

# Non-Blocking Collectives for MPI-2

– overlap at the highest level –

Torsten Höfler

Department of Computer Science  
Indiana University / Technical University of Chemnitz

Dresden University of Technology  
Center for Information Services and High Performance Computing (ZIH)  
Dresden, Germany  
17th October 2007



# Outline

- 1 Some Considerations about Interconnects
- 2 Why Non blocking Collectives?
- 3 LibNBC
- 4 And Applications?
- 5 Ongoing Efforts

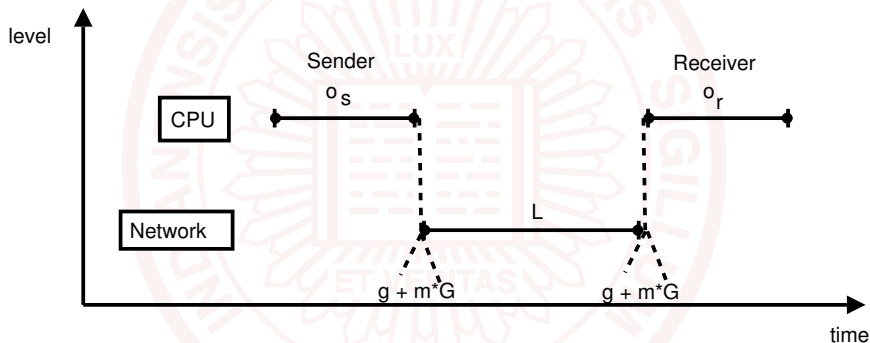


# Outline

- 1 **Some Considerations about Interconnects**
- 2 Why Non blocking Collectives?
- 3 LibNBC
- 4 And Applications?
- 5 Ongoing Efforts



# The LogGP Model



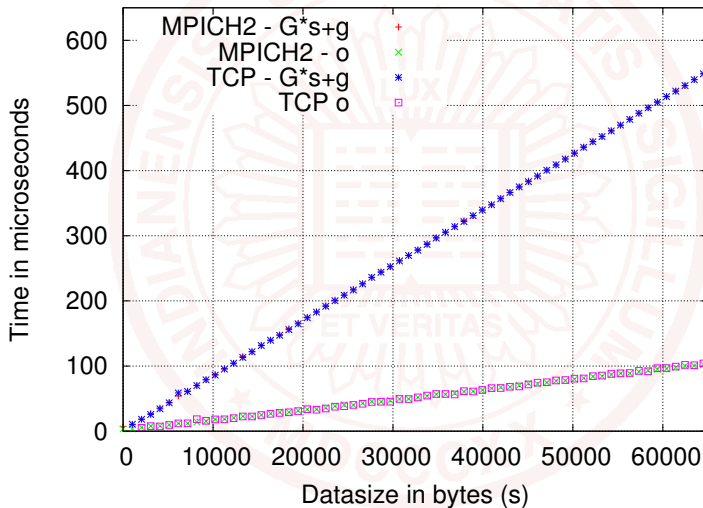
# Interconnect Trends

## Technology Change

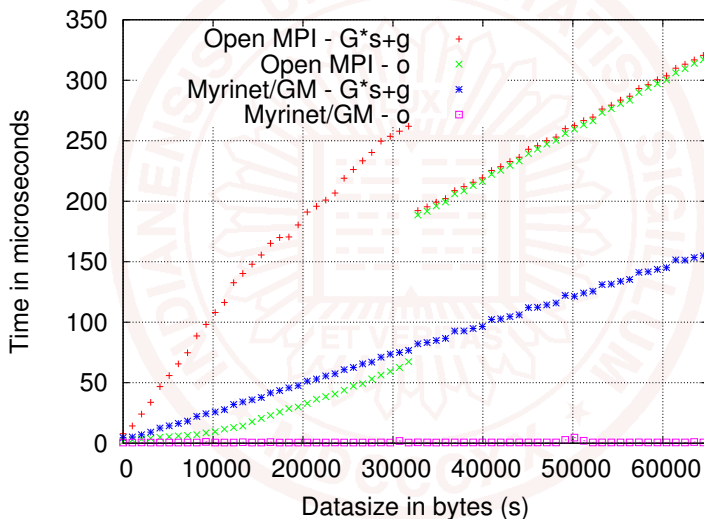
- modern interconnects offload communication to co-processors (Quadrics, InfiniBand, Myrinet)
- TCP/IP is optimized for lower host-overhead (e.g., Gamma)
- even Ethernet supports protocol offload
- $L + g + m \cdot G \gg o$

