# Hybrid Parallel Programming: Performance Problems and Chances on Cray X1, NEC SX-6 and Other Platforms

Rolf Rabenseifner
rabenseifner@hlrs.de

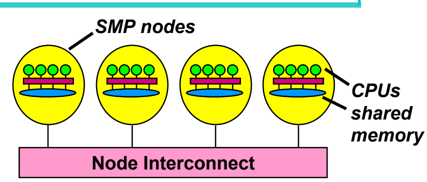
University of Stuttgart,
High Performance Computing Center Stuttgart (HLRS)
www.hlrs.de

**CUG SUMMIT 2003**
May 12–16, Columbus, Ohio, USA



## Motivation

- HPC systems
  - often clusters of SMP nodes
  - i.e., hybrid architectures



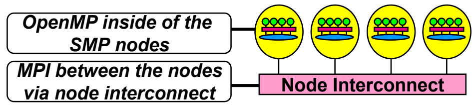

- Using the communication bandwidth of the hardware
- Minimizing synchronization = idle time

} **optimal usage of the hardware**

- 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

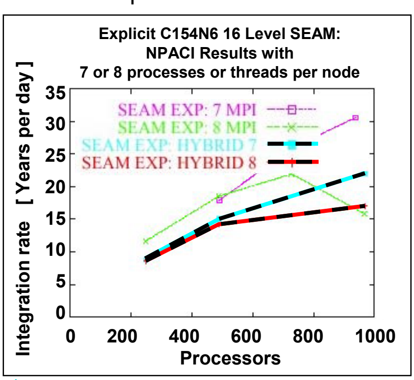


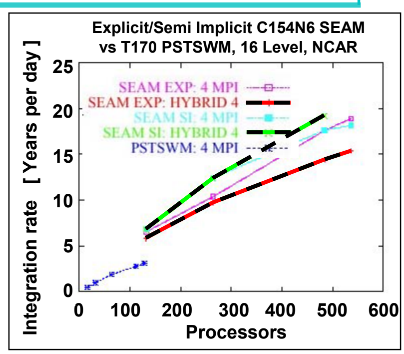

- Other: Virtual shared memory systems, HPF, …
- Often **hybrid programming (MPI+OpenMP)** slower than **pure MPI**
  - why?



## Example from SC 2001

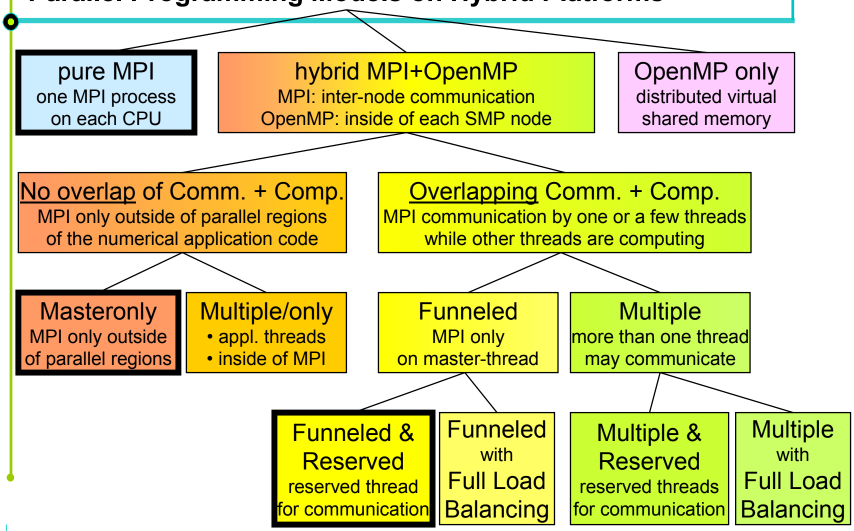
- Pure MPI versus
  Hybrid MPI+OpenMP (Masteronly)
- What‘s better?
  → it depends on?




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.

