio: 5 x 16
- On system with 10 x dual socket x quad-core
- 2 x 5 domain decomposition



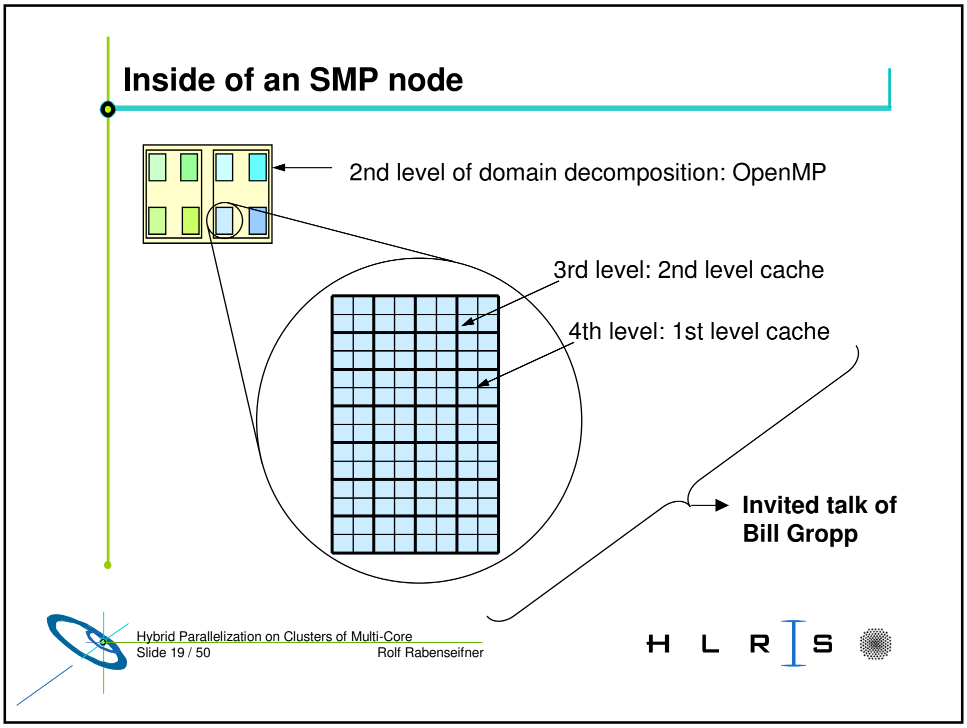
Application
MPI Level
OpenMP

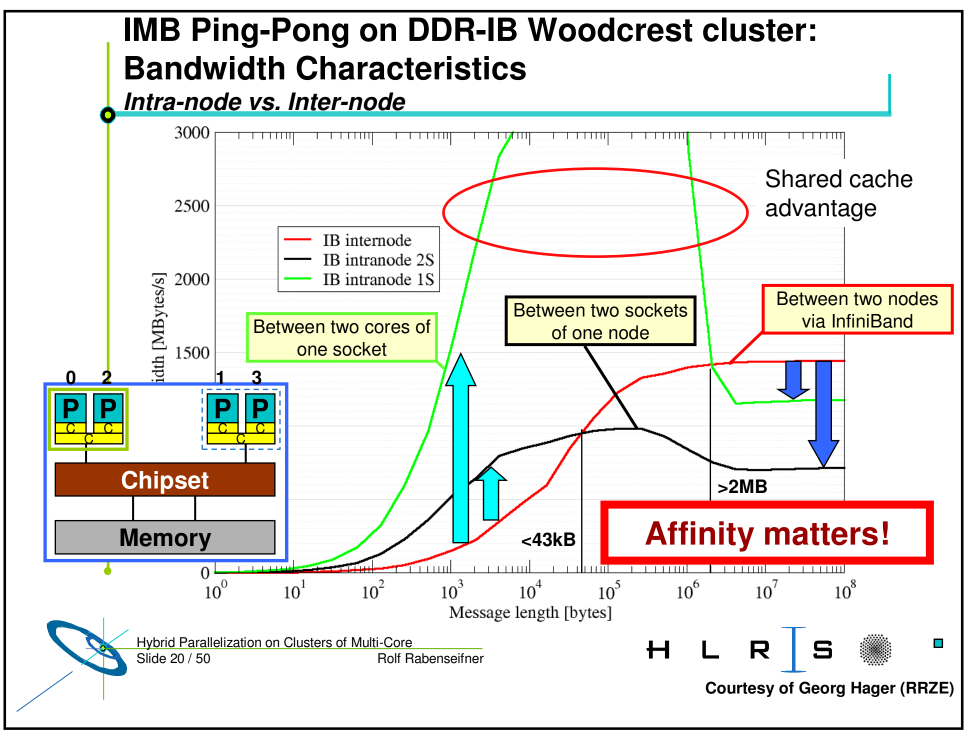


+ 3 x inter-node connections per node, but ~ 4 x more traffic
+ 2 x inter-socket connection per node

Hybrid Parallelization on
Slide 18 / 50

Affinity of cores to thread ranks !!!

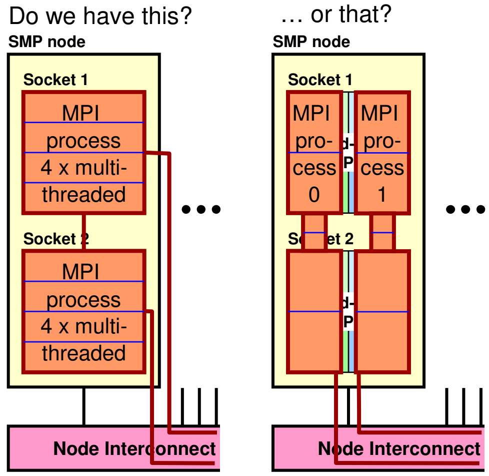

Inside of an SMP node
2nd level of domain decomposition: OpenMP
3rd level: 2nd level cache
4th level: 1st level cache
Invited talk of
Bill Gropp
Hybrid Parallelization on Clusters of Multi-Core
Slide 19 / 50
Rolf Rabenseifner
H L R S


IMB Ping-Pong on DDR-IB Woodcrest cluster:
Bandwidth Characteristics
Intra-node vs. Inter-node
Shared cache
advantage
IB internode
IB intranode 2S
IB intranode 1S
Between two cores of
one socket
Between two sockets
of one node
Between two nodes
via InfiniBand
0 2
1 3
P P
C C
C
Chipset
Memory
>2MB
<43kB
Affinity matters!
Message length [bytes]
Hybrid Parallelization on Clusters of Multi-Core
Slide 20 / 50
Rolf Rabenseifner
H L R S
Courtesy of Georg Hager (RRZE)

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., lbrun **numactl** option on Sun