⇒ we prove our expectations with benchmarks of the user CPU overhead

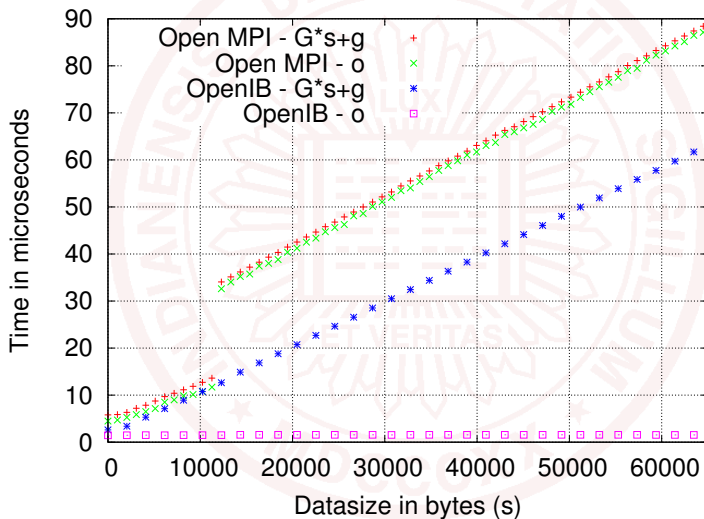
# LogGP Model Examples - TCP



# LogGP Model Examples - Myrinet/GM



# LogGP Model Examples - InfiniBand/OpenIB





# Literature

- [1] T. HOEFLER, A. LICHEI, AND W. REHM: *Low-Overhead LogGP Parameter Assessment for Modern Interconnection Networks. In Proceedings of the 21st IEEE International Parallel & Distributed Processing Symposium*
- [2] T. HOEFLER, J. SQUYRES, G. FAGG, G. BOSILCA, W. REHM AND A. LUMSDAINE: *A New Approach to MPI Collective Communication Implementations. In proceedings of the 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems*
- [3] T. HOEFLER, M. REINHARDT, F. MIETKE, T. MEHLAN, AND W. REHM: *Low Overhead Ethernet Communication for Open MPI on Linux Clusters. In Chemnitzer Informatik Berichte CSR-06, Nr. 06*



# Outline

- 1 Some Considerations about Interconnects
- 2 Why Non blocking Collectives?**
- 3 LibNBC
- 4 And Applications?
- 5 Ongoing Efforts



# Modelling the Benefits

## LogGP Models for Collective Operations

$$t_{barr} = (2o + L) \cdot \lceil \log_2 P \rceil$$

$$t_{allred} = 2 \cdot (2o + L + m \cdot G) \cdot \lceil \log_2 P \rceil + m \cdot \gamma \cdot \lceil \log_2 P \rceil$$

$$t_{bcast} = (2o + L + m \cdot G) \cdot \lceil \log_2 P \rceil$$

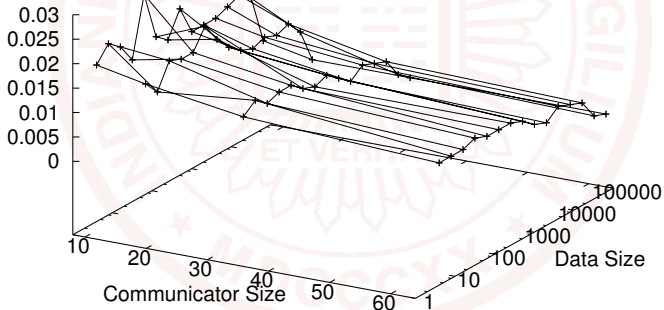
## Split into CPU and Network parts

$$\begin{array}{ll}
 t_{barr}^{CPU} = 2o \cdot \lceil \log_2 P \rceil & t_{barr}^{NET} = L \cdot \lceil \log_2 P \rceil \\
 t_{allred}^{CPU} = (4o + m \cdot \gamma) \cdot \lceil \log_2 P \rceil & t_{allred}^{NET} = 2 \cdot (L + m \cdot G) \cdot \lceil \log_2 P \rceil \\
 t_{bcast}^{CPU} = 2o \cdot \lceil \log_2 P \rceil & t_{bcast}^{NET} = (L + m \cdot G) \cdot \lceil \log_2 P \rceil
 \end{array}$$

# CPU Overhead Benchmarks

## LAM/MPI 7.1.2/TCP over GigE

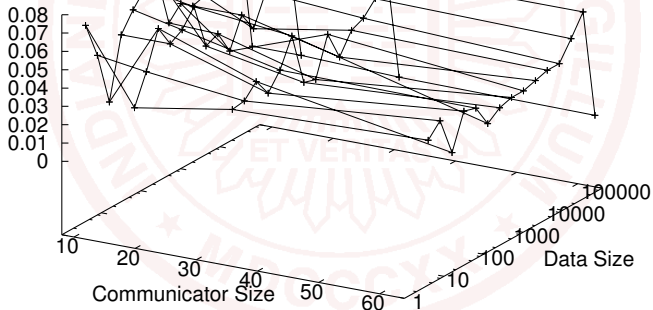
CPU Usage (share)



# CPU Overhead Benchmarks

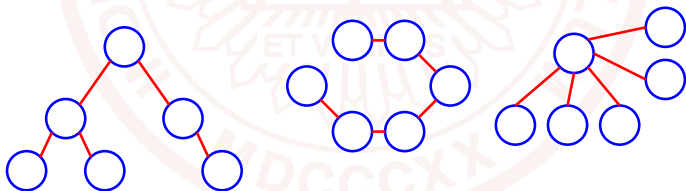
## MPICH2 1.0.3/TCP over GigE

CPU Usage (share)



# Why non blocking Collectives

- many collectives synchronize unnecessarily
- scale typically with  $O(\log_2 P)$  sends
- wasted CPU time:  $\log_2 P \cdot (L + G_{all})$ 
  - Fast Ethernet:  $L = 50-60$
  - Gigabit Ethernet:  $L = 15-20$
  - InfiniBand:  $L = 2-7$
  - $1\mu s \approx 4000$  FLOPs on a 2GHz Machine



