

- PART 1: Introduction
- **PART 2: MPI+OpenMP**
- PART 3: PGAS Languages
- ANNEX

Programming Models and Languages for Clusters of Multi-core Nodes

Part 2: Hybrid MPI and OpenMP

Alice Koniges – NERSC, Lawrence Berkeley National Laboratory

Rolf Rabenseifner – High Performance Computing Center Stuttgart (HLRS), Germany

Gabriele Jost – Texas Advanced Computing Center, The University of Texas at Austin

*Georg Hager – Erlangen Regional Computing Center (RRZE), University of Erlangen-Nuremberg, Germany

*author only—not speaking

Tutorial at SciDAC Tutorial Day
June 19, 2009, San Diego, CA



SciDAC 2009 Tutorial © Koniges, Rabenseifner, Jost, Hager & others

<https://fs.hlr.de/projects/rabenseifner/publ/SciDAC2009-Part2-Hybrid.pdf>

Hybrid Programming – Outline

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit
- Summary

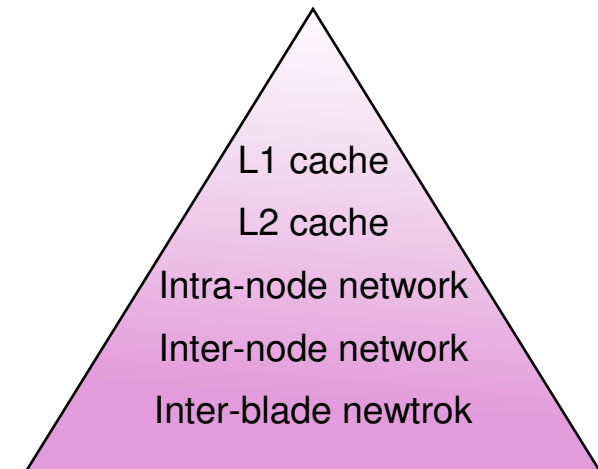
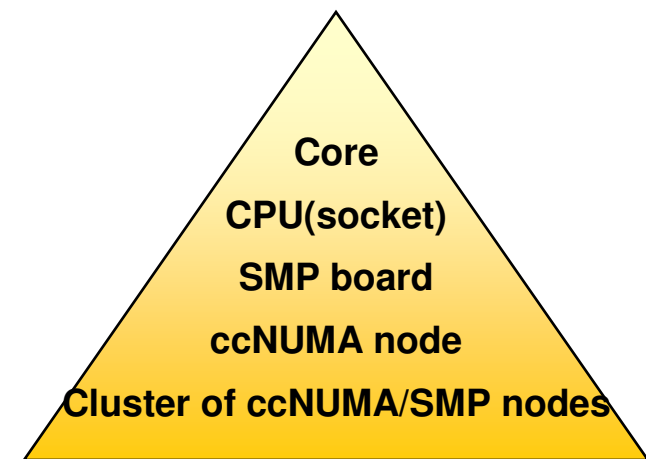
- Introduction / Motivation
- Programming Models on Clusters of SMP nodes
- Practical “How-To” on hybrid programming
- Mismatch Problems & Pitfalls
- Application Categories that Can Benefit from Hybrid Parallelization/Case Studies
- Summary on hybrid parallelization

<https://fs.hlr.de/projects/rabenseifner/publ/SciDAC2009-Part2-Hybrid.pdf>

- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit
- Summary

Goals of this part of the tutorial

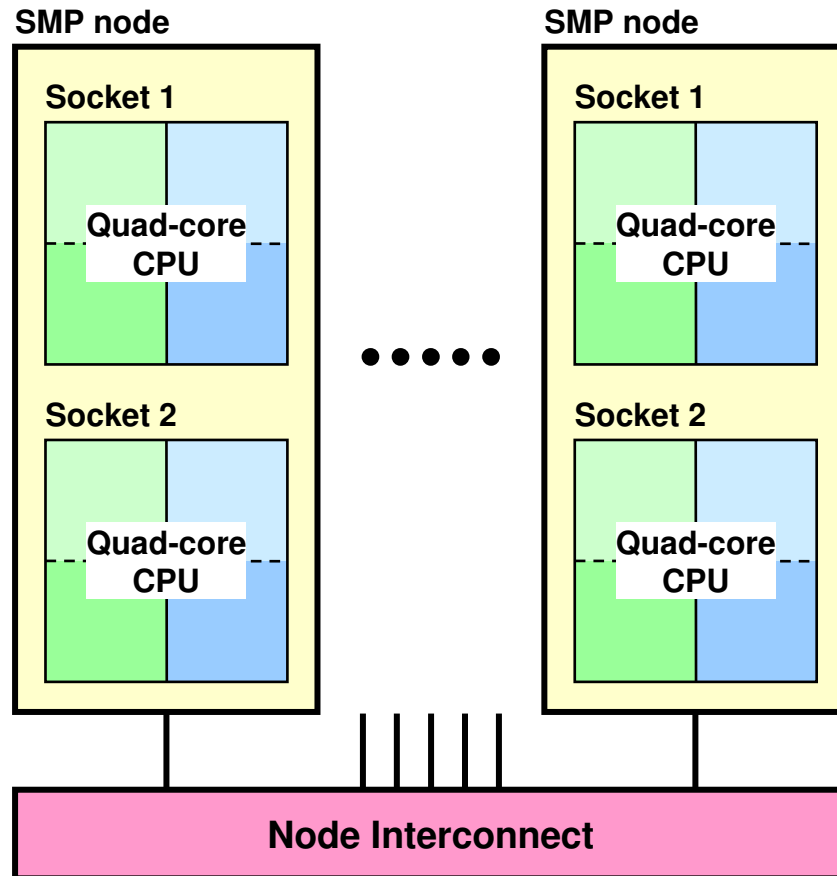
- **Effective methods for clusters of SMP node**
→ Mismatch problems & Pitfalls
- **Technical aspects of hybrid programming**
→ Programming models on clusters
→ “How-To”
- **Opportunities with hybrid programming** → Application categories that can benefit from hybrid parallelization
→ Case studies



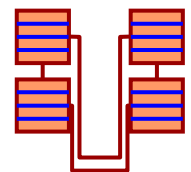
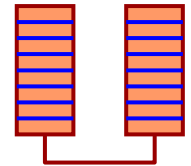
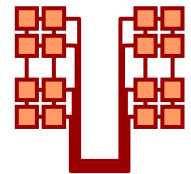
Motivation

Hybrid MPI/OpenMP programming seems natural

- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit
- Summary

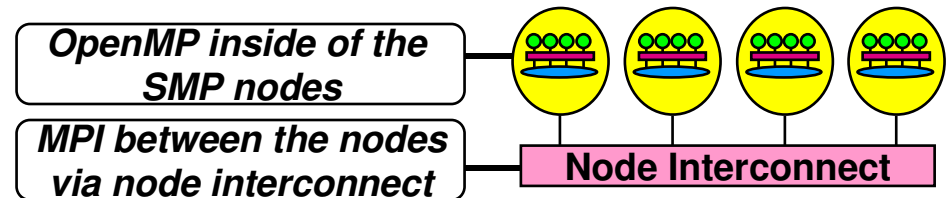


- Which programming model is fastest?
- MPI everywhere?
- Fully hybrid MPI & OpenMP?
- Something between? (Mixed model)
- Often hybrid programming **slower** than pure MPI
 - Examples, Reasons,

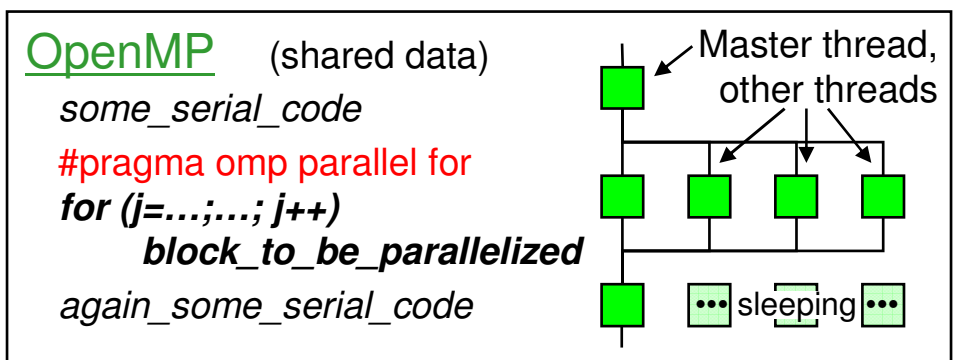
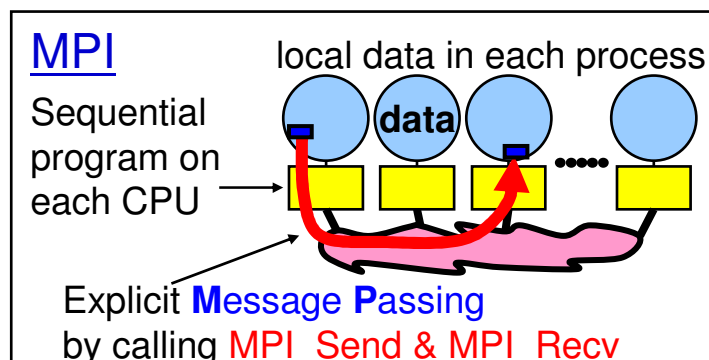


