Communication Characteristics and Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-core SMP Nodes

Georg Hager, Erlangen Regional Computing Center (RRZE)
Gabriele Jost, Texas Advanced Computing Center (TACC)
Rolf Rabenseifner, High Performance Computing Center Stuttgart (HLRS)
Neil Stringfellow, Swiss National Supercomputing Centre (CSCS), presenter

Cray User Group
Atlanta, GA, May 4-7, 2009

Aspects & Outline

• High Performance Computing (HPC) systems
  – Always hierarchical hardware design
  – Programming models on hierarchical hardware

• Mismatch problems
  – Programming models are not suited for hierarchical hardware

• Performance opportunities with MPI+OpenMP hybrid programming
  – NPB BT/SP-MZ benchmark results on Cray XT5

• Optimization always requires knowledge about the hardware
  – … and appropriate runtime support
  – It's a little more complicated than make; mpirun

High Performance Computing (HPC) systems → 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 the best programming model?

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

• Lore: hybrid programming slower than pure MPI
  – Why?
Example from SC

- Pure MPI versus Hybrid MPI+OpenMP (Masteronly)
- What’s better? → What does it depend on?


Explicit/Semi implicit C154N6 SEAM vs T170 PSTSWM, 16 Level, NCAR

Parallel Programming Models on Hybrid Platforms

- pure MPI
  - one MPI process on each CPU
- hybrid MPI+OpenMP
  - MPI: inter-node communication
  - OpenMP: inside of each SMP node
- OpenMP only
  - distributed virtual shared memory

No overlap of Comm. + Comp.
- MPI only outside of parallel regions of the numerical application code

Overlapping Comm. + Comp.
- MPI communication by one or a few threads while other threads are computing

“Masteronly” mode
- This can get ugly…

See also

Pure MPI

- Advantages
  - No modifications on existing MPI codes
  - MPI library need not to support multiple threads
- Major problems
  - Does MPI library internally use different protocols?
    - Network communication between the nodes
    - Shared memory inside of the SMP nodes
      - Usually true today, but see later
  - Does application topology fit on hardware topology?
  - MPI-communication inside of SMP nodes – unnecessary?

Hybrid Masteronly

- Advantages
  - No message passing inside SMP nodes
  - No intra-node topology problem
  - (but watch thread placement)

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!
  - Inter-node bandwidth saturation?
  - As of MPI 2.1, MPI lib must support at least MPI_THREAD_FUNNELED
    (there is no MPI_THREAD_MASTERONLY)
Overlapping Communication and Computation

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

```c
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)

- 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)

- Communication of modified parts of pages
- 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 last section
- Quantitative implications → depends on the application

Examples:

<table>
<thead>
<tr>
<th>Benefit through hybrid (see next section)</th>
<th>No.1</th>
<th>No.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>30%</td>
<td>10%</td>
<td>-10%</td>
</tr>
<tr>
<td>-25%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>+20%</td>
<td>-15%</td>
</tr>
</tbody>
</table>

In most cases: Both categories!

---

The Topology Problem with pure MPI

- one MPI process on each CPU

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 CPU

Application example on 80 cores:
- Cartesian application with 5 x 16 = 80 sub-domains
- On system with 10 x dual socket x quad-core

Round robin ranking of MPI_COMM_WORLD

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

Never trust the default !!!

Two levels of domain decomposition
- 10 x inter-node connections per node
- 4 x inter-socket connection per node

Bad affinity of cores to ranks

The Topology Problem with pure MPI
one MPI process on each CPU

Application example on 80 cores:
- Cartesian application with 5 x 16 = 80 sub-domains
- On system with 10 x dual socket x quad-core

Round robin ranking of MPI_COMM_WORLD

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

Two levels of domain decomposition
- 10 x inter-node connections per node
- 4 x inter-socket connection per node

Good affinity of cores to ranks – best solution if intra-node MPI is “fast”

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)
The Topology Problem with \textit{hybrid MPI+OpenMP}

- 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 \sim 4 x more traffic
+ 2 x inter-socket connections per node

**Affinity matters!**

The Mapping Problem with \textit{mixed model}

- Do we have this? ...
- or that?

Several multi-threaded MPI process per SMP node:

- Problem: Where are your processes and threads really located?
- Solution:
  - Use platform-dependent tools!
  - \textit{e.g., ibrun numactl} option on Sun

\rightarrow case-study on Cray XT5 with BT-MZ and SP-MZ

Intra-node communication issues

- Problem:
  - If several MPI processes on each SMP node -> unnecessary (and inefficient?) intra-node communication

- Remarks:
  - MPI library must use appropriate fabrics / protocol for intra-node communication
  - Intra-node bandwidth/latency probably much better than inter-node -> problem may be small
  - MPI implementation may cause unnecessary data copying -> waste of memory bandwidth

**Quality aspects of the MPI library**

Realities of intra-node MPI: IMB Ping-Pong on Cray XT5 – Latency

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{chart.png}
\caption{Latency comparison between inter-node, inter-socket, and intra-socket communication.}
\end{figure}

**Affinity matters!**
### Realities of intra-node MPI:

**IMB Ping-Pong on Cray XT5 – Bandwidth**

- **Shared cache advantage**
  - Between two cores of one socket
  - Between two nodes via interconnect fabric
  - Between two sockets of one node (cache effects eliminated)

**Affinity matters!**

### Sleeping threads and network saturation with Masteronly

- **MPI only outside of parallel regions**

### 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

