Non-Blocking Collectives for MPI

– overlap at the highest level –

Torsten Höfler

Open Systems Lab
Indiana University
Bloomington, IN, USA

Institut für Wissenschaftliches Rechnen
Technische Universität Dresden
Dresden, Germany

27th May 2008

Outline

Outline

Fundamental Assumptions (I)

We need more powerful machines!

- Solutions for real-world scientific problems need huge processing power (more than available)

Capabilities of single PEs have fundamental limits

- The scaling/frequency race is currently stagnating
- Moore's law is still valid (number of transistors/chip)
- Instruction level parallelism is limited (pipelining, VLIW, multi-scalar)

Explicit parallelism seems to be the only solution

- Single chips and transistors get cheaper
- Implicit transistor use (ILP, branch prediction) have their limits

Fundamental Assumptions (II)

Parallelism requires communication

- Local or even global data-dependencies exist
- Off-chip communication becomes necessary
- Bridges a physical distance (many PEs)

Communication latency is limited

- It's widely accepted that the speed of light limits data-transmission
- Example: minimal 0-byte latency for $1m \approx 3.3ns \approx 13$ cycles on a $4GHz$ PE

Bandwidth can hide latency only partially

- Bandwidth is limited (physical constraints)
- The problem of "scaling out" (especially iterative solvers)

Assumptions about Parallel Program Optimization

Collective Operations

- Collective Operations (COs) are an optimization tool
- CO performance influences application performance
- optimized implementation and analysis of CO is non-trivial

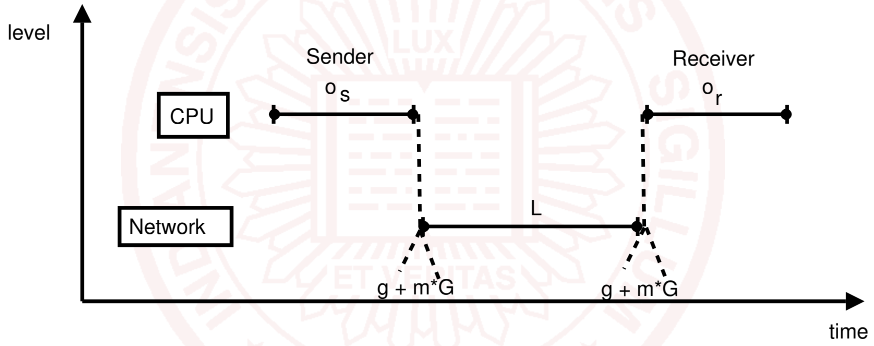
Hardware Parallelism

- More PEs handle more tasks in parallel
- Transistors/PEs take over communication processing
- Communication and computation could run simultaneously