- Pure MPI (one MPI process on each CPU)
- Hybrid MPI+OpenMP

- shared memory OpenMP
- distributed memory MPI



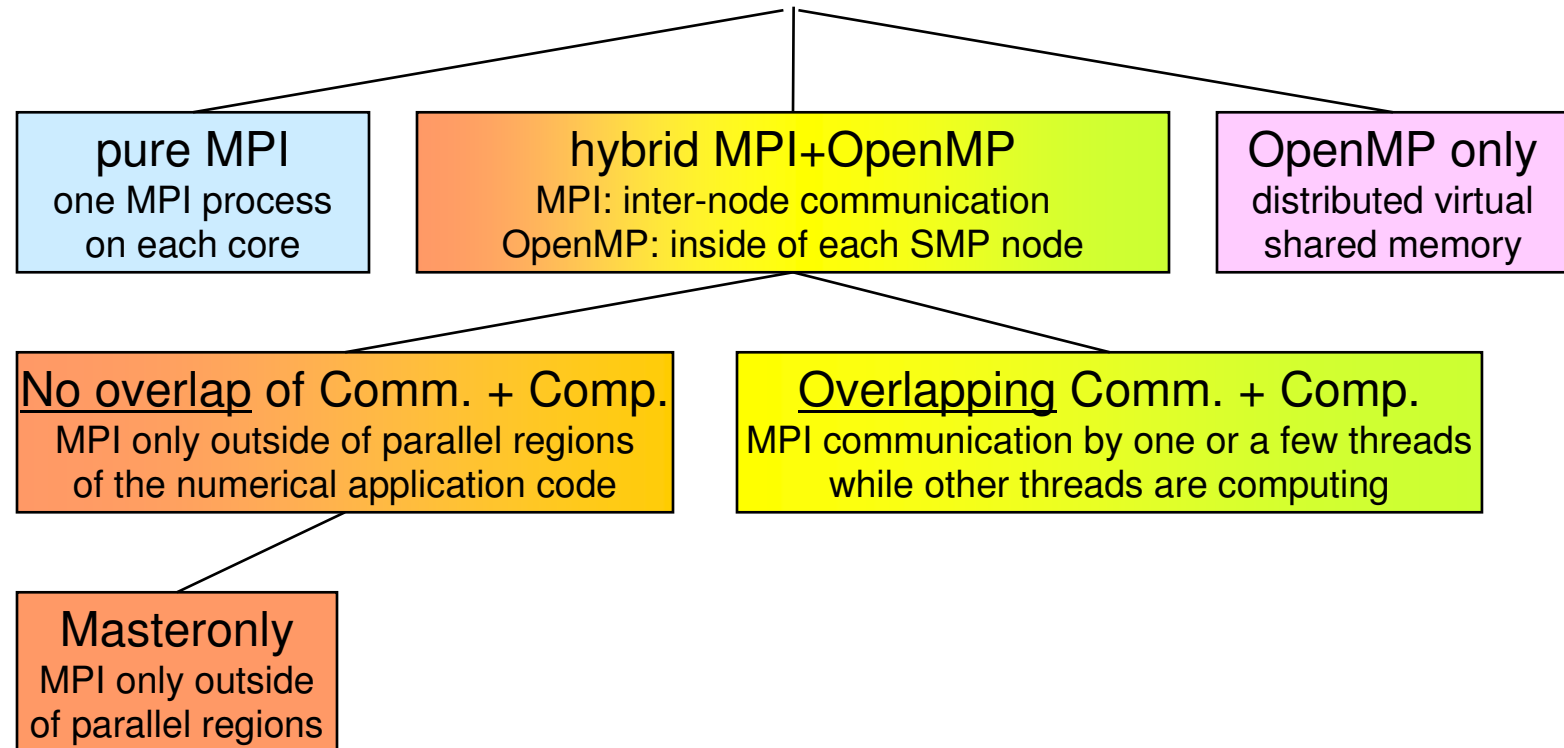
- Other: Virtual shared memory systems, PGAS, HPF, ...
- Often hybrid programming (MPI+OpenMP) slower than pure MPI
 - why?



MPI and OpenMP Programming Models

PART 2: Hybrid MPI+OpenMP

- Introduction
- **Programming Models**
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit
- Summary



Pure MPI

pure MPI
one MPI process
on each core

Advantages

- MPI library need not to support multiple threads

Major problems

- Does MPI library use 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**

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?

Comparison of MPI and OpenMP

- **MPI**
- **Memory Model**
 - Data private by default
 - Data accessed by multiple processes needs to be explicitly communicated
- **Program Execution**
 - Parallel execution from start to beginning
- **Parallelization**
 - Domain decomposition
 - Explicitly programmed by user
- **OpenMP**
- **Memory Model**
 - Data shared by default
 - Access to shared data requires synchronization
 - Private data needs to be explicitly declared
- **Program Execution**
 - Fork-Join Model
- **Parallelization**
 - Thread based
 - Incremental, typically on loop level
 - Based on compiler directives

Support of Hybrid Programming

- **MPI**

- MPI-1 no concept of threads
- MPI-2:
 - Thread support
 - MPI_Init_thread

- **OpenMP**

- None
- API only for one execution unit, which is one MPI process
- For example: No means to specify the total number of threads across several MPI processes.

```
Syntax: call MPI_Init_thread(                                irequired,      iprovided, ierr)
        int MPI_Init_thread(int *argc, char ***argv, int required, int *provided)
        int MPI::Init_thread(int& argc, char**& argv, int required)
```

Support Levels	Description
<code>MPI_THREAD_SINGLE</code>	Only one thread will execute.
<code>MPI_THREAD_FUNNELED</code>	Process may be multi-threaded, but only main thread will make MPI calls (calls are "funneled" to main thread). Default
<code>MPI_THREAD_SERIALIZE</code>	Process may be multi-threaded, any thread can make MPI calls, but threads cannot execute MPI calls concurrently (all MPI calls must be " serialized ").
<code>MPI_THREAD_MULTIPLE</code>	Multiple threads may call MPI, no restrictions.

If supported, the call will return provided = required.
Otherwise, the highest level of support will be provided.

Overlapping Communication and Work

PART 2: Hybrid MPI+OpenMP

- Introduction
- **Programming Models**
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit
- Summary

- One core can saturate the PCI-e \leftrightarrow network bus. Why use all to communicate?
- **Communicate with one** or several cores.
- **Work with others** during communication.
- Need at least **MPI_THREAD_FUNNELED** support.
- Can be difficult to manage and load balance!

Overlapping Communication and Work

PART 2: Hybrid MPI+OpenMP

- Introduction
- **Programming Models**
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit
- Summary

Fortran

```
include 'mpi.h'
program hybover

call mpi_init_thread(MPI_THREAD_FUNNELED,...)

!$OMP parallel

  if (ithread .eq. 0) then
    call MPI_<whatever>(...,ierr)
  else
    <work>
  endif

!$OMP end parallel
end
```

C

```
#include <mpi.h>
int main(int argc, char **argv){
  int rank, size, ierr, i;

  ierr= MPI_Init_thread(...)

  #pragma omp parallel
  {
    if (thread == 0){
      ierr=MPI_<Whatever>(...)
    }
    if(thread != 0){
      work
    }
  }
}
```

Thread-rank Communication

```

:
call mpi_init_thread( MPI_THREAD_MULTIPLE, iprovided,ierr)
call mpi_comm_rank(MPI_COMM_WORLD, irank, ierr)
call mpi_comm_size(MPI_COMM_WORLD,nranks, ierr)
:
!$OMP parallel private(i, ithread, nthreads)
:
  nthreads=OMP_GET_NUM_THREADS()
  ithread =OMP_GET_THREAD_NUM()
  call pwork(ithread, irank, nthreads, nranks...)
  if(irank == 0) then
    call mpi_send(ithread,1,MPI_INTEGER, 1, thread,MPI_COMM_WORLD, ierr)
  else
    call mpi_recv(      j,1,MPI_INTEGER, 0, thread,MPI_COMM_WORLD, istatus,ierr)
    print*, "Yep, this is ",irank," thread ", ithread," I received from ", j
  endif

!$OMP END PARALLEL
end

```

Communicate between ranks.

Threads use tags to differentiate.

Hybrid Programming – Outline

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

- Introduction / Motivation
 - Programming models on clusters of SMP nodes
- Practical “How-To” on hybrid programming
 - Mismatch Problems & Pitfalls
- Application categories that can benefit from hybrid parallelization/Case Studies
 - Summary on hybrid parallelization

Compile and Linking Hybrid Codes

- **Use MPI include files and link with MPI library:**
 - Usually achieved by using MPI compiler script
- **Use OpenMP compiler switch for **compiling** AND **linking****
- **Examples:**
 - PGI (Portlan Group compiler)
 - `mpif90 -fast -tp barcelona-64 -mp (AMD Opteron)`
 - Pathscale for AMD Opteron:
 - `mpif90 -Ofast -openmp`
 - Cray (based on pgf90)
 - `ftn -fast -tp barcelona-64 -mp`
 - IBM Power 6:
 - `mpxlf_r -O4 -qarch=pwr6 -qtune=pwr6 -qsmp=omp`
 - Intel
 - `mpif90 -openmp`

Running Hybrid Codes

- **Running the code**

- Highly non-portable! Consult system docs
- Things to consider:
 - Is environment available for MPI Processes:
 - E.g.: `mpirun -np 4 OMP_NUM_THREADS=4 ... a.out`
instead of your binary alone may be necessary
 - How many MPI Processes per node?
 - How many threads per MPI Process?
 - Which cores are used for MPI?
 - Which cores are used for threads?
 - Where is the memory allocated?

Running the code *efficiently*?

