Non-Blocking Collectives for MPI-2

– overlap at the highest level –

Torsten Höfler

Department of Computer Science
Indiana University / Technical University of Chemnitz

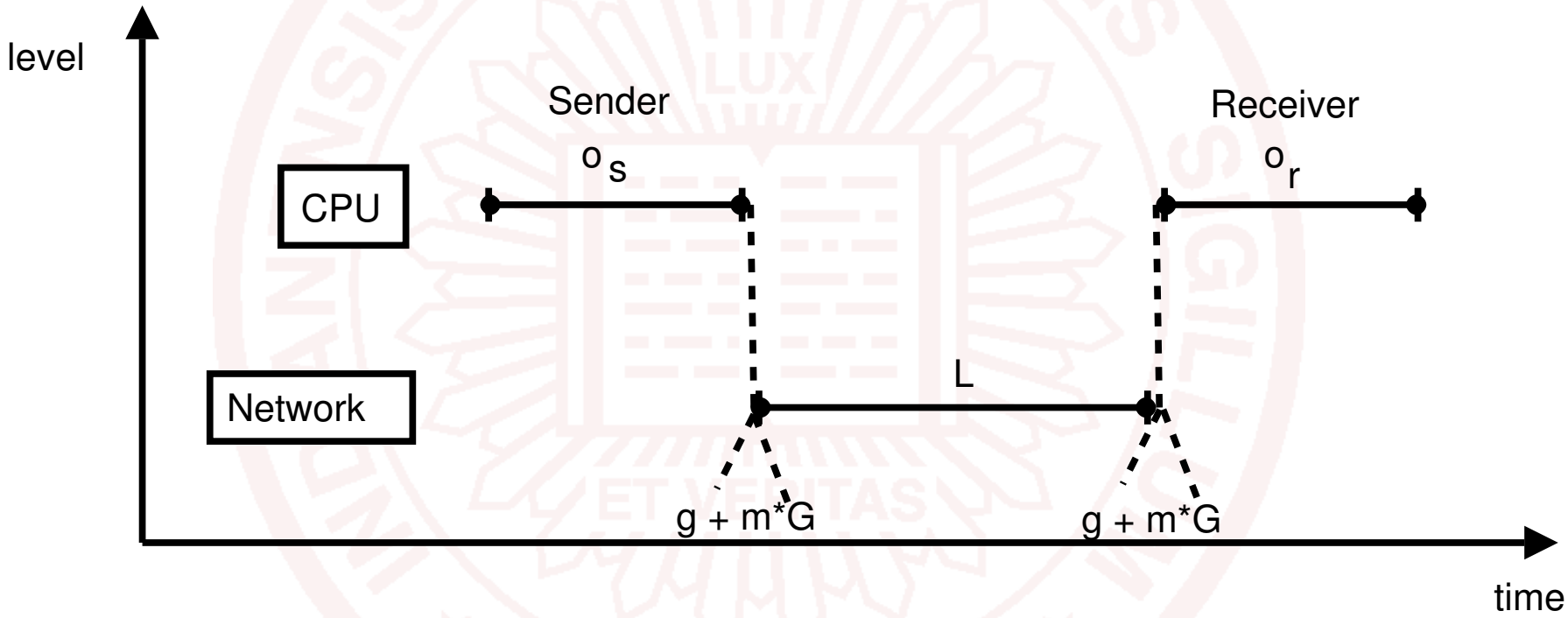
Commissariat à l'Énergie Atomique
Direction des applications militaires (CEA-DAM)
Bruyéres-le-chatel, France

