Hybrid Parallel Programming on HPC Platforms

Rolf Rabenseifner
rabenseifner@hlrs.de
University of Stuttgart, High Performance Computing Center Stuttgart (HLRS)
www.hlrs.de
EWOMP 2003
Sep. 22–26, Aachen, Germany

Motivation

- HPC systems
  - often clusters of SMP nodes
  - i.e., hybrid architectures
- Using the communication bandwidth of the hardware
- Minimizing synchronization = idle time
- Appropriate parallel programming models / Pros & Cons
Major Programming models on hybrid systems

- Pure MPI (one MPI process on each CPU)
- Hybrid MPI+OpenMP
  - shared memory OpenMP
  - distributed memory MPI
- Other: Virtual shared memory systems, HPF, ...
- Often hybrid programming (MPI+OpenMP) slower than pure MPI
  - why?

```
#pragma omp parallel for
for (j=…;…; j++)
  block_to_be_parallelized
again_some_serial_code
```

Master thread, other threads

MPI
Sequential program on each CPU
Explicit Message Passing by calling MPI_Send & MPI_Recv

OpenMP
(shared data)

Node Interconnect

Hybrid Parallel Programming Rolf Rabenseifner
Slide 3 / 34 High Perf. Comp. Center, Univ. Stuttgart

Example from SC 2001

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

Explicit C154N6 16 Level SEAM: NPACI Results with 7 or 8 processes or threads per node

Figures: Richard D. Loft, Stephen J. Thomas, John M. Dennis:
Terascale Spectral Element Dynamical Core for Atmospheric General Circulation Models.
Fig. 9 and 10.
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**
  - MPI only outside of parallel regions

- **Multiple/only**
  - • appl. threads inside of MPI

- **Multiple & Reserved**
  - reserved thread for communication

- **Funneled & Reserved**
  - reserved thread for communication

- **Funneled**
  - MPI only on master-thread

- **Multiple & Reserved**
  - reserved threads for communication

- **Multiple with Full Load Balancing**

Mismatch Problems

- **Topology problem** [with pure MPI]
- **Unnecessary intra-node communication** [with pure MPI]
- **Inter-node bandwidth problem** [with hybrid MPI+OpenMP]
- **Sleeping threads and saturation problem** [with masteronly]
- **Additional OpenMP overhead** [with hybrid MPI+OpenMP]
  - Thread startup / join
  - Cache flush (data source thread — communicating thread — sync. → flush)
- **Overlapping communication and computation** [with hybrid MPI+OpenMP]
  - an application problem → separation of local or halo-based code
  - a programming problem → thread-ranks-based vs. OpenMP work-sharing
  - a load balancing problem, if only some threads communicate / compute

» no silver bullet
  - each parallelization scheme has its problems
The Topology Problem with Pure MPI

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

Problems
- To fit application topology on hardware topology

Solutions for Cartesian grids:
- E.g. choosing ranks in MPI_COMM_WORLD ???
  - round robin (rank 0 on node 0, rank 1 on node 1, ...)
  - Sequential (ranks 0-7 on 1st node, ranks 8-15 on 2nd, ...)

... in general
- load balancing in two steps:
  - all cells among the SMP nodes (e.g. with ParMetis)
  - inside of each node: distributing the cells among the CPUs
- or ... → using hybrid programming models

Unnecessary intra-node communication

Alternative:
- Hybrid MPI+OpenMP
- No intra-node messages
- Longer inter-node messages
- Really faster ???????

(... wait 2 slides)
Programming Models on Hybrid Platforms: Hybrid Masteronly

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

**Problems**
- MPI-lib must support MPI_THREAD_FUNNELED

**Disadvantages**
- do we get full inter-node bandwidth? ... next slide
- all other threads are sleeping while master thread communicates

→Reason for implementing overlapping of communication & computation

```c
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 */
```