- **Memory access not uniform on node level**
 - NUMA (AMD Opteron, SGI Altix, IBM Power6 (p575), Sun Blades, Intel Nehalem)
- **Multi-core, multi-socket**
 - Shared vs. separate caches
 - Multi-chip vs. single-chip
 - Separate/shared buses
- **Communication bandwidth not uniform between cores, sockets, nodes**

Running code on *Hierarchical Systems*

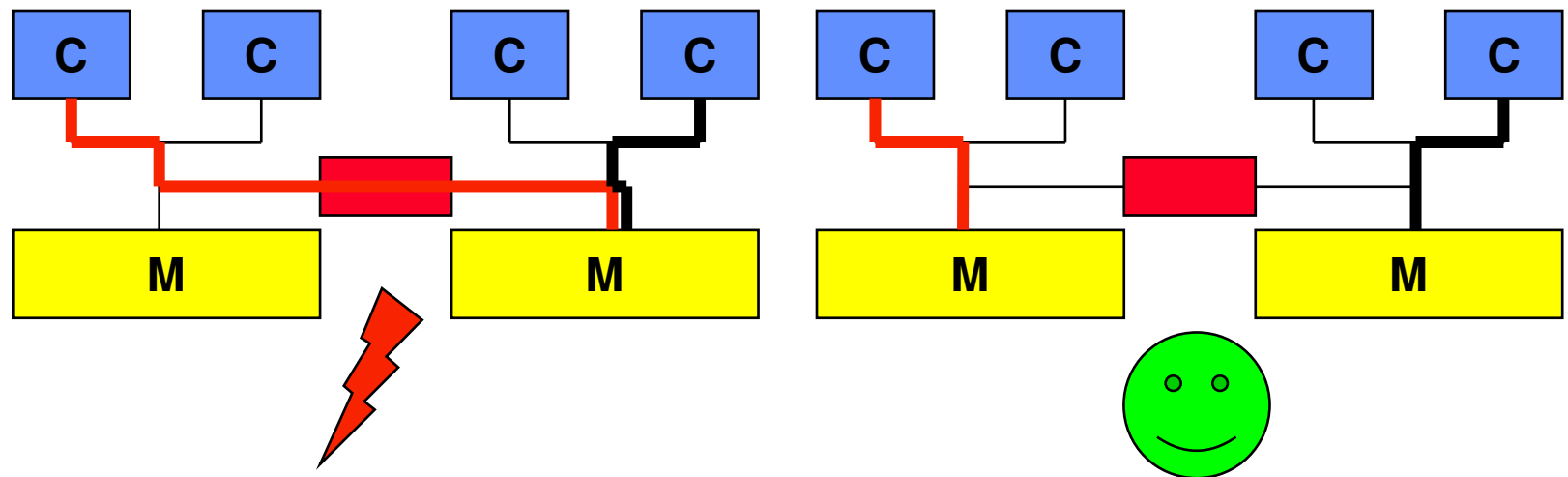
PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

- **Multi-core:**
 - NUMA locality effects
 - Shared vs. separate caches
 - Separate/shared buses
 - Placement of MPI buffers
- **Multi-socket effects/multi-node/multi-rack**
 - Bandwidth bottlenecks
 - Intra-node MPI performance
 - Core ↔ core; socket ↔ socket
 - Inter-node MPI performance
 - node ↔ node within rack; node ↔ node between racks
- **OpenMP performance depends on placement of threads**

A short introduction to ccNUMA

- ccNUMA:
 - whole memory is **transparently accessible** by all processors
 - but **physically distributed**
 - with **varying bandwidth and latency**
 - and **potential contention** (shared memory paths)



ccNUMA Memory Locality Problems

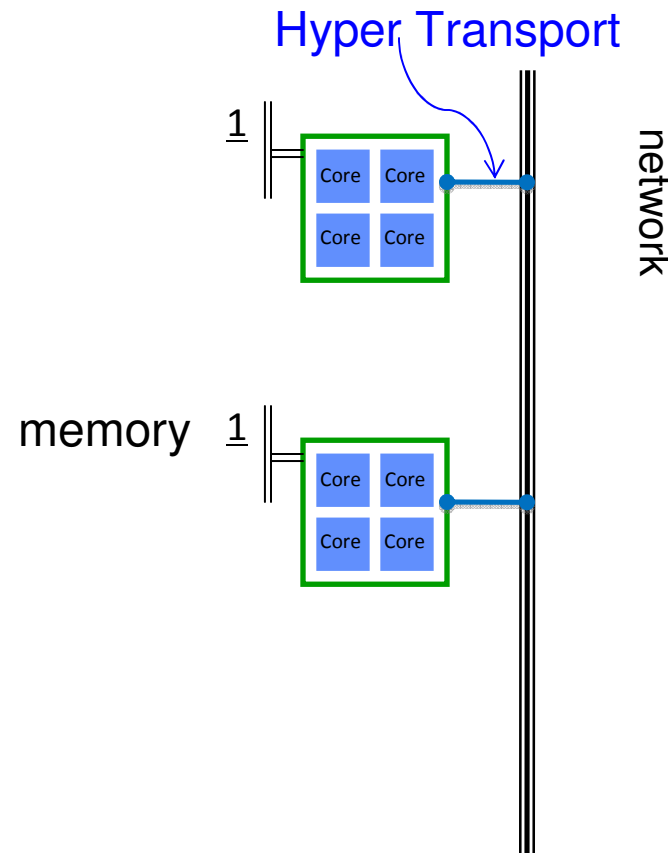
- **Locality of reference** is key to scalable performance on ccNUMA
 - Less of a problem with pure MPI, but see below
- **What factors can destroy locality?**
- **MPI programming:**
 - processes lose their association with the CPU the mapping took place on originally
 - OS kernel tries to maintain strong affinity, but sometimes fails
- **Shared Memory Programming (OpenMP, hybrid):**
 - threads losing association with the CPU the mapping took place on originally
 - improper initialization of distributed data
 - Lots of extra threads are running on a node, especially for hybrid
- **All cases:**
 - Other agents (e.g., OS kernel) may fill memory with data that prevents optimal placement of user data

Example 1: Running Hybrid on Cray XT4

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
 - **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

- **Shared Memory:**
 - Cache-coherent 4-way Node
- **Distributed memory:**
 - Network of nodes
 - Core-to-Core
 - Node-to-Node



Process and Thread Placement on Cray XT4

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

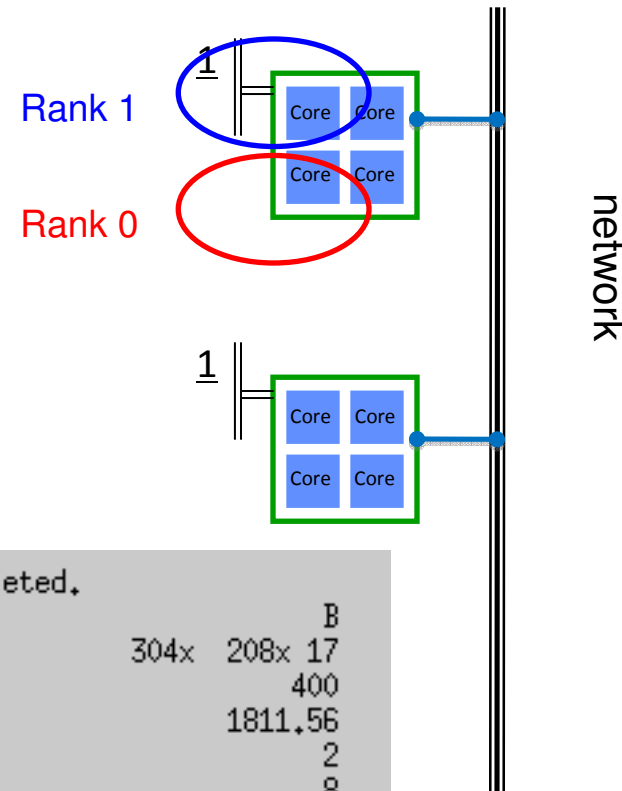
```
export OMP_NUM_THREADS=4
export MPICH_RANK_REORDER_DISPLAY=1
```

```
aprun -n 2 sp-mz.B.2
```

1 node, 4 cores, 8 threads

```
[PE_0]: rank 0 is on nid01759;
```

```
[PE_0]: rank 1 is on nid01759;
```



```
SP-MZ Benchmark Completed.
Class          = B
Size           = 304x 208x 17
Iterations      = 400
Time in seconds = 1811.56
Total processes = 2
Total threads   = 8
Mop/s total     = 167.45
Mop/s/thread    = 20.93
Operation type  = floating point
Verification    = SUCCESSFUL
Version         = 3.3
Compile date    = 28 May 2009
```

Process and Thread Placement on Cray XT4

PART 2: Hybrid MPI+OpenMP

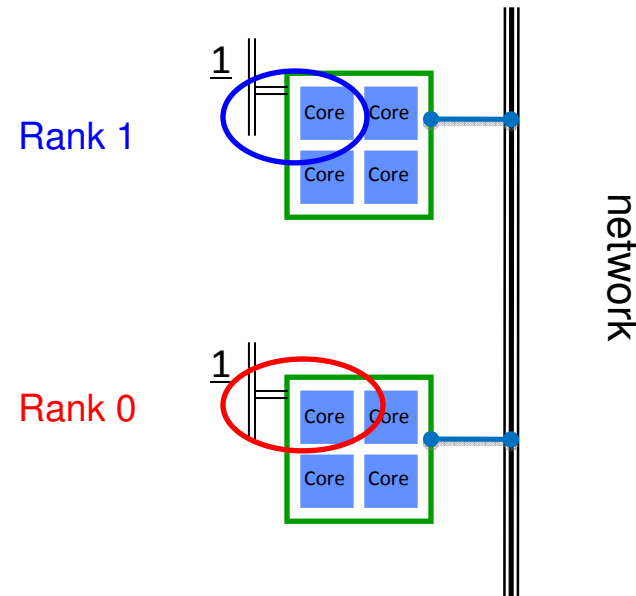
- Introduction
- Programming Models
- **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

```
export OMP_NUM_THREADS=4
export MPICH_RANK_REORDER_DISPLAY=1
```

```
aprun -n 2 -N 1 sp-mz.B.2
```

2 nodes, 8 cores, 8 threads

```
[PE_0]: rank 0 is on nid01759;
[PE_0]: rank 1 is on nid01882;
```



```
SP-MZ Benchmark Completed.
Class           = B
Size            = 304x 208x 17
Iterations      = 400
Time in seconds = 47.20
Total processes = 2
Total threads   = 8
Mop/s total     = 6427.18
Mop/s/thread    = 803.40
Operation type  = floating point
Verification    = SUCCESSFUL
Version         = 3.3
Compile date    = 28 May 2009
```


Example Batch Script Cray XT4

Cray XT4 at ERDC:

- 1 quad-core AMD Opteron per node
- `ftn -fastsse -tp barcelona-64 -mp -o bt-mz.128`

```
#!/bin/csh
#PBS -q standard
#PBS -l mppwidth=512
#PBS -l walltime=00:30:00
module load xt-mpt
cd $PBS_O_WORKDIR
setenv OMP_NUM_THREADS 4
aprun -n 128 -N 1 -d 4 ./bt-mz.128
setenv OMP_NUM_THREADS 2
aprun -n 256 -N 2 -d 2 ./bt-mz.256
```

