Collective Operations and Topologies WG

May 2010 Meeting

Torsten Hoefler

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Update on NBC

- Received three thorough reviews
  - Thanks to Fab, Alexander, and Dave

- Will “clean” the document and get it ready to be released asap
  - Will use pdf diff tools to ensure consistency
  - Do we have editing guidelines for MPI-3 yet?

NBC: Implementation Issues

Rich mentioned some problems?

Discussion

(Non)Scalable Graph Topologies

- We added the distributed graph interface to MPI-2.2
  - Effectively replaces the old one
  - Is better in **all cases**!
- We postponed the deprecation
  - We should do it now
  - What is the Forum's opinion?

Sparse Collectives

- Worked with IBM (Sameer Kumar) on optimized implementation in DCMF
  - Trying to separate communication pattern (sparse colls) from buffer binding (persistent colls) and one-sided semantics
  - DCMF does all in one, synchronization based on collectives between iterations

DCMF Talk

Present IBM’s DCMF talk here!

What fits MPI best

- Three separate principles:
  1. Sparse collectives (SC)
     - Enables neighborhood definitions
  2. Persistent collectives (PC)
     - Enables static buffer binding
  3. One Sided collectives (OSC)
     - Enables looser synchronization
       - Didn't really talk about this yet and it might migrate to the (too busy?) OS working group

Step by step

- Dependency chain (for neighborhoods)
  - SC <- PC <- OSC
    - (PC could stand on it’s own)
- Step-wise process:
  - Propose SC
  - Propose PC
  - Propose OSC
- Forum decides where to stop ☺

SC: Progress

- Reference implementation exists in LibNBC
  - Well understood, non-blocking, proved useful in one application (TDDFT/Octopus)
- Proposal-draft finished
  - Soliciting initial feedback from Forum
  - Will be up for public comments soon

SC: Proposal

- Add as extension to Chapter 7
  - 7.6 Sparse Collective Communication on Process Topologies
- Why not in Chap 5 (Collectives)?
  - SC depends on Process topologies (PTs)
  - PTs are not introduced in Chap 5
  - Will back-reference to Chap 5

SC: MPI_Neighbor_gather

- int MPI_Neighbor_gather(void* sendbuf,
  int sendcount, MPI_Datatype sendtype,
  void* recvbuf, int recvcount, MPI_Datatype recvtype,
  MPI_Comm comm)

- Sends the **same** data to each neighbor
  - Vector variant for receiving different sizes

SC: MPI_Neighbor_alltoall

- int MPI_Neighbor_alltoall(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm)

- Sends **personalized** data to each neighbor
  - Vector variant for different size comms
  - W-variant for optimized DDT layouts

SC: MPI_Neighbor_reduce

- int MPI_Neighbor_reduce(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Op op, MPI_Comm comm)

- Vector variant for overlapping neighborhoods
- It's not directly applicable to stencils
  - Clear use-case needs to be found
  - We might decide to exclude it for now (?)

PC: Persistent Collectives

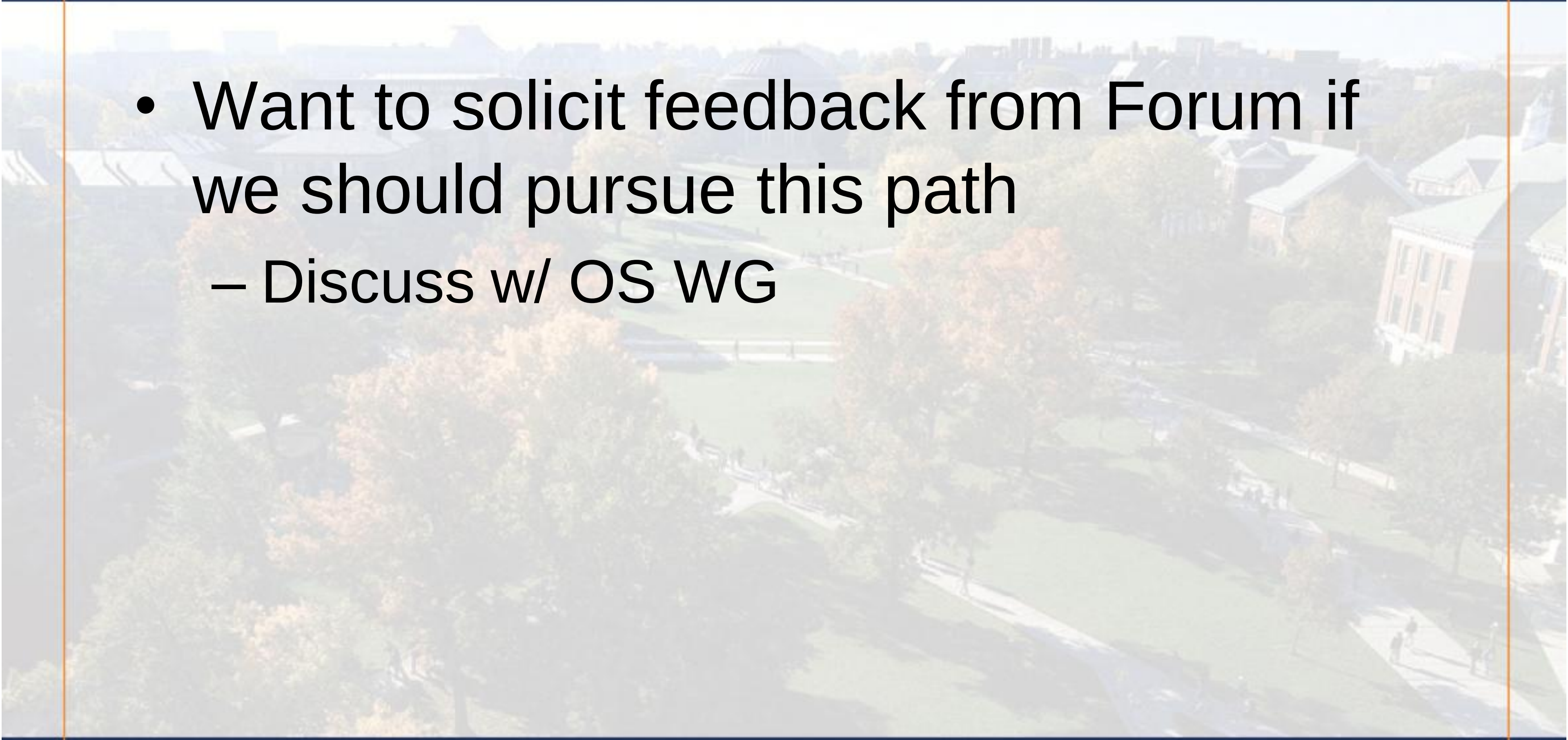
- They seem very useful – Tony?
  - Exploit temporal locality
    - Most (iterative) applications re-use buffers often
  - Really easy to specify