Overlap of Communication and Computation

- Overlap can hide latency
- Improves application performance

The LogGP Model



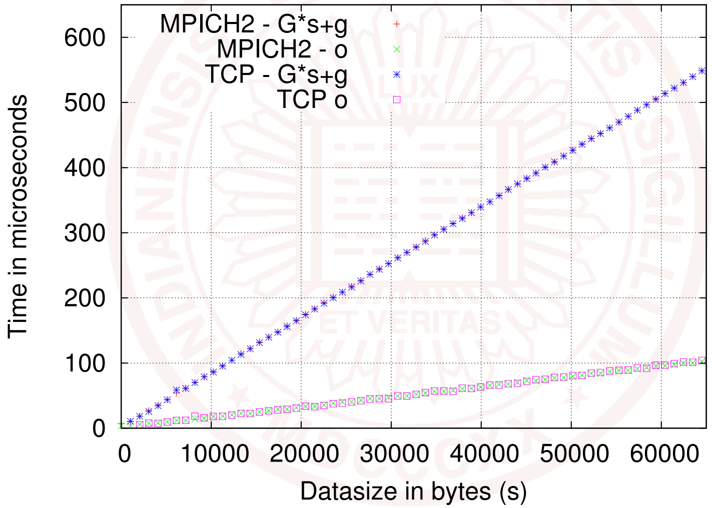
Interconnect Trends

Technology Change

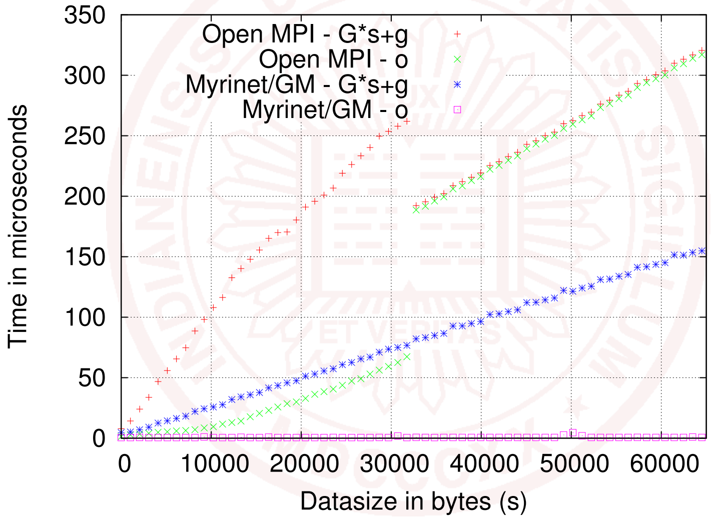
- modern interconnects offload communication to co-processors (Quadrics, InfiniBand, Myrinet)
- TCP/IP is optimized for lower host-overhead (e.g., Gamma)
- even legacy Ethernet supports protocol offload
- $L + g + m \cdot G >> o$

$\Rightarrow$ we prove our expectations with benchmarks of the user CPU overhead

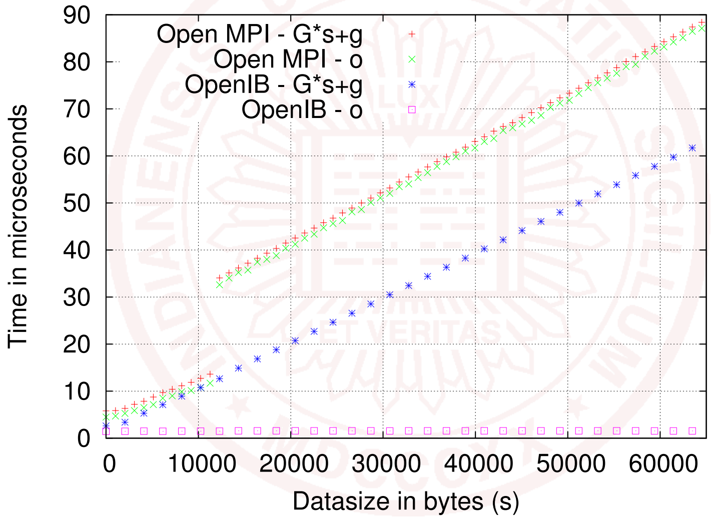
LogGP Model Examples - TCP



LogGP Model Examples - Myrinet/GM



LogGP Model Examples - InfiniBand/OpenIB



Outline

Isend/Irecv is there - Why Collectives?

- Gorlach, ’04: “Send-Receive Considered Harmful”
- $\Leftrightarrow$ 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

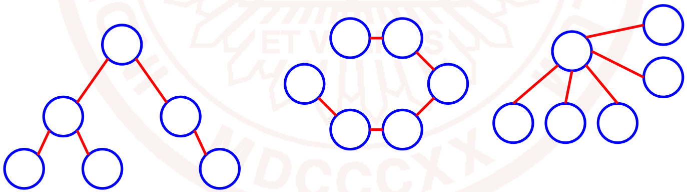
Sparse Collectives

But my algorithm only needs nearest neighbor communication!?
$\Rightarrow$ this is a collective too, just sparse (cf. sparse BLAS)

- sparse communication with neighbors on process topologies
- graph topology makes it generic
- many optimization possibilities (process placing, overlap, message scheduling/forwarding)
- easy to implement
- not part of MPI but fully implemented in LibNBC and proposed to the MPI Forum