## Parallel Programming Models on Hybrid Platforms






## Mismatch Problems

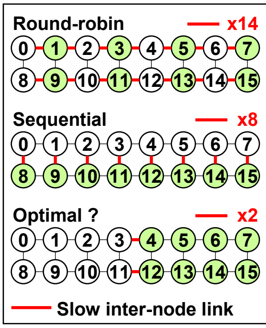
- **Topology problem** [with pure MPI]
- **Unnecessary intra-node communication** [with pure MPI]
- **Inter-node bandwidth problem** [with hybrid MPI+OpenMP]
- **Sleeping threads and saturation problem** [with masteronly] [with pure MPI]
- **Additional OpenMP overhead** [with hybrid MPI+OpenMP]
  - Thread 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

**pure MPI**
one MPI process
on each CPU

Exa.: 2 SMP nodes, 8 CPUs/node




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

Problems
- To fit application topology on hardware topology

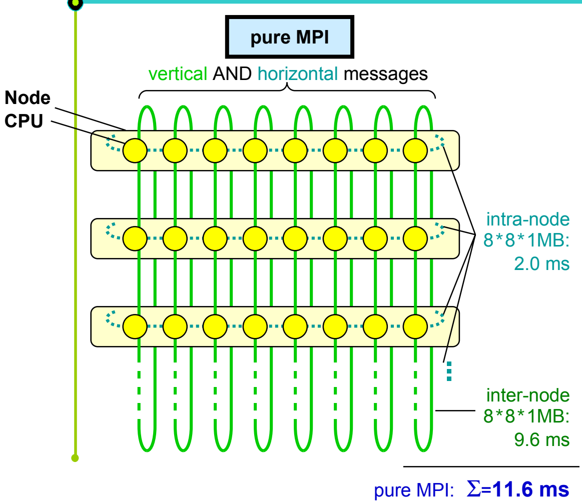
Solutions for Cartesian grids:
- E.g. choosing ranks in MPI_COMM_WORLD ???
  - **round robin (rank 0 on node 0, rank 1 on node 1, ... )**
  - **Sequential (ranks 0-7 on 1st node, ranks 8-15 on 2nd …)**

… 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**

Hybrid Parallel Programming Rolf Rabenseifner
Slide 7 / 32 High Perf. Comp. Center, Univ. Stuttgart

HLRS

## Unnecessary intra-node communication




Alternative:
- Hybrid MPI+OpenMP
- No intra-node messages
- Longer inter-node messages
- **Really faster ???????**
  **(… wait 2 slides)**

Timing:
Hitachi SR8000, MPI_Sendrecv
8 nodes, each node with 8 CPUs

Hybrid Parallel Programming Rolf Rabenseifner
Slide 8 / 32 High Perf. Comp. Center, Univ. Stuttgart