18th January 2007

Outline

Outline

The LogGP Model



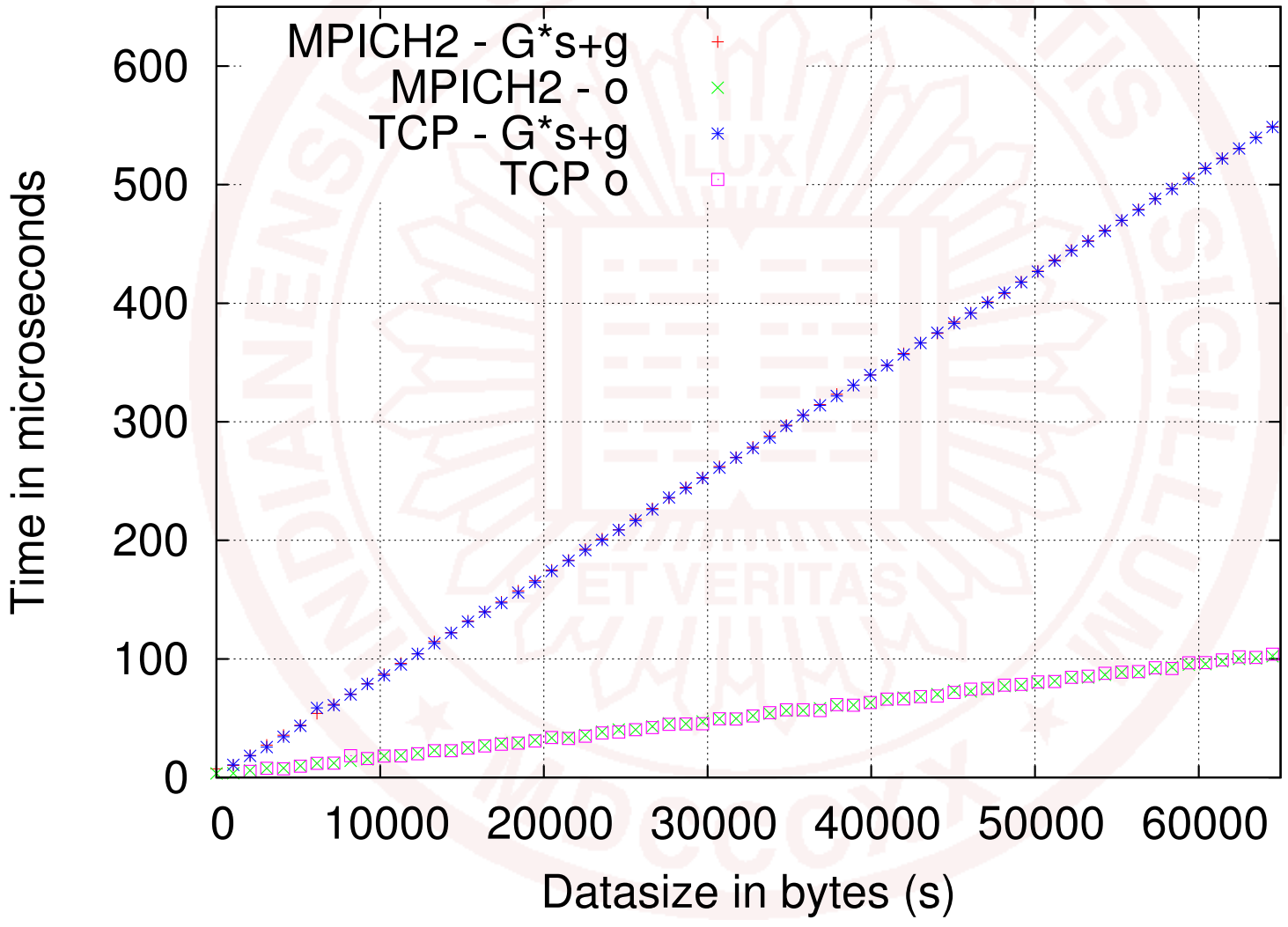
Interconnect Trends

Technology Change

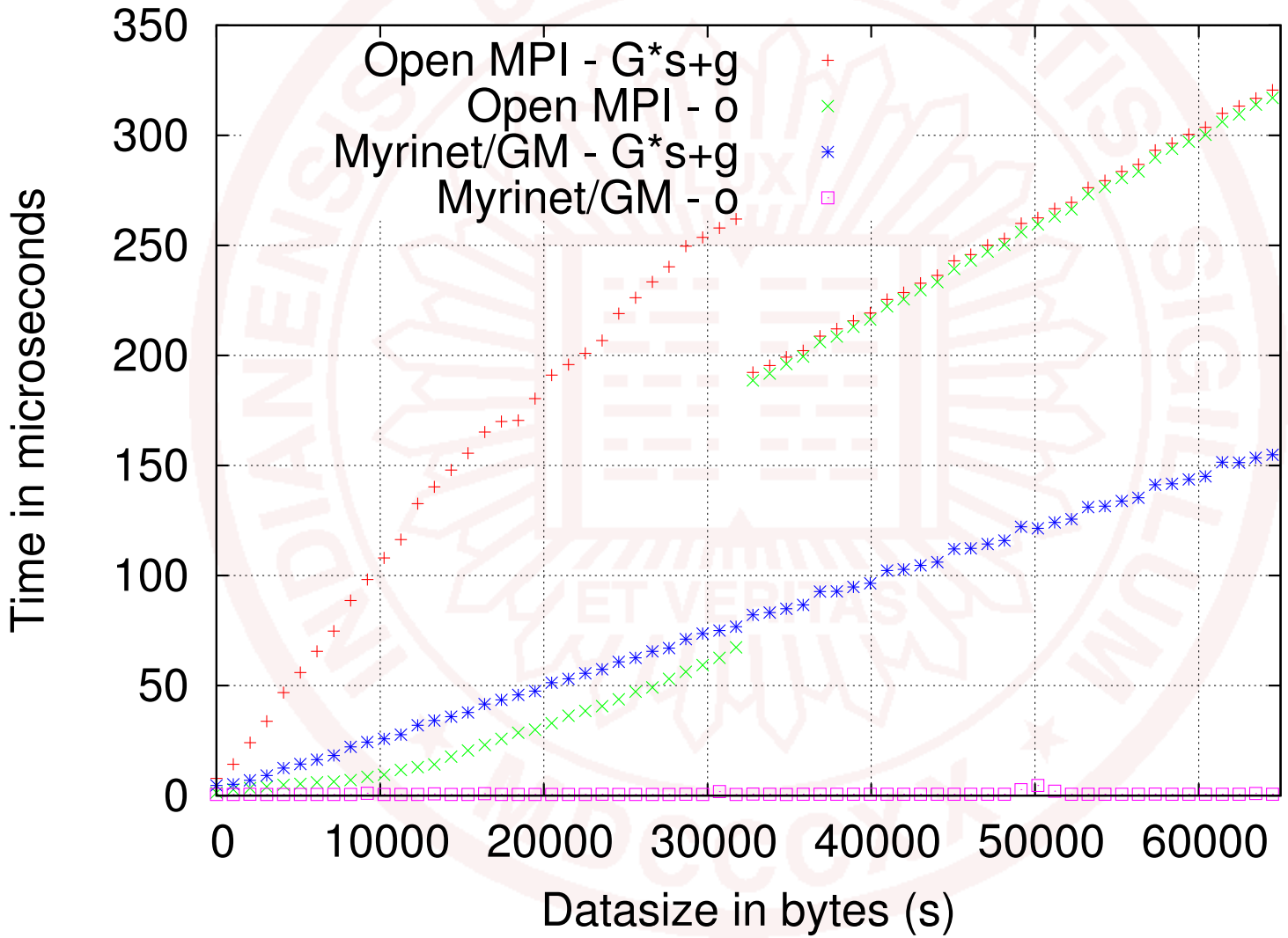
- modern interconnects have co-processors (Quadrics, InfiniBand, Myrinet)
- TCP/IP is optimized for lower host-overhead (see our work)
- Ethernet protocol offload
- $L + g + m \cdot G >> o$

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

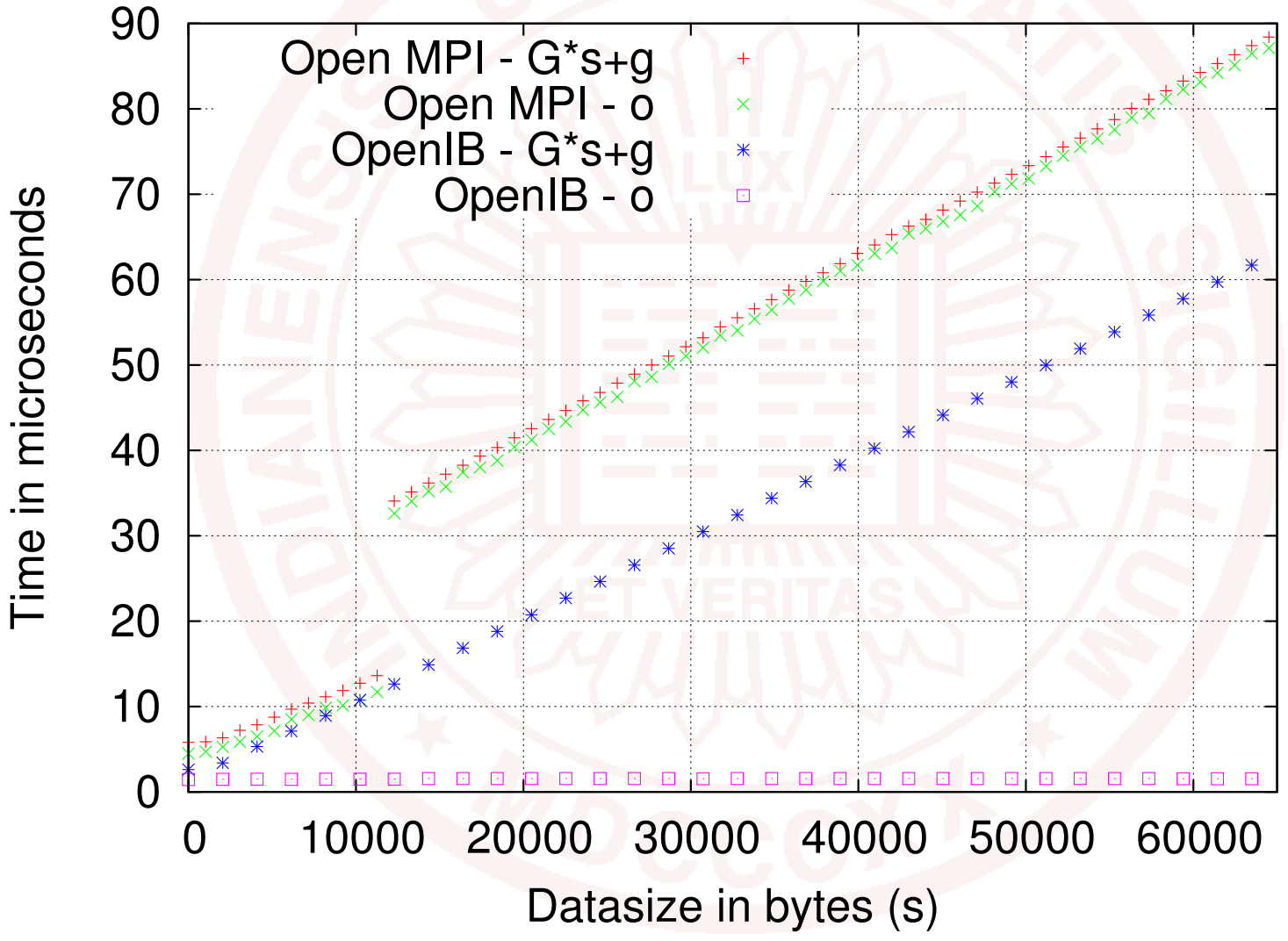
LogGP Model Examples - TCP



LogGP Model Examples - Myrinet/GM



LogGP Model Examples - InfiniBand/OpenIB



Literature

[1] **T. Hoefler**, A. LICHEI, AND W. REHM: *Low-Overhead LogGP Parameter Assessment for Modern Interconnection Networks*. *Accepted at PMEO-PDS 2007 in conjunction with IPDPS 2007*