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

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

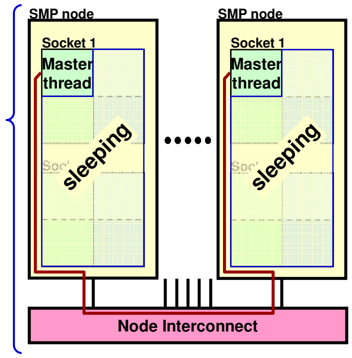
Quality aspects of the MPI library

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
- On ccNUMA SMP nodes:
  - E.g. in 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?

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 Communication and Computation

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

Subteams

- Important proposal for OpenMP 3.x or OpenMP 4.x

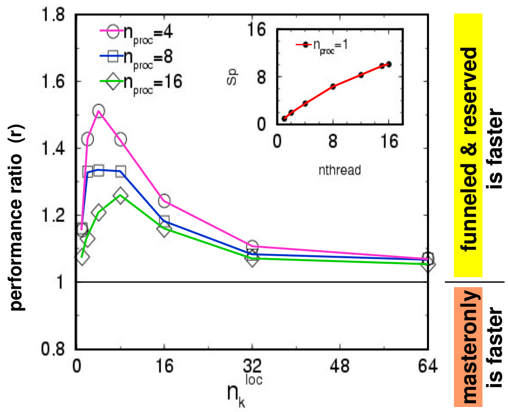
Barbara Chapman et al.: Toward Enhancing OpenMP's Work-Sharing Directives. In proceedings, W.E. Nagel et al. (Eds.): Euro-Par 2006, LNCS 4128, pp. 645-654, 2006.

```
#pragma omp parallel
{
#pragma omp single onthreads( 0 )
  {
    MPI_Send/Recv....
  }
#pragma omp for onthreads( 1 : omp_get_numthreads()-1 )
   for (........)
   { /* work without halo information */
   }  /* barrier at the end is only inside of the subteam */
  ...
#pragma omp barrier
#pragma omp for
   for (........)
   { /* work based on halo information */
   }
} /*end omp parallel */
```