Why non blocking Collectives

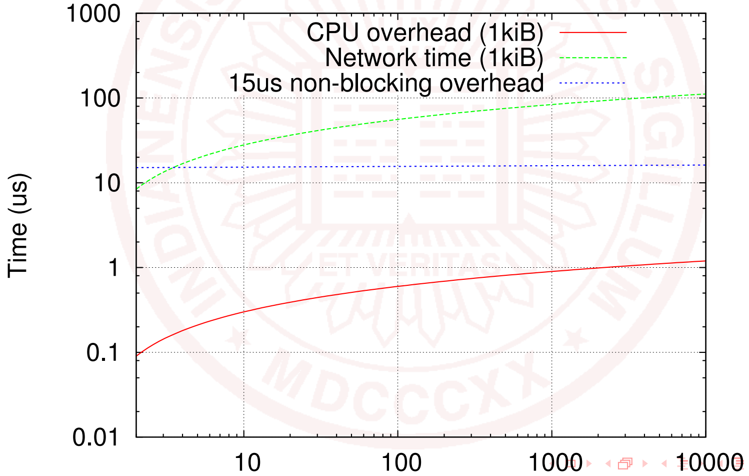
- scale typically with $O(log_2 P)$ sends
- wasted CPU time: $log_2 P \cdot (L + G_{all})$
  - Fast Ethernet: L = 50-60
  - Gigabit Ethernet: L = 15-20
  - InfiniBand: L = 2-7
  - $1 \mu s \approx$ 6000 FLOP on a 3GHz Machine
- ... and many collectives synchronize unneccessarily



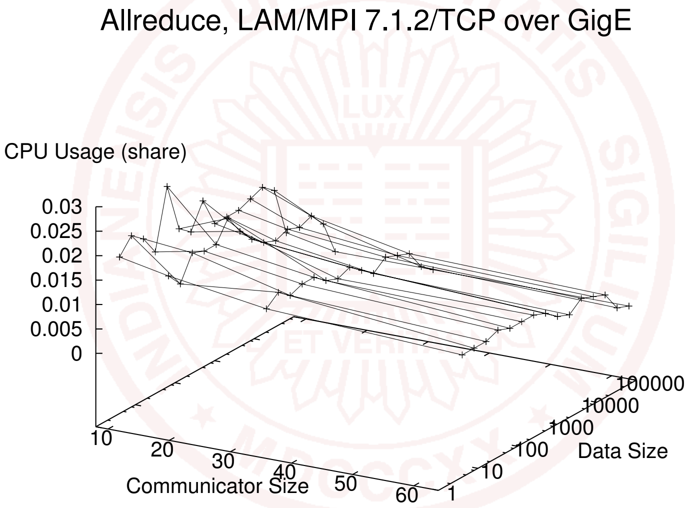
Modelling the Benefits

LogGP Model for Allreduce:

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



CPU Overhead Benchmarks



Performance Benefits

overlap

- leverage hardware parallelism (e.g. InfiniBand$^{\text{TM}}$)
- overlap similar to non-blocking point-to-point