[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*

Outline

Modelling the Benefits

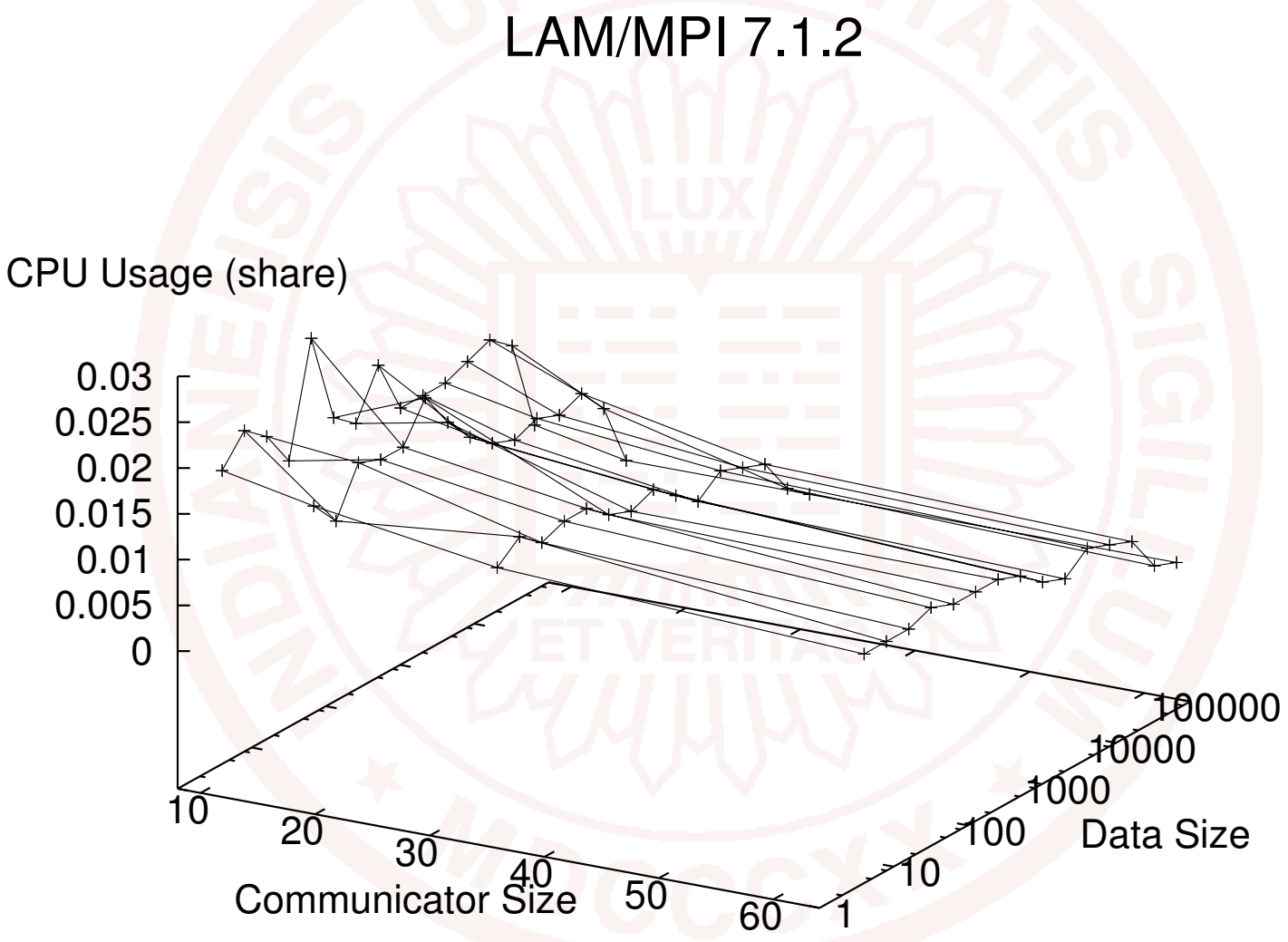
LogGP Models - general

$$
\begin{aligned}
t_{barr} &= (2o + L) \cdot \lceil log_2 P \rceil \\
t_{allred} &= 2 \cdot (2o + L + m \cdot G) \cdot \lceil log_2 P \rceil + m \cdot \gamma \cdot \lceil log_2 P \rceil \\
t_{bcast} &= (2o + L + m \cdot G) \cdot \lceil log_2 P \rceil
\end{aligned}
$$

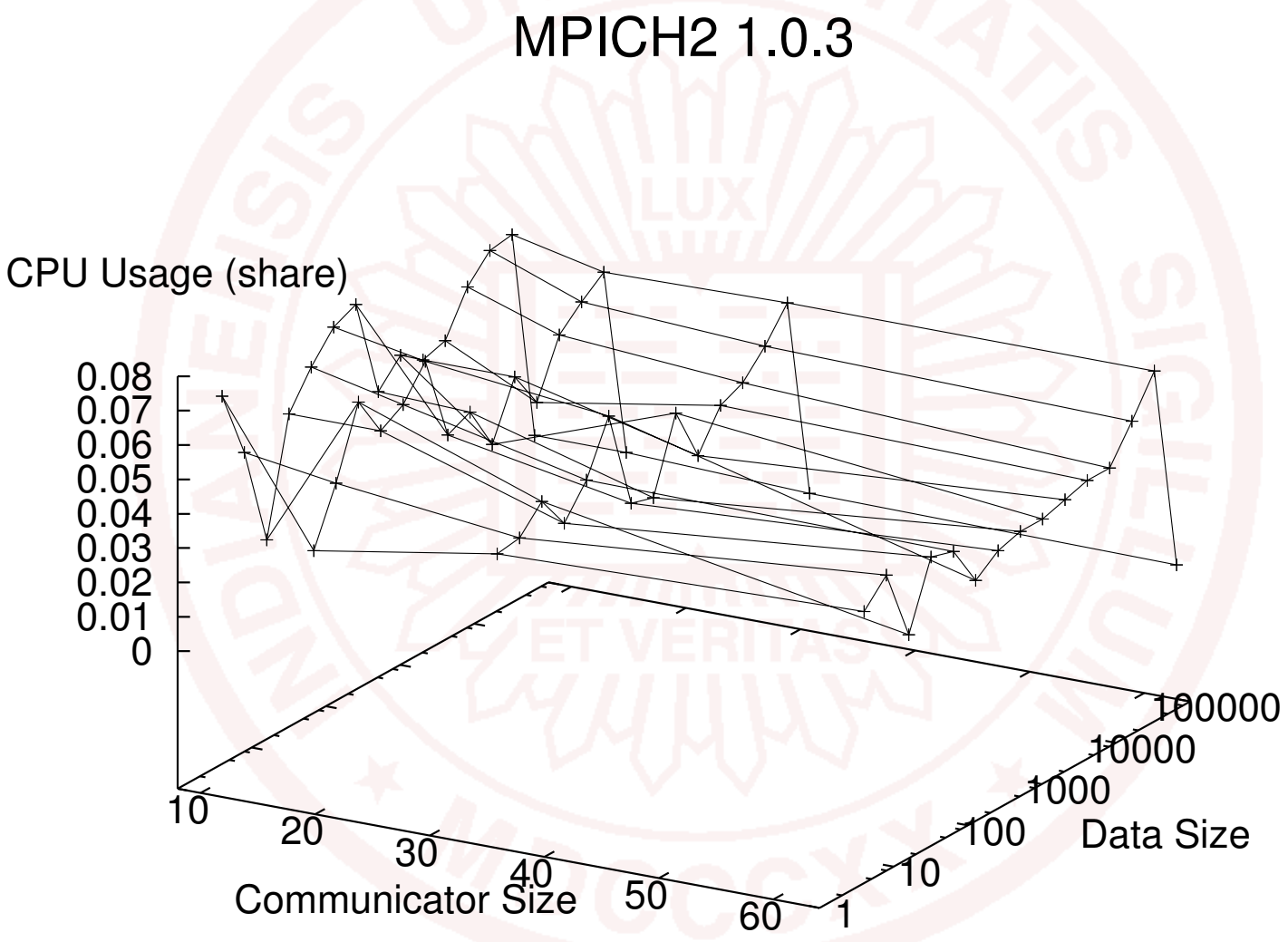
CPU and Network LogGP parts

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

User Overhead Benchmarks



User Overhead Benchmarks



Send/Recv 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
   do i=1,p
     call MPI_SEND(...)
   enddo
else
   call MPI_RECV(...)
endif
```

vs. collective