Maximum of 4 threads
per MPI process on XT4

4 threads per MPI Proc

Number of MPI Procs per Node:
1 Proc per node allows for 4 threads per Proc

2 MPI Procs per node, 2
threads per MPI Proc

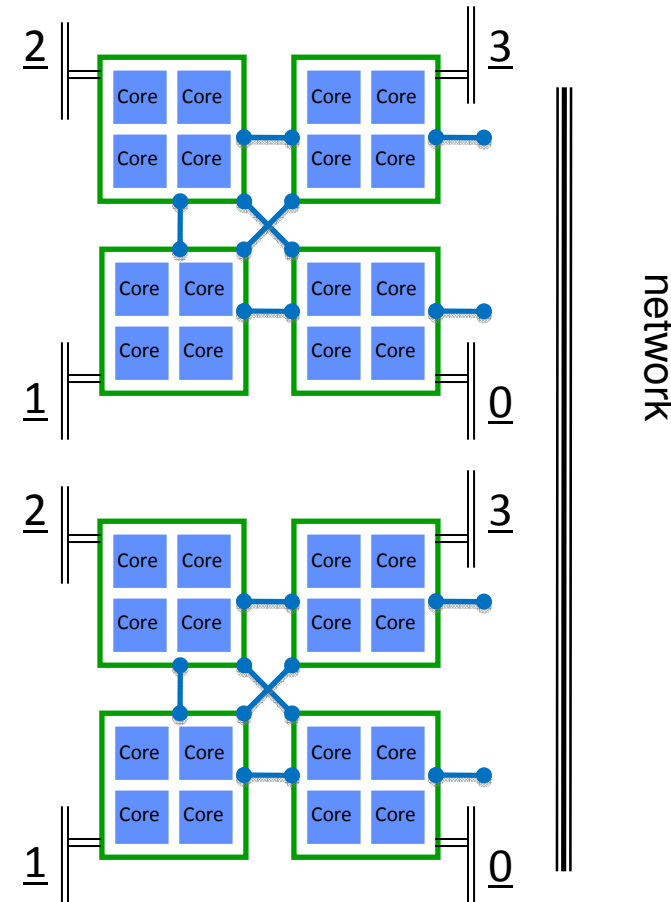
Hybrid Pa

Example 2: Running hybrid on Sun Constellation Cluster Ranger

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
 - **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

- **Highly hierarchical**
- **Shared Memory:**
 - Cache-coherent, Non-uniform memory access (ccNUMA) 16-way Node (Blade)
- **Distributed memory:**
 - Network of ccNUMA blades
 - Core-to-Core
 - Socket-to-Socket
 - Blade-to-Blade
 - Chassis-to-Chassis

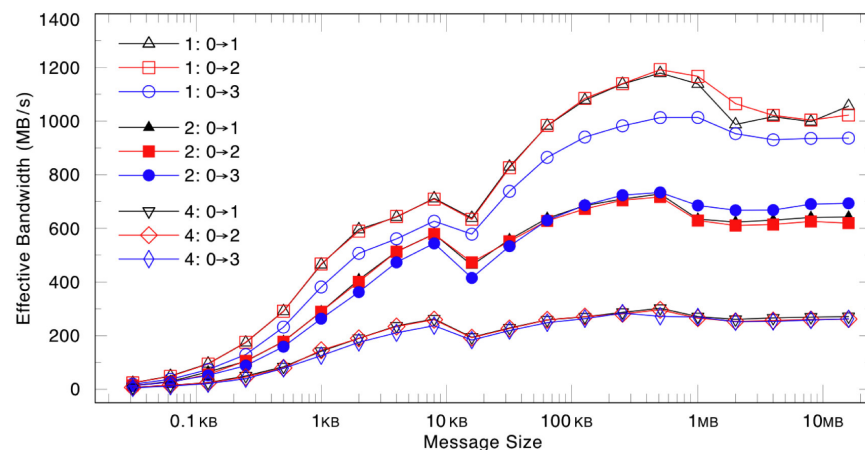


Ranger Network Bandwidth

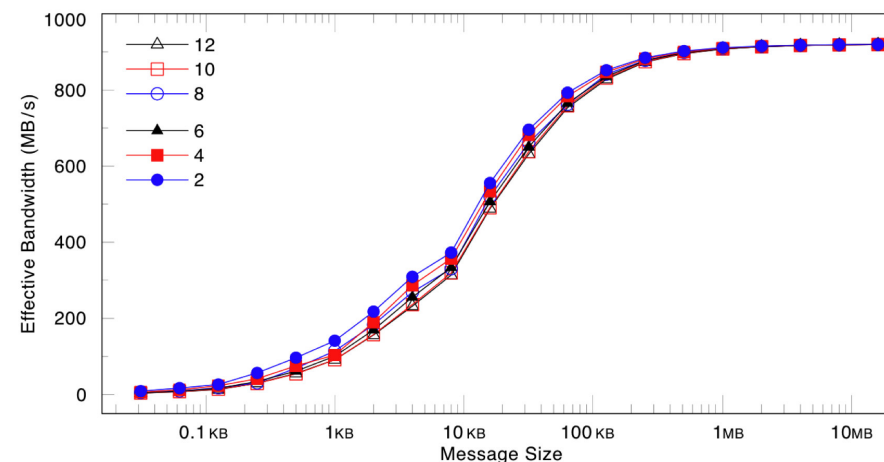
PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

On-Node Communication Scaling (between 2 Sockets)



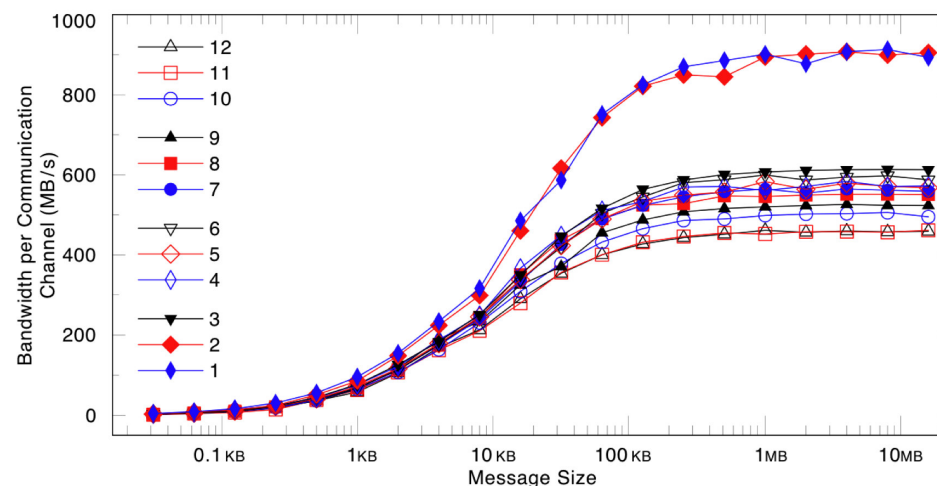
On NEM Node-2-Node Communication Scaling



MPI ping-pong micro benchmark results

“Exploiting Multi-Level Parallelism on the Sun Constellation System”.
L. Koesterke, et. al., TACC,
TeraGrid08 Paper

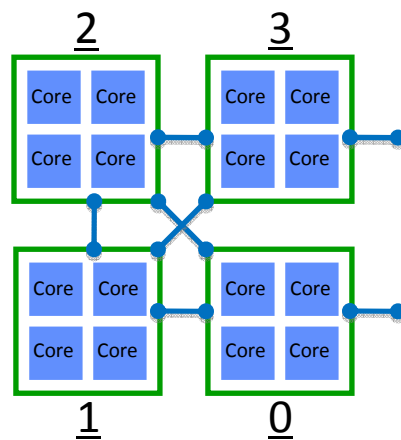
NEM to NEM Scaling Performance



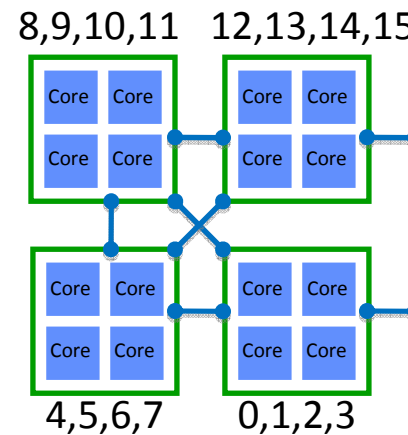
NUMA Control: Process Placement

- Affinity and Policy can be changed externally through **numactl** at the socket and core level.

Command: `numactl <options> ./a.out`



Socket References

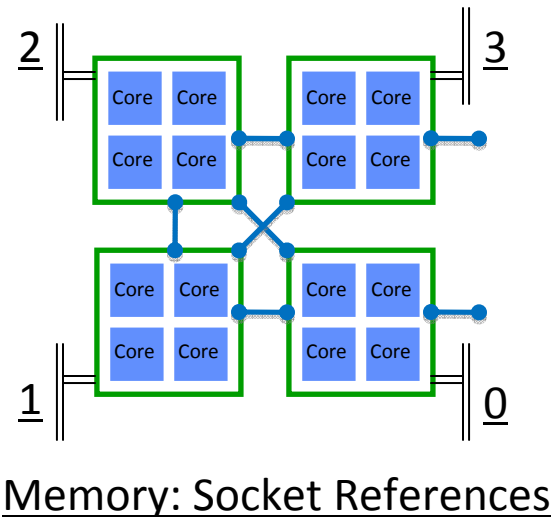


Core References

NUMA Operations: Memory Placement

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
 - **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary



- **Memory allocation:**
- **MPI – local allocation is best**
- **OpenMP**
 - Interleave best for large, completely shared arrays that are randomly accessed by different threads
 - local best for private arrays
- **Once allocated, a memory structure's is fixed**

- Introduction
- Programming Models
- **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

NUMA Operations (cont. 3)

	cmd	option	arguments	description
Socket Affinity	numactl	-N	{0,1,2,3}	Only execute process on cores of this (these) socket(s).
Memory Policy	numactl	-l	{no argument}	Allocate on current socket.
Memory Policy	numactl	-i	{0,1,2,3}	Allocate round robin (interleave) on these sockets.
Memory Policy	numactl	--preferred=	{0,1,2,3} select only one	Allocate on this socket; fallback to any other if full .
Memory Policy	numactl	-m	{0,1,2,3}	Only allocate on this (these) socket(s).
Core Affinity	numactl	-C	{0,1,2,3, 4,5,6,7, 8,9,10,11, 12,13,14,15}	Only execute process on this (these) Core(s).