# Isend/Irecv is there - Why Collectives?

- Gorlach, '04: "Send-Receive Considered Harmful"
- ⇔ Dijkstra, '68: "Go To Statement Considered Harmful"

## point to point

```
if ( rank == 0 ) then
  call MPI_SEND(...)
else
  call MPI_RECV(...)
end if
```

## vs. collective

```
call MPI_GATHER(...)
```

cmp. math libraries vs. loops



# Putting Everything Together

- non blocking collectives?
- JoD mentions "split collectives"
- example:
  - `MPI_Bcast_begin(...)`
  - `MPI_Bcast_end(...)`
- no nesting with other colls
- very limited
- not in the MPI-2 standard
- votes: 11 yes, 12 no, 2 abstain





# Performance Benefits

## overlap

- leverage hardware parallelism (e.g. InfiniBand™)
- overlap similar to non-blocking point-to-point

## pseudo synchronization

- avoidance of explicit pseudo synchronization
- limit the influence of OS noise

⇒ we analyze Barrier, Allreduce and Bcast



# Process Skew

- caused by OS interference or unbalanced application
- worse if processors are overloaded
- multiplies on big systems
- can cause dramatic performance decrease
- all nodes wait for the last

## Example

Petrini et. al. (2003) *"The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q"*



# Process Skew

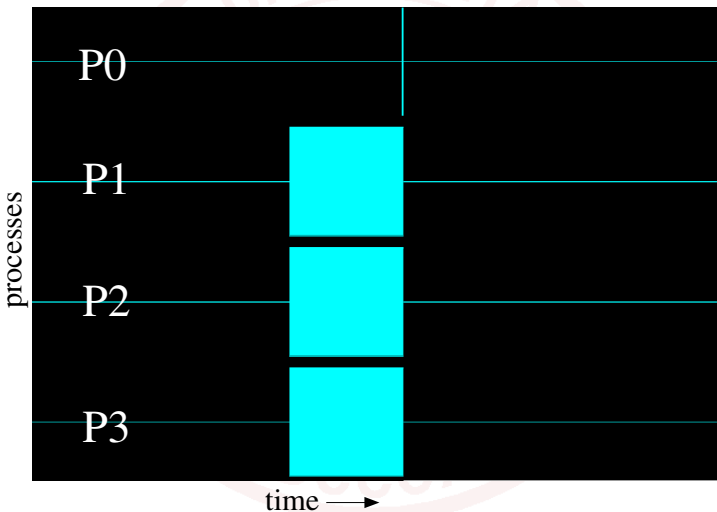
- caused by OS interference or unbalanced application
- worse if processors are overloaded
- multiplies on big systems
- can cause dramatic performance decrease
- all nodes wait for the last

## Example

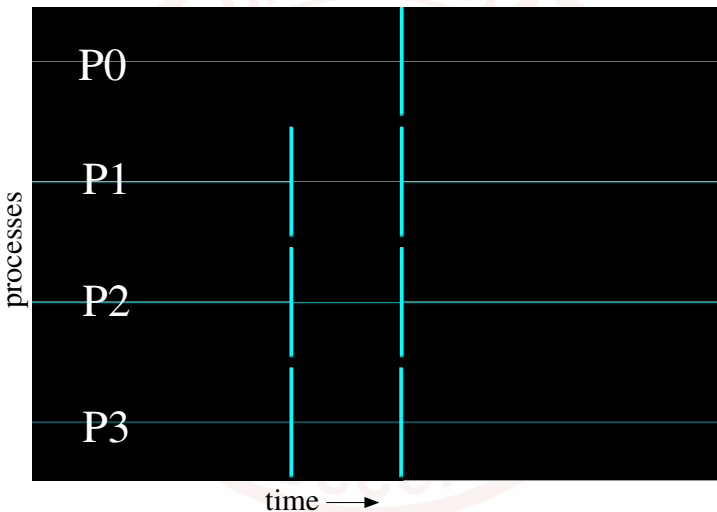
Petrini et. al. (2003) *"The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q"*



# MPI\_Bcast with P0 delayed - Jumpshot



# MPI\_Ibcast with P0 delayed + overlap - Jumpshot



# Literature

[4] T. HOEFLER, J. SQUYRES, W. REHM, AND A. LUMSDAINE: *A Case for Non-Blocking Collective Operations. In Frontiers of High Performance Computing and Networking, pages 155-164, Springer Berlin / Heidelberg, ISBN: 978-3-540-49860-5 Dec. 2006*

[5] T. HOEFLER, J. SQUYRES, G. BOSILCA, G. FAGG, A. LUMSDAINE, AND W. REHM: *Non-Blocking Collective Operations for MPI-2. Open Systems Lab, Indiana University. presented in Bloomington, IN, USA, School of Informatics, Aug. 2006*



# Outline

- 1 Some Considerations about Interconnects
- 2 Why Non blocking Collectives?
- 3 LibNBC**
- 4 And Applications?
- 5 Ongoing Efforts



