Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Optimization Principles for Collective Neighborhood Communications

Torsten Hoefler, Timo Schneider
ETH Zürich
2012 Joint Lab Workshop, Argonne, IL, USA

Parallel Application Classes

- We distinguish five classes of applications with regards to their communication patterns:

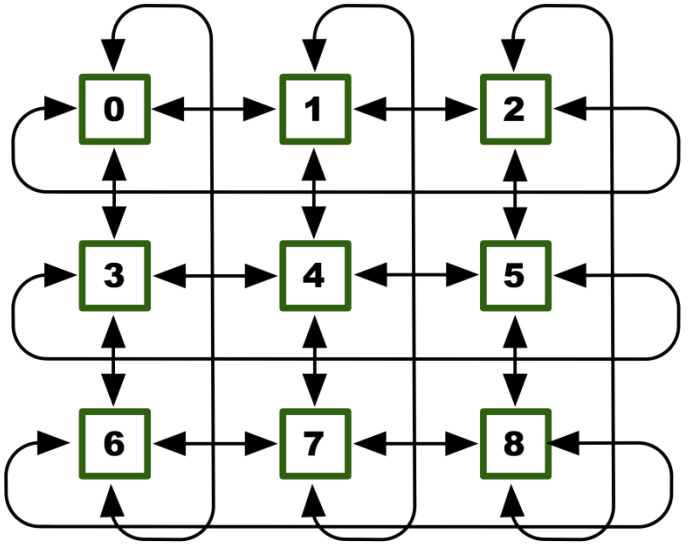
1. Compile-time static (fixed at compile time)
2. Run-time static (fixed after problem input)
3. Run-time flexible (changes slowly during runtime)
4. Dynamic (completely unstructured)
5. “Embarrassingly” parallel (insignificant)

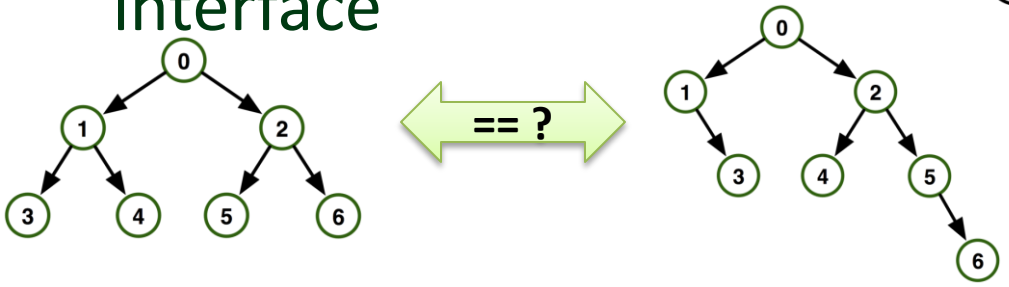
MPI-3.0 IS HERE!

- Ratified in September!
  - MPICH 3.0 released Nov. 13
- Many new features, e.g.:
  - MPIT (Tools) interface
  - New one sided operations
  - Shared memory windows (cf. EuroMPI'12)
  - Noncollective comm. Creation (cf. EuroMPI'11)
  - Nonblocking collectives (cf. SC07)
  - Neighborhood collectives (this work)
  - … and many more features!

MPI 3.0's Neighborhood Collectives

- Idea: "build your own collective"
  - MPI library optimizes it during runtime
  - Interesting challenges and opportunities
  - Utilizes process topology interface

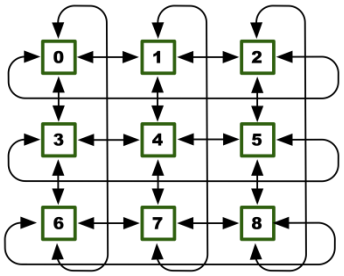


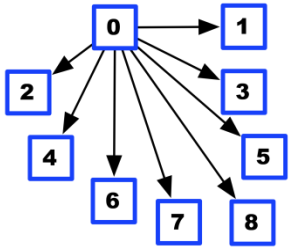


Hoefler et al.: The Scalable Process Topology Interface of MPI 2.2, CCPE 2012

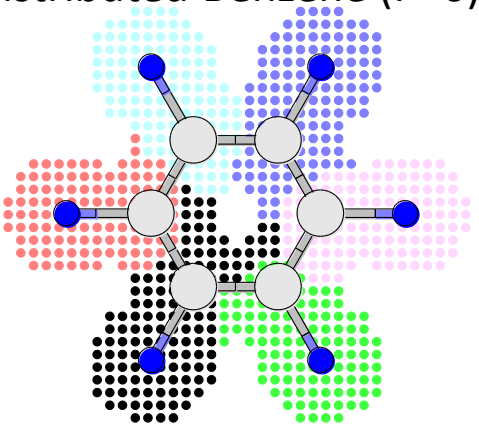
Process Topologies

- Constructors:
  - MPI_Cart_create()
  - MPI_Dist_graph_create()
- Topology mapping
- Accept info arguments
  - Provide optimization hints/assertions

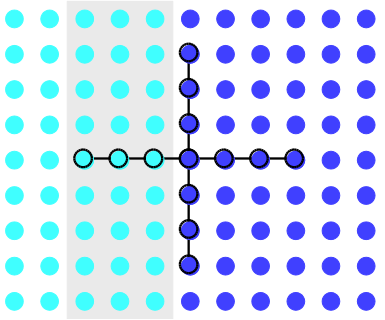




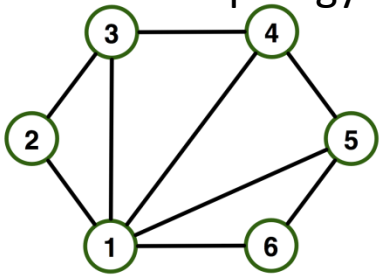
Distributed Benzene (P=6)