```
call MPI_SCATTER(...)
```

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^{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
- especially if processors are overloaded
- worse for 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"*

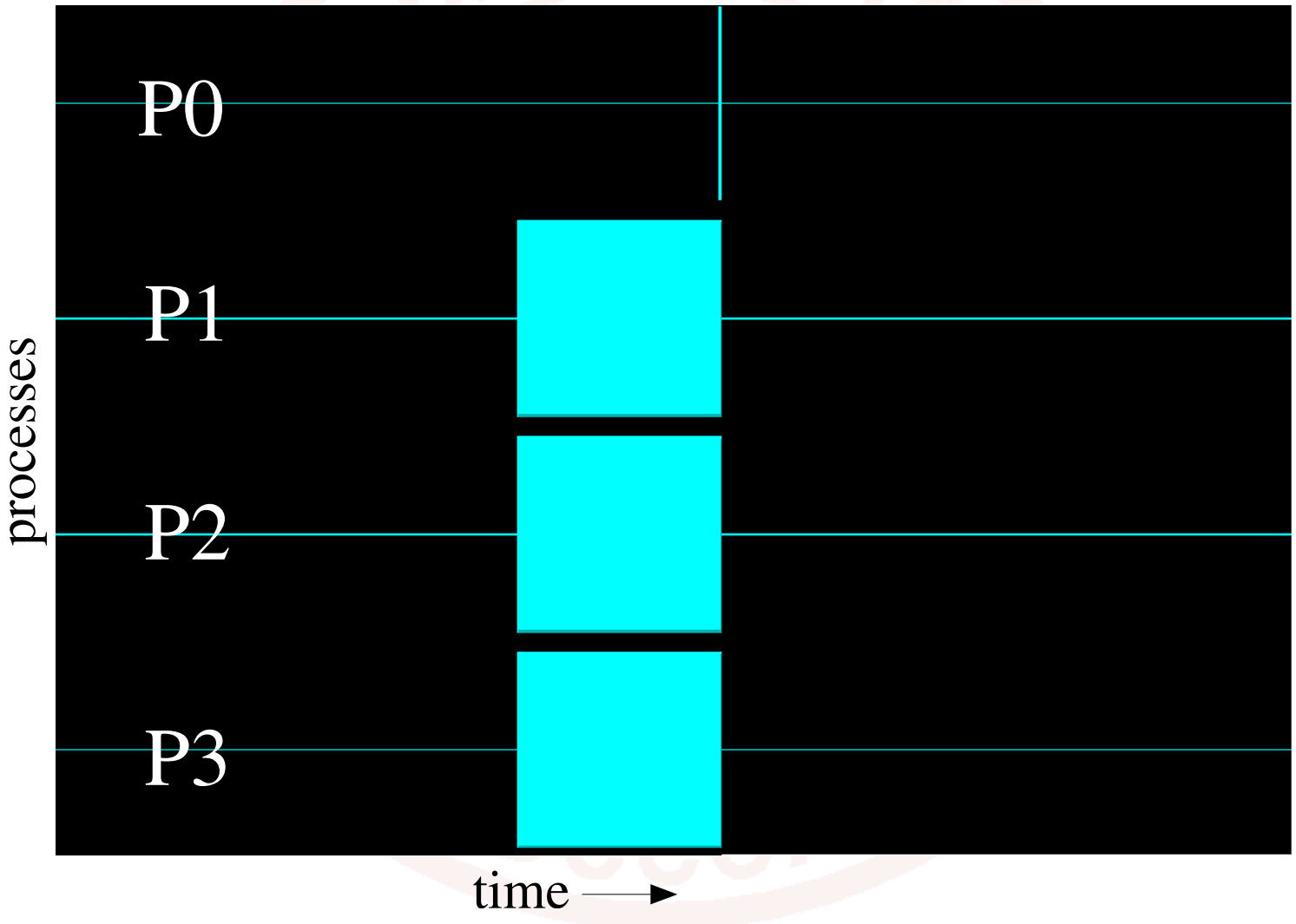
Process Skew

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

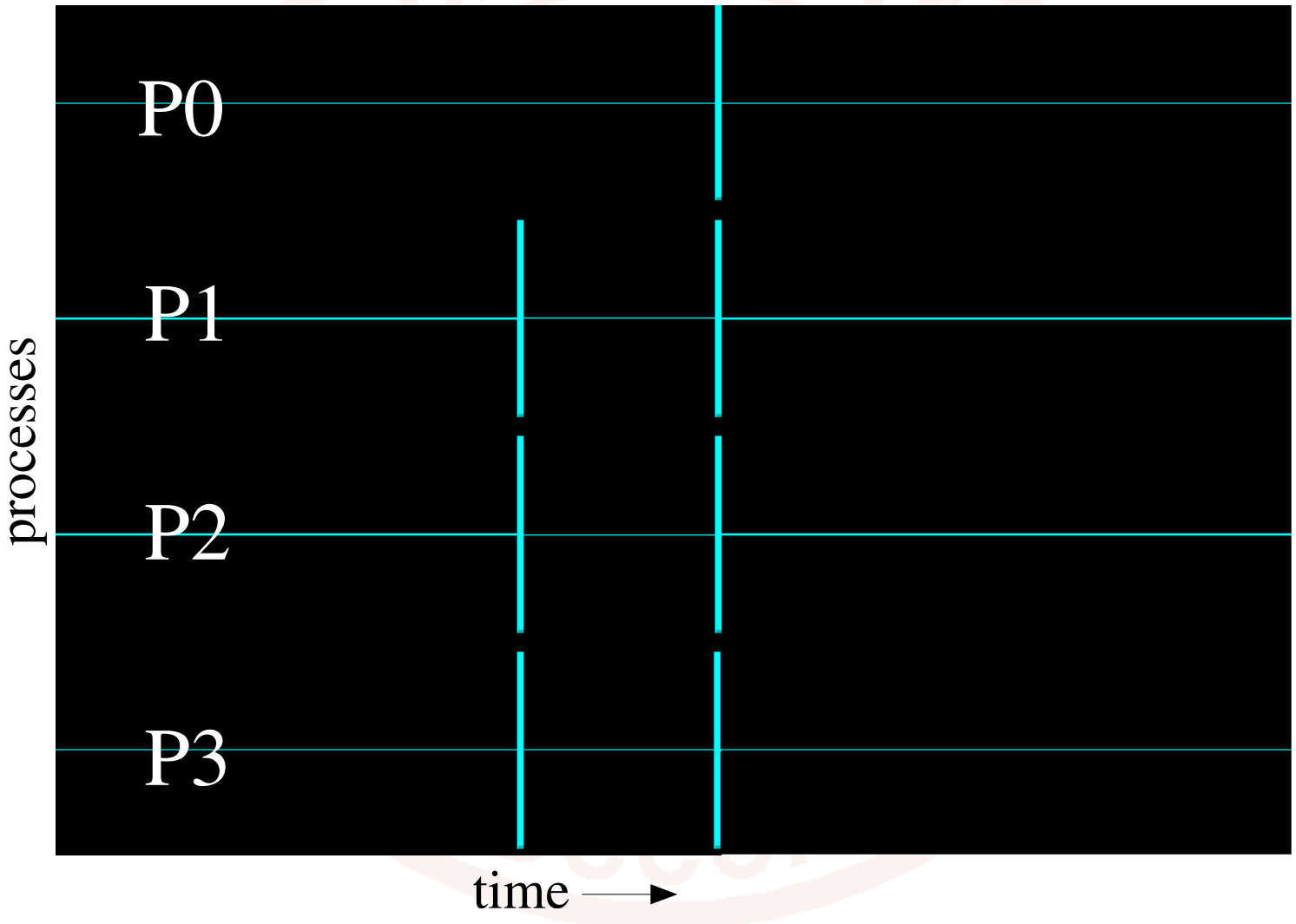
Example

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

Process Skew - MPI Example - Jumpshot



Process Skew - NBC Example - Jumpshot



Literature

[3] **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*