# Non-Blocking Collectives - Interface

- extension to MPI-2
- "mixture" between non-blocking ptp and collectives
- uses MPI\_Requests and MPI\_Test/MPI\_Wait

```
MPI_Ibcast(buf1, p, MPI_INT, 0, MPI_COMM_WORLD, &req);  
MPI_Wait(&req);
```

## Proposal

Hoefler et. al. (2006): *"Non-Blocking Collective Operations for MPI-2"*





## Non-Blocking Collectives - Interface

- extension to MPI-2
- "mixture" between non-blocking ptp and collectives
- uses MPI\_Requests and MPI\_Test/MPI\_Wait

```
MPI_Ibcast(buf1, p, MPI_INT, 0, MPI_COMM_WORLD, &req);  
MPI_Wait(&req);
```

### Proposal

Hoefler et. al. (2006): *"Non-Blocking Collective Operations for MPI-2"*



# Non-Blocking Collectives - Implementation

- implementation available with LibNBC
- written in ANSI-C and uses only MPI-1
- central element: collective schedule
- a coll-algorithm can be represented as a schedule
- trivial addition of new algorithms

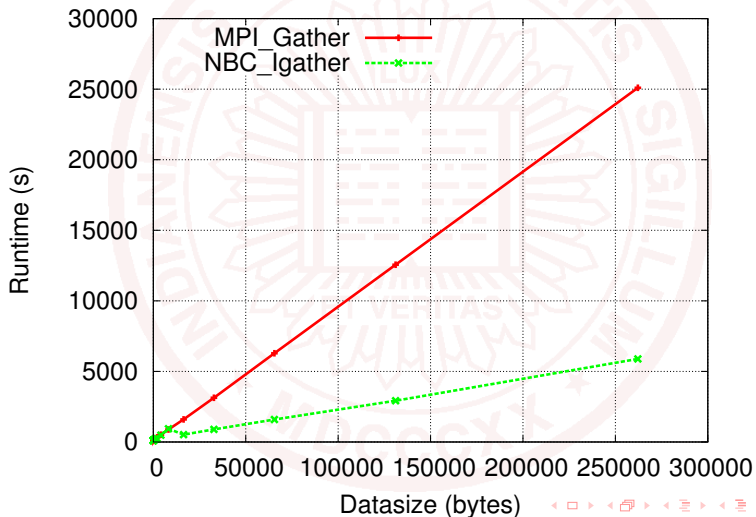
Example: dissemination barrier, 4 nodes, node 0:

send to 1	recv from 3	end	send to 2	recv from 2	end
-----------	-------------	-----	-----------	-------------	-----

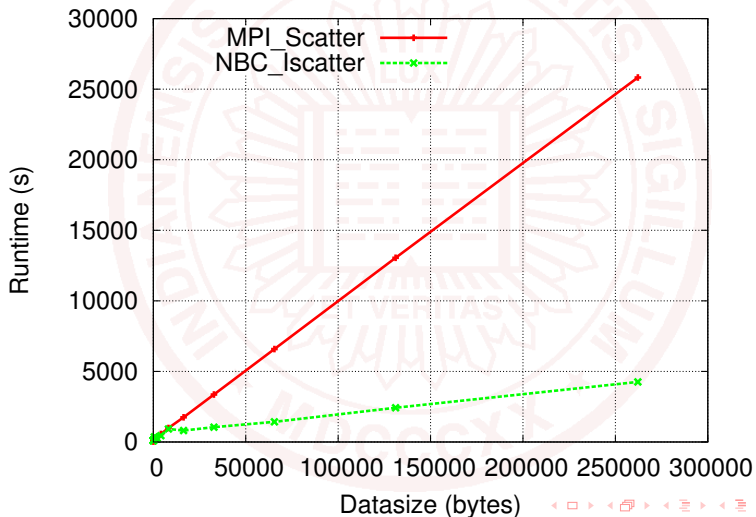
LibNBC download: <http://www.unixer.de/NBC>



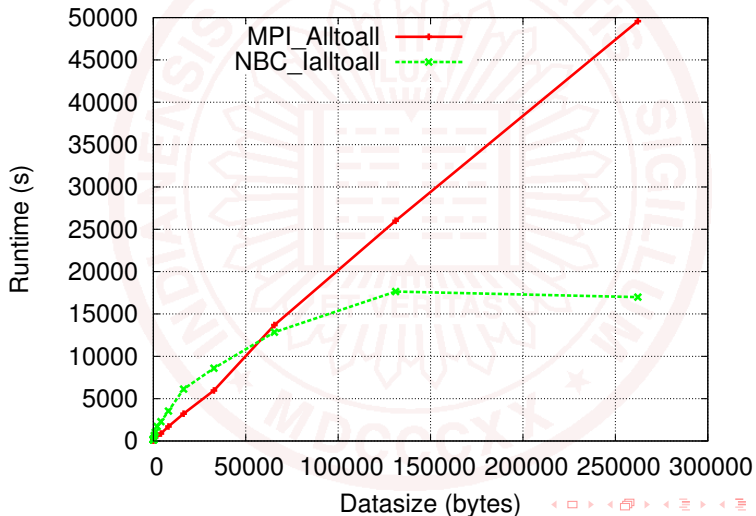
# Overhead Benchmarks - Gather with InfiniBand/MVAPICH on 64 nodes