Hybrid Batch Script

4 tasks, 4 threads/task

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- **How-To on hybrid prog.**
- Mismatch Problems
- Application ... can benefit
- Summary

<p>job script (Bourne shell)</p> <pre>... #!/-pe 4way 32 ... export OMP_NUM_THREADS=4 ibrun numa.sh</pre>	<p>job script (C shell)</p> <pre>... #!/-pe 4way 32 ... setenv OMP_NUM_THREADS 4 ibrun numa.csh</pre>
<p>numa.sh</p> <pre>#!/bin/bash export MV2_USE_AFFINITY=0 export MV2_ENABLE_AFFINITY=0 export VIADEV_USE_AFFINITY=0 #TasksPerNode TPN=`echo \$PE sed 's/way//'` [! \$TPN] && echo TPN NOT defined! [! \$TPN] && exit 1 socket=\$((\$PMI_RANK % \$TPN)) numactl -N \$socket -m \$socket ./a.out</pre>	<p>numa.csh</p> <pre>#!/bin/tcsh setenv MV2_USE_AFFINITY 0 setenv MV2_ENABLE_AFFINITY 0 setenv VIADEV_USE_AFFINITY 0 #TasksPerNode set TPN = `echo \$PE sed 's/way//'` if(! \${%TPN}) echo TPN NOT defined! if(! \${%TPN}) exit 0 @ socket = \$PMI_RANK % \$TPN numactl -N \$socket -m \$socket ./a.out</pre>

for mvapich2

The Topology Problem with pure MPI

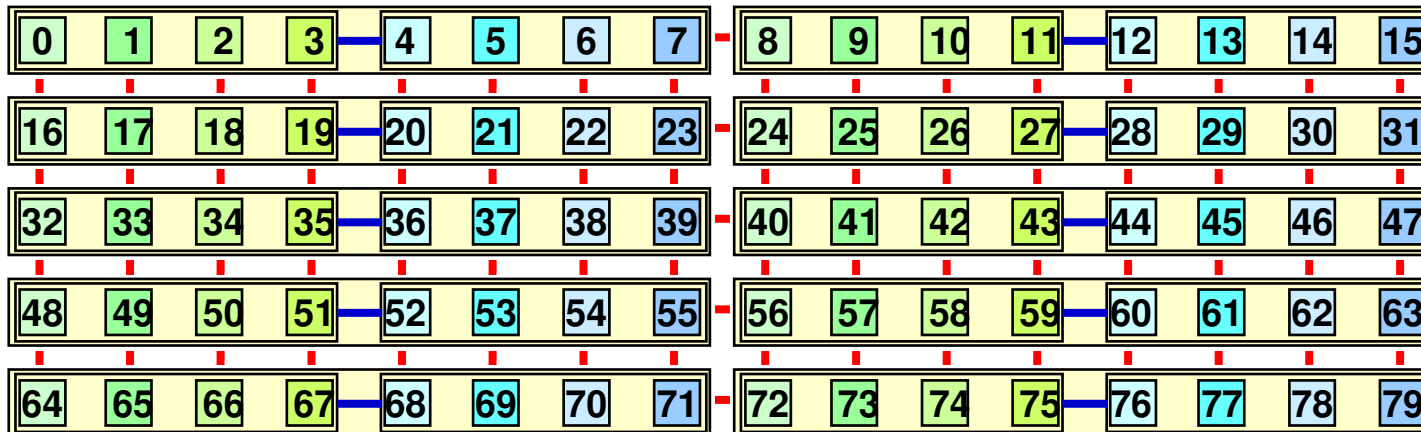
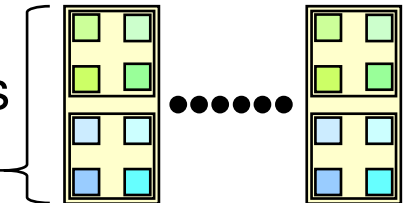
one MPI process
on each core

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- **Mismatch Problems**
- Application ... can benefit
- Summary

Application example on 80 cores:

- Cartesian application with $5 \times 16 = 80$ 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

The Topology Problem with pure MPI

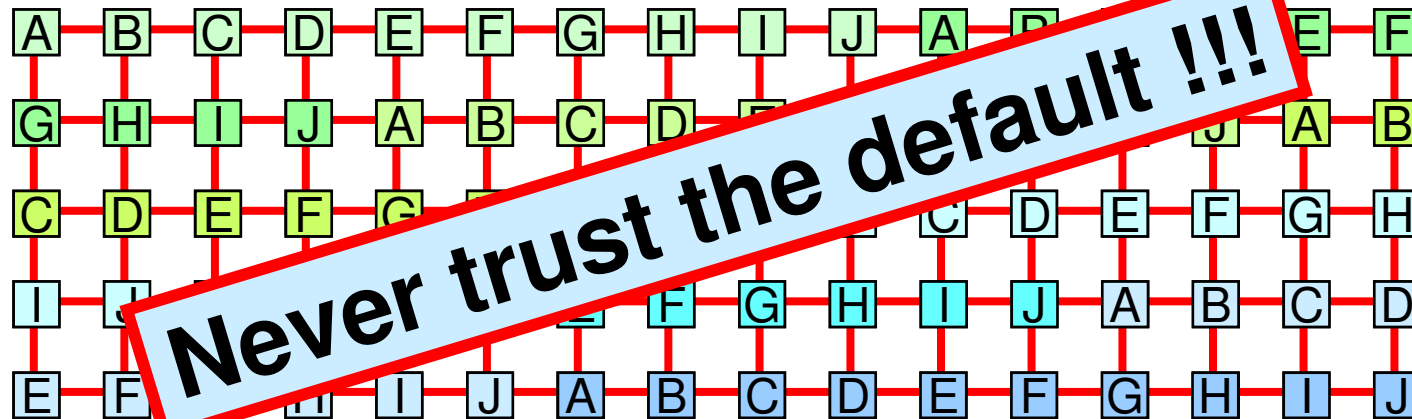
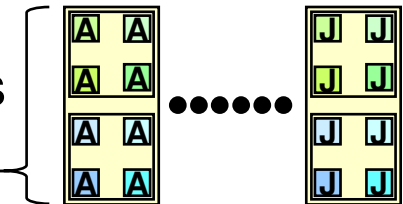
one MPI process
on each core

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- **Mismatch Problems**
- Application ... can benefit
- Summary

Application example on 80 cores:

- Cartesian application with $5 \times 16 = 80$ 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



The Topology Problem with pure MPI

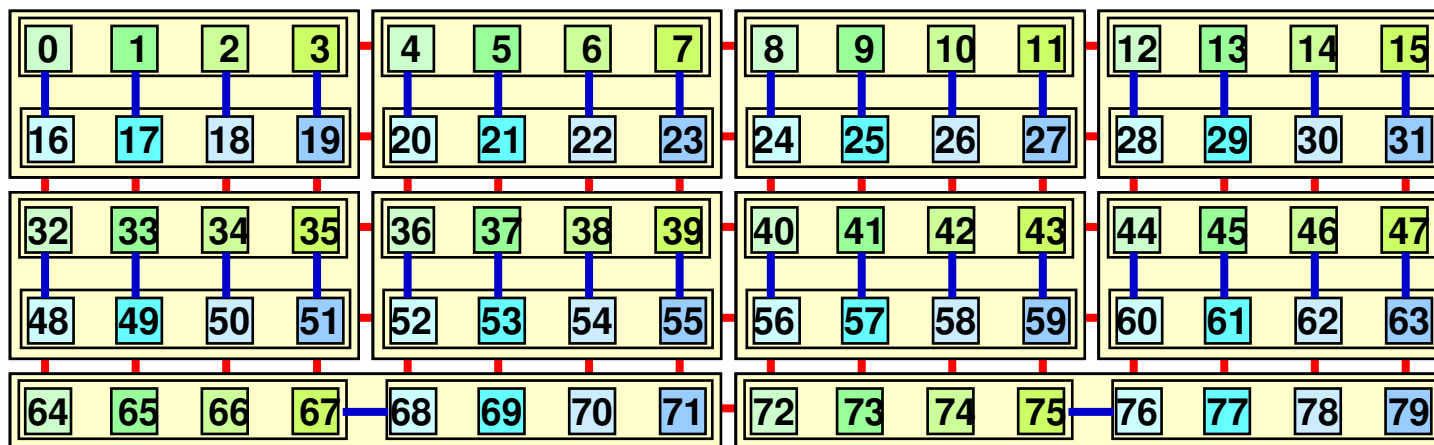
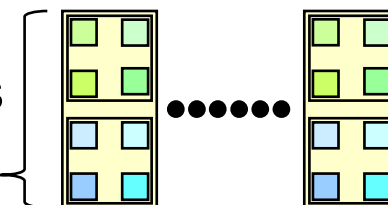
one MPI process
on each core

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- **Mismatch Problems**
- Application ... can benefit
- Summary

Application example on 80 cores:

- Cartesian application with $5 \times 16 = 80$ 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