[4] **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

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

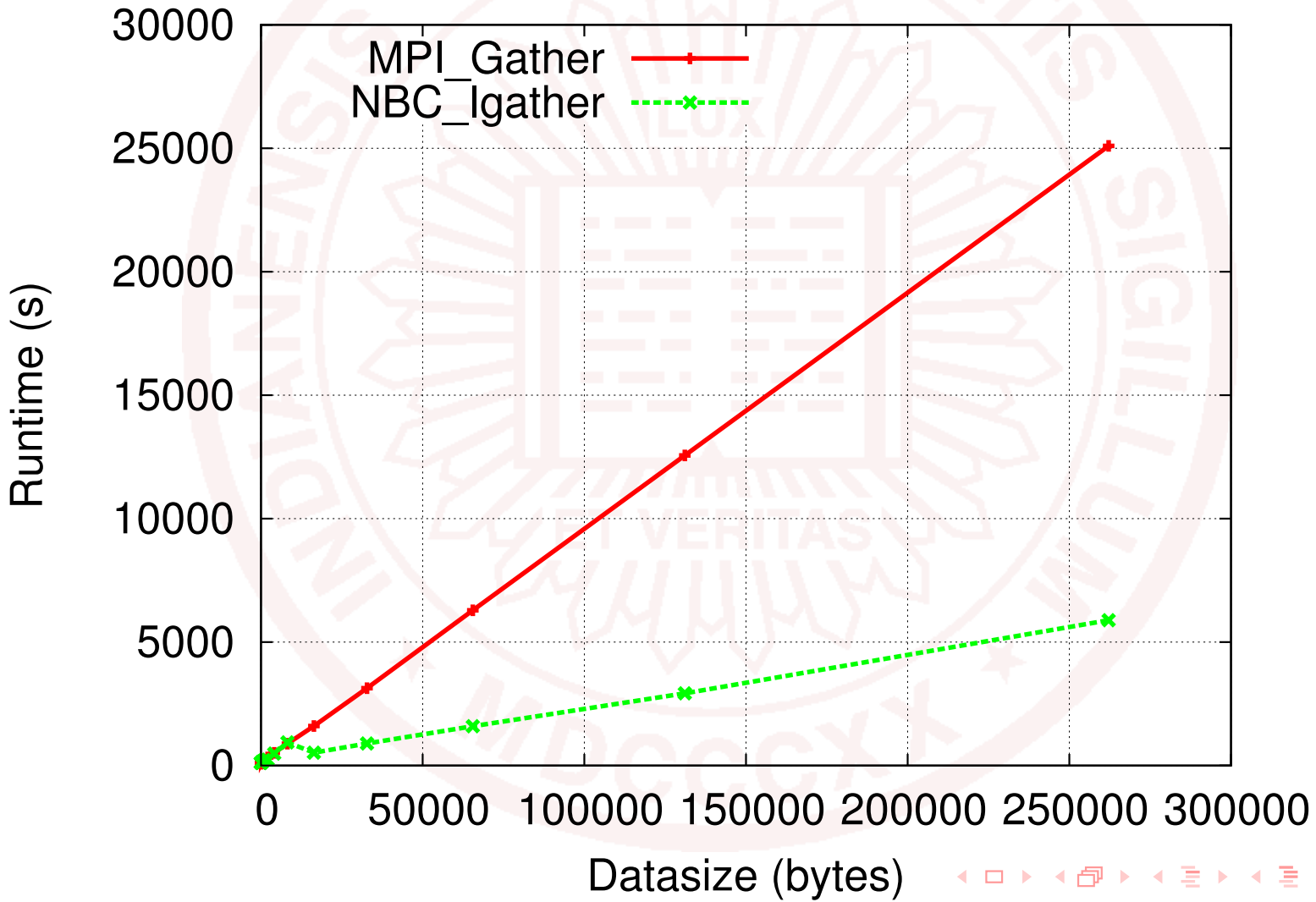
- implementation available with LibNBC
- written in ANSI-C and uses only MPI-1
- central element: collective schedule
- a coll-algorithm can be represented as a schedule
- trivial addition of new algorithms

Example: dissemination barrier, 4 nodes, node 0:

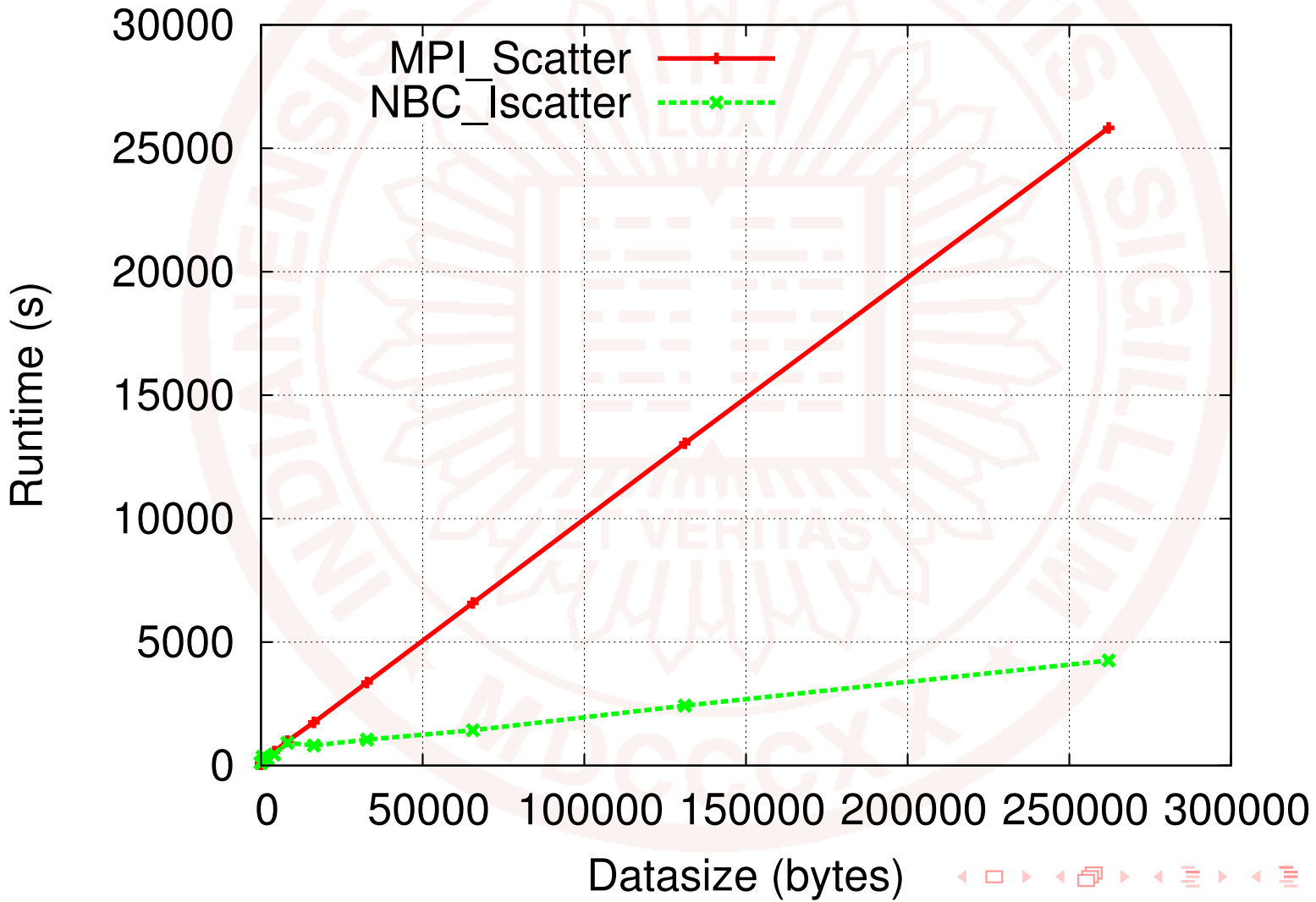
send to 1	recv from 3	end	send to 2	recv from 2	end

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