+ 13 point stencil



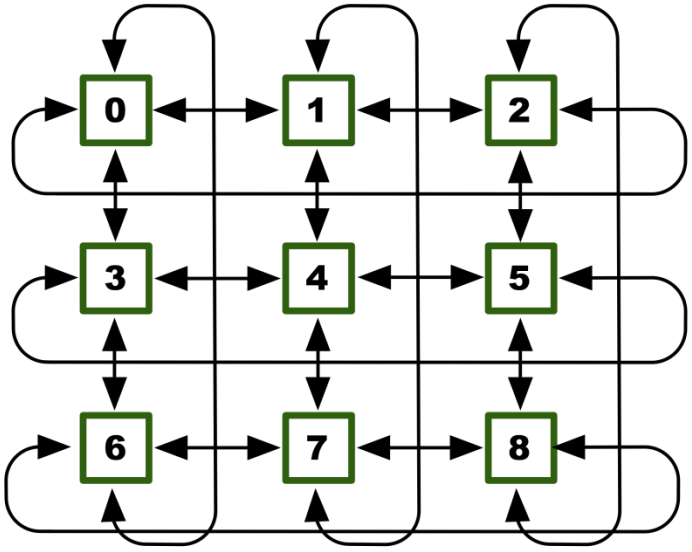
= Process Topology

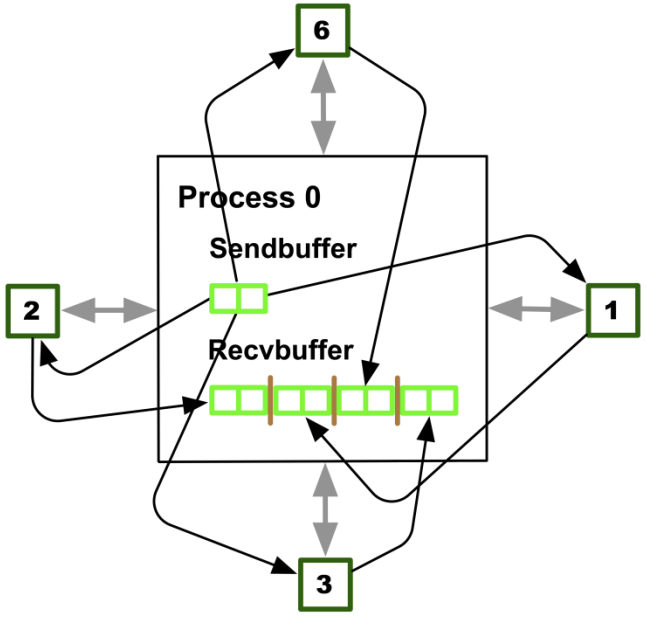


Hoefler et al.: The Scalable Process Topology Interface of MPI 2.2, CCPE 2012

NEIGHBORHOOD ALLGATHER

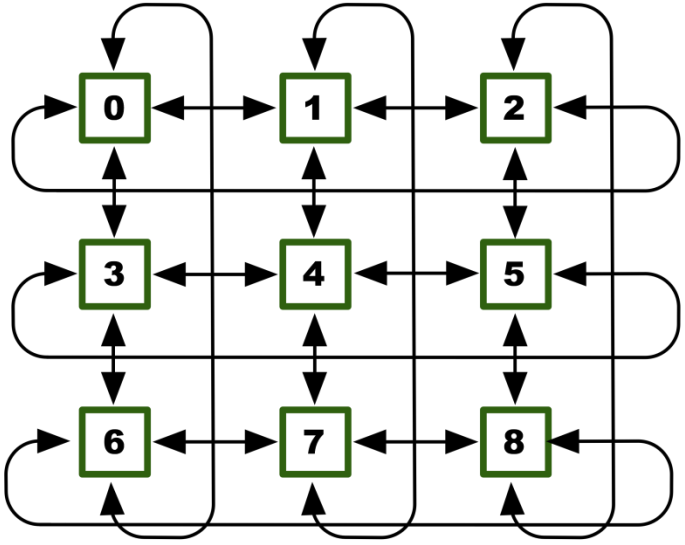
```
int dims[2]={3,3}, periods[2]={1,1};
MPI_Comm comm_cart;
MPI_Cart_create(MPI_COMM_WORLD, 2, dims, periods, 0, &comm_cart);
int sbuf[1], rbuf[4];
MPI_Neighbor_allgather(sbuf, 1, MPI_INT, rbuf, 1, MPI_INT,
    comm_cart);
```

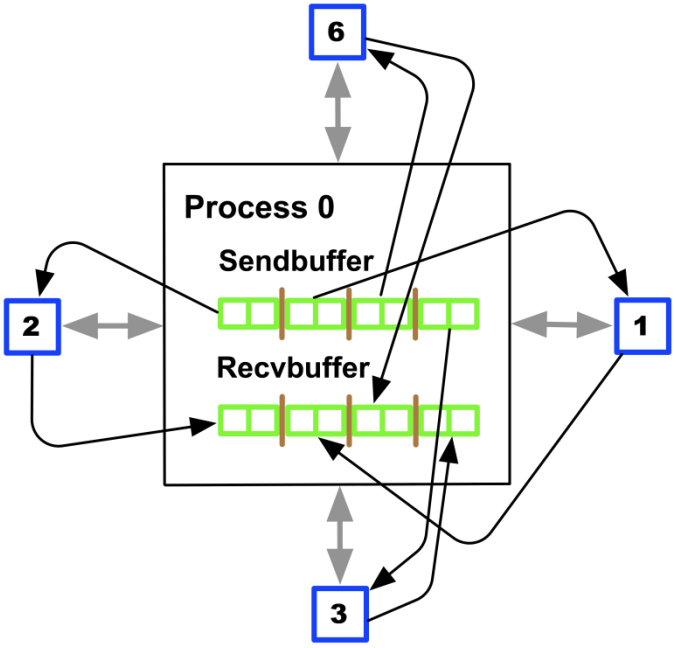




NEIGHBORHOOD ALLTOALL

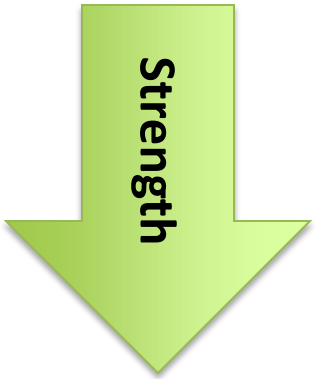
```
int dims[2]={3,3}, periods[2]={1,1};
MPI_Comm comm_cart;
MPI_Cart_create(MPI_COMM_WORLD, 2, dims, periods, 0, &comm_cart);
int sbuf[4], rbuf[4];
MPI_Neighbor_alltoall(sbuf, 1, MPI_INT, rbuf, 1, MPI_INT,
    comm_cart);
```