→ Already seen in invited talk of Barbara Chapman

Masteronly
funneled & reserved

Experiment: Matrix-vector-multiply (MVM)



- Jacobi-Davidson-Solver 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 .

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

Aspects & Outline

- Future High Performance Computing (HPC)
  → always hierarchical hardware design
- **Mismatches and opportunities with current MPI based programming models**
  → **Some new features are needed → e.g., OpenMP subteams**
  → **Some optimizations can be done best by the application itself**
- Optimization always requires knowledge on the hardware:
  → Qualitative and quantitative information is needed
  → through a standardized interface
- The MPI-3 Forum tries to address those aspects
  → MPI-2.1 is only a starting point:
  combination of MPI-1.1 and 2.0 in one book

Opportunities of hybrid parallelization (MPI & OpenMP)

Overview

- 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"* problems
  → Significantly reduced number of MPI processes
- … **(→ slide on "Further Opportunities")**

Nested Parallelism

- Example NPB: BT-MZ (Block tridiagonal simulated CFD application)
  - Outer loop:
    - **limited number of zones → limited parallelism**
    - **zones with different workload → speedup < $\frac{\text{Sum of workload of all zones}}{\text{Max workload of one zone}}$**
  - 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

Benchmark Characteristics

- Aggregate sizes and zones:
  - Class B: 304 x 208 x 17 grid points, 64 zones
  - Class C: 480 x 320 x 28 grid points, 256 zones
  - Class D: 1632 x 1216 x 34 grid points, 1024 zones
  - Class E: 4224 x 3456 x 92 grid points, 4096 zones
- **BT-MZ:**
  Block tridiagonal simulated CFD application
  - 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
  - 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/NPS)

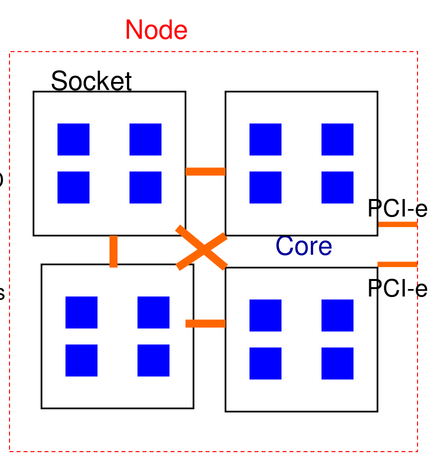
Sun Constellation Cluster Ranger (1)

- Located at the Texas Advanced Computing Center (TACC), University of Texas at Austin (http://www.tacc.utexas.edu)
- 3936 Sun Blades, 4 AMD Quad-core 64bit 2.3GHz processors per node (blade), 62976 cores total
- 123TB aggregrate memory
- Peak Performance 579 Tflops
- InfiniBand Switch interconnect
- Sun Blade x6420 Compute Node:
  - 4 Sockets per node
  - 4 cores per socket
  - HyperTransport System Bus
  - 32GB memory

Courtesy of Gabriele Jost (TACC/NPS)