The Topology Problem with pure MPI

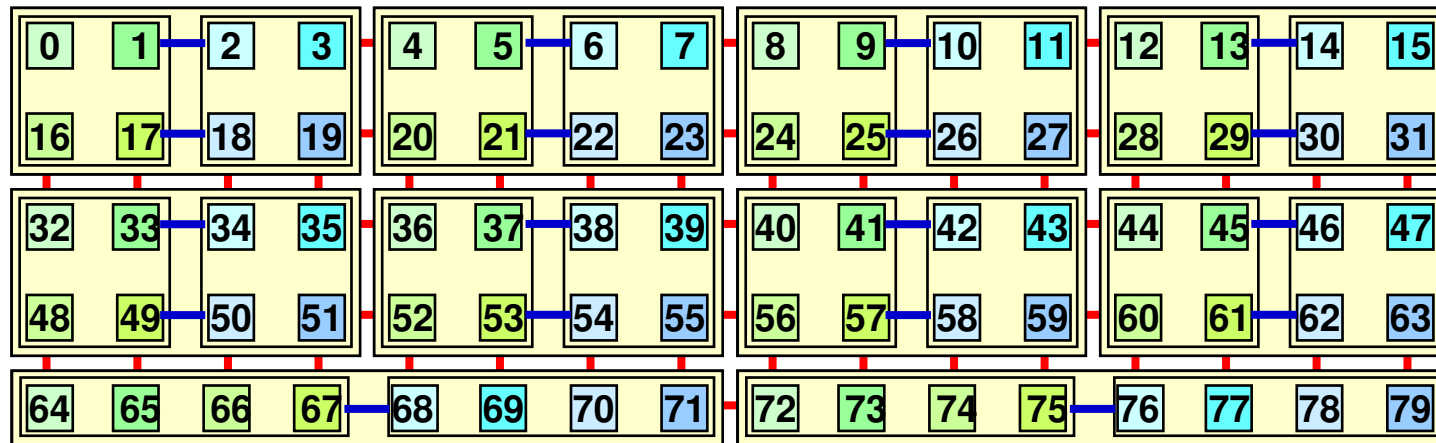
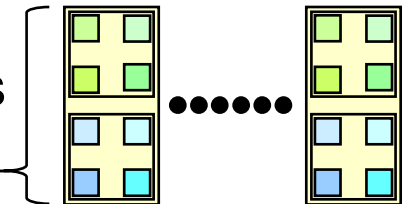
one MPI process
on each core

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- **Mismatch Problems**
- Application ... can benefit
- Summary

Application example on 80 cores:

- Cartesian application with $5 \times 16 = 80$ 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

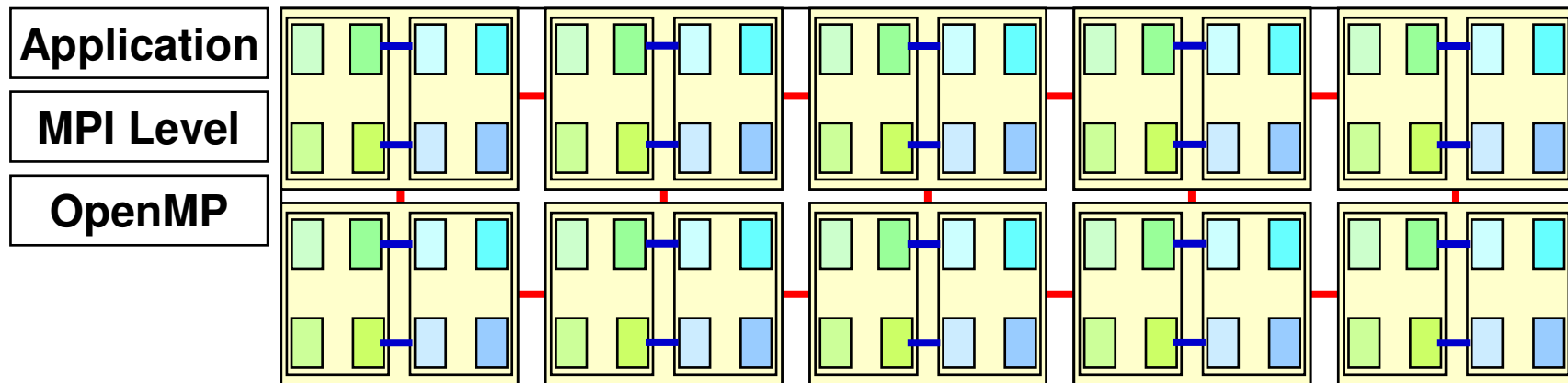
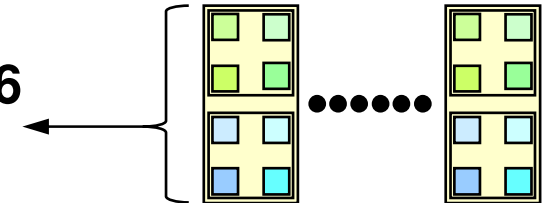
Good affinity of cores to thread ranks

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



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

+ 2 x inter-socket connection per node

Affinity of cores to thread ranks !!!

skipped

Intra-node MPI characteristics: IMB Ping-Pong benchmark

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- **Mismatch Problems**
- Application ... can benefit
- Summary

- Code (to be run on 2 processors):

```
wc = MPI_WTIME()

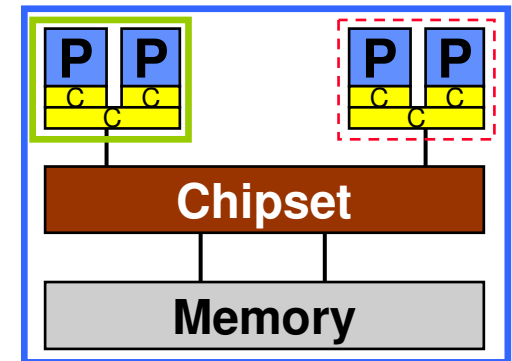
do i=1,NREPEAT

  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
```



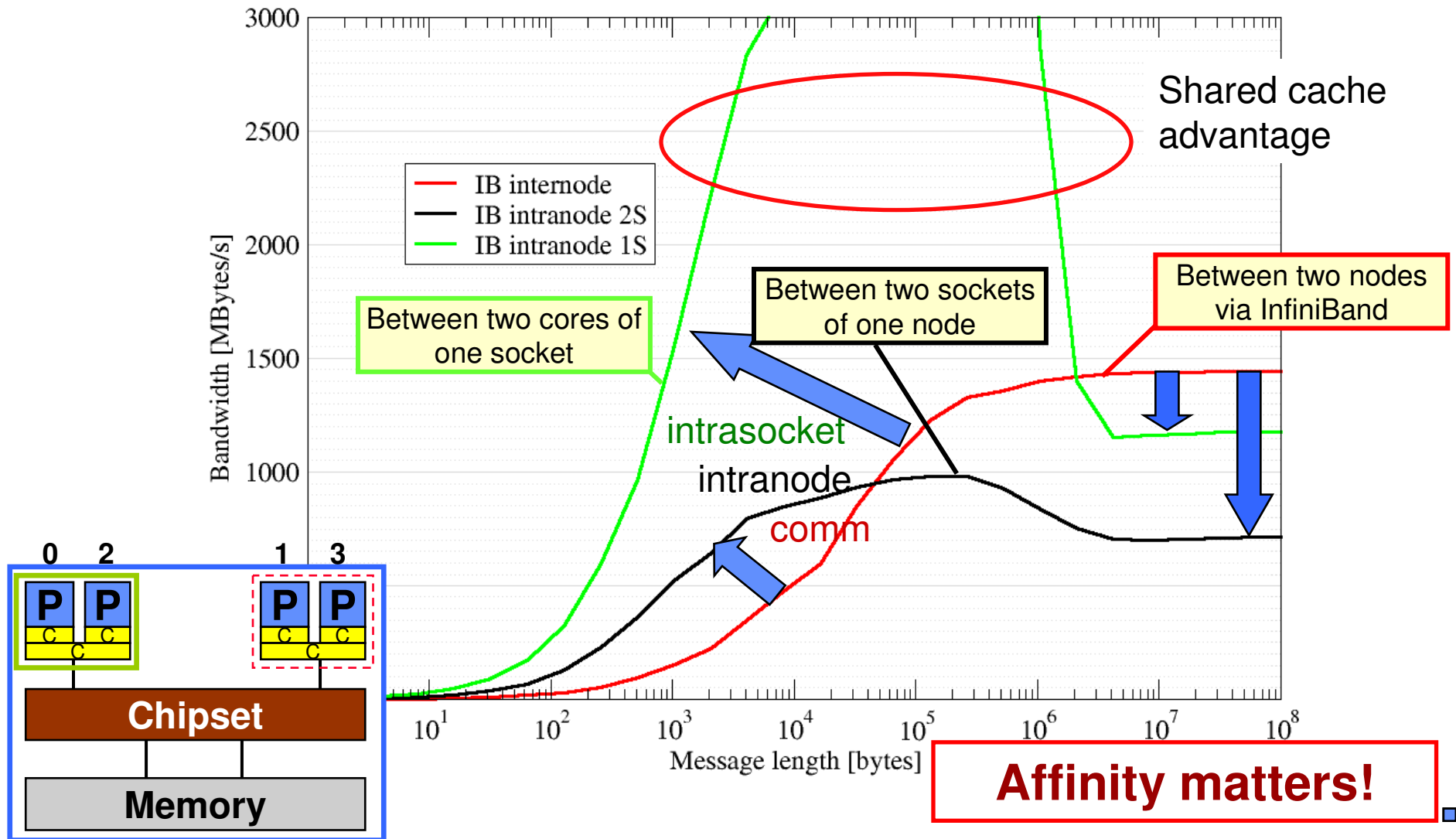
- Intranode (1S): `mpirun -np 2 -pin "1 3" ./a.out`
- Intranode (2S): `mpirun -np 2 -pin "2 3" ./a.out`
- Internode: `mpirun -np 2 -pernode ./a.out`

IMB Ping-Pong on DDR-IB Woodcrest cluster: Bandwidth Characteristics

Intra-Socket vs. Intra-node vs. Inter-node

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- **Mismatch Problems**
- Application ... can benefit
- Summary



OpenMP: Additional Overhead & Pitfalls

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- **Mismatch Problems**
- Application ... can benefit
- Summary