Communication Persistence

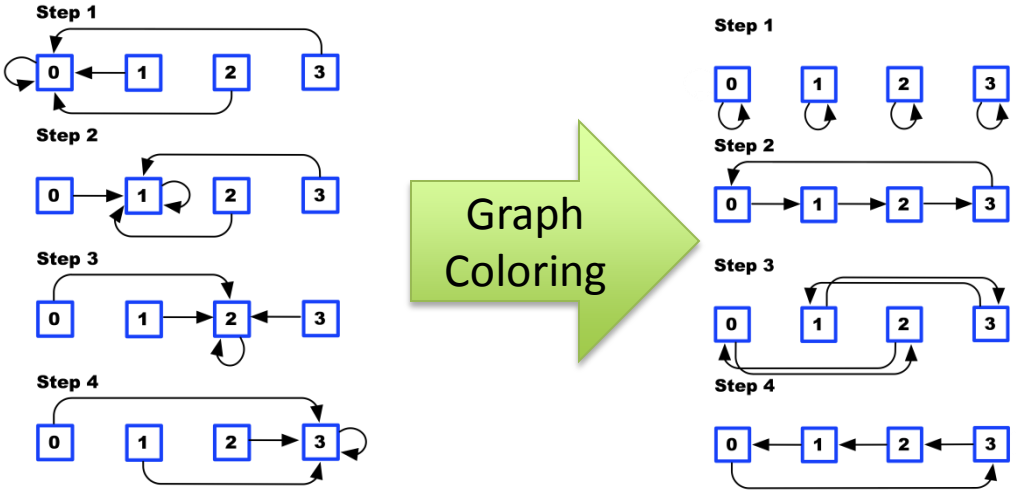
- Three persistence hierarchy levels:
  - Communication topology
  - Message sizes
  - Communication buffers
- Communicated via info arguments
  - Per collective and communicator
  - Side-effect: persistent collectives!



Persistent Communication Topology

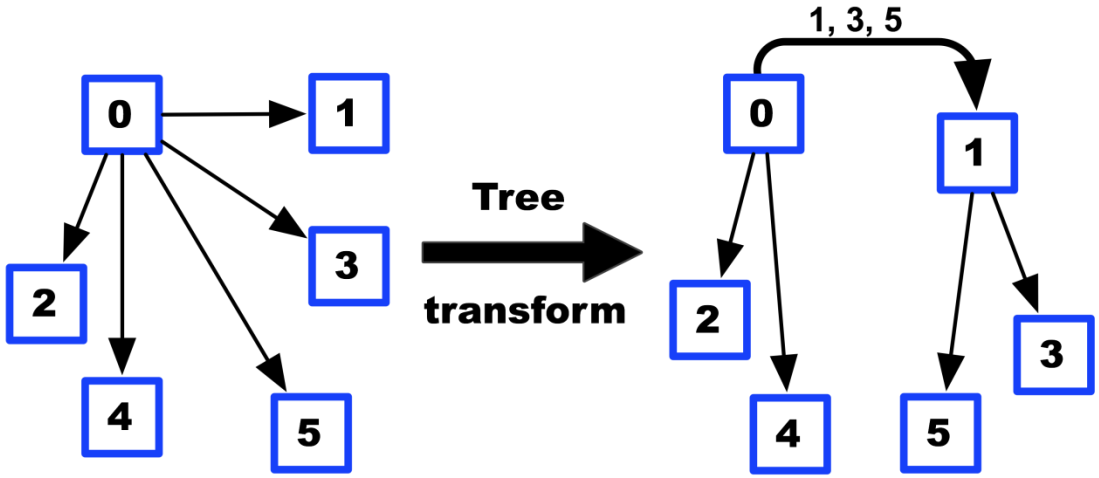
- Enables:
  - Fixed channel semantics (pre-connect)
  - RDMA synchronization trees
  - Communication scheduling

```
for(int i=0; i<p; ++i) {
  MPI_Irecv(..., src=i, ...);
  MPI_Isend(..., dst=i, ...);
}
MPI_Waitall(...);
```