HLRS

## Programming Models on Hybrid Platforms: Hybrid Masteronly

**Masteronly**
MPI only outside of parallel regions

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

  /* on master thread only */
     MPI_Send (original data
       to halo areas
       in other SMP nodes)
     MPI_Recv (halo data
       from the neighbors)
} /*end for loop
```

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

Problems
- MPI-lib must support MPI_THREAD_FUNNELED

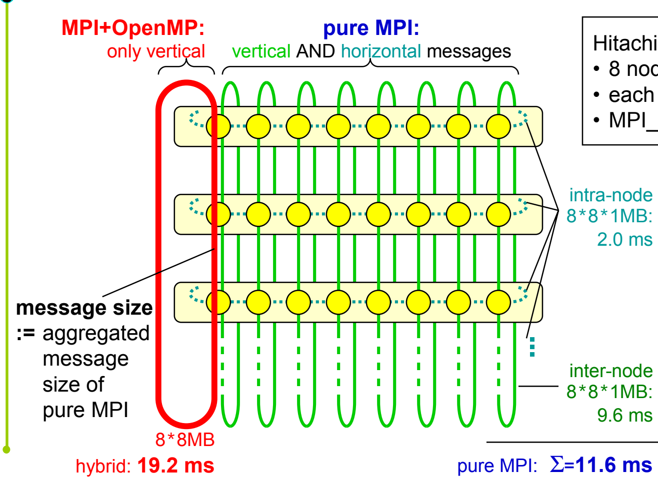
Disadvantages
- do we get full inter-node bandwidth? **… next slide**
- all other threads are sleeping while master thread communicates

**→Reason for implementing overlapping of communication & computation**



## Experiment: Orthogonal parallel communication

pure MPI ↔ Masteronly

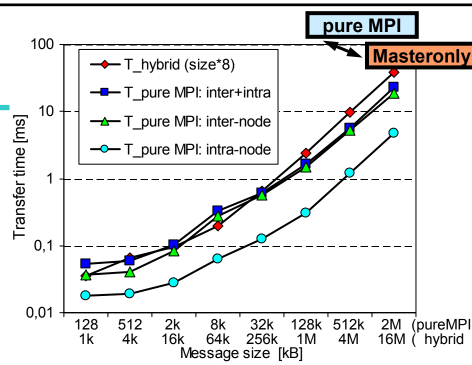


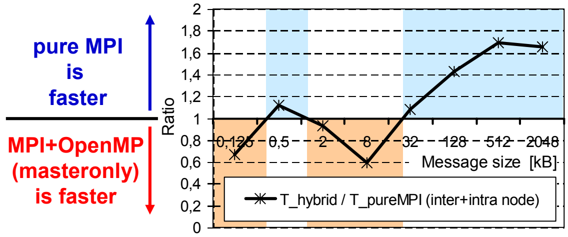

Hitachi SR8000
- 8 nodes
- each node with 8 CPUs
- MPI_Sendrecv

→ **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: $T_{hybrid} / T_{pureMPI} > 1.6$