- **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?**

Hybrid Programming – Outline

PART 2: Hybrid MPI+OpenMP

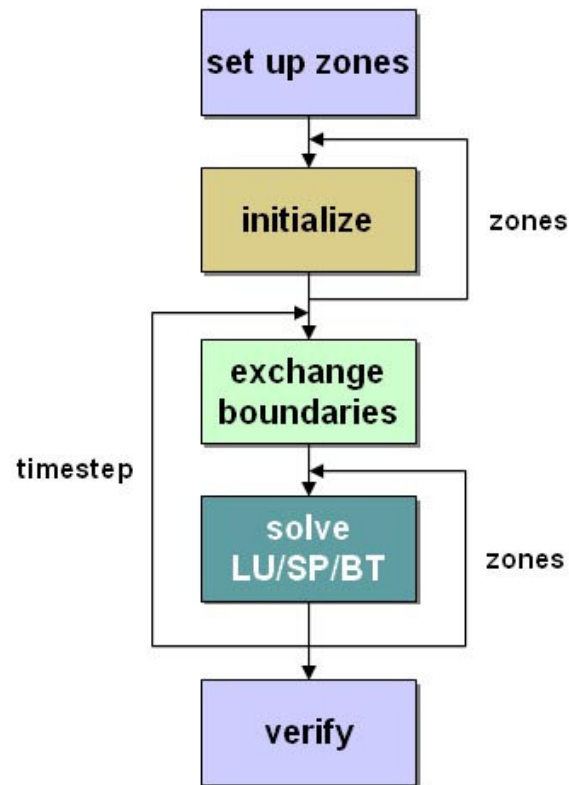
- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- **Application ... can benefit**
- Summary

- Introduction / Motivation
- Programming Models on Clusters of SMP nodes
- Practical “How-To” on hybrid programming
- Mismatch Problems & Pitfalls
- Application Categories that Can Benefit from Hybrid Parallelization/Case Studies
- Summary on hybrid parallelization

The Multi-Zone NAS Parallel Benchmarks

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- **Application ... can benefit**
- Summary



	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

Benchmark Characteristics

- **Aggregate sizes:**

- Class C: 480 x 320 x 28 grid points
- Class D: 1632 x 1216 x 34 grid points
- Class E: 4224 x 3456 x 92 grid points

Expectations:

- **BT-MZ: (Block-tridiagonal Solver)**

- #Zones: 256 (C), 1024 (D), 4096 (E)
- Size of the zones varies widely:
 - large/small about 20
 - requires multi-level parallelism to achieve a good load-balance

Pure MPI: Load-balancing problems!

Good candidate for MPI+OpenMP

- **LU-MZ: (Lower-Upper Symmetric Gauss Seidel Solver)**

- #Zones: 16 (C, D, and E)
- Size of the zones identical:
 - no load-balancing required
 - limited parallelism on outer level

Limited MPI Parallelism:
→ MPI+OpenMP increases Parallelism

LU not used in this study because of small number of cores on the systems

- **SP-MZ: (Scalar-Pentadiagonal Solver)**

- #Zones: 256 (C), 1024 (D), 4096 (E)
- Size of zones identical
 - no load-balancing required

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

BT-MZ based on MPI/OpenMP

PART 2: Hybrid MPI+OpenMP

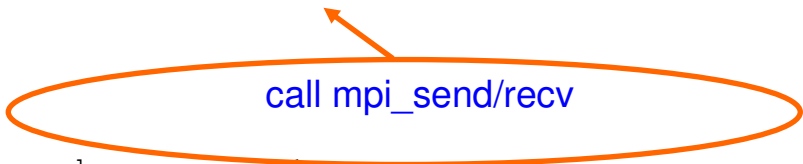
- ...
- Mismatch Problems
- **Application ... can benefit**
- Summary

Coarse-grain MPI Parallelism

```
call omp_set_numthreads (weight)
do step = 1, itmax
  call exch_qbc(u, qbc, nx,...)

  call mpi_send/recv

  do zone = 1, num_zones
    if (iam .eq.pzone_id(zone))
then
      call comp_rhs(u,rsd,...)
      call x_solve (u, rhs,...)
      call y_solve (u, rhs,...)
      call z_solve (u, rhs,...)
      call add (u, rhs,...)
    end if
  end do
end do
...
```

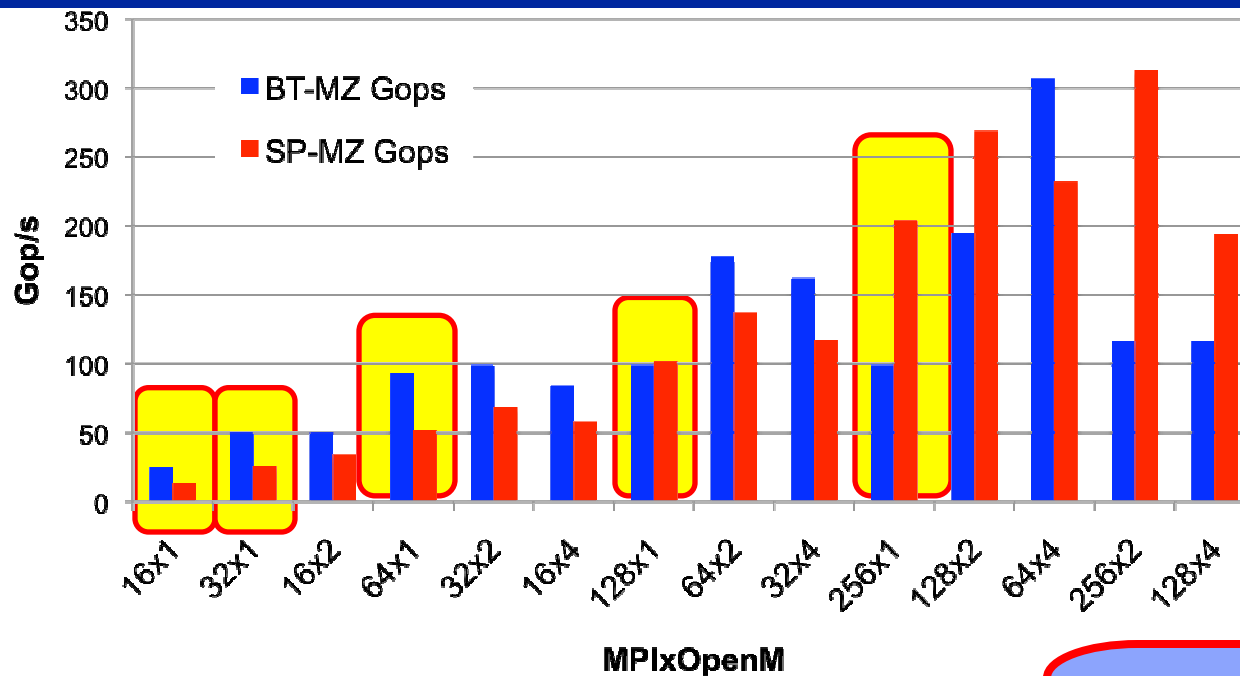


Fine-grain OpenMP Parallelism

```
subroutine x_solve (u, rhs,
!$OMP PARALLEL DEFAULT(SHARED)
!$OMP& PRIVATE(i,j,k, isize...)
isize = nx-1
!$OMP DO
  do k = 2, nz-1
    do j = 2, ny-1
      ...
      call lhsinit (lhs, isize)
      do i = 2, nx-1
        lhs(m,i,j,k)= ..
      end do
      call matvec ()
      call matmul ()....
    end do
  end do
end do
!$OMP END DO nowait
!$OMP END PARALLEL
```

NPB-MZ Class C Scalability on Cray XT4

No 512x1 since
#zones = 256 !!



- Results reported for 16-512 cores
 - SP-MZ pure MPI scales up to 256 cores
 - SP-MZ MPI/OpenMP scales to 512 cores
 - SP-MZ MPI/OpenMP outperforms pure MPI for 256 cores
 - BT-MZ MPI does not scale
 - BT-MZ MPI/OpenMP does not scale to 512 cores

Expected:
#MPI Processes limited

Unexpected!

Expected: Good load-
balance requires 64 x8

Sun Constellation Cluster Ranger (1)

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- **Application ... can benefit**
- Summary

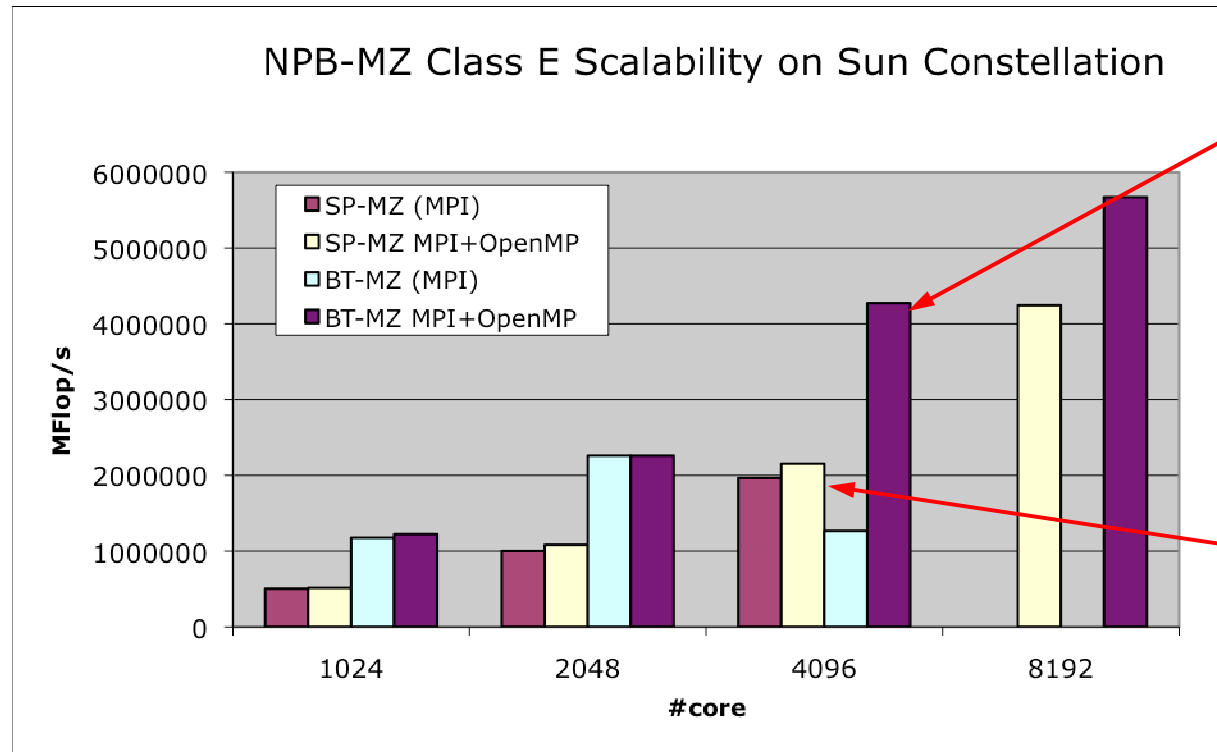
- Located at the Texas Advanced Computing Center (TACC), University of Texas at Austin (<http://www.tacc.utexas.edu>)
- 3936 Sun Blades, 4 AMD Quad-core 64bit 2.3GHz processors per node (blade), 62976 cores total
- 123TB aggregate memory
- Peak Performance 579 Tflops
- InfiniBand Switch interconnect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

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.sh bt-mz.exe
- numactl controls
 - Socket affinity: select sockets to run
 - Core affinity: select cores within socket
 - Memory policy: where to allocate memory

Default script for process
placement available on
Ranger

NPB-MZ Class E Scalability on Ranger



BT-MZ

Significant improvement (235%):
Load-balancing issues solved with MPI+OpenMP

SP-MZ

Pure MPI is already load-balanced.
But hybrid programming 9.6% faster

Unexpected!