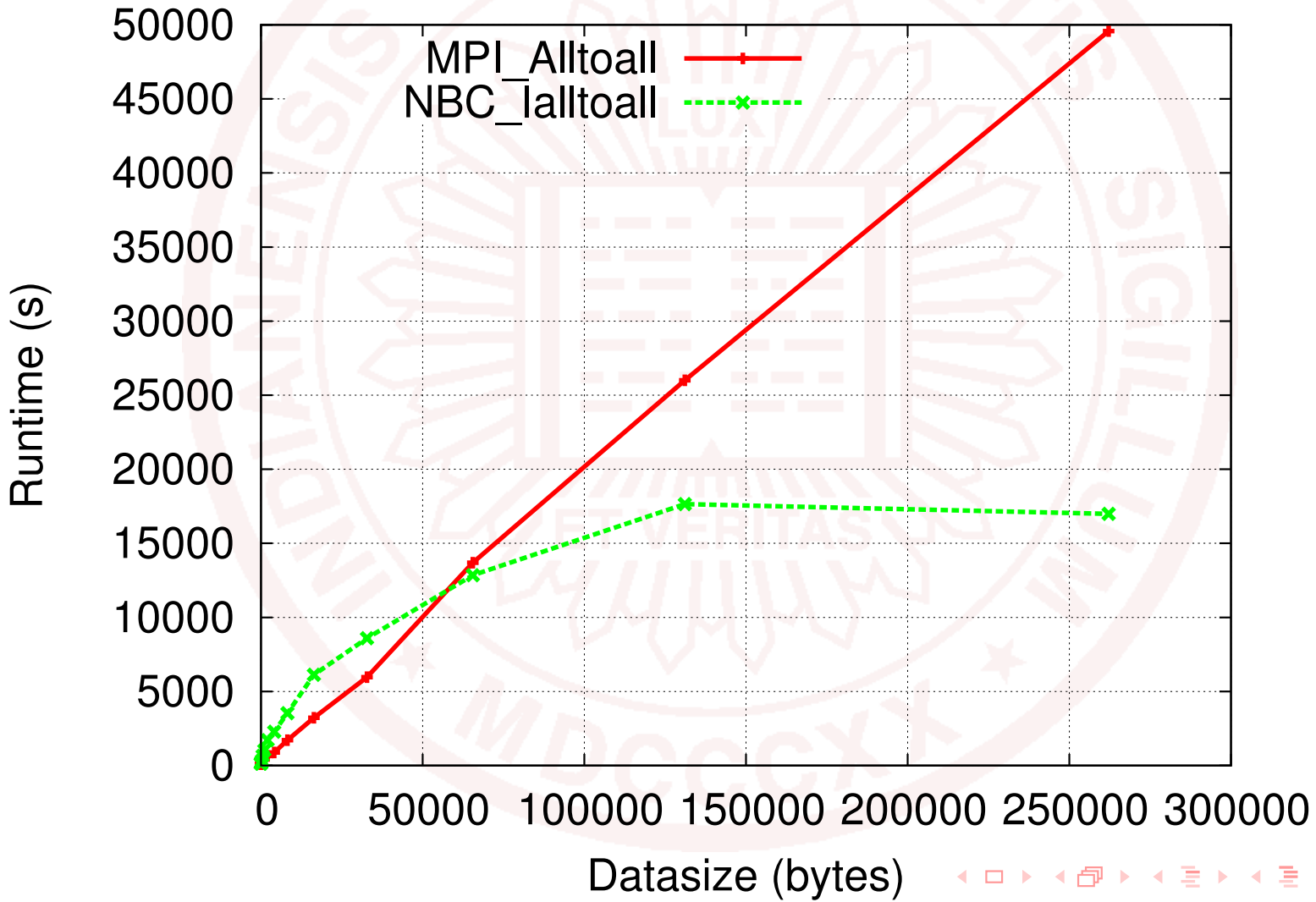
LibNBC Benchmarks - Gather with InfiniBand/MVAPICH on 64 nodes



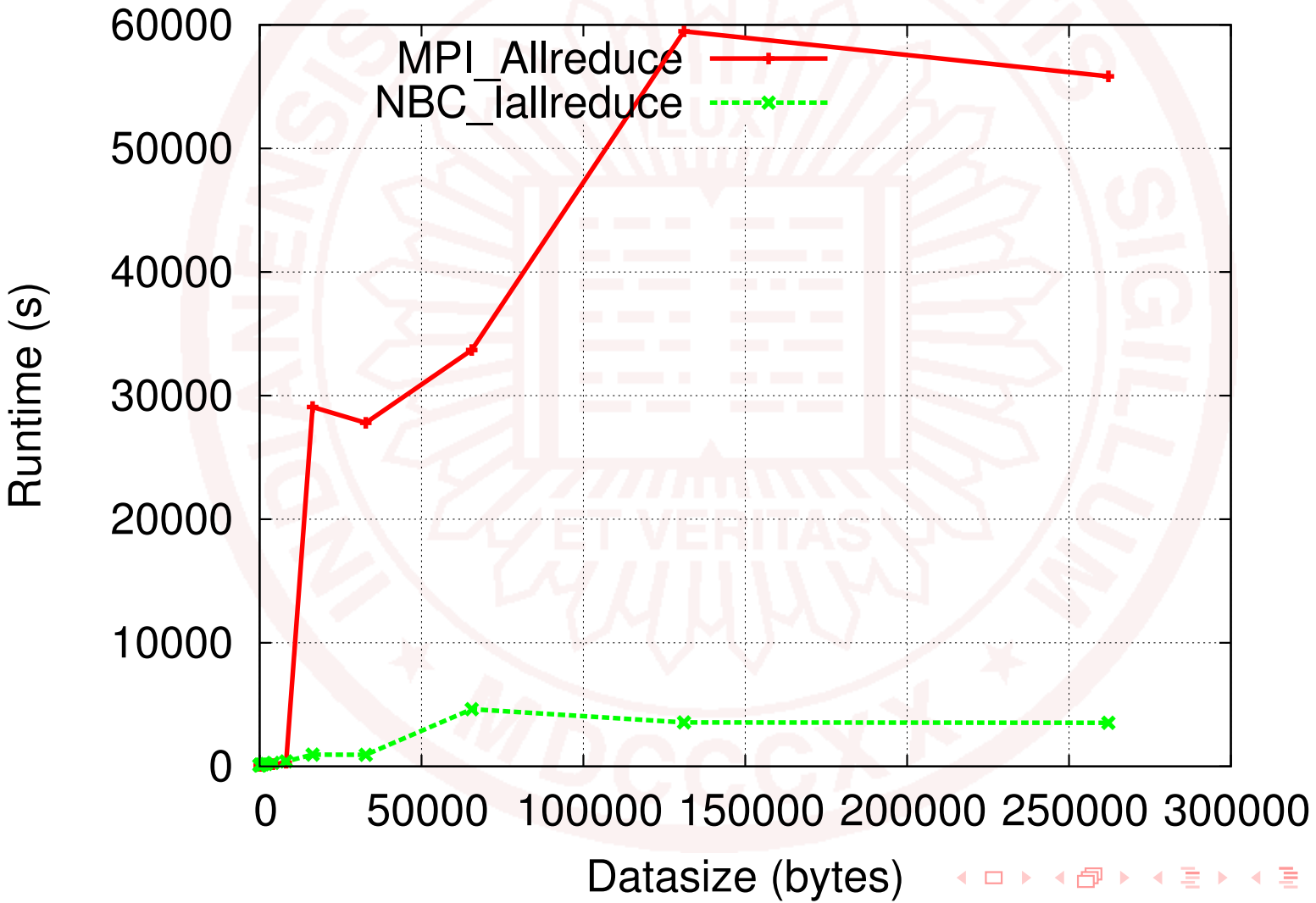
LibNBC Benchmarks - Scatter with InfiniBand/MVAPICH on 64 nodes



LibNBC Benchmarks - Alltoall with InfiniBand/MVAPICH on 64 nodes



LibNBC Benchmarks - Allreduce with InfiniBand/MVAPICH on 64 nodes



Literature

[5] **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*

[6] **T. Hoefler**, A. LUMSDAINE AND W. REHM: *Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI. Under submission (ask me for a copy)*

Outline

Pseudocode Overlap with Double Buffering