- OpenMP master thread cannot saturate the inter-node network bandwidth

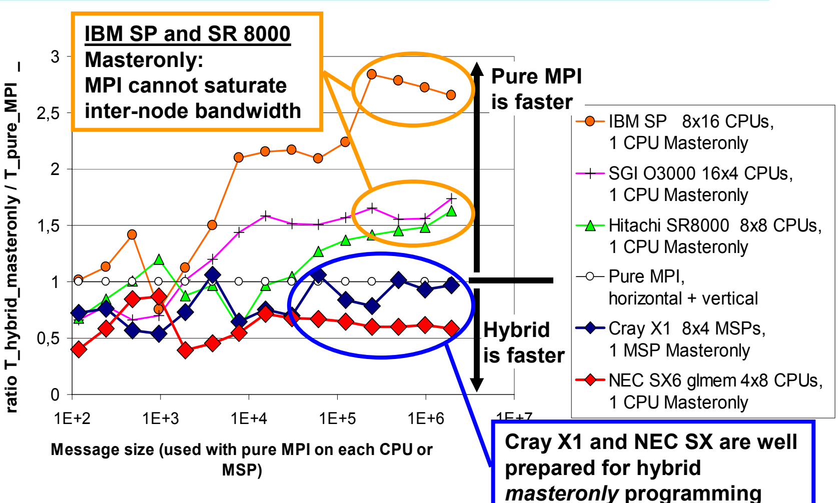



Hybrid Parallel Programming — Rolf Rabenseifner
Slide 11 / 32 — High Perf. Comp. Center, Univ. Stuttgart

H L R S

## Ratio on several platforms




Hybrid Parallel Programming — Rolf Rabenseifner
Slide 12 / 32 — High Perf. Comp. Center, Univ. Stuttgart

H L R S

Cray X1 and SGI 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?

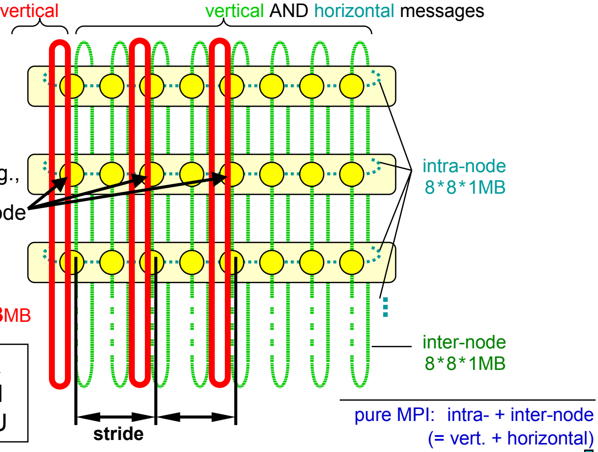
**➔ Let's look at parallel bandwidth results**



## Multiple inter-node communication paths

**MPI+OpenMP:** only vertical

pure MPI: vertical AND horizontal messages

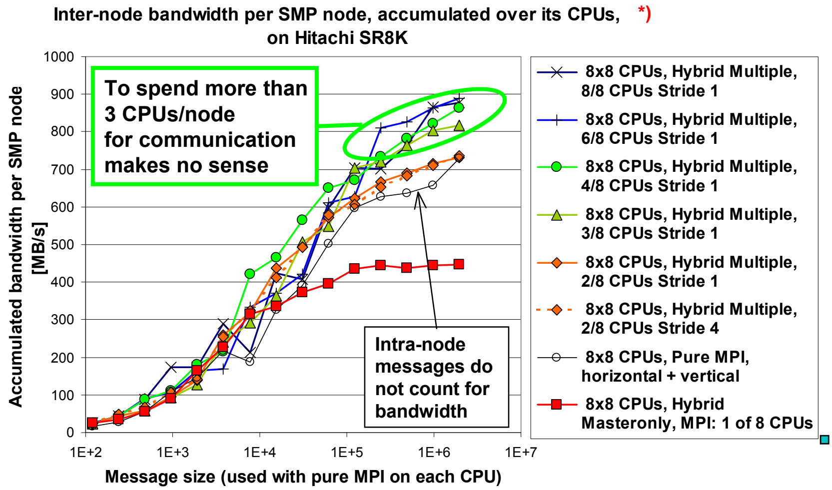



Multiple vertical communication paths, e.g.,
- **3** of 8 CPUs in each node
- **stride 2**

hybrid: **3***8 * 8/**3**MB

Following benchmark results: with one MPI process on each CPU

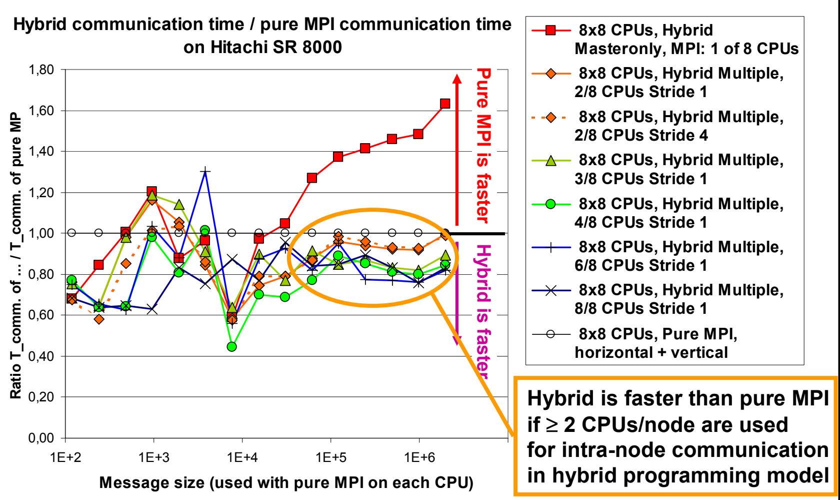