**Experiment: Orthogonal parallel communication**

- **pure MPI**
- **Masteronly**

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

- **8x8MB**: 2.0 ms
- **8x1MB**: 9.6 ms

- **hybrid**: 19.2 ms
- **pure MPI**: $\Sigma \leq 11.6$ ms

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

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

### Ratio on several platforms

<table>
<thead>
<tr>
<th>Platform</th>
<th>Masteronly: MPI cannot saturate inter-node bandwidth</th>
<th>Pure MPI is faster</th>
<th>Hybrid is faster</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM SP, 8x16 CPUs, 1 CPU Masteronly</td>
<td>Pure MPI, horizontal + vertical</td>
<td>IBM SP</td>
<td>Cray X1 and NEC SX are well prepared for hybrid masteronly programming</td>
</tr>
<tr>
<td>SGI O3000, 16x4 CPUs, 1 CPU Masteronly</td>
<td>Cray X1 8x4 MSPs, 1 MSP Masteronly</td>
<td>SGI</td>
<td></td>
</tr>
<tr>
<td>Hitachi SR8000, 8x8 CPUs, 1 CPU Masteronly</td>
<td>NEC SX6 glmem 4x8 CPUs, 1 CPU Masteronly</td>
<td>Hitachi</td>
<td></td>
</tr>
<tr>
<td>Pure MPI, horizontal + vertical</td>
<td>Pure MPI, horizontal + vertical</td>
<td>Pure MPI, horizontal + vertical</td>
<td></td>
</tr>
</tbody>
</table>

Cray X1 and NEC SX results are preliminary.
Possible Reasons

- **Hardware:**
  - is one CPU able to saturate the inter-node network?

- **Software:**
  - internal MPI buffering may cause additional memory traffic
    → memory bandwidth may be the real restricting factor?

→ Let’s look at parallel bandwidth results

Multiple inter-node communication paths

- **MPI+OpenMP:** only vertical
- **pure MPI:** vertical AND horizontal messages

Multiple vertical communication paths, e.g.,
- 3 of 8 CPUs in each node
- stride 2

**hybrid:** 3×8 × 8/3MB

**intra-node** 8×8×1MB

**pure MPI:** intra- + inter-node (= vert. + horizontal)

Following benchmark results with one MPI process on each CPU
Multiple inter-node communication paths: Hitachi SR8000

Inter-node bandwidth per SMP node, accumulated over its CPUs, *)
on Hitachi SR8K

To spend more than 3 CPUs/node for communication makes no sense

Intra-node messages do not count for bandwidth

*) Bandwidth per node: totally transferred bytes on the inter-node network / wall clock time / number of nodes

Multiple inter-node communication paths: Hitachi SR 8000

Hybrid communication time / pure MPI communication time on Hitachi SR 8000

Hybrid is faster than pure MPI if ≥ 2 CPUs/node are used for intra-node communication in hybrid programming model
Multiple inter-node communication paths: **IBM SP**

Inter-node bandwidth per SMP node, accumulated over its CPUs, on IBM at NERSC (16 Power3+ CPUs/node)

- More than 4 CPUs per node needed to achieve full inter-node bandwidth
- With 3 CPUs similar to pure MPI

The second CPU doubles the accumulated bandwidth

**Intra-node messages do not count for bandwidth**

Inverses:
- More CPUs = less bandwidth

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

Multiple inter-node communication paths: **NEC SX-6** (using global memory)

Inter-node bandwidth per SMP node, accumulated over its CPUs, on NEC SX6 (with MPI_Alloc_mem)

Inverse:
- More CPUs = less bandwidth

Measurements: Thanks to Holger Berger, NEC.

© Rolf Rabenseifner: Hybrid Parallel Programming on HPC Platforms.
EWOMP 2003, Sep. 22–26, RWTH, Aachen, Germany. Page 9
Multiple inter-node communication paths: Cray X1, used with 4 MSPs/node (preliminary results)

Inter-node bandwidth per SMP node, accumulated over its CPUs, on Cray X1, 4 MSPs / node (1 MSP = 4 CPUs)

1 MSP achieves already 80% of full inter-node bandwidth

Intra-node messages do not count for bandwidth

Multiple inter-node communication paths: Cray X1, used with 4 MSPs/node, shmem put (instead MPI)

Inter-node bandwidth per SMP node, accumulated over its CPUs, on Cray X1, 4 MSPs / node (1 MSP = 4 CPUs), shmem put