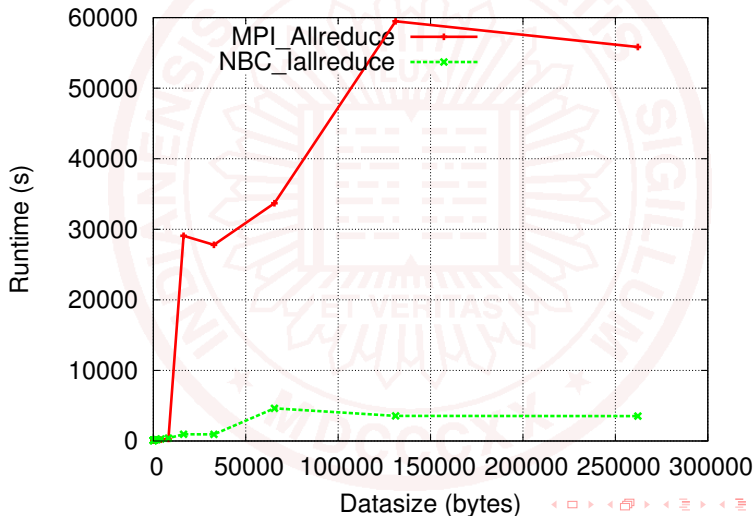
# Overhead Benchmarks - Scatter with InfiniBand/MVAPICH on 64 nodes



# Overhead Benchmarks - Alltoall with InfiniBand/MVAPICH on 64 nodes



# Overhead Benchmarks - Allreduce with InfiniBand/MVAPICH on 64 nodes



# Literature

- [6] T. HOEFLER, A. LUMSDAINE AND W. REHM: *Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI*. Accepted for publication at the Supercomputing 2007 (SC07)
- [7] T. HOEFLER, P. KAMBADUR, R. L. GRAHAM, G. SHIPMAN AND A. LUMSDAINE: *A Case for Standard Non-Blocking Collective Operations*. In *Proceedings of the 14th European PVM/MPI User's Group Meeting 2007*
- [8] T. HOEFLER AND A. LUMSDAINE: *Design, Implementation, and Usage of LibNBC*. Open Systems Lab, Indiana University. presented in Bloomington, IN, USA, School of Informatics, Aug. 2006



# Outline

- 1 Some Considerations about Interconnects
- 2 Why Non blocking Collectives?
- 3 LibNBC
- 4 And Applications?**
- 5 Ongoing Efforts





# Linear Solvers - Domain Decomposition

## First Example

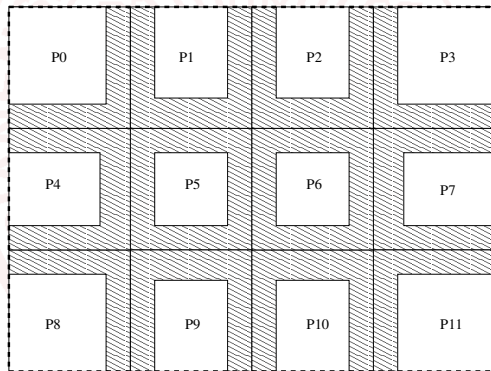
### Naturally Independent Computation - 3D Poisson Solver

- iterative linear solvers are used in many scientific kernels
- often used operation is vector-matrix-multiply
- matrix is domain-decomposed (e.g., 3D)
- only outer (border) elements need to be communicated
- can be overlapped



# Domain Decomposition

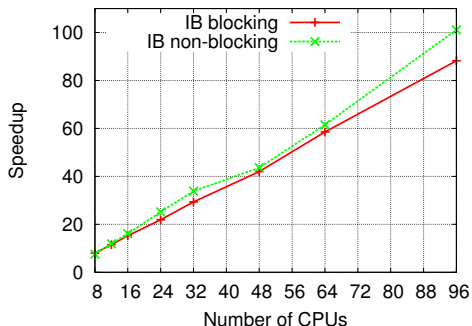
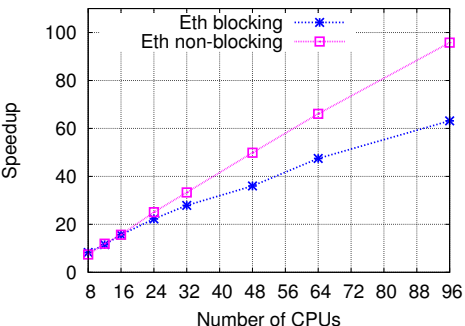
- nearest neighbor communication
- can be implemented with MPI\_Alltoallv



□ Process-local data    □ 2D Domain  
▨ Halo-data



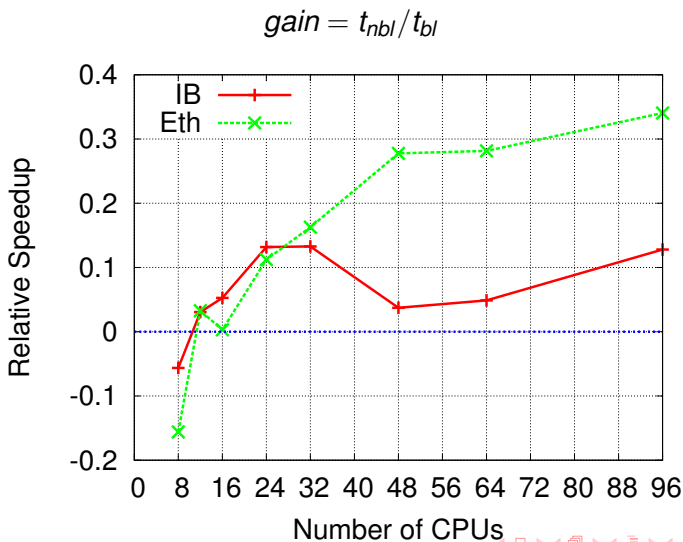
# Parallel Speedup (Best Case)