pseudo synchronization

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

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”*

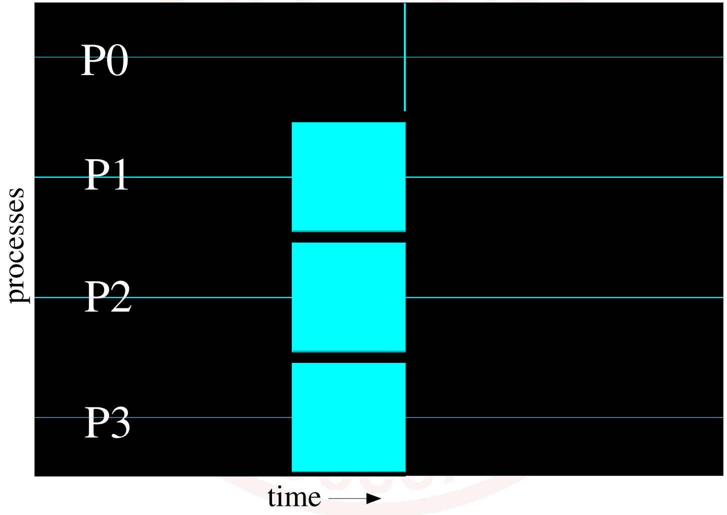
Process Skew

- 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

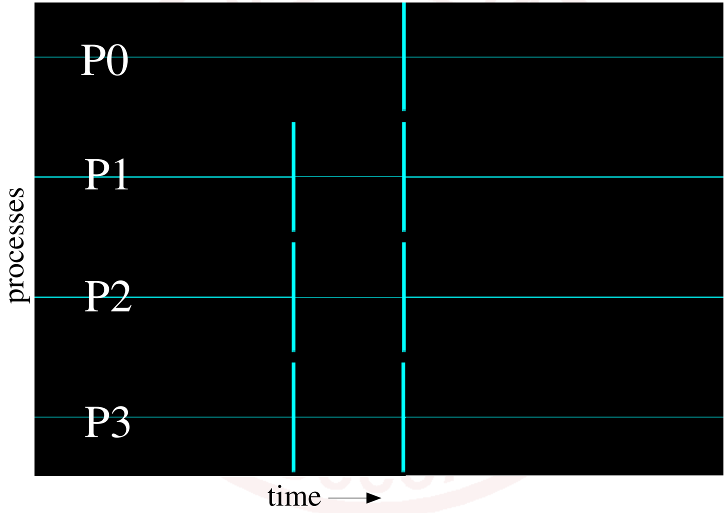
Example

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



Outline

Non-Blocking Collectives - Interface

- extension to MPI-2
- ”mixture” between non-blocking ptp and collectives
- uses MPI_Requests and MPI_Test/MPI_Wait

Interface

```
MPI_Ibcast(buf, count, MPI_INT, 0, comm, &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

Interface