```c
if (my_thread_rank < 1) {
    MPI_SendRecv....
} 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)*my_range;
    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**
  - proposal for OpenMP 3.x?
  - for OpenMP 4.x


- **Tasking** (OpenMP 3.0)
  - works only if app can cope with dynamic scheduling

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

#### For further examples and performance case studies see:
- R. Rabenseifner, G. Hager, G. Jost, and R. Keller: Hybrid MPI and OpenMP Parallel Programming. SC08 Tutorial M09
OpenMP: Additional Overhead & Pitfalls

- Using OpenMP
  - may prohibit compiler optimization
  - may cause significant loss of computational performance
- Thread fork / join, implicit barriers (see next slide)
- On ccNUMA SMP nodes:
  - E.g. in the masteronly scheme:
    - One thread produces data
    - Master thread sends the data with MPI
  - data may be communicated between NUMA domains
- Amdahl’s law for each level of parallelism
- Using MPI-parallel application libraries?
  - Are they prepared for hybrid?

OpenMP Overhead

- As with intra-node MPI, OpenMP loop start overhead varies with the mutual position of threads in a team
- Possible variations
  - Intra-socket vs. inter-socket
  - Different overhead for ‘parallel for’ vs. plain “for”
  - If one multi-threaded MPI process spans multiple sockets,
    - ... are neighboring threads on neighboring cores?
    - ... or are threads distributed “round-robin” across cores?
- Test benchmark: Vector triad

```c
#pragma omp parallel
for(int j=0; j < NITER; j++){
  #pragma omp (parallel) for (nowait)
  for(i=0; i < N; ++i)
    a[i]=b[i]+c[i]*d[i];
  if(OBSCURE)
    dummy(a,b,c,d);
}
```

Look at performance for small array sizes!

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 → see below
- In the NPB-MZ case studies
  - We tried to use an 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 by using appropriate system tools
Hybrid MPI/OpenMP

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
  - Significant reduction of replicated data on MPI level

- Chances, if MPI speedup is limited due to "algorithmic" problems
  - Significantly reduced number of MPI processes
  - OpenMP threading makes each process "faster", even if code is already Amdahl-limited

Nested Parallelism

- Example NPB: BT-MZ (Block tridiagonal simulated CFD application)
  - Outer loop:
    - limited number of zones → limited parallelism
    - zones with different workload → speedup < 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

Cray XT5 Experiments

- Results obtained by the courtesy of the HPCMO Program and the Engineer Research and Development Center Major Shared Resource Center, Vicksburg, MS (http://www.erdc.hpc.mil/index)
- Cray XT5 is located at the Arctic Region Supercomputing Center (ARSC)
  - 432- Cray XT5 compute nodes with
    - 32 GB of shared memory per node (4 GB per core)
    - 2 - quad core 2.3 GHz AMD Opteron processors per node.
    - 1 - Seastar2+ Interconnect Module per node.
  - Cray Seastar2+ Interconnect between all compute and login nodes

- Compilation:
  - Cray ftn compiler based on PGI pgf90 7.2.2
  - ftn -fastsse -tp barcelona-64 -r8 -mp=nonuma

- Execution:
  - MPICH based MPI-2
  - export OMP_NUM_THREADS={8,4,2,1}
  - aprun -n NPROCS=N 1 -d 8 ./a.out
  - aprun -n NPROCS=S {1,2,4} -d {4,2,1} ./a.out
Cray XT5 Process Placement

- `export OMP_NUM_THREADS=8`
- `aprun -n 64 -N 1 -d 8 bt-mz.C.64x8`
- `export OMP_NUM_THREADS=4`
- `aprun -n 128 -S 1 -d 4 bt-mz.C.128x4`

**Conclusions & outlook**

- Future High Performance Computing (HPC)
  - always hierarchical hardware design
- Mismatches and chances 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?

- ... and don’t forget the usual OpenMP pitfalls
  - Fork/join, barriers, NUMA placement

**NPB-MZ Class C Scalability on Cray XT5**

- Results reported for 16-512 cores
  1. SP-MZ pure MPI scales up to 256 cores
  2. SP-MZ MPI/OpenMP scales to 512 cores
  3. SP-MZ MPI/OpenMP outperforms pure MPI for 128, 256 cores
  4. BT-MZ MPI does not scale
  5. BT-MZ MPI/OpenMP scales to 512 cores, outperforms pure MPI

- Maximum for same number of cores

- Expected: #MPI Processes limited
- Unexpected!

- Expected: Best load-balance for 64x8!