```
do ....
  ! send communication buffer
  MPI_Ialltoall(buffer_comm, ...,, request)

  ! do useful work
  do_work(buffer_work)

  ! finish communication
  MPI_Wait(request, ...)

  ! swap the buffers
  buffer_tmp = buffer_comm
  buffer_comm = buffer_work
  buffer_work = buffer_tmp
enddo
```

Linear Solvers - Domain Decomposition

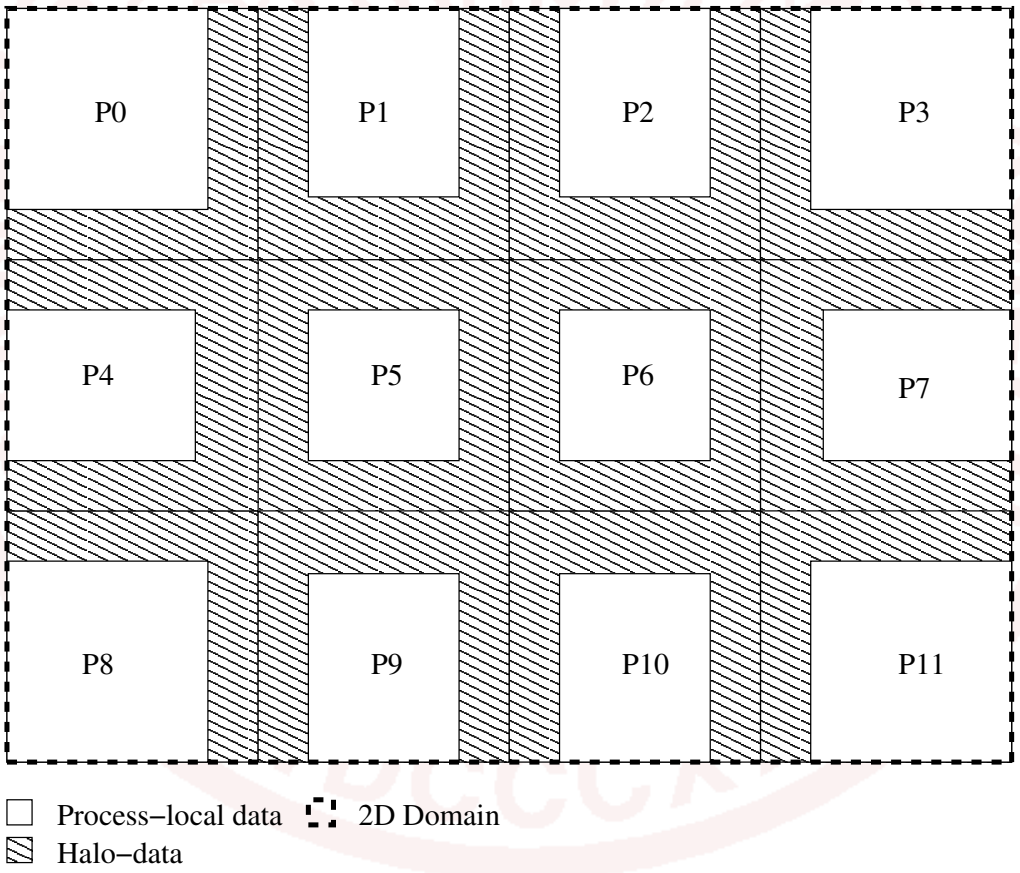
First Example

Naturally Independent Computation - Linear Solvers

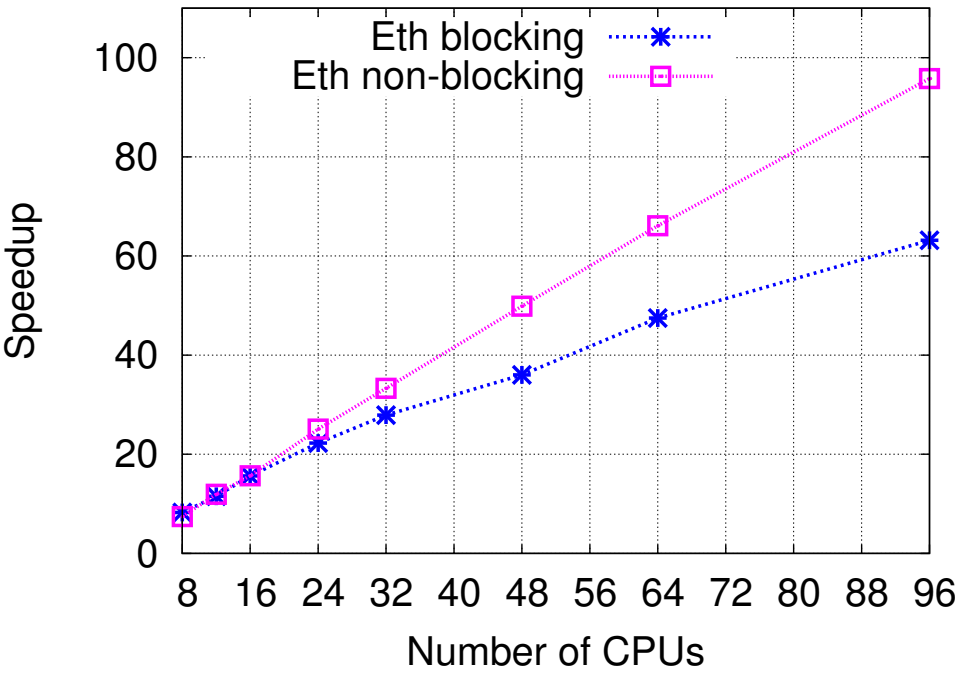
- iterative linear solvers are used in many scientific kernels
- often used operation is vector-matrix-multiply
- matrix is domain-decomposed (e.g., 3D)
- only outer (border) elements need to be communicated
- can be overlapped

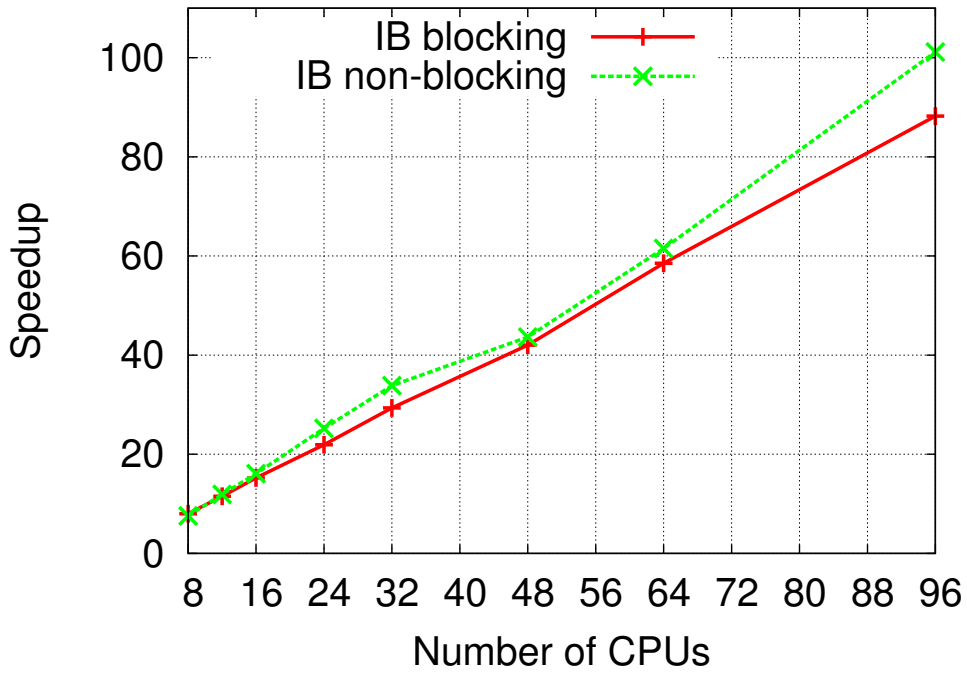
Domain Decomposition

- nearest neighbor communication
- can be implemented with MPI_Alltoallv



Parallel Speedup (Best Case)

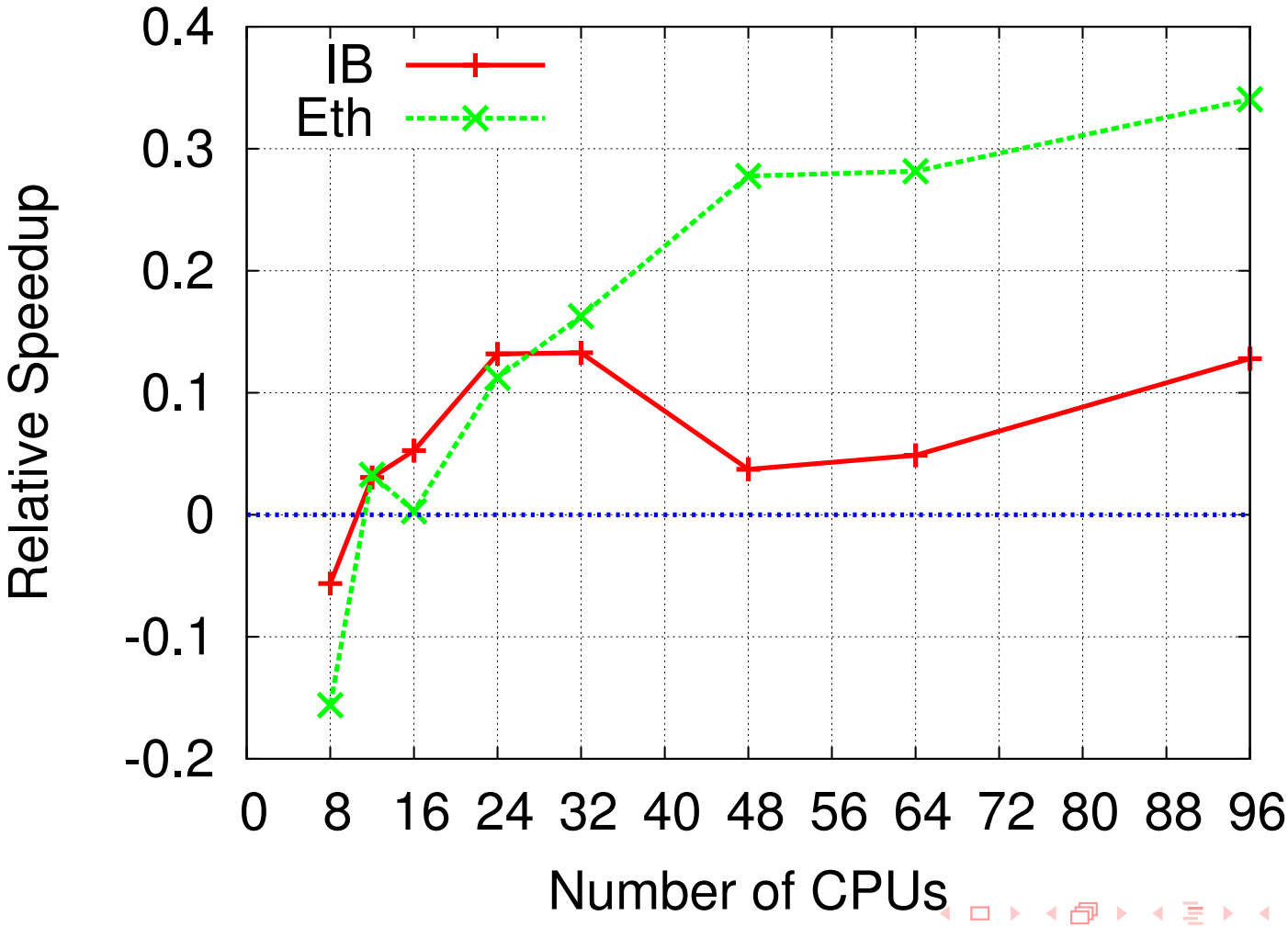




- Cluster: 128 2 GHz Opteron 246 nodes
- Interconnect: Gigabit Ethernet, InfiniBand™
- System size 800x800x800 (1 node ≈ 5300s)

Parallel Gain with Non-Blocking Communication

$$gain = t_{nbl} / t_{bl}$$



Linear Solvers - Domain Decomposition

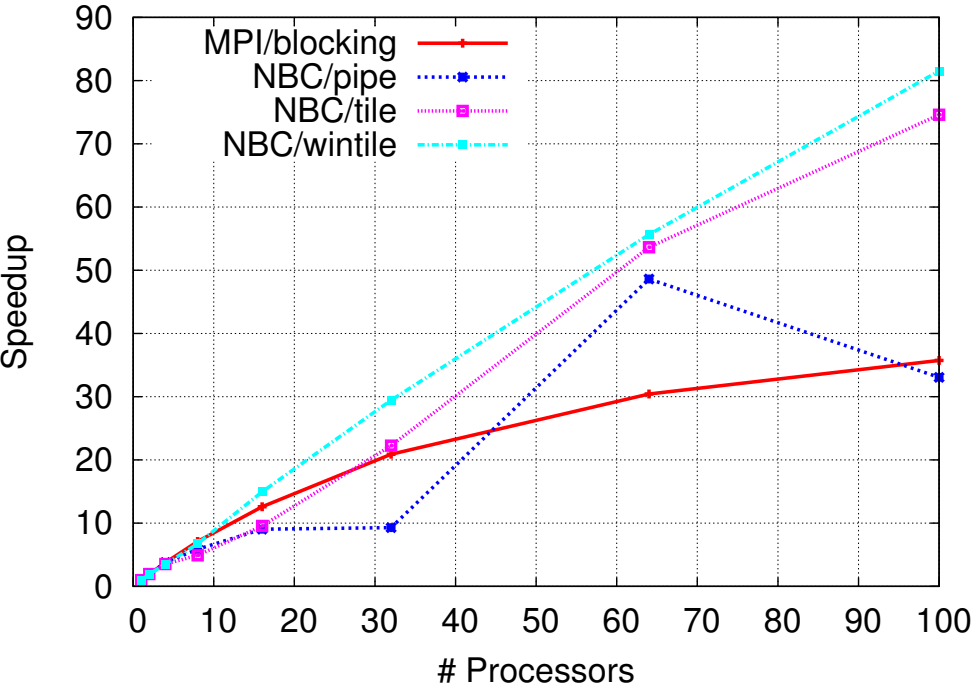
Second Example

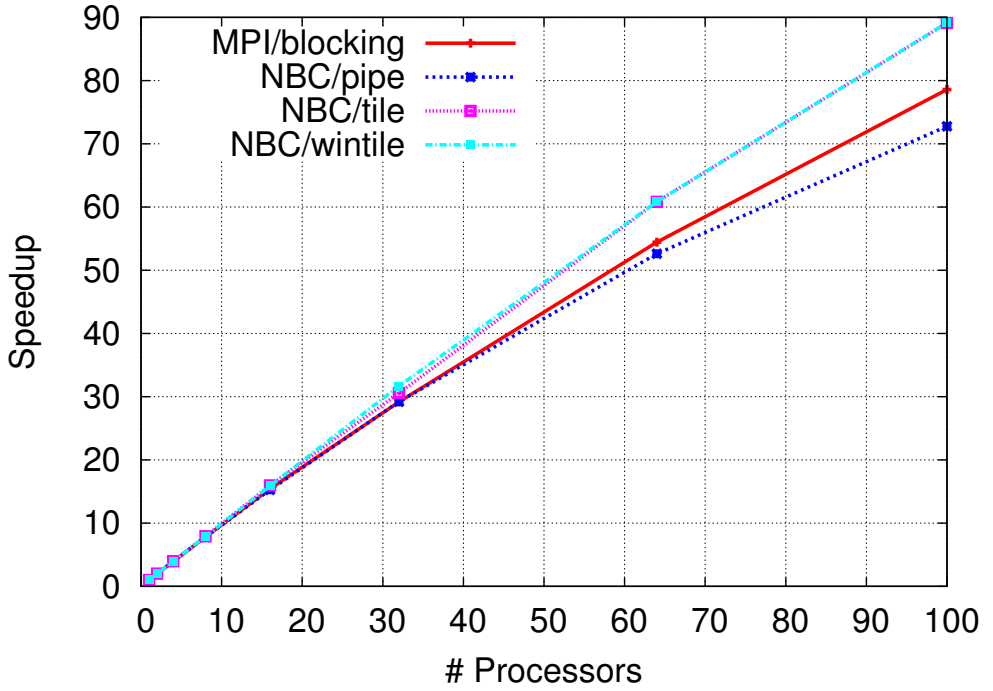
Data Parallel Loops - Parallel Compression

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