1 MSP achieves already 75% of full inter-node bandwidth

Intra-node messages do not count for bandwidth

Measurements:
Thanks to Monika Wierse and Wilfried Oed, CRAY.
Comparison

Inter-node bandwidth per SMP node, accumulated over its CPUs *)

accumulated message size from node to node

*) Bandwidth per node: totally transferred bytes on the inter-node network / wall clock time / number of nodes

Comparison (as percentage of maximal bandwidth and #CPUs)

Nearly full bandwidth
- with 1 MSP on Cray
- with 1 CPU on NEC

50 % and less on the other platforms

Nearly all platforms:
>80% bandwidth with 25% of CPUs/node
Comparison (only 960 kB aggregated message size)

Similar behavior on Cray X1 and NEC SX-6

Myrinet Cluster

• 1 CPU can achieve full inter-node bandwidth
• Myrinet cluster is as well prepared for hybrid masteronly programming
Myrinet Cluster (only 64 nodes)

Inter-node bandwidth per SMP node, accumulated over its CPUs, on HELICS, 2 CPUs / node, Myrinet

- 1 CPU can achieve full inter-node bandwidth
- Myrinet cluster is as well prepared for hybrid masteronly programming

The sleeping-threads and the saturation problem

- Masteronly:
  - all other threads are sleeping while master thread calls MPI
    → wasting CPU time
    → → → wasting plenty of CPU time
    if master thread cannot saturate the inter-node network

- Pure MPI:
  - all threads communicate,
    but already 1-3 threads could saturate the network
  → wasting CPU time

→ Overlapping communication and computation
Overlapping Communication and Computation