```
MPI_Ibcast(buf, count, MPI_INT, 0, comm, &req);
MPI_Wait(&req);
```

Proposal

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

Non-Blocking Collectives - Implementation

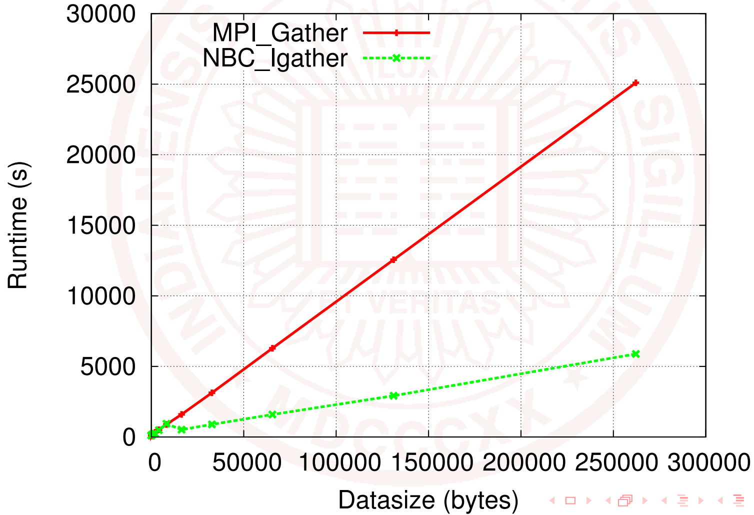
- implementation available with LibNBC
- written in ANSI-C and uses only MPI-1
- central element: collective schedule
- a coll-algorithm can be represented as a schedule
- trivial addition of new algorithms

Example: dissemination barrier, 4 nodes, node 0:

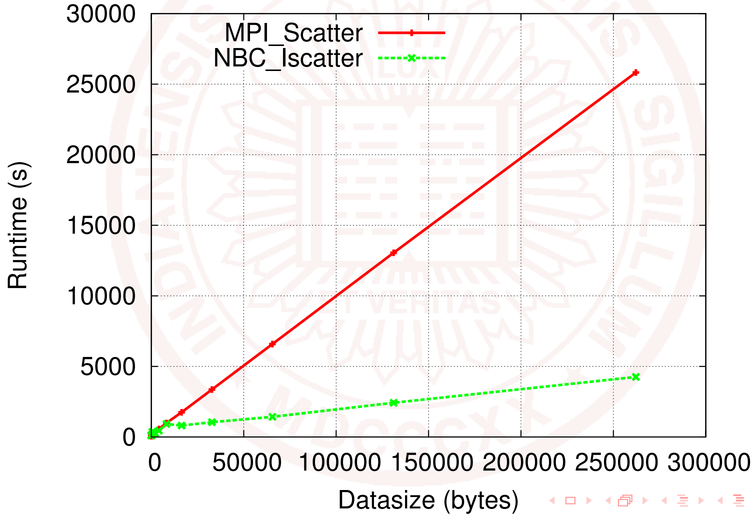
send to 1	recv from 3	end	send to 2	recv from 2	end

LibNBC download: `http://www.unixer.de/NBC`

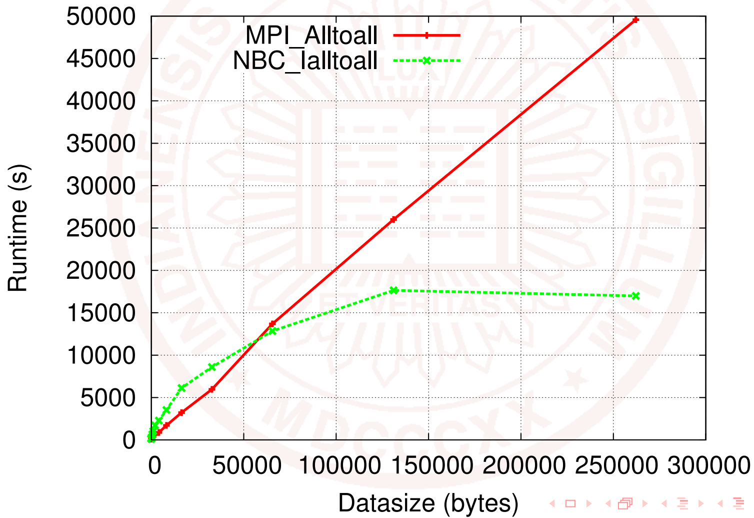
Overhead Benchmarks - Gather with InfiniBand/MVAPICH on 64 nodes



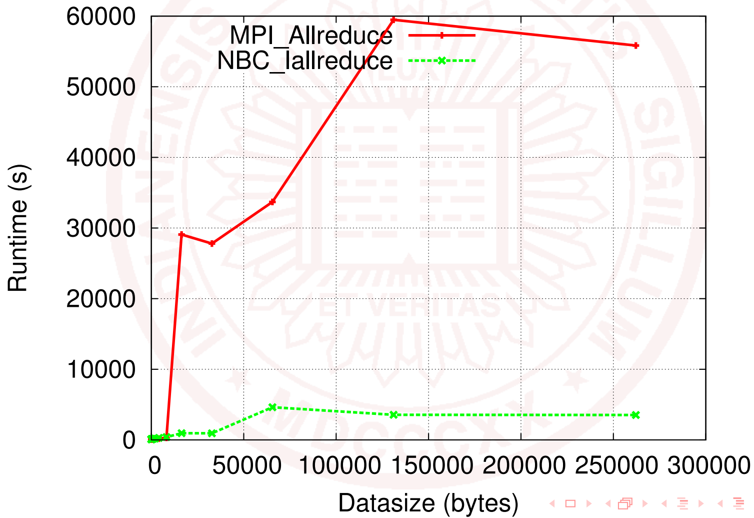
Overhead Benchmarks - Scatter with InfiniBand/MVAPICH on 64 nodes



Overhead Benchmarks - Alltoall with InfiniBand/MVAPICH on 64 nodes



Overhead Benchmarks - Allreduce with InfiniBand/MVAPICH on 64 nodes



Outline

Linear Solvers - Domain Decomposition

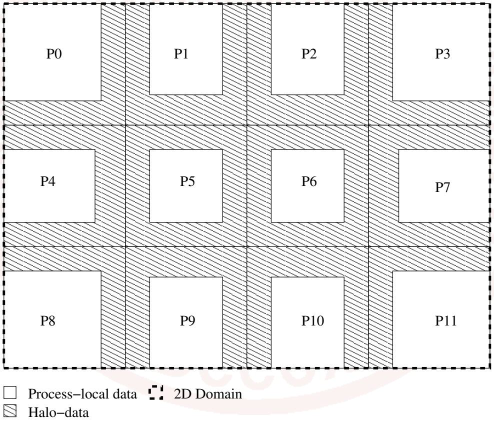
First Example

Naturally Independent Computation - 3D Poisson Solver

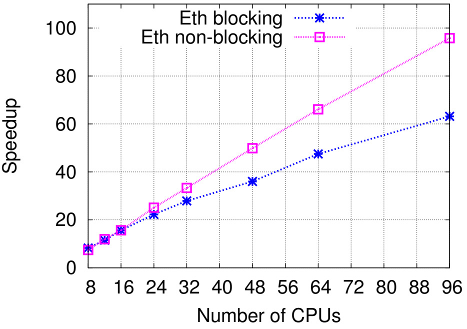