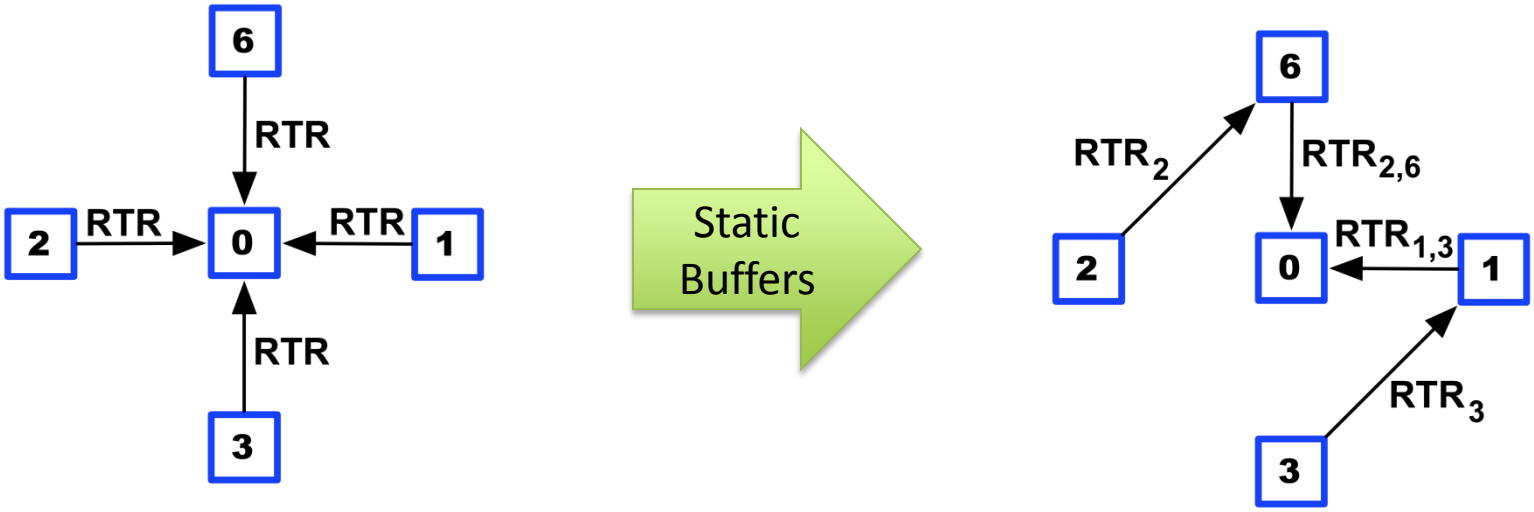
Persistent Message Sizes

- Enables:
  - Balance communications
  - Tree transformations



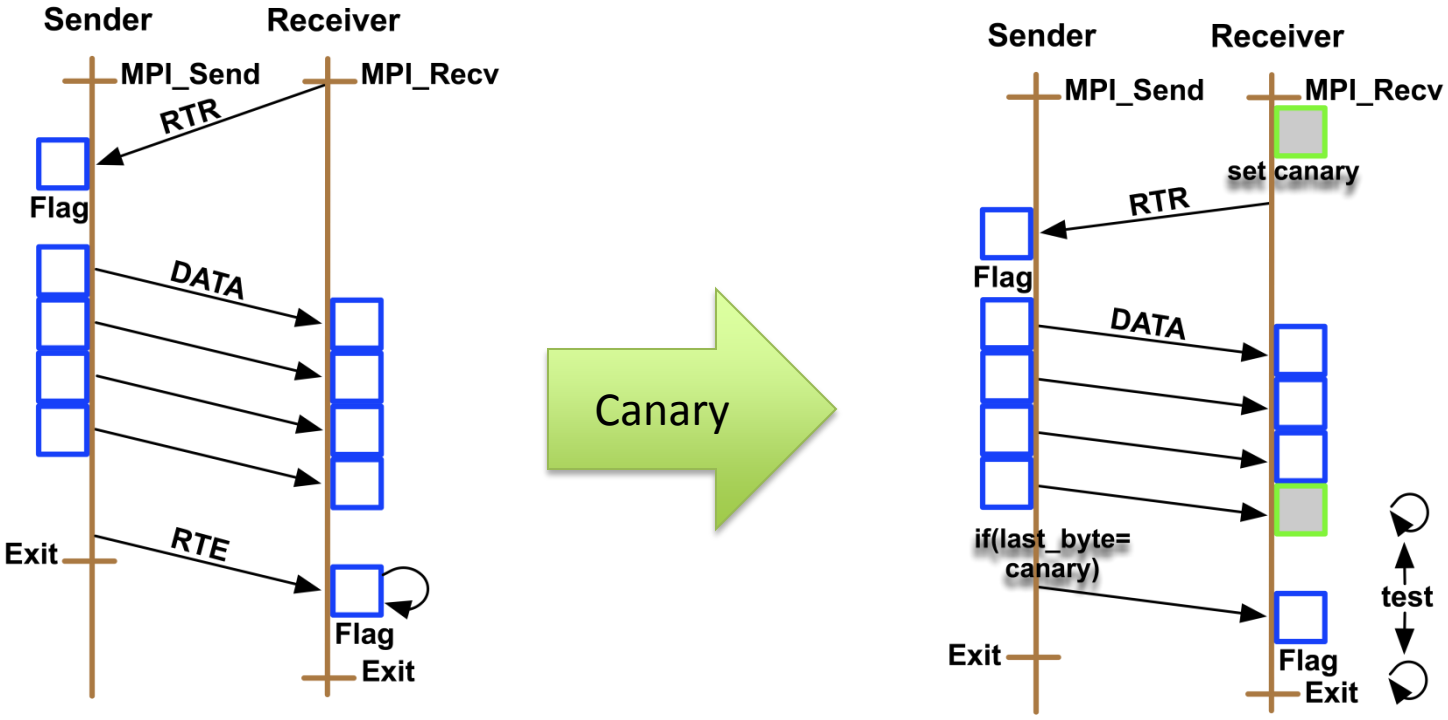
Persistent Message Buffers

- Enables:
  - Static (persistent) RDMA regions
  - Collective RDMA RTR protocols



Persistent Message Buffers

- Canary RTE protocol (system dependent)

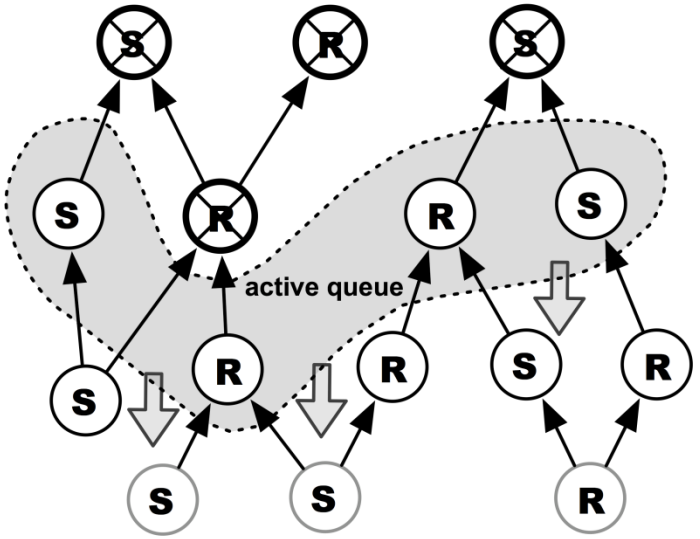


Implementation

- LibNBC acts as reference implementation
  - Naïve post all recvs, then all sends, waitall
  - We compare to hard-coded MPI versions!
- Two low-level interfaces:
  - Cray DMAPP
    - Canary protocol up to 64 bytes
  - XPMEM (shared memory)
    - Linux kernel module enables page sharing

cDAG – Communication DAG

- Express arbitrary communication relations as directed acyclic graph (DAG)
  - Easy translation from MPI calls
  - Enables DAG transformations
  - Highly optimized scheduled execution