- 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

- the load balancing problem

```c
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 (cont’d)

- the load balancing problem:
  - some threads communicate, others not
  - balance work on both types of threads
  - strategies:
    - Funneled & Reserved
      reserved threads for communication
  ➔ a good chance !!! … see next slide
    - Multiple & Reserved
      reserved threads for communication

- reservation of one or a fixed amount of threads (or portion of a thread) for communication
- see example on previous slide: 1 thread was reserved for communication

- Full Load Balancing

- Very hard to do !!!
Hybrid Parallel Programming

Performance ratio (theory)

\[ \varepsilon = \frac{1}{\frac{T_{\text{hybrid, funneled & reserved}}}{T_{\text{hybrid, masteronly}}}} \]

\( \varepsilon > 1 \Rightarrow \text{funneled & reserved is faster} \)

\( \varepsilon < 1 \Rightarrow \text{masteronly is faster} \)

Good chance of funneled & reserved:

\[ \varepsilon_{\text{max}} = 1 + m(1 - \frac{1}{n}) \]

Small risk of funneled & reserved:

\[ \varepsilon_{\text{min}} = 1 - \frac{m}{n} \]

Hybrid Programming on Cray X1: MSP based usage

- pure MPI or hybrid masteronly MPI+OpenMP
  - same communication time
- 1 MSP already achieves 80% of maximum bandwidth (contiguous data)
  - Are CPU-intensive MPI routines (Reduce, strided data) efficient & multi-threaded?
- Hybrid programming → 4 layers of parallelism
  - MPI between nodes (e.g. domain decomposition)
  - OpenMP between MSPs (e.g. outer loops)
  - Automatic parallelization (e.g. inner loops)
  - Vectorization (e.g. most inner loops)
- risk of Amdahl’s law on each level!
- Hybrid & overlapping communication and computation
  - horrible programming interface (but standardized)
  - but chance to use sleeping MSPs while master MSP communicates

Hybrid Programming on Cray X1: SSP based

- Communication is hardware-bound to SSP
  - 1 SSP can get only 1/4 of 1 MSP’s inter-node bandwidth
  - with `shmem put`
    all SSPs of a node can together achieve full inter-node bandwidth
    (12.3 GB/s of 12.8 GB/s hardware specification)

- Hybrid MPI+OpenMP, masteronly style
  - optimized MPI library needed with same bandwidth
    as on 1 or 4 MSP
  - e.g., internally thread-parallel

- Multiple communicating user-threads are not supported

- pure MPI
  - efficient MPI implementation under development

Comparing inter-node bandwidth
with CPU performance

<table>
<thead>
<tr>
<th>System</th>
<th>Master-only, inter-node bw [GB/s]</th>
<th>Master-only bw / max. intra-node bw [GB/s]</th>
<th>pure MPI, inter-node bw [GB/s]</th>
<th>pure MPI, intra-node bw [GB/s]</th>
<th>memo-ry bandwidth [GB/s]</th>
<th>Peak &amp; Linpack perfor- mance Gflop/s</th>
<th>max.inter-node bw / peak &amp; Linpack perf. B/Flop</th>
<th>nodes*CPUs</th>
</tr>
</thead>
</table>
| Cray X1, `shmem_put`
 preliminary results          | 9.27                              | 12.34                                      | 75 %                          | 33.0                          | 136                      | 51.2 45.03                          | 0.241 0.274                                 | 8 * 4 MSPs |
| Cray X1, MPI
 preliminary results          | 4.52                              | 5.52                                       | 82 %                          | 19.5                          | 136                      | 51.2 45.03                          | 0.108 0.123                                 | 8 * 4 MSPs |
| NEC SX-6 global memory       | 7.56                              | 4.98                                       | 100 %                         | 78.7                          | 256                      | 64 61.83                            | 0.118 0.122                                 | 4 * 8 CPUs |
| NEC SX-5Be local memory      | 2.27                              | 2.50                                       | 91 %                          | 35.1                          | 512                      | 64 60.50                            | 0.039 0.041                                 | 2 *16 CPUs |
| Hi-Cchi SR8000               | 0.45                              | 0.91                                       | 49 %                          | 5.0                           | 8                        | 64 6.32                             | 0.114 0.133                                 | 8 * 8 CPUs |
| IBM SP Power3+               | 0.16                              | 0.57                                       | 28 %                          | 2.0                           | 16                       | 24 14.27                            | 0.023 0.040                                 | 8 *16 CPUs |
| SGI O3000, 600MHz            | 0.43                             | 1.74                                      | 25 %                          | 1.73                          | 4.8                      | 3.64 0.363                          | 0.478                                      | 16 *4 CPUs |
| SUN-fire (prelim.)           | 0.15                              | 0.85                                       | 18 %                          | 1.68                          |                          | 2.80 1.61                          | 0.043 0.074                                 | 4 *24 CPUs |
| HELICS Dual-PC cluster with Myrinet | 0.118 | 0.119                                | 100 %                         | 0.104                         |                          | 2.80 1.61                          | 0.043 0.074                                 | 128 *2 CPUs |

All values: aggregated over one SMP nodes. *) mess. size: 1 MB a) 2 MB

Bandwidth per node: totally transferred bytes on the network / number of nodes / wall clock time

© Rolf Rabenseifner: Hybrid Parallel Programming on HPC Platforms.
EWOMP 2003, Sep. 22–26, RWTH, Aachen, Germany. Page 16
Acknowledgements

- I want to thank
  - Gerhard Wellein, RRZE
  - Monika Wierse, Wilfried Oed, and Tom Goozen, CRAY
  - Holger Berger, NEC
  - Reiner Vogelsang, SGI
  - Gabriele Jost, NASA
  - Dieter an Mey, RZ Aachen
  - Horst Simon, NERSC
  - my colleges at HLRS

Conclusions

- Cray X1 with MSPs (1 node = 4 MSPs), NEC SX-5/6, and Myrinet-cluster:
  - well designed hybrid MPI+OpenMP masteronly scheme
- Cray X1 with SSPs (1 node = 16 SSPs) and others (e.g., IBM, SGI, SUN)
  - hybrid programming:
    - masteronly style cannot saturate inter-node bandwidth
    - thread-parallel MPI library is needed
- Pure MPI and hybrid masteronly:
  - idling CPUs (while one is communicating)
- Optimal performance:
  - overlapping of communication & computation
    - extreme programming effort
  - optimal throughput
    - reuse of idling CPUs by other applications
    - single threaded, vectorized, low-priority, small-medium memory needs

See also www.hlrs.de/people/rabenseifner → list of publications