## Multiple inter-node communication paths: Hitachi SR8000





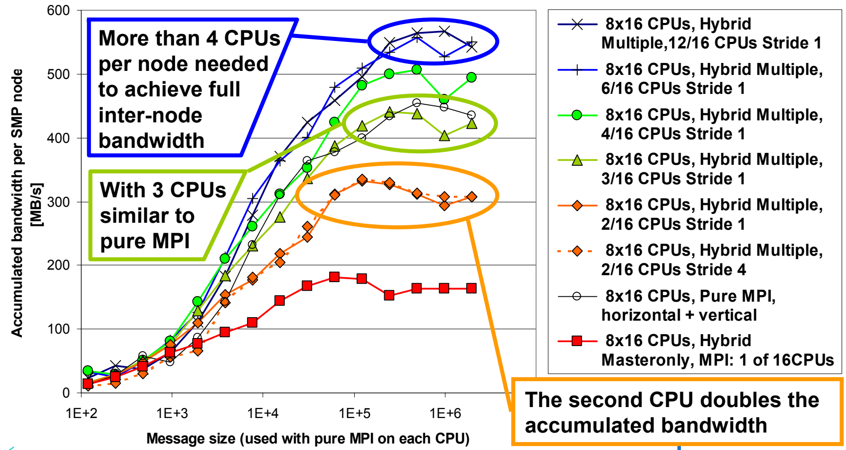

*) Bandwidth per node: totally transferred bytes on the inter-node network / wall clock time / number of nodes

## Multiple inter-node communication paths: Hitachi SR 8000



## 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)



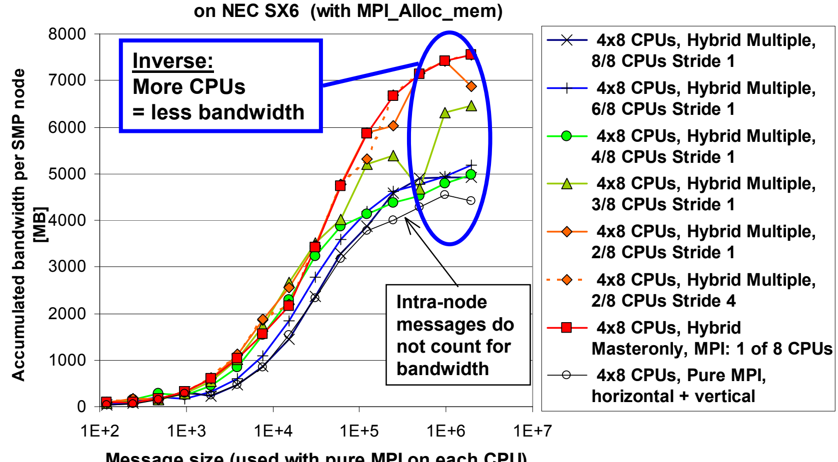

Hybrid Parallel Programming — Rolf Rabenseifner
Slide 17 / 32 — High Perf. Comp. Center, Univ. Stuttgart

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

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)

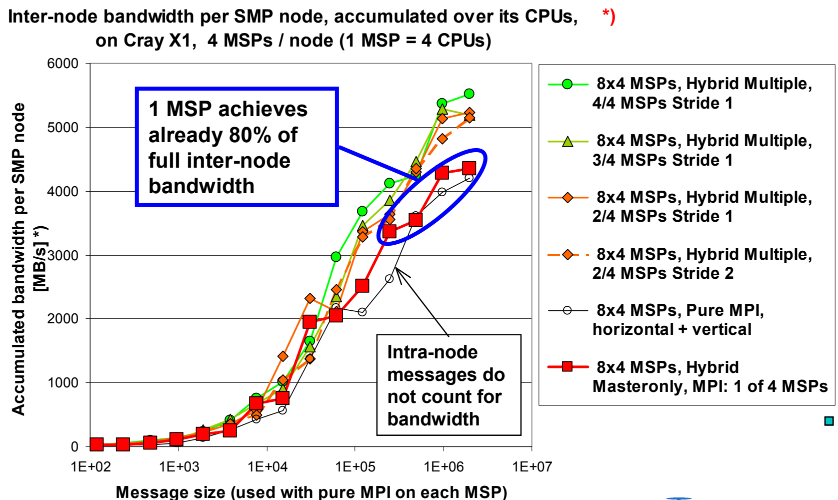



Hybrid Parallel Programming — Rolf Rabenseifner
Slide 18 / 32 — High Perf. Comp. Center, Univ. Stuttgart

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

Measurements: Thanks to Holger Berger, NEC.

## Multiple inter-node communication paths: Cray X1, used with 4 MSPs/node (preliminary results)



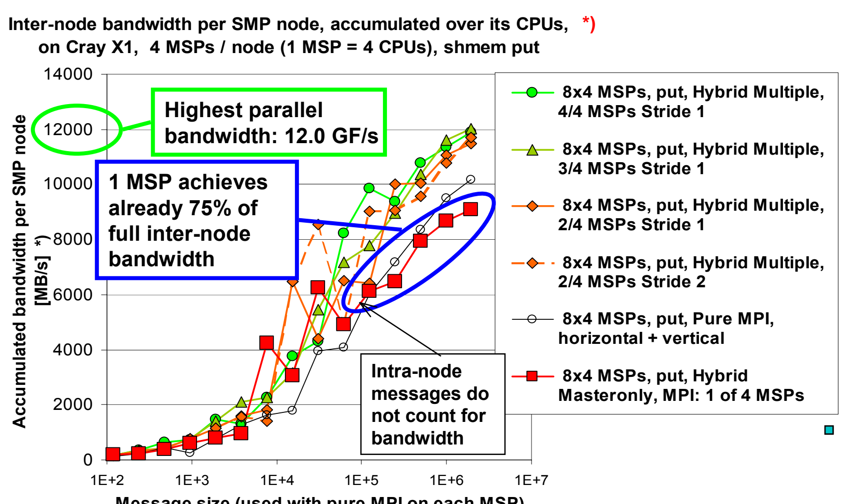

Hybrid Parallel Programming Rolf Rabenseifner