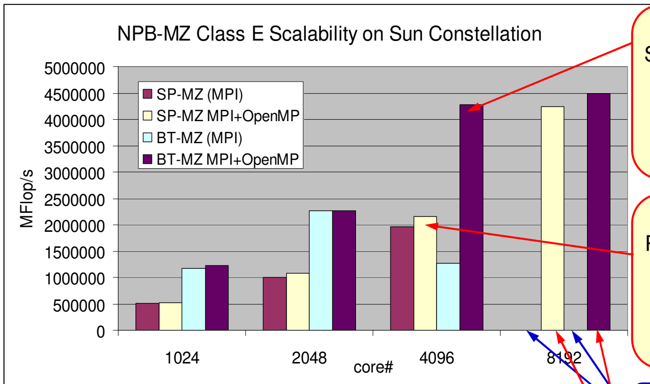
Sun Constellation Cluster Ranger (2)

- **Compilation:**
  - PGI pgf90 7.1
  - mpif90 -tp barcelona-64 -r8
- **Cache optimized benchmarks Execution:**
  - MPI MVAPICH
  - setenv OMP_NUM_THREAD NTHREAD
  - Ibrun numactl bt-mz.exe
- **numactl** controls
  - Socket affinity: select sockets to run
  - Core affinity: select cores within socket
  - Memory policy:where to allocate memory
  - http://www.halobates.de/numaapi3.pdf



Courtesy of Gabriele Jost (TACC/NPS)

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 programming 9.6% faster

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

Hybrid Parallelization on Clusters of Multi-Core
Slide 35 / 50 Rolf Rabenseifner

Courtesy of Gabriele Jost (TACC/NPS)

Next opportunity: 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
- 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**

Hybrid Parallelization on Clusters of Multi-Core
Slide 36 / 50 Rolf Rabenseifner

Memory consumption

- Shared nothing
  - Heroic theory
  - In practice: Some data is duplicated
- **MPI & OpenMP**
  With n threads per MPI process:
  - Duplicated data is reduced by factor n

Memory consumption (continued)

- Future:
  With 100+ cores per chip the memory per core is limited.
  - Data reduction through usage of shared memory may be a key issue
  - Domain decomposition on each hardware level
    - **Maximizes**
      - Data locality
      - Cache reuse
    - **Minimizes**
      - CCnuma accesses
      - Message passing
  - No halos between domains inside of SMP node
    - **Minimizes**
      - Memory consumption

How many multi-threaded MPI processes per SMP node

- SMP node = 1 Chip
  - 1 MPI process per SMP node
- SMP node is n-Chip ccNUMA node
  - With x NICs (network interface cards) per node
- How many MPI processes per SMP node are optimal?
  - somewhere between 1 and n

In other words:

- How many threads (i.e., cores) per MPI process?
  - Many threads
    → overlapping of MPI and computation may be necessary,
    → some NICs unused?
  - Too few threads
    → too much memory consumption (see previous slides)

Opportunities, if MPI speedup is limited due to *"algorithmic"* problems

- Algorithmic opportunities due to larger physical domains inside of each MPI process
  → If multigrid algorithm only inside of MPI processes
  → If separate preconditioning inside of MPI nodes and between MPI nodes
  → If MPI domain decomposition is based on physical zones

Further Opportunities

- Reduced number of MPI messages, reduced aggregated message size } compared to pure MPI
- Functional parallelism
  → e.g., I/O in an other thread
- MPI shared memory fabrics not loaded if whole SMP node is parallelized with OpenMP

Aspects & Outline

- Future High Performance Computing (HPC)
  → always hierarchical hardware design
- Mismatches and opportunities with current MPI based programming models
  → Some new features are needed → e.g., OpenMP subteams
  → Some optimizations can be done best by the application itself
- **Optimization always requires knowledge on the hardware:**
  → Qualitative and quantitative information is needed
  → through a standardized interface
- The MPI-3 Forum tries to address those aspects
  → MPI-2.1 is only a starting point: combination of MPI-1.1 and 2.0 in one book

Which hardware topology information

- Structure of the cluster and memory hierarchy
- Data exchange „speed“
  (e.g., transmission time for a given data size)

Where to get this information

- Currently, this information is accessible through different interfaces
  - E.g., numalib / numctl
  - Linux processor information
  - …
- Most information must be measured by the application

What is needed

- A standardized interface
  - Independent of the operating system

Similar to the beginning of the MPI standardization:
Where to get wall-clock-time with
- high accuracy,
- little overhead

Proposal

- Let's include in MPI-3 standardization

What about quantitative information?

