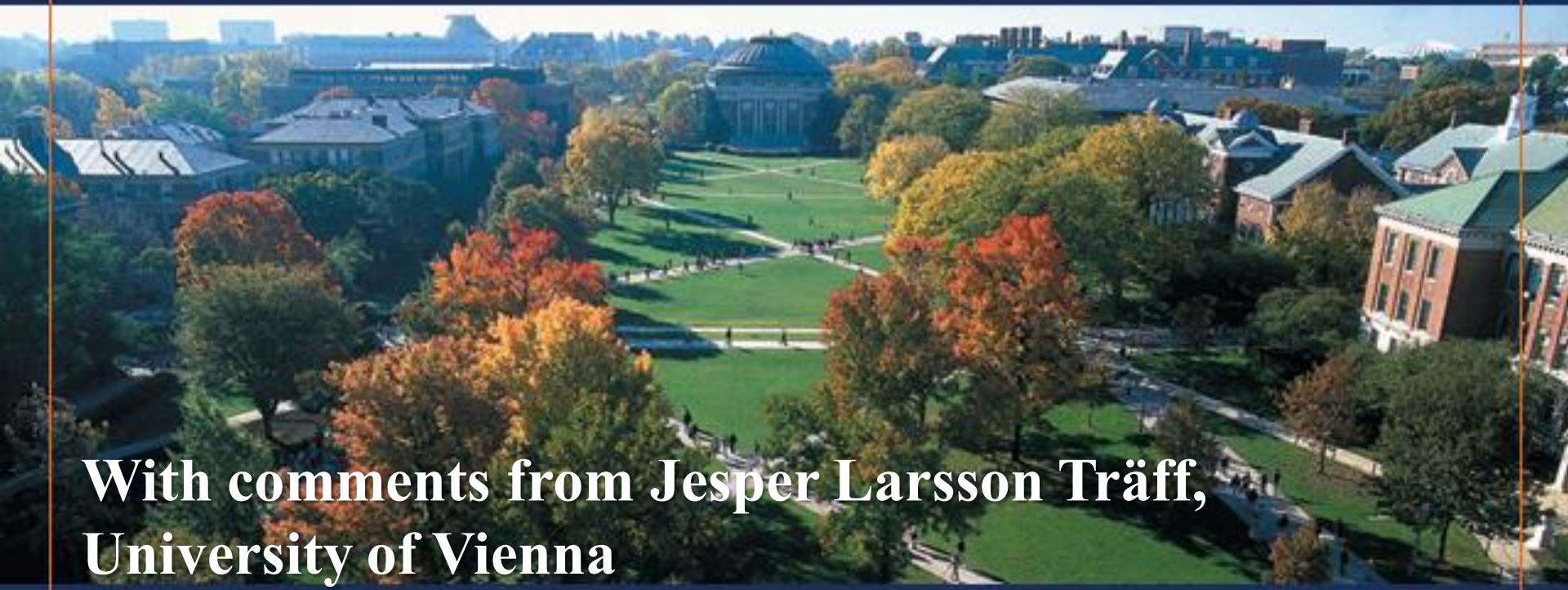
One Sided Proposal

Version 2

Torsten Hoefler



With comments from Jesper Larsson Träff, University of Vienna

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

What changed from v1 (high-level)

- Point-to-point windows dropped
  - Forum decided that user has to do memory management in “collective” windows
- Added collective memory allocation
  - cf. MPI_Alloc_mem(), more later
- Disentangled unlock semantics
  - Added separate flush() call
- Collective Operation registration

Collective Memory Allocation

- Allows to allocate symmetric memory
  - much simpler RDMA implementations
  - symmetric to MPI_Alloc_mem()

- MPI_Win_allocate(size, disp, info, comm, base, win)
  - accepts similar infos like win_create **and** alloc_mem
  - base is now an out argument instead of in
  - memory will be freed in MPI_Win_free()
  - Info arguments must be the same on all callers

Collective Op Allocation

- MPI_Rma_op_create(op, win)
  - collectively allocates operation in win
  - we have to either bind it to window or communicator (to enable libraries)
  - Pro window: can make use of special memory (symmetric allocation)
  - Con window: no registration for each window necessary (once per comm).
  - Straw Vote!
  - Should we return an