    -
      ```
      MPI_Alltoall_init(…, &req)
      MPI_Start(&req)
      /* computation */
      MPI_Wait(&req, MPI_STATUS_IGNORE)
      ```
  - Trivial to implement w/o optimization

OC: One Sided Collectives

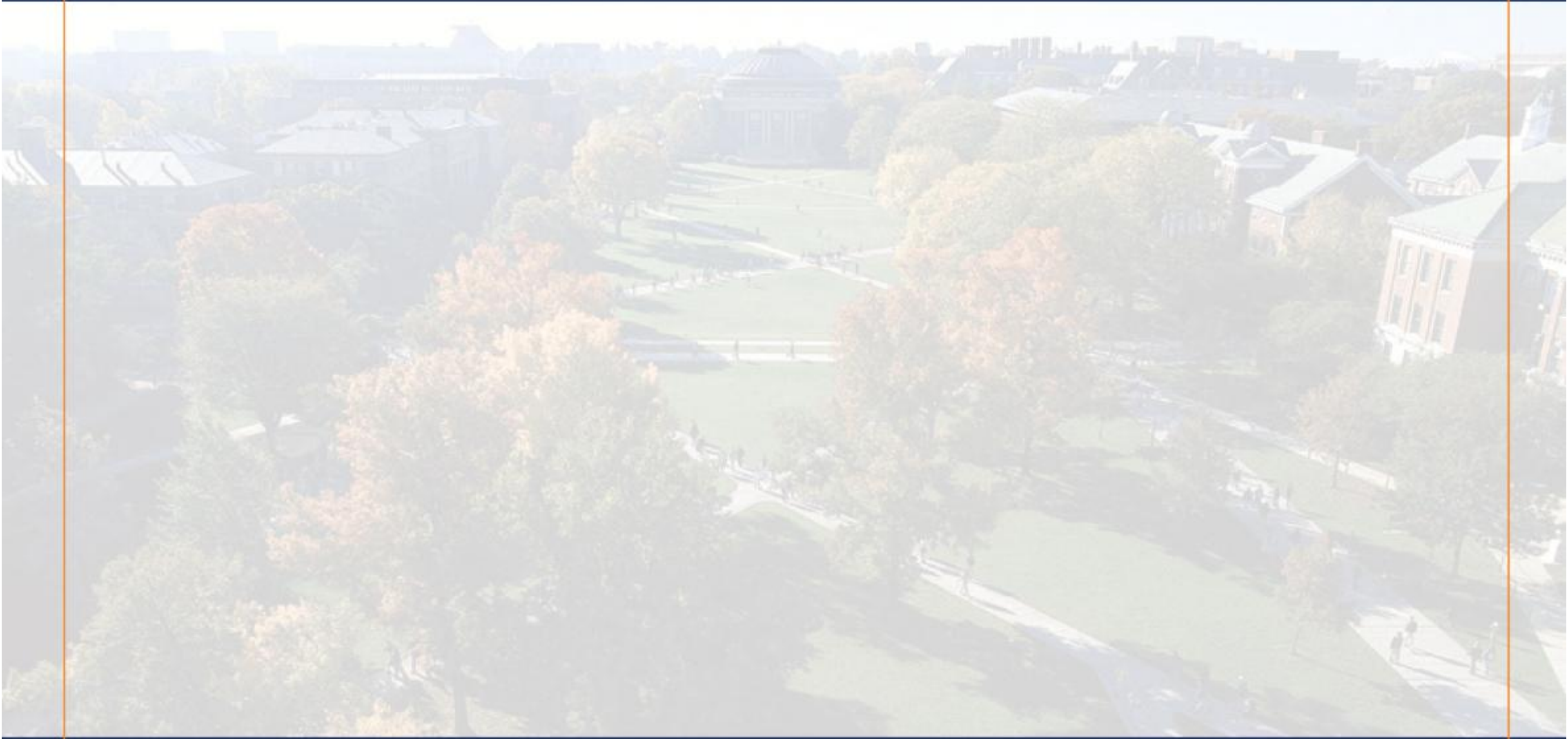
- Relax synchronization (cf. NBC)
  - Synch. Cannot be avoided!
- Buffers have to be ready all the time
  - Sender can just RDMA into recv-buffers
- Synchronization is tricky
  - When are the buffers valid? When can they be overwritten?
- DCMF implementation is very specific

OC: Discussion



- Want to solicit feedback from Forum if we should pursue this path
  - Discuss w/ OS WG

Questions/Discussion?



UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

illinois.edu