- iterative linear solvers are used in many scientific kernels
- often used operation is vector-matrix-multiply
- matrix is domain-decomposed (e.g., 3D)
- only outer (border) elements need to be communicated
- can be overlapped

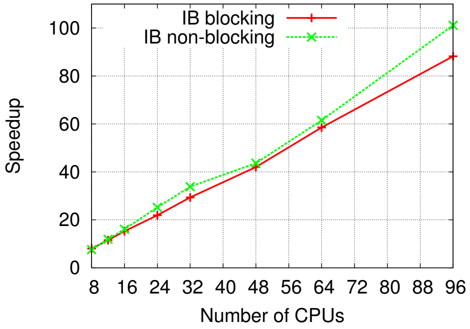
Domain Decomposition

- nearest neighbor communication
- can be implemented with MPI_Alltoallv or sparse collectives



Parallel Speedup (Best Case)





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

Parallel Data Compression

Second Example

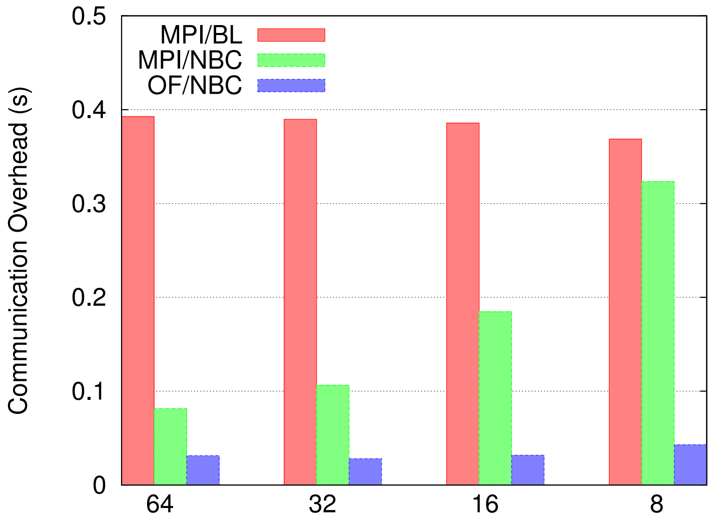
Data Parallel Loops - Parallel Compression

Automatic transformations (C++ templates)

typical loop structure:

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

Parallel Compression Communication Overhead



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_Ialltoall 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_Ialltoall 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

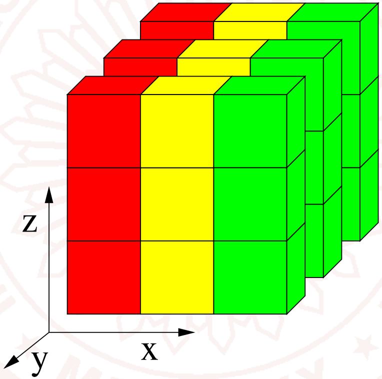
- start communication as early as possible
- achieve maximum overlap time

Solution

- start MPI_Ialltoall 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