- Scalability in Mflops with increasing number of cores
- MPI/OpenMP: Best Result over all MPI/OpenMP combinations for a fixed number of cores
- Use of numactl essential to achieve scalability

Numactl: Using Threads across Sockets

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- **Application ... can benefit**
- Summary

bt-mz.1024x8 yields
best load-balance

```
-pe 2way 8192  
export OMP_NUM_THREADS=8
```

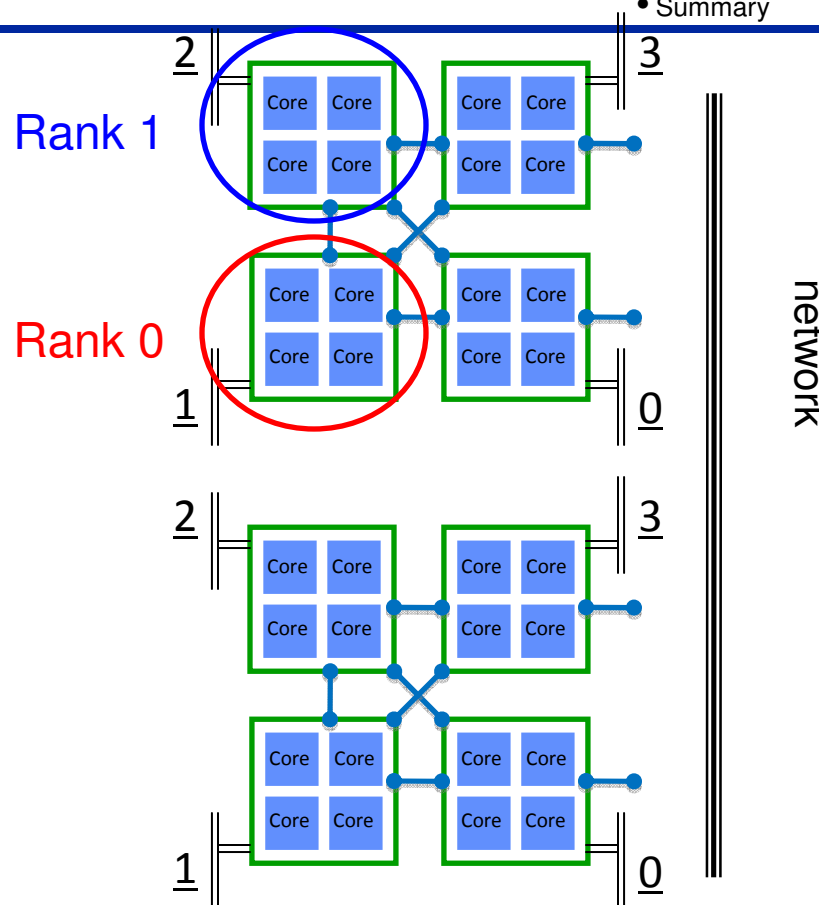
```
my_rank=$PMI_RANK  
local_rank=$(( $my_rank % $myway ))  
numnode=$(( $local_rank + 1 ))
```

Original:

```
-----  
numactl -N $numnode -m $numnode $*
```

Bad performance!

- Each process runs 8 threads on 4 cores
- Memory allocated on one socket



Numactl: Using Threads across Sockets

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- **Application ... can benefit**
- Summary

bt-mz.1024x8

```
export OMP_NUM_THREADS=8
```

```
my_rank=$PMI_RANK  
local_rank=$(( $my_rank % $myway ))  
numnode=$(( $local_rank + 1 ))
```

Original:

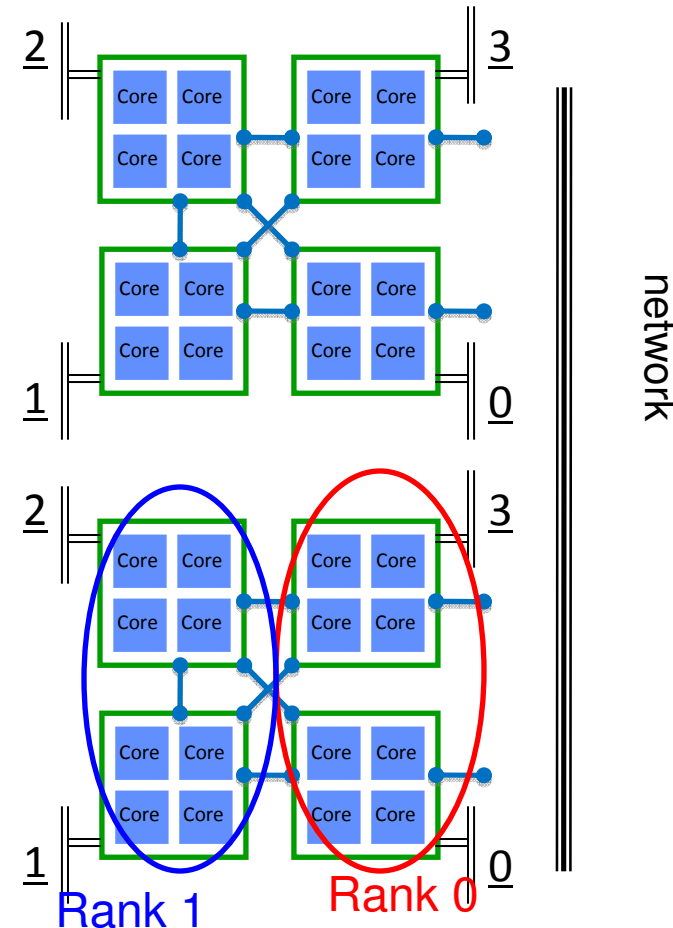
```
-----  
numactl -N $numnode -m $numnode $*
```

Modified:

```
-----  
if [ $local_rank -eq 0 ]; then  
    numactl -N 0,3 -m 0,3 $*  
else  
    numactl -N 1,2 -m 1,2 $*  
fi
```

Achieves Scalability!

- Process uses cores and memory across 2 sockets
- Suitable for 8 threads



Hybrid Programming – Outline

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit
- **Summary**

- Introduction / Motivation
- Programming Models on Clusters of SMP nodes
- Practical “How-To” on hybrid programming & Case Studies
- Mismatch Problems & Pitfalls
- Application Categories that Can Benefit from Hybrid Parallelization/Case Studies
- Summary on Hybrid Parallelization

Elements of Successful Hybrid Programming

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit
- **Summary**

- **System Requirements:**
 - Some level of shared memory parallelism, such as within a multi-core node
 - Runtime libraries and environment to support both models
 - Thread-safe MPI library
 - Compiler support for OpenMP directives, OpenMP runtime libraries
 - Mechanisms to map MPI processes onto cores and nodes
- **Application Requirements:**
 - Expose multiple levels of parallelism
 - Coarse-grained and fine-grained
 - Enough fine-grained parallelism to allow OpenMP scaling to the number of cores per node
- **Performance:**
 - Highly dependent on optimal process and thread placement
 - No standard API to achieve optimal placement
 - Optimal placement may not be known beforehand (i.e. optimal number of threads per MPI process) or requirements may change during execution
 - Memory traffic yields resource contention on multi-core nodes
 - Cache optimization more critical than on single core nodes

Recipe for Successful Hybrid Programming

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit

➤ Summary

- **Familiarize yourself with the layout of your system:**
 - Blades, nodes, sockets, cores?
 - Interconnects?
 - Level of Shared Memory Parallelism?
- **Check system software**
 - Compiler options, MPI library, thread support in MPI
 - Process placement
- **Analyze your application:**
 - Does MPI scale? If not, why?
 - Load-imbalance => OpenMP might help
 - Too much time in communication? Load-imbalance? Workload too small?
 - Does OpenMP scale?
- **Performance Optimization**
 - Optimal process and thread placement is important
 - Find out how to achieve it on your system
 - Cache optimization critical to mitigate resource contention

Hybrid Programming: Does it Help?

PART 2: Hybrid MPI+OpenMP

- Introduction
- Programming Models
- How-To on hybrid prog.
- Mismatch Problems
- Application ... can benefit

➤ Summary

- **Hybrid Codes provide these opportunities:**
 - Lower communication overhead
 - Few multi-threaded MPI processes vs Many single-threaded processes
 - Fewer number of calls and smaller amount of data communicated
 - Lower memory requirements
 - Reduced amount of replicated data
 - Reduced size of MPI internal buffer space
 - May become more important for systems of 100's or 1000's cores per node
 - Provide for flexible load-balancing on coarse and fine grain
 - Smaller #of MPI processes leave room to assign workload more even
 - MPI processes with higher workload could employ more threads
 - Increase parallelism
 - Domain decomposition as well as loop level parallelism can be exploited

YES, IT CAN!

<https://fs.hlr.de/projects/rabenseifner/publ/SciDAC2009-Part2-Hybrid.pdf>