Slide 19 / 32 High Perf. Comp. Center, Univ. Stuttgart

HLRS

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

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

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



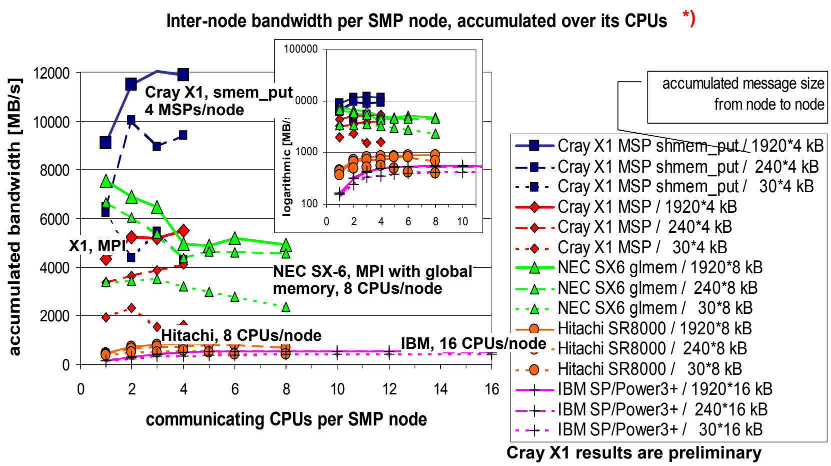

Hybrid Parallel Programming Rolf Rabenseifner
Slide 20 / 32 High Perf. Comp. Center, Univ. Stuttgart

HLRS

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

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

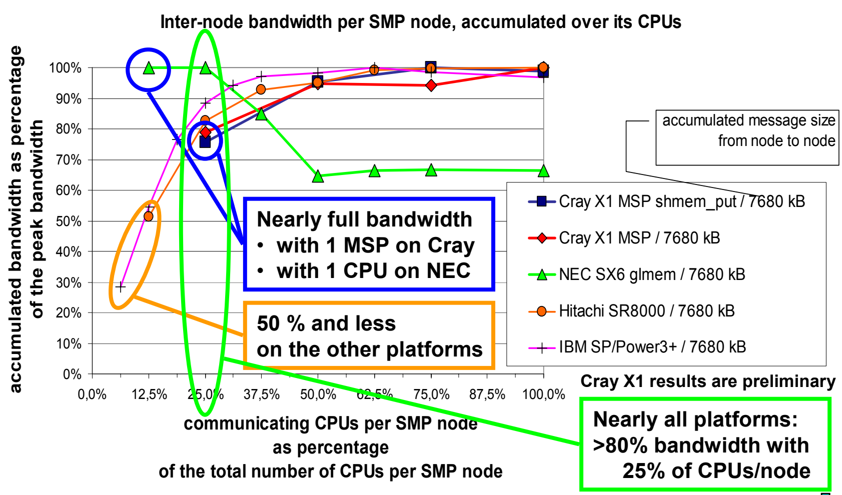
## Comparison



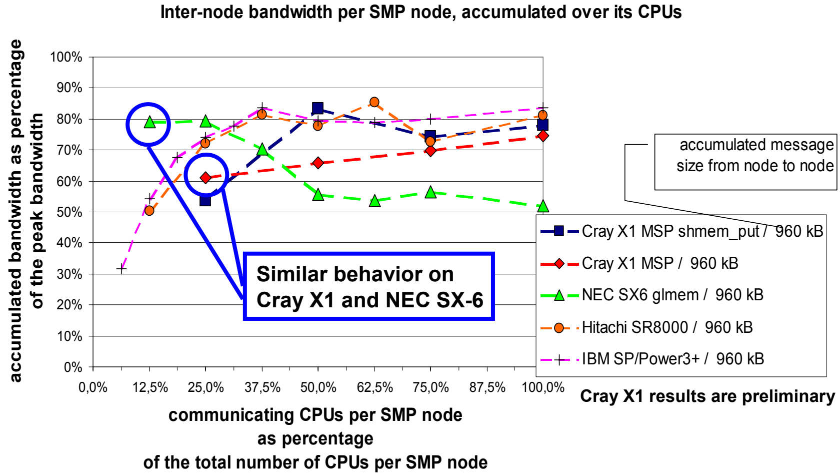



*) Bandwidth per node: totally transferred bytes on the inter-node network / wall clock time / number of nodes

## Comparison (as percentage of maximal bandwidth and #CPUs)



## Comparison (only 960 kB aggregated message size)






## 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

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

- 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

```
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++) {
          ....
   }
}
```



skipped

## 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 thread for communi.) | Multiple & Reserved (reserved threads for communic.) |
|---|---|

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

➔ **a good chance !!!** **... see next slide**

| Funneled with Full Load Balancing | Multiple with Full Load Balancing |
|---|---|

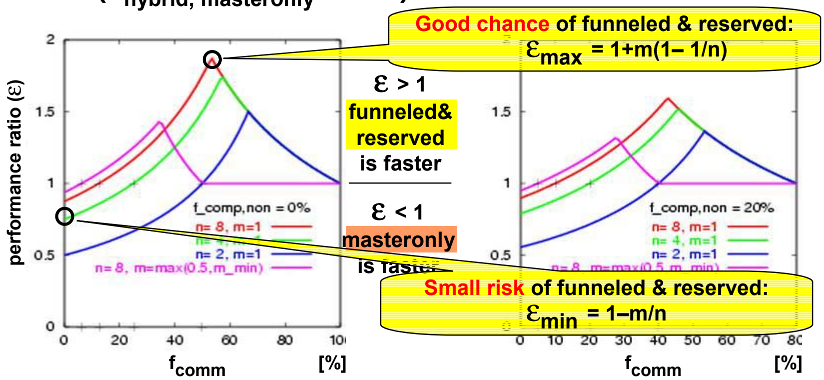
➔ **very hard to do !!!**

skipped

Masteronly ↔ funneled & reserved

## Performance ratio (theory)

- $\varepsilon = \left(\frac{T_{\text{hybrid, funneled\&reserved}}}{T_{\text{hybrid, masteronly}}}\right)^{-1}$




$T_{\text{hybrid, masteronly}} = (f_{comm} + f_{\text{comp, non-overlap}} + f_{\text{comp, overlap}})\, T_{\text{hybrid, masteronly}}$

n = # threads per SMP node, m = # reserved threads for MPI communication



See also Rabenseifner/Wellein, *The International Journal of High Performance Computing Applications*, Volume 17, No. 1, Spring 2003, © 2003 Sage Publications → www.hlrs.de/people/rabenseifner

## 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 peak CPU performance

| All values: aggregated over one SMP nodes. *) mess. size: 16 MB +) 2 MB | Master -only, inter-node [GB/s] | pure MPI, inter-node [GB/s] | Master-only bw / max. intra-node bw | pure MPI, intra-node [GB/s] | memo -ry band-width [GB/s] | Peak perfor-mance Gflop/s | max. inter-node bw / peak perf. B/Flop | nodes*CPUs |
|---|---|---|---|---|---|---|---|---|
| Cray X1, **shmem_put** **preliminary results** | 9.27 | **12.34** | **75 %** | 33.0 | 136 | **51.2** | **0.241** | 8 * 4 MSPs |
| Cray X1, MPI **preliminary results** | 4.52 | **5.52** | **82 %** | 19.5 | 136 | **51.2** | **0.108** | 8 * 4 MSPs |
| NEC SX-6 **global memory** | **7.56** | 4.98 | **100 %** | 78.7 93.7+) | 256 | **64** | **0.118** | 4 * 8 CPUs |
| NEC SX-5Be local memory | 2.27 | **2.50** a) | **91 %** | 35.1 | 512 | **64** | **0.039** | 2 *16 CPUs a) only with 8 |
| Hitachi SR8000 | 0.45 | **0.91** | **49 %** | 5.0 | 32 store 32 load | **8** | **0.114** | 8 * 8 CPUs |
| IBM SP Power3+ | 0.16 | **0.57+)** | **28 %** | 2.0 | 16 | **24** | **0.023** | 8 *16 CPUs |
| SGI Origin 3000 **preliminary results** | 0.10 | **0.30+)** | **33 %** | 0.39+) | 3.2 | **4.8** | **0.063** | 16 *4 CPUs |
| SUN-fire **(prelimi.)** | 0.15 | **0.85** | **18 %** | 1.68 | | | | 4 *24 CPUs |



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

## Acknowledgements


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




## Conclusions

- **Cray X1 with MSPs (1 node = 4 MSPs) and NEC SX-5/6:**
  - well designed hybrid MPI+OpenMP masteronly scheme
- **Cray X1 with SSPs (1 node = 16 SSPs)**
  - hybrid programming: 1 SSP cannot saturate inter-node bandwidth
- **Other platforms**
  - masteronly style cannot saturate inter-node bandwidth
- **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