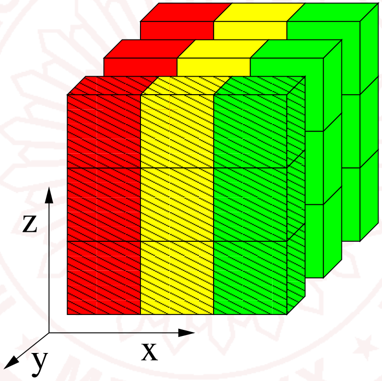
Data already transformed in y direction



1 block = 1 double value (3x3x3 grid)

Transformation in z Direction

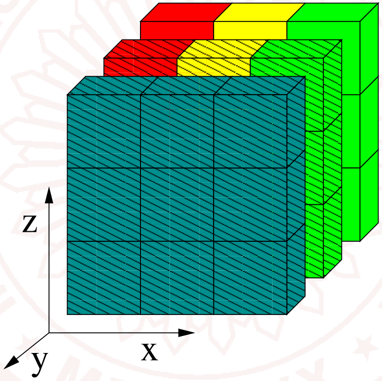
Transform first xz plane in z direction



pattern means that data was transformed in y and z direction

Transformation z Direction

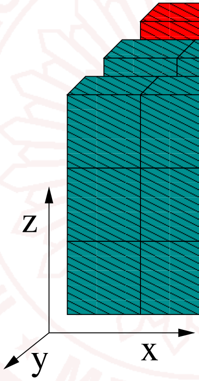
start MPI_Ialltoall of first xz plane and transform second plane



cyan color means that data is communicated in the background

Transformation in z Direction

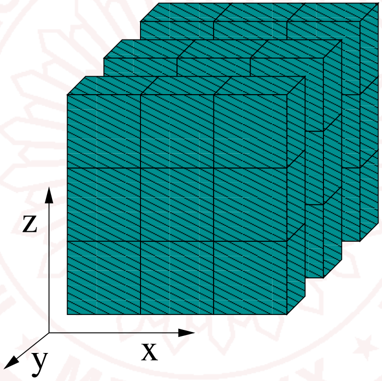
start MPI_Ialltoall of second xz plane and transform third plane



data of two planes is not accessible due to communication

Transformation in x Direction

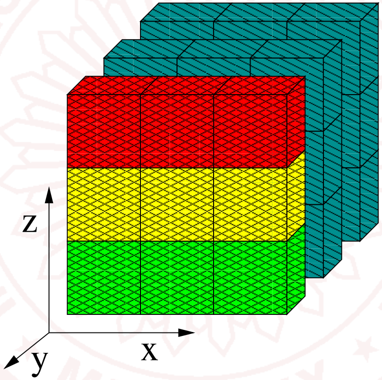
start communication of the third plane and ...



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

Transformation in x Direction

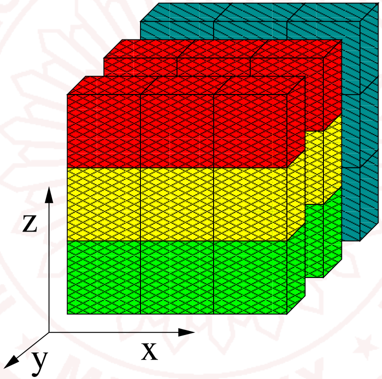
... so MPI_Wait for the first MPI_Ialltoall!



and transform first plane (new pattern means xyz transformed)

Transformation in x Direction

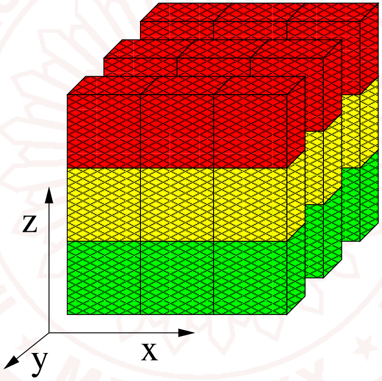
Wait and transform second xz plane