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

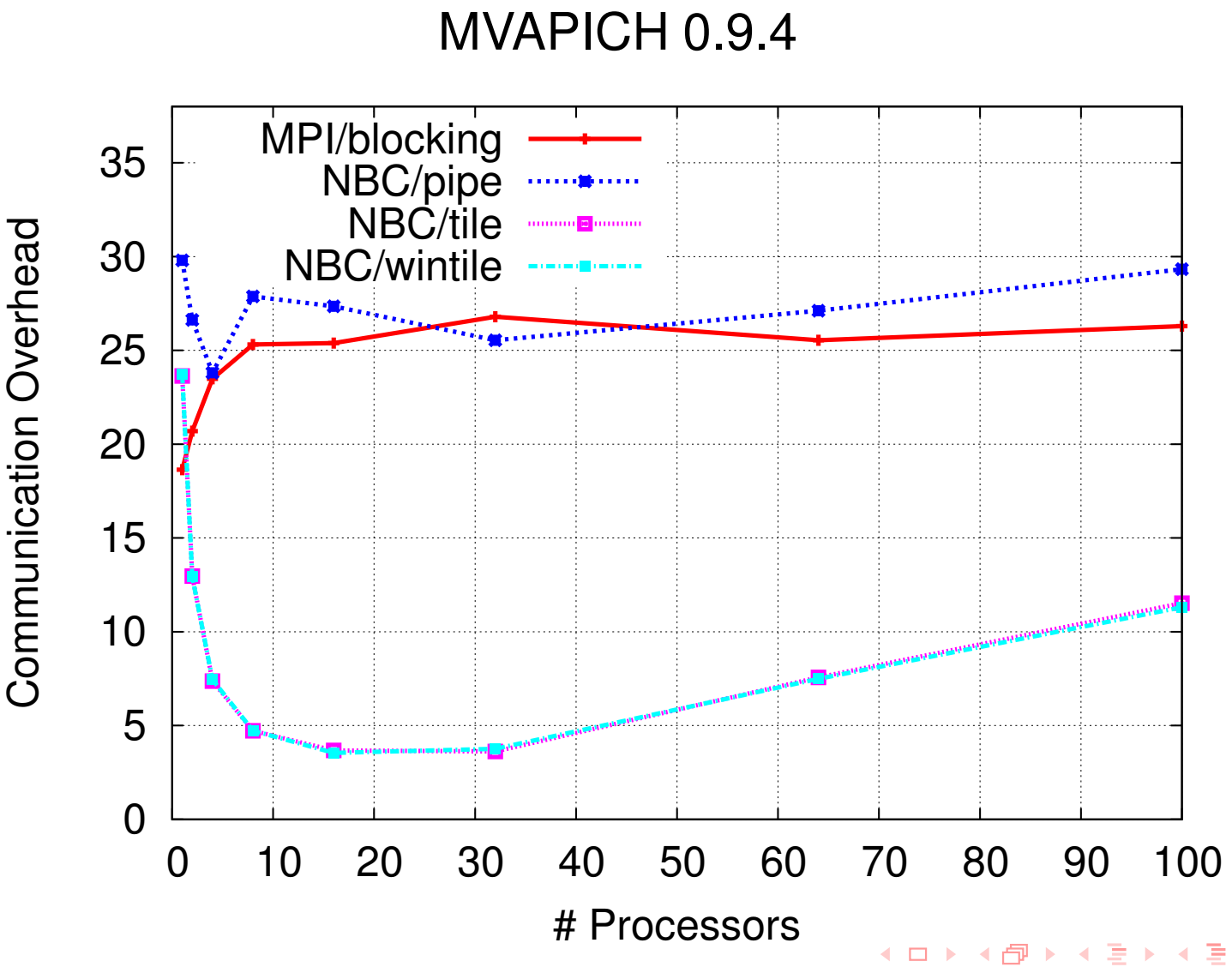
Parallel Speedup (Best Case)





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

Communication Overhead



Literature

[7] **T. Hoefler** P. GOTTSCHLING, W. REHM AND A. LUMSDAINE: *Optimizing a Conjugate Gradient Solver with Non-Blocking Collective Operations. In 13th European PVM/MPI User's Group Meeting, Proceedings, LNCS 4192, presented in Bonn, Germany, pages 374-382, Springer, ISSN: 0302-9743, ISBN: 3-540-39110-X Sep. 2006*

[8] **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

Ongoing Work

3D-FFT

- optimized version of 3D-FFT with full overlap or pipelined
- still in development (ABINIT version)
- very promising

LOBPCG Method in ABINIT

- developed by G. Zérah (CEA)
- could use NBC for parallel matrix-matrix multiplication

LibNBC

- Fortran bindings for LibNBC
- optimized collectives

Ongoing Work (continued)

Collective Communication

- optimized collectives for InfiniBand™
- using special hardware support

Network Modelling

- refined LogGP model parametrization
- modelling of collective algorithms

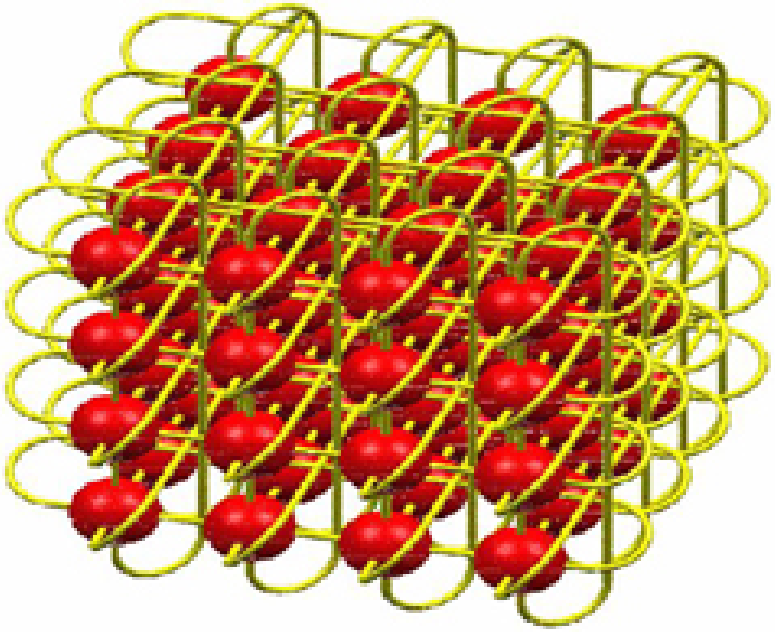
Collaborate with Application Engineers

- I want to help to apply NBC to your problem!

Discussion

THE END

Questions?



Thank you for your attention!