- Cluster: 128 2 GHz Opteron 246 nodes
- Interconnect: Gigabit Ethernet, InfiniBand™
- System size 800x800x800 (1 node  $\approx$  5300s)



# Parallel Gain with Non-Blocking Communication



# Parallel Data Compression

## Second Example

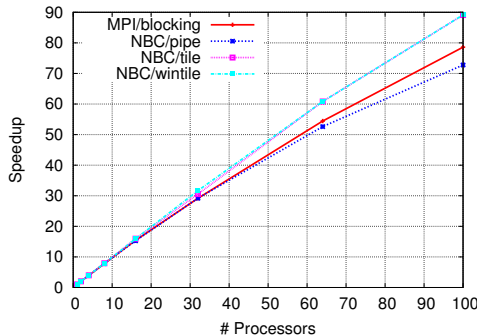
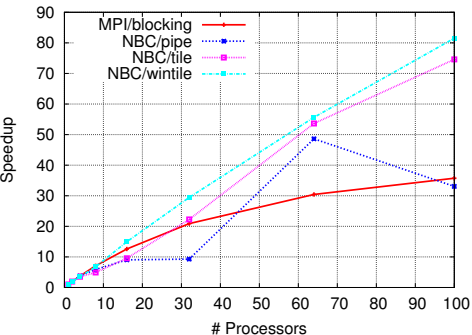
### Data Parallel Loops - Parallel Compression

automatic transformations (C++ templates), typical loop structure:

```
for (i=0; i < N/P; i++) {  
    compute(i);  
}  
comm(N/P);
```



# Parallel Speedup (Best Case)

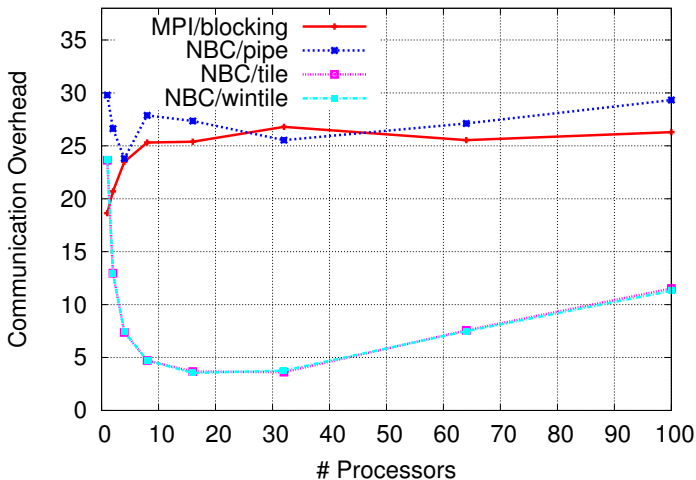


- Cluster: 64 2 GHz Opteron 246 nodes
- Interconnect: Gigabit Ethernet, InfiniBand™
- System size 64\*50 MB



# Communication Overhead

## MVAPICH 0.9.4



# Parallel 3d Fast Fourier Transform

## Third Example

Specialized Algorithms - A parallel 3d-FFT with overlap

Specialized design to achieve the highest overlap. Less cache-friendly!





# Non-blocking Collectives - 3D-FFT

## Derivation from “normal” implementation

- distribution identical to “normal” 3D-FFT
- first FFT in z direction and index-swap identical

## Design Goals to Minimize Communication Overhead

- start communication as early as possible
- achieve maximum overlap time

## Solution

- start MPI\_lalltoall as soon as first xz-plane is ready
- calculate next xz-plane
- start next communication accordingly ...
- collect multiple xz-planes (tile factor)



# Non-blocking Collectives - 3D-FFT

## Derivation from “normal” implementation

- distribution identical to “normal” 3D-FFT
- first FFT in z direction and index-swap identical

## Design Goals to Minimize Communication Overhead

- start communication as early as possible
- achieve maximum overlap time

## Solution

- start `MPI_lalltoall` as soon as first xz-plane is ready
- calculate next xz-plane
- start next communication accordingly ...
- collect multiple xz-planes (tile factor)



# Non-blocking Collectives - 3D-FFT

## Derivation from “normal” implementation

- distribution identical to “normal” 3D-FFT
- first FFT in z direction and index-swap identical