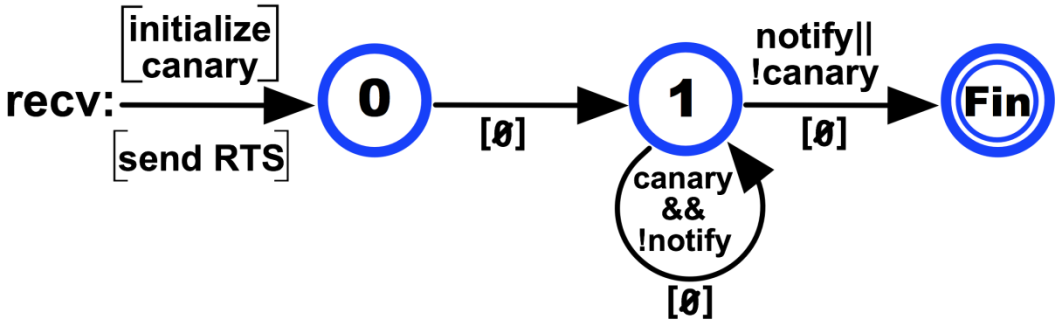


When to apply Optimizations?

- Persistent communication topology
  - During communicator creation
- Persistent message sizes
  - At first call of collective
  - Remember schedules/sizes for later calls
- Persistent communication buffers
  - At first call of collective
  - Remember schedules/buffers for later calls
  - Auto-tune?

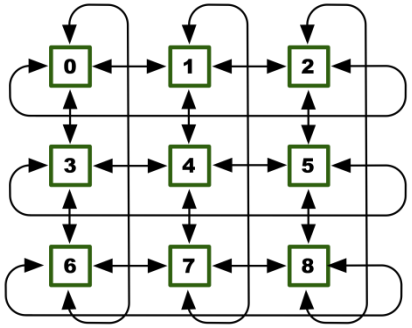
Protocol Drivers

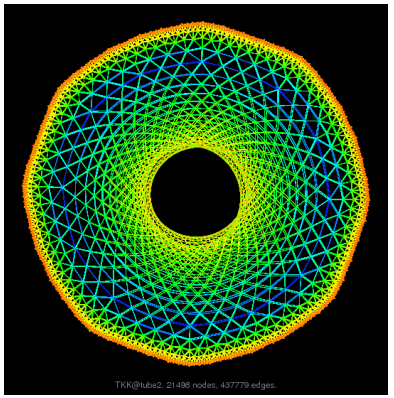
- cDAG transforms execution schedule
  - Scheduler executes on DMAPP and XPMEM
  - Represented by state machines
    - E.g., DMAPP small message recv:



Evaluation

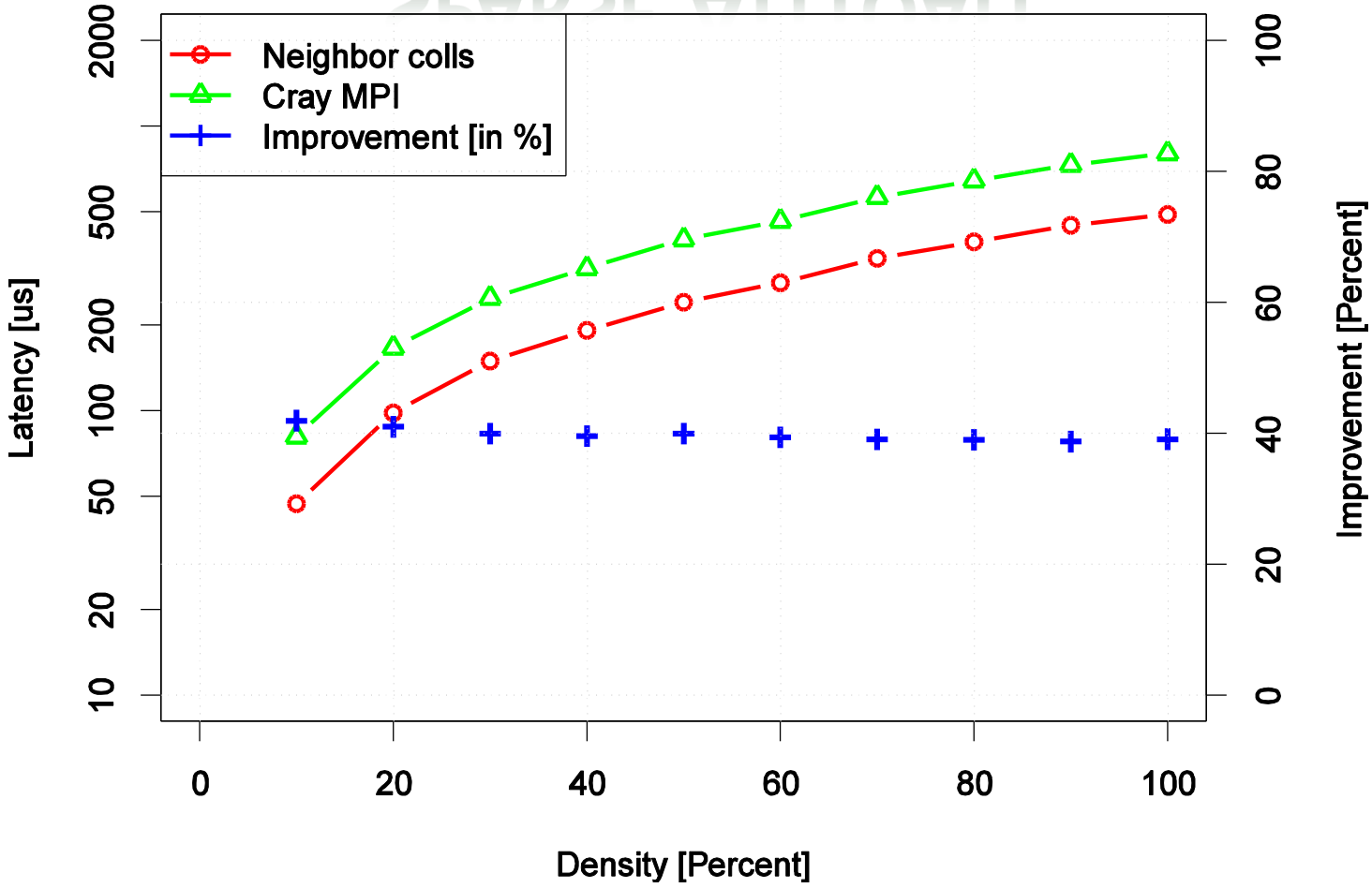
- Test system: Blue Waters test machine (JYC)
  - 50 nodes Cray XE6, ~1600 cores
  - Cray CCE 4.0.46
- Microbenchmark patterns
  - Sparse alltoall + Cartesian stencil
- Application patterns
  - WRF+ UFL sparse matvec