first plane's data could be accessed for next operation

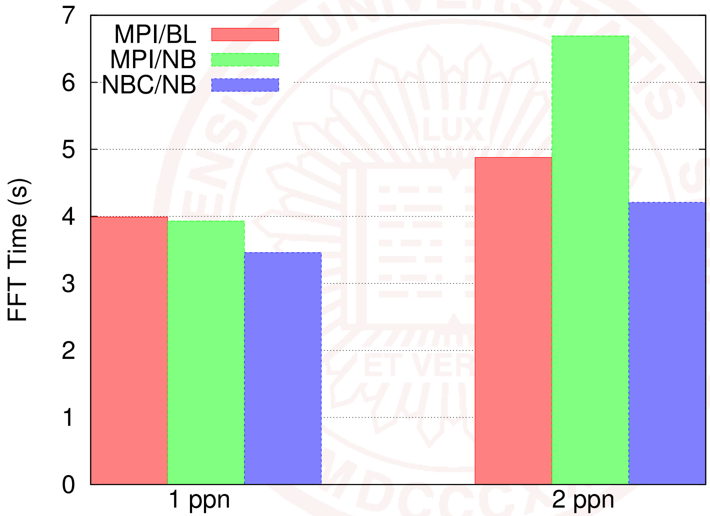
Transformation in x Direction

wait and transform last xz plane



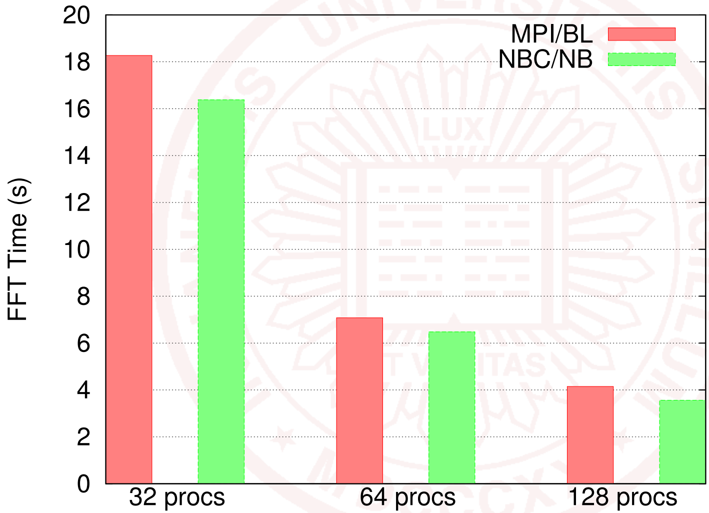
done! → 1 complete 1D-FFT overlaps a communication

1024^3 3d-FFT over InfiniBand



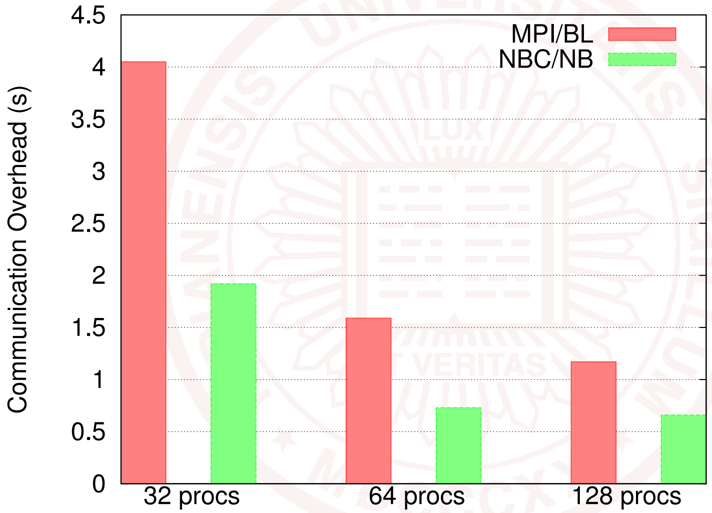
- P=128, "Coyote"@LANL - 128/64 dual socket 2.6GHz Opteron node

1024^3 3d-FFT on the XT4



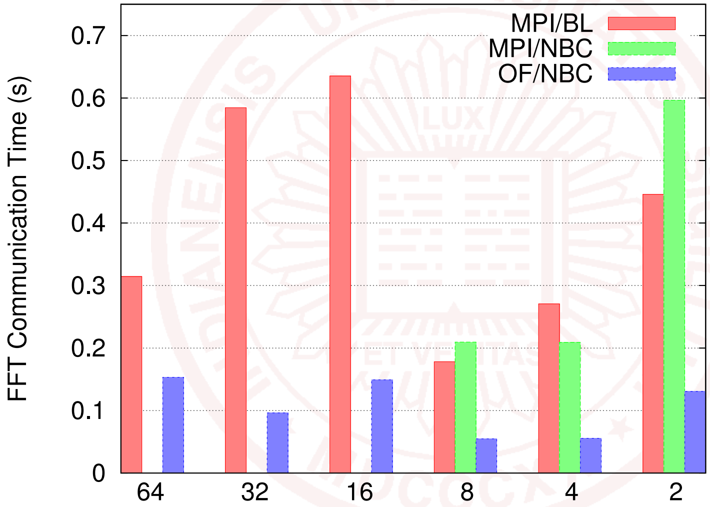
- “Jaguar”@ORNL - Cray XT4, dual socket dual core 2.6GHz Opteron

1024^3 3d-FFT on the XT4 (Communication Overhead)



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

640^3 3d-FFT InfiniBand (Communication Overhead)



- “Odin”@IU - dual socket dual core 2.0GHz Opteron InfiniBand

Outline

Ongoing Work

LibNBC

- analysis of multi-threaded implementation
- optimized collectives

Collective Communication

- optimized collectives for InfiniBand^{TM} (topology-aware)
- using special hardware support

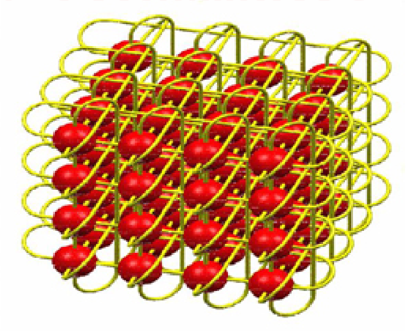
Applications

- work on more applications
- $\Rightarrow$ interested in collaborations (ask me!)

Discussion

THE END

Questions?



Thank you for your attention!