Sparse Collective Operations for MPI

Torsten Hoefler
OpenSystems Lab, Indiana University, Bloomington, IN, USA
Jesper Larsson Träff
NEC Laboratories Europe, St. Augustin, Germany

© NEC Laboratories Europe

MPI 3.0 Standardization process has started: …

Improve usefulness, efficiency, suitability of MPI with

- More/better collectives support (non-blocking collectives, additional functionality)
- Better/additional one-sided communication
- More topology support (for applications and systems – MPI 2.2)
- Hybrid programming (thread safety/support, mixing models)
- Fault-tolerance
- Tool support

Visit: www.mpi-forum.org

© NEC Laboratories Europe

MPI 3.0 Standardization process has started: …

and is pursued by the MPI Forum:

(mostly implementers and library/tool builders)

© NEC Laboratories Europe

Problems with current (full) collectives

- Scalability
- Do not support some (naturally sparse) applications
- HPC Systems have/may have sparse communication networks

MPI_Allgatherv(sendbuf,…,
recvbuf,recvcounts[],recvdispls[],recvtype,comm);

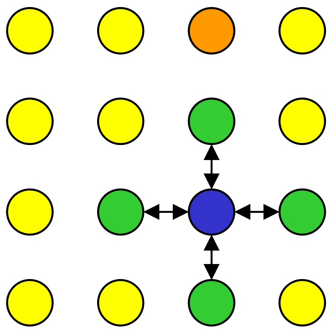
Array of p entries

all processes p involved

…most of which may be 0

© NEC Laboratories Europe

Applications (Qbox, TDDF, QCD codes, POP, …) often exhibit sparse (collective) communication patters



Simultaneous MPI_Gatherv on subcommunicators: deadlock!

MPI_Allgatherv gathers too much (all data on all processes)

MPI_Alltoallv too powerful, and wasteful

…

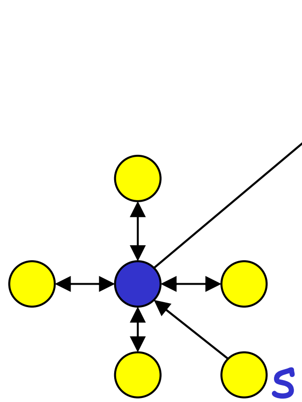
Sparse analogue of MPI_Allgatherv:

Process x gathers data from all mesh neighbors (and sends same data to all neighbors) cannot readily be expressed with existing collectives

© NEC Laboratories Europe

Proposal: sparse collective operations for MPI 3.0

MPI_Neighbor_gather(sendbuf,…,recvbuf,recvcount,…,comm);



Calling processes receives (different) data from a set of source neighbors s

Calling process sends same data to a set of target neighbors t

sendbuf:

recvbuf:

s_0	s_1	s_2	s_3	s_4

© NEC Laboratories Europe

For completeness (in analogy with current, dense collectives):

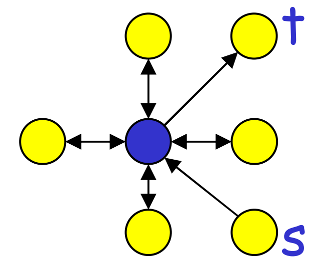
"irregular" (vector) variant:

processes can send and receive different amounts of data from different neighbors

```
MPI_Neighbor_Gatherv(sendbuf,...,
                     recvbuf,recvcounts[],recvdispls[],...,comm);
```


© NEC Laboratories Europe

Semantics and neighborhoods



Semantics:

If process j is a neighbor (source/target) of process i then process i must be a neighbor of process j (with multiplicities)

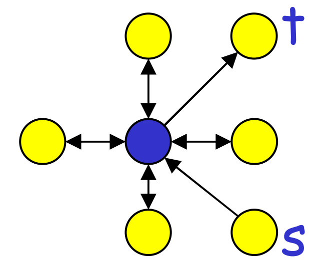
Semantics:

Datasizes between neighbors must match (same type signature)

Semantics:

If process i calls sparse collective C, then all neighbors of i must eventually call C (and no other collectives on the same communicator in between)

© NEC Laboratories Europe



Semantics:

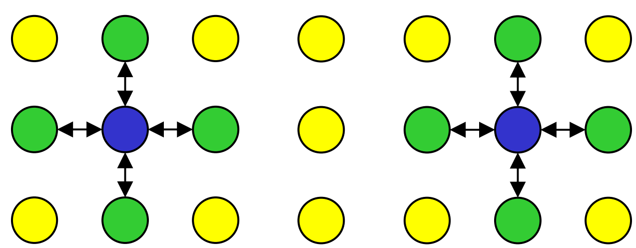
Sparse collectives are blocking (like current, dense collectives).

Note (for completeness):

Non-blocking sparse collectives will be proposed analogous to the non-blocking, dense collectives for MPI 3.0

Note: disjoint neighborhoods allowed to be "out of sync"

MPI_Neighbor_gather(…,comm);



MPI_Neighbor_gather(…,comm);

MPI_Neighbor_gather(…,comm);

© NEC Laboratories Europe

Experiment 1: naïve vs. scheduled implementation

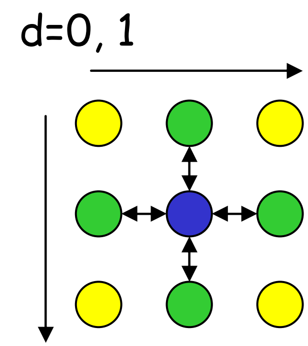
Naïve: post non-blocking send-receives to all neighbors, and wait…

```
MPI_Neighbor_Alltoall(sendbuf,….,recvbuf,…., comm)
{
  // for all source neighbors s:
  MPI_Irecv(recvbuf+s*recvextent,….,comm);
  // for all target neighbtors t:
  MPI_Isend(sendbuf+t*sendextent,….,comm);
  MPI_Waitall(…,comm);
}
```


© NEC Laboratories Europe

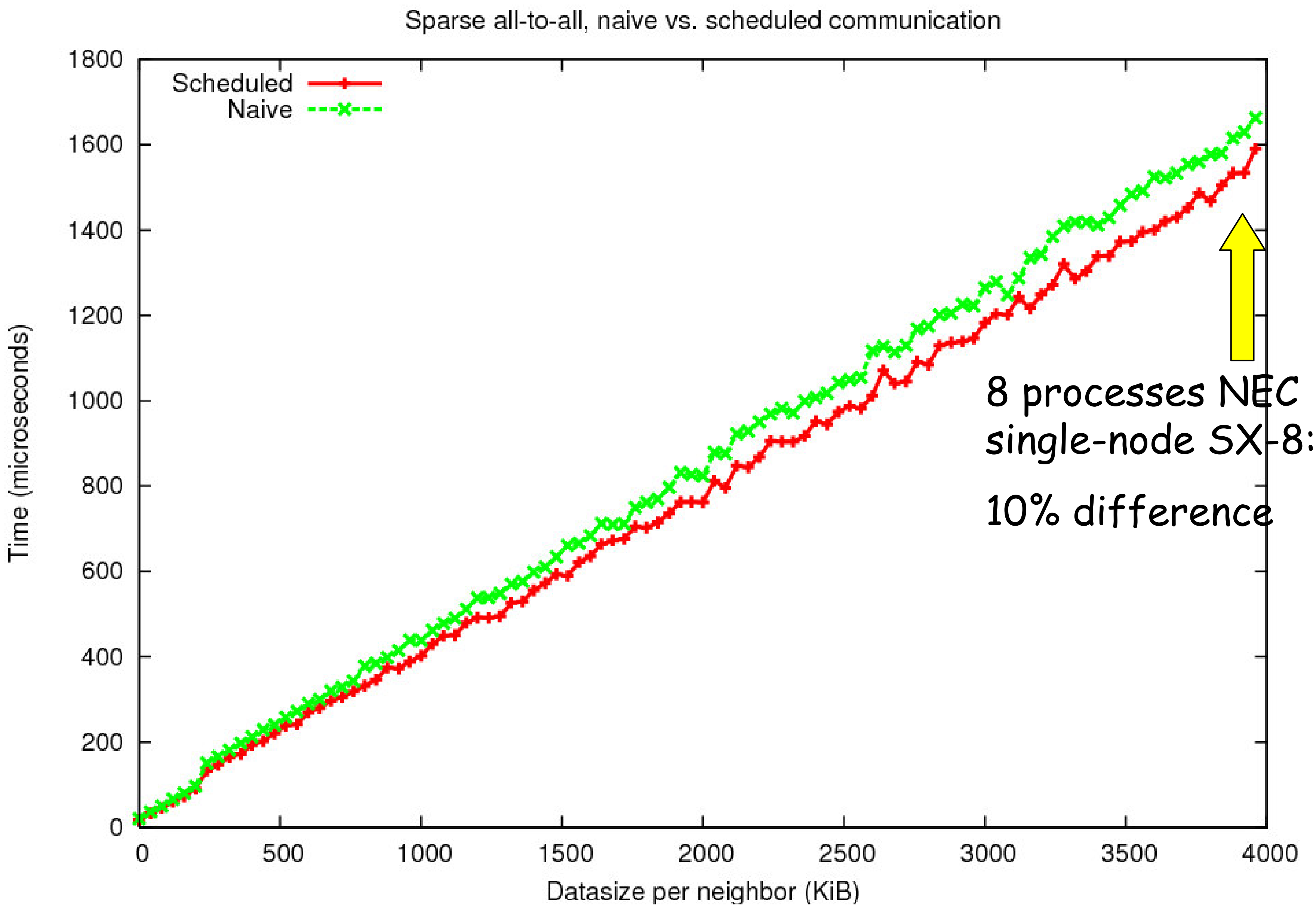
Special scheduled implementation for Cartesian grids:

use dimensions

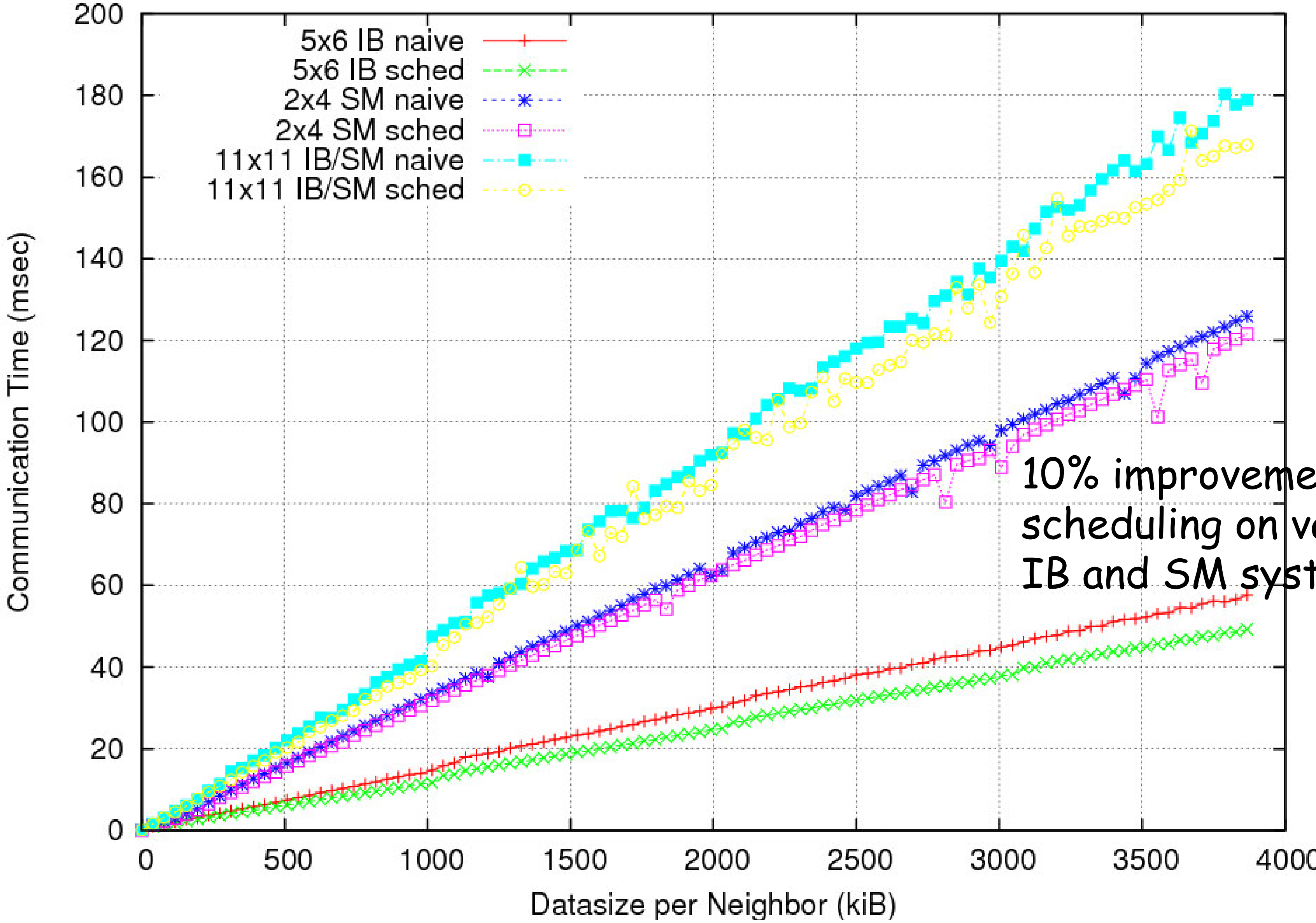



```
MPI_Neighbor_Alltoall(sendbuf,…,recvbuf,…, comm)
{
  for (d=0; d<dim; d++) {
    MPI_Cart_shift(comm,d,1,&down,&up);
    MPI_Sendrecv(sendbuf+s*sendextent,…,up,
                        recvbuf+t*recvextent,…,down,…,comm); s++; t++;
    MPI_Sendrecv(sendbuf+s*sendextent,…,down,
                        recvbuf+t*recvextent,…,up,…comm); s++; t++;
}
```


© NEC Laboratories Europe



© NEC Laboratories Europe



10% improvement by scheduling on various IB and SM systems

© NEC Laboratories Europe

Lesson 1:

Collective hook for the MPI implementation to schedule communication based on global view is needed!

- Also to discover “global view” (e.g. mesh)

Neighborhoods cannot be specified on a call by call basis

- Overhead would kill performance (and conflict with semantics)

© NEC Laboratories Europe

Neighborhoods a):

MPI_Neighborhood_gather(s,sources,sourceweights,
t,targets,targetweights,
info,comm);

s, sources, sourceweights: list of sources of calling process

t, targets, targetweights: list of targets of calling process

Semantics:

Collective function (all processes must call). For each sparse collective, associates neighborhood with communicator. Processes may appear multiple times. Weights proportional to data sizes in subsequent sparse calls.

© NEC Laboratories Europe

Neighborhoods b):

Compact version:

MPI_Neighborhood(MPI_NEIGHBOR_GATHER,
s,sources,sourceweights,
t,targets,targetweights,
info,comm);

Operation type MPI_NEIGBOR_GATHER, MPI_NEIGHBOR_ALLTOALL, … for each sparse collective

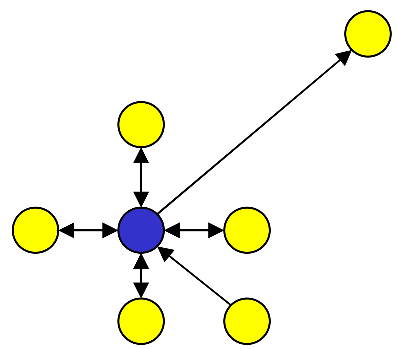
© NEC Laboratories Europe

Neighborhoods c):

Use virtual topology interface to specify neighborhoods:

MPI_Cart_create(comm,dims,...,&cartcomm);

MPI_Graph_create(comm,degrees,edges,...,&graphcomm);



© NEC Laboratories Europe

Neighborhoods a): vs.

Neighborhoods b): vs.

Neighborhoods c):

Left for discussion after the talk, and for the MPI Forum:

Discussions are ongoing, see

www.mpi-forum.org

Completeness: the other sparse collective operations

MPI_Neighbor_alltoall(sendbuf,…,recvbuf,comm)

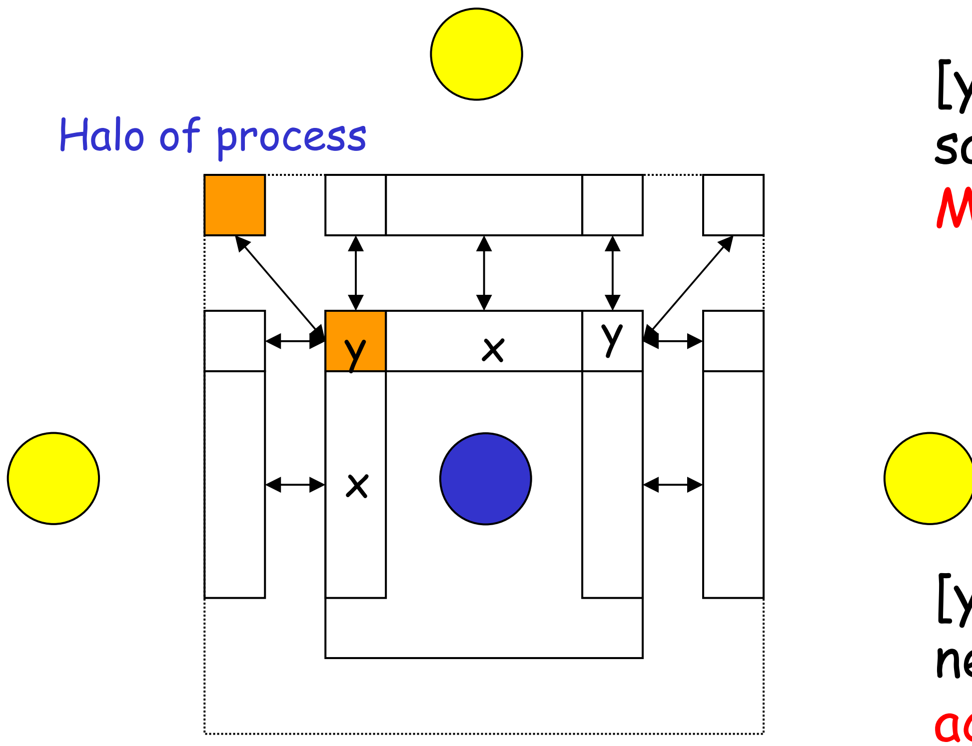
MPI_Neighbor_alltoallv(sendbuf,senddispls[],…,
recvbuf,recvdispls[],…comm)

MPI_Neighbor_alltoallw(sendbuf,senddispls[],…,sendtypes[]…,
recvbuf,recvdispls[],…,recvtypes[],…,comm)

All-to-all like exchanges in neighborhood

© NEC Laboratories Europe

Example: Halo exchange



A good MPI library can optimize communication of diagonal [y] blocks to piggyback on horizontal or vertical blocks

MPI_Neighbor_alltoallw:

[y,x,y] blocks sent to same neighbor: Multiplicities required

[y] blocks sent to 3 neighbors: Multiple access to same buffer

Horizontal and vertical [x] blocks may have different layout in memory: Need for datatypes

© NEC Laboratories Europe

Completeness: the final sparse collective operations

MPI_Neighbor_reduce(sendbuf,…,recvbuf,…,op,comm)

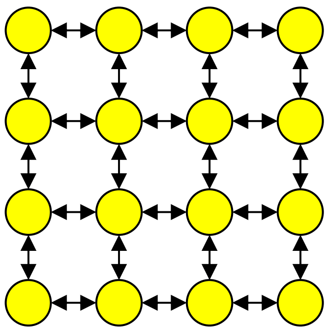
MPI_Neighbor_reducev(sendbuf,senddispls[],…,
recvbuf,recvdispls[],…,op,comm)

Sparse reduction collectives gather data from neighborhood and perform MPI reduction (built-in like MPI_SUM, or user-defined)

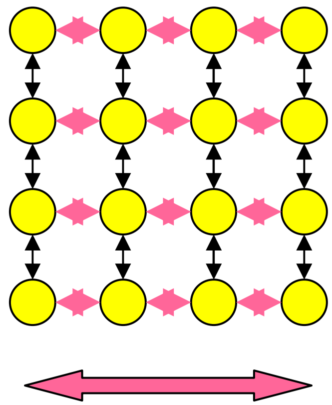
© NEC Laboratories Europe

Experiment 2: need for communication weights

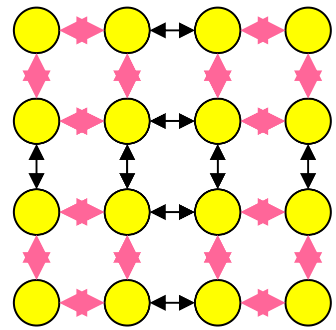
homogeneous



horizontal



circular



Same datasize along all edges

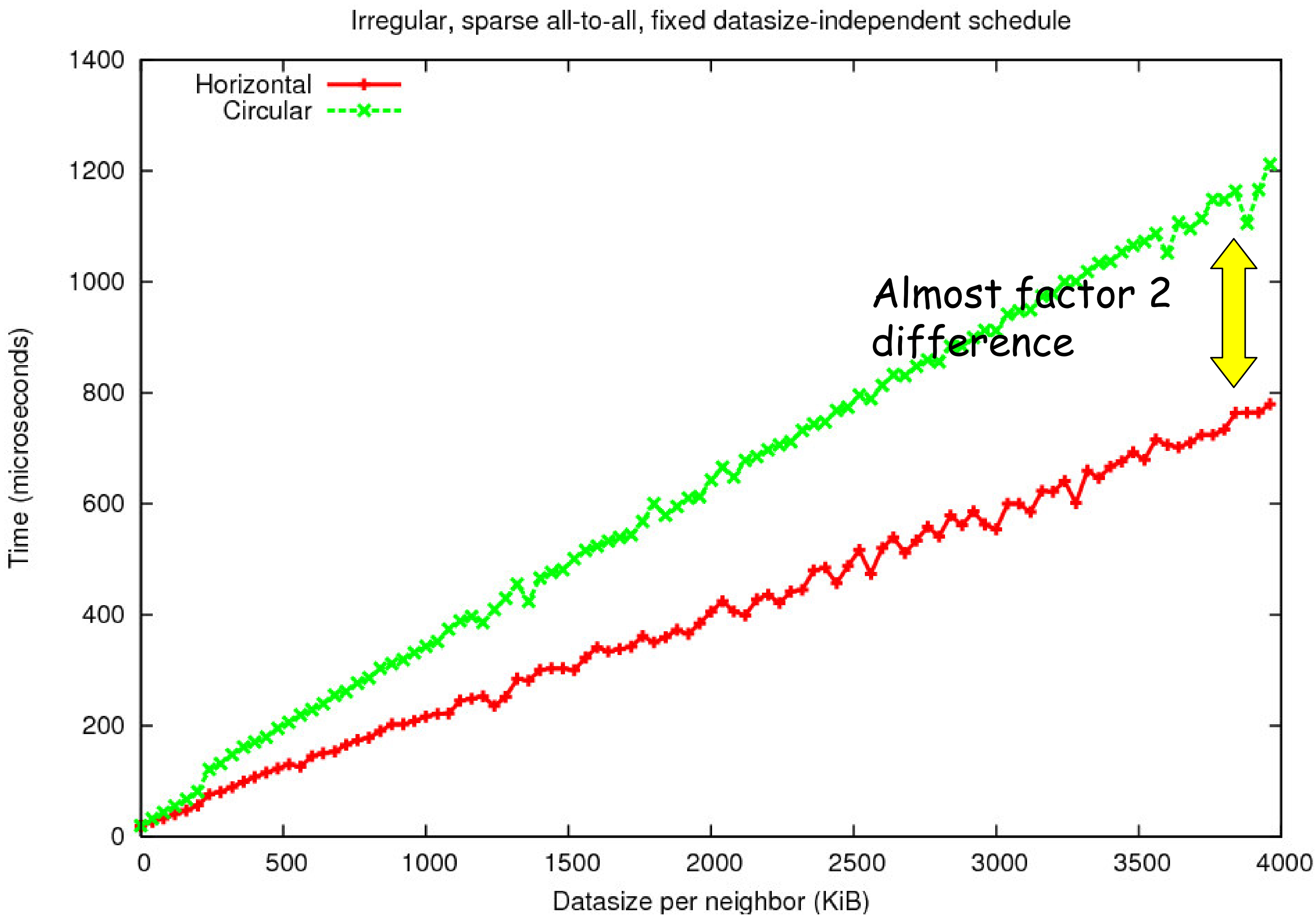
Heavy edges: more communication along these

Dimension based MPI_Neighbor_alltoallv()

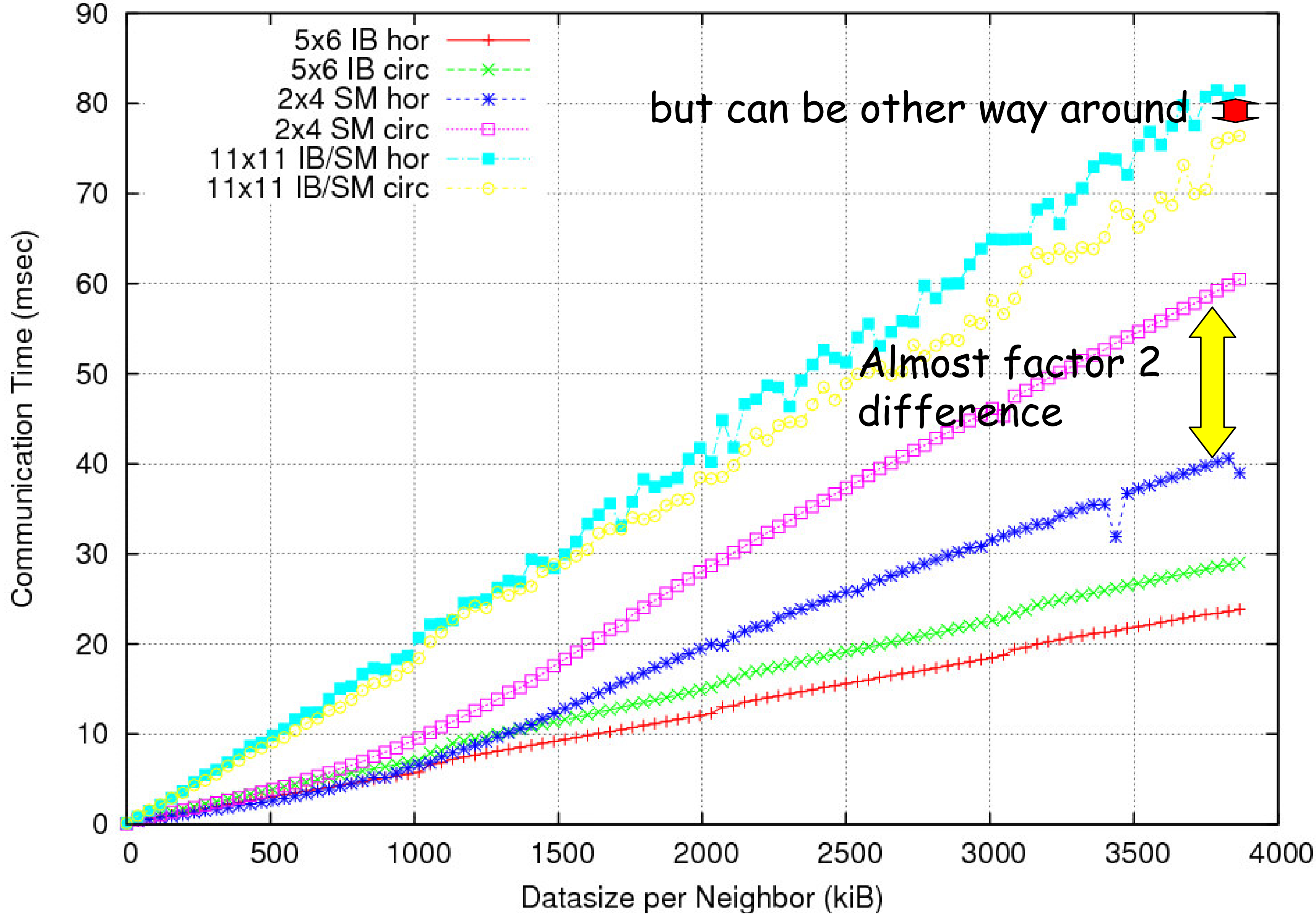
2 heavy rounds

4 heavy rounds

© NEC Laboratories Europe



© NEC Laboratories Europe



© NEC Laboratories Europe

Lesson 2:

Weights on neighbor-edges necessary to guide optimization

"Best" schedules:

Horizontal pattern: dimension based exchange

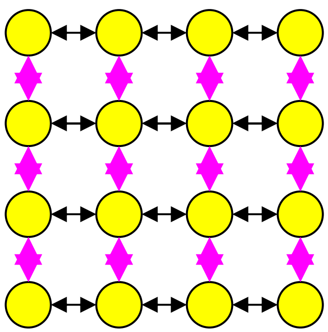
Circular pattern: left-right circular exchange

© NEC Laboratories Europe

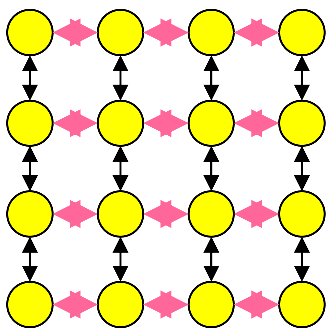
Experiment 3:sensitivity to mapping on non-homogeneous system

Horizontal, vertical, and circular pattern on 32 processes, mapped as 4x8 processes on SMP system.

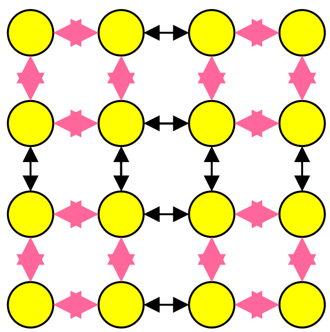
vertical



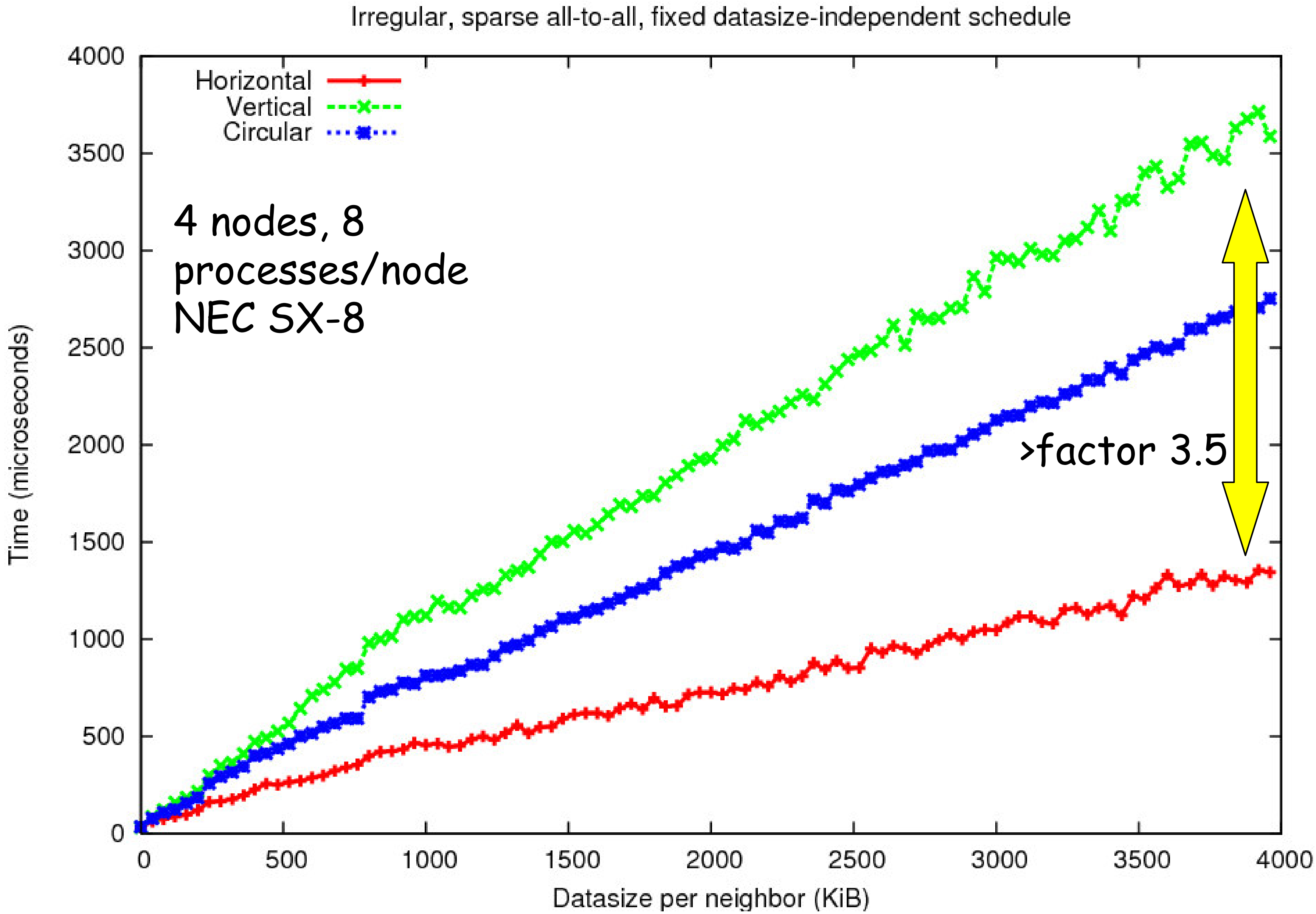
horizontal



circular



© NEC Laboratories Europe



© NEC Laboratories Europe

Lesson 3:

System topology must be taken into account in optimization

MPI library has the topology information to do this

© NEC Laboratories Europe

Summary:

Need for sparse collective support in MPI: usefulness, efficiency, suitability.

Proposed interface separates functionality and sparsity pattern information, makes it possible for MPI library to perform scheduling optimizations and tyke system topology into account.

Simple experiments supports this design

© NEC Laboratories Europe