- The affinity slide has clearly shown, that this is needed
- Can *"benchmark data"* be returned by a standardized library?

Yes, it can!
MPI_Wtick is such an information.
It is returned by MPI since days of MPI-1!

Let's do it in MPI-3

- Contribution by the MPI community are welcome!

Aspects & Outline

- Future High Performance Computing (HPC)
  - → always hierarchical hardware design
- Mismatches and opportunities with current MPI based programming models
  - → Some new features are needed
  - → Some optimizations can be done best by the application itself
- Optimization always requires knowledge on the hardware:
  - → Qualitative and quantitative information is needed
  - → through a standardized interface
- **The MPI-3 Forum tries to address those aspects**
  - **→ MPI-2.1 is only a starting point: combination of MPI-1.1 and 2.0 in one book**

MPI-3 Forum

- MPI-2.1 (done): Merging MPI-1.1 and MPI-2.0 to one book
- MPI-2.2: Small additions (Sep. 2009)
- MPI-3.0: Major new features (2010/2011), e.g.,
  - **Non-blocking collectives (→overlap of computation and communication)**
  - **Fault-tolerant MPI**
  - **New efficient remote memory access interface**
  - **Fortran interface with argument checking**
  - **Tools support**
  - **Hybrid MPI&OpenMP programming**
- If you have interest / ideas / …

  → **please contact one of the members of the MPI Forum**
  - Several members are here at the conference!
  - They represent
    - **Industry**
    - **Academics**
    - **Labs**

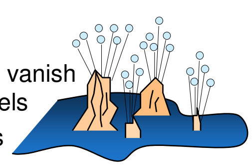
    } **MPI users and developers from USA, Europe, and Asia**

I didn't mention …

- Other parallelization models:
  - Partitioned Global Address Space (PGAS) languages
    (Unified Parallel C (UPC), Co-array Fortran (CAF), Chapel, Fortress, Titanium, and X10).
  - High Performance Fortran (HPF)

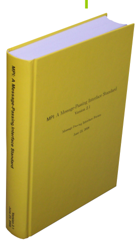
  → Many rocks in the cluster-of-SMP-sea do not vanish into thin air by using new parallelization models

  → Area of interesting research in the next years



Further information

- **SC08 Tutorial S02, Sunday, Nov. 16, 2008, Austin Texas.**
  Alice Koniges, David Eder, Bill Gropp, Ewing (Rusty) Lusk, and Rolf Rabenseifner:
  Application Supercomputing and the Many-Core Paradigm Shift.
- **SC08 Tutorial M09, Monday, Nov. 17, 2008, Austin Texas.**
  Rolf Rabenseifner, Georg Hager, Gabriele Jost, and Rainer Keller:
  Hybrid MPI and OpenMP Parallel Programming.
- **MPI-2.1** (**June 23, 2008 – finally voted at MPI Forum meeting, Sep. 4, 2008**)
  - Electronically via www.mpi-forum.org
  - As hardcover book (608 pages)
    - The book was printed by HLRS
    - As a service for the MPI community.
    - High-quality sewn binding.
    - Sold at costs – 17 Euro
    - Available via www.mpi-forum.org/docs.
    - Not via normal book stores!



Conclusions

- Future High Performance Computing (HPC)
  → always hierarchical hardware design
- Mismatches and opportunities
  with current MPI based programming models
  → Some new features are needed
  → Some optimizations can be done best by the application itself

MPI + OpenMP:
- Often hard to solve the mismatch problems
- May be a significant opportunity for performance

→(huge) amount of work

- Optimization always requires knowledge on the hardware:
  → Qualitative and quantitative information is needed
  → through a standardized interface

A new standard may assist the research community, and vice versa.

- The MPI-3 Forum tries to address those aspects
  → MPI-2.1 is only a starting point: combination of MPI-1.1 and 2.0 in one book

You may join in or you may share your ideas with the MPI Forum

This slides – via my publications list (in a few hours) at www.hlrs.de/people/rabenseifner/