## Design Goals to Minimize Communication Overhead

- start communication as early as possible
- achieve maximum overlap time

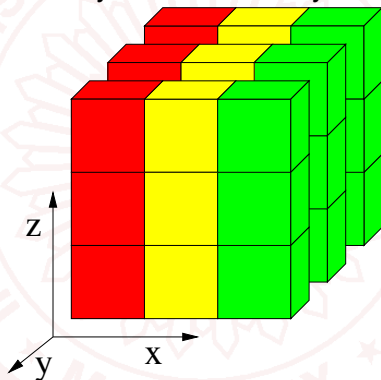
## Solution

- start MPI\_lalltoall as soon as first xz-plane is ready
- calculate next xz-plane
- start next communication accordingly ...
- collect multiple xz-planes (tile factor)



# Transformation in z Direction

Data already transformed in y direction

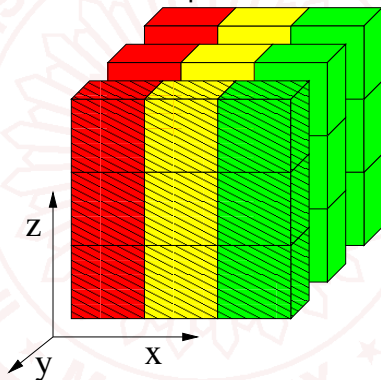


1 block = 1 double value (3x3x3 grid)



# Transformation in z Direction

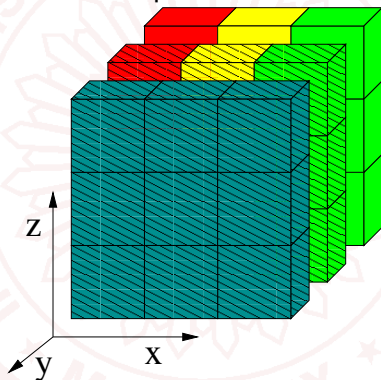
Transform first xz plane in z direction



pattern means that data was transformed in y and z direction

## Transformation z Direction

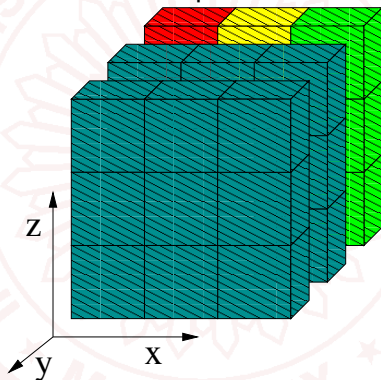
start MPI\_lalltoall of first xz plane and transform second plane



cyan color means that data is communicated in the background

## Transformation in z Direction

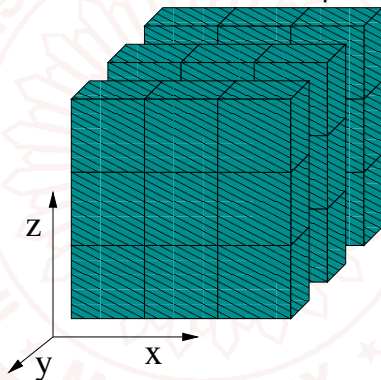
start MPI\_lalltoall of second xz plane and transform third plane



data of two planes is not accessible due to communication

# Transformation in x Direction

start communication of the third plane and ...

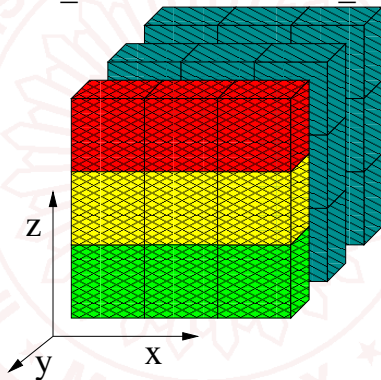


we need the first xz plane to go on ...



# Transformation in x Direction

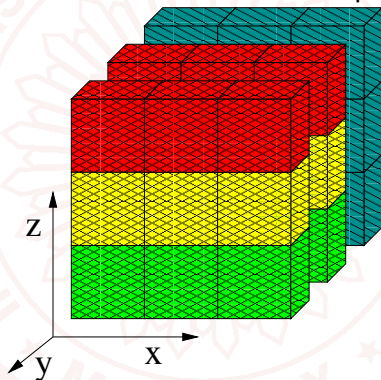
... so MPI\_Wait for the first MPI\_Ialltoall!



and transform first plane (new pattern means xyz transformed)