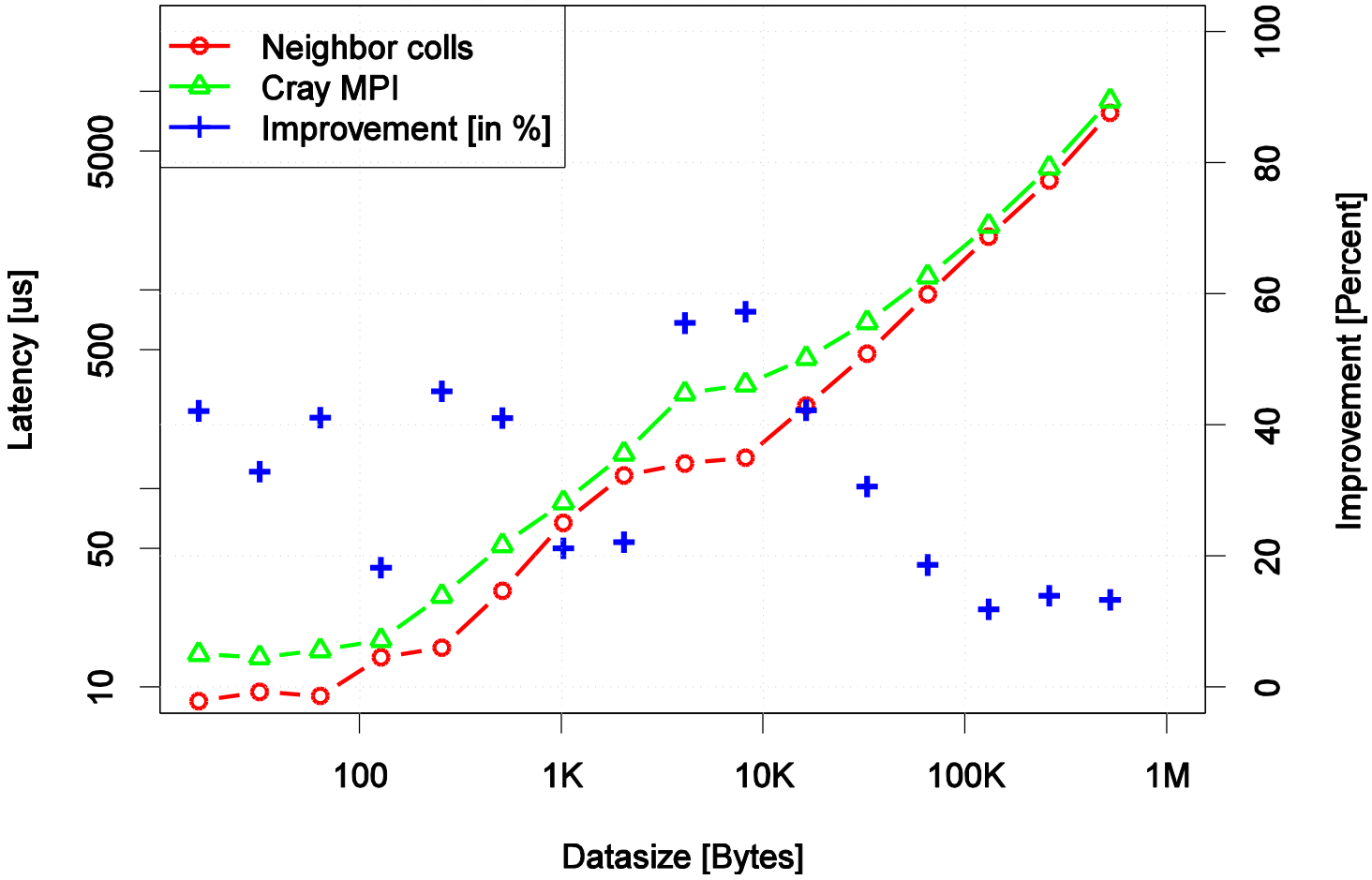
Credits: UFL collection

Sparse Alltoall



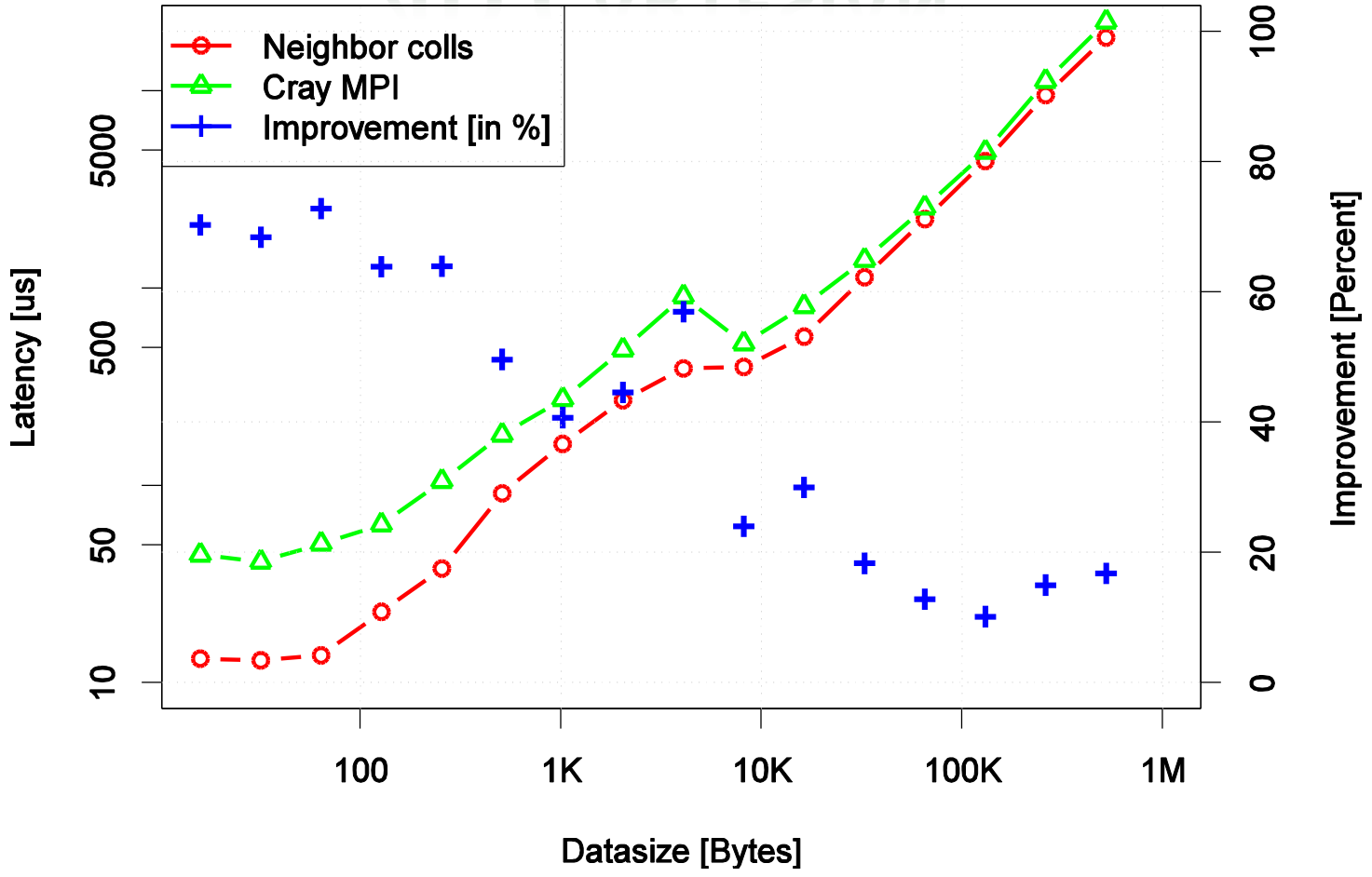
- 1024 processes, 16 Bytes per process

2D Cartesian



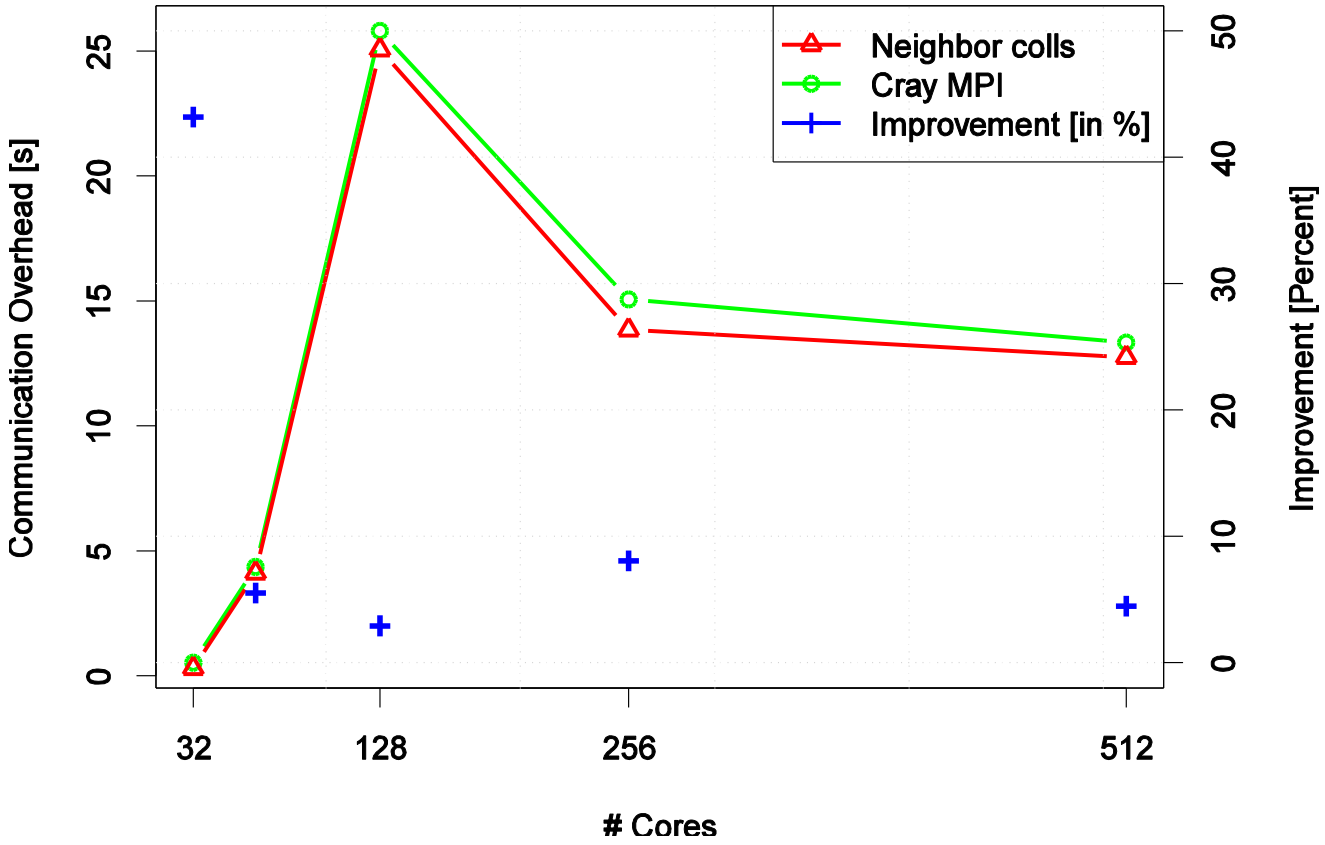
- 512 processes, varying size

4D Cartesian



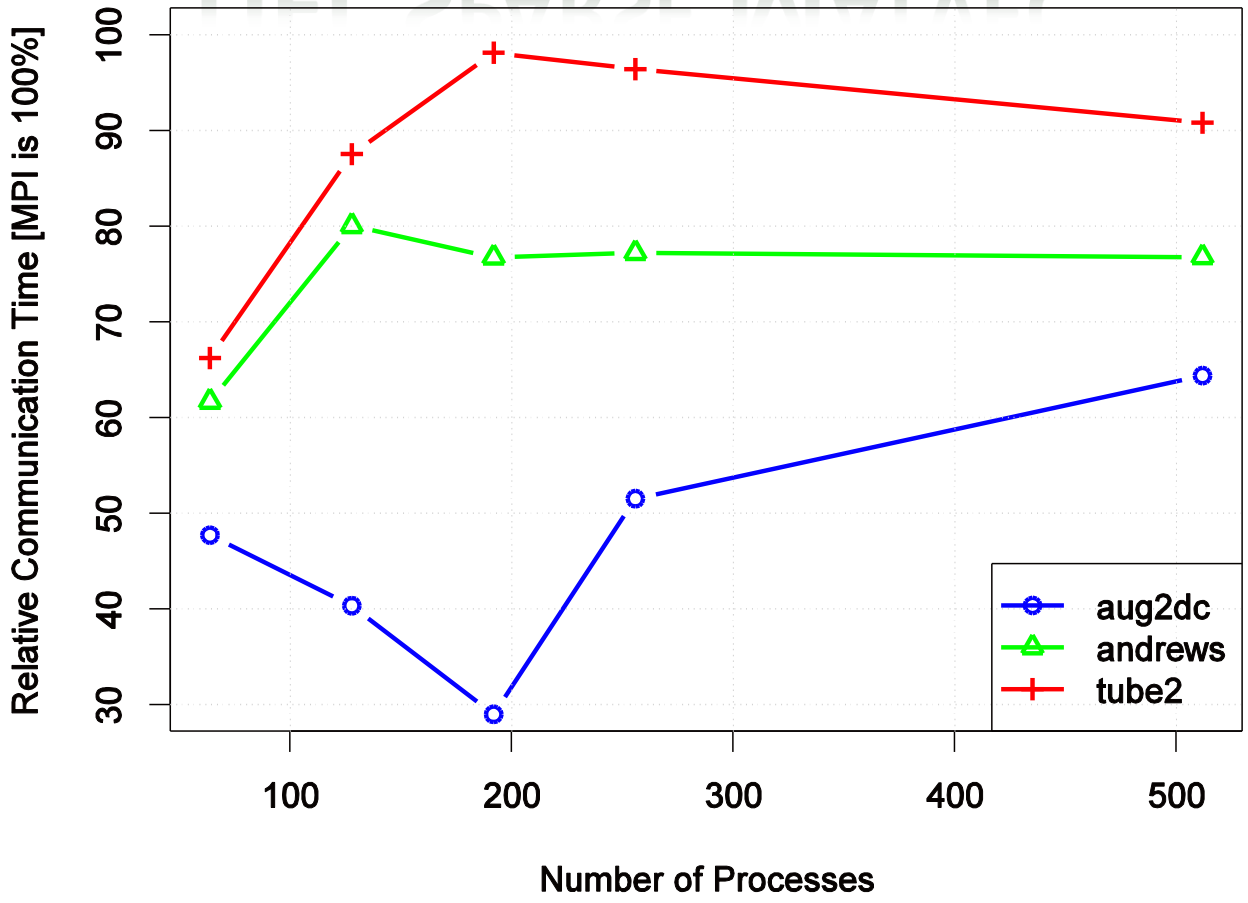
- 512 processes, varying size

WRF COMMUNICATION



- Em_b_wave input, five simulation days, 200k points
- Up to 40% improvement (14% app), average 7-10% (3-5% app)

UFL Sparse Matvec



- METIS partitioned, strong scaling

Discussion

- Most applications have 3-7 neighbors
  - [Vetter, Mueller, JPDC 2003]
- Some applications have up to 66 neighbors
  - [Kamil et al., TPDS 2010]
- Collective optimization is well understood
  - Very limited interface
  - Neighborhood collectives extend to runtime
  - Specialized hardware allows for optimizations
- Our scheme also enables standard-compliant persistent collectives

Collaboration Opportunities

- Use neighborhood collectives in MPI applications
  - Looking for users ☺
- Optimize neighborhood collectives in MPICH?
  - MPICH 3.0 offers an unoptimized version
- Other opportunities
  - Derive specification automatically (compiler)
  - Experiment with auto-tuning (feedback-driven online schedule transformations)

Acknowledgments & Questions

- The MPI Forum
  - Especially the collective WG!
  - Cray (Larry, Duncan, & Howard)

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

NSF

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

Sandia National Laboratories