# Transformation in x Direction

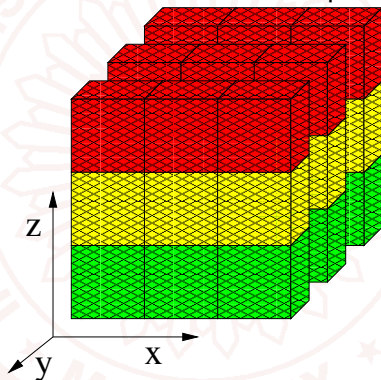
Wait and transform second xz plane



first plane's data could be accessed for next operation

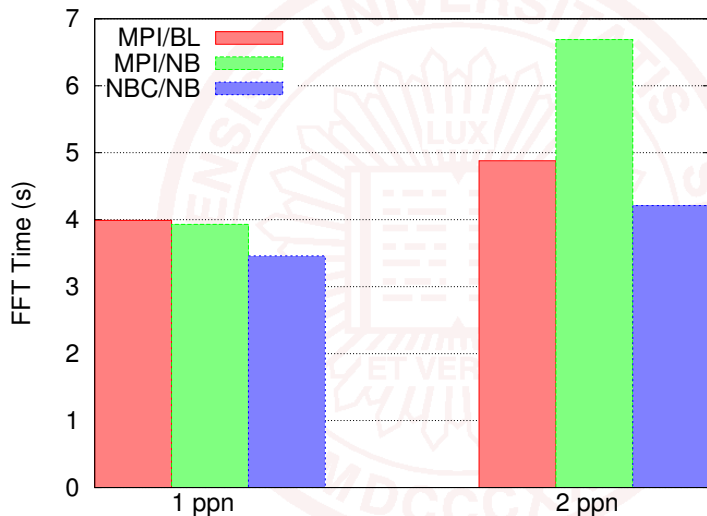
# Transformation in x Direction

wait and transform last xz plane



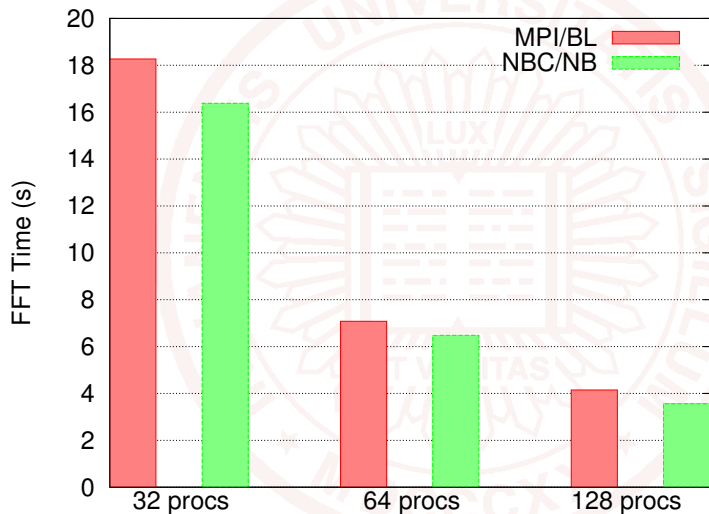
done! → 1 complete 1D-FFT overlaps a communication

# 1024<sup>3</sup> 3d-FFT over InfiniBand



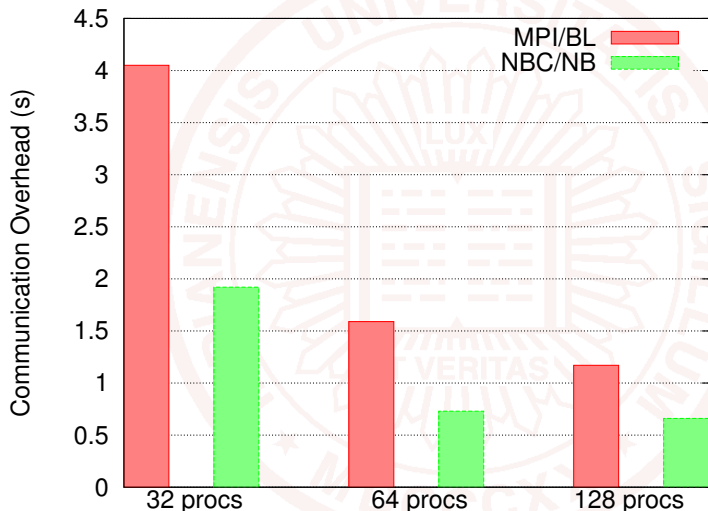
- P=128, "Coyote"@LANL - 128/64 dual socket 2.6GHz Opteron nodes

# 1024<sup>3</sup> 3d-FFT on the XT4



- “Jaguar”@ORNL - Cray XT4, dual socket dual core 2.6GHz Opteron

# 1024<sup>3</sup> 3d-FFT on the XT4 (Communication Overhead)



- “Jaguar”@ORNL - Cray XT4, dual socket dual core 2.6GHz Opteron

# Literature

- [9] T. HOEFLER P. GOTTSCHLING, W. REHM AND A. LUMSDAINE:  
*Optimizing a Conjugate Gradient Solver with Non-Blocking Collective Operations. Elsevier Journal of Parallel Computing (PARCO). Vol 33, Nr. 9, pages 624-633*
- [10] T. HOEFLER, P. GOTTSCHLING AND A. LUMSDAINE:  
*Transformations for enabling non-blocking collective communication in high-performance applications. Under submission (ask me for a copy)*



# Outline

- 1 Some Considerations about Interconnects
- 2 Why Non blocking Collectives?
- 3 LibNBC
- 4 And Applications?
- 5 **Ongoing Efforts**





# Ongoing Work

## LibNBC

- distribute as part of Open MPI 1.3
- optimized collectives

## Collective Communication

- optimized collectives for InfiniBand™
- using special hardware support

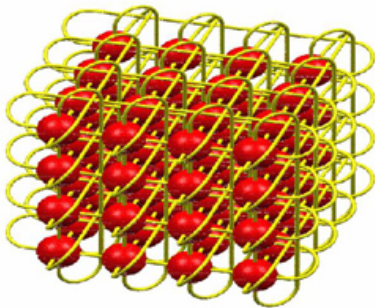
## Network Modelling

- refined LogGP model parametrization
- modelling of collective algorithms

# Discussion

# THE END

Questions?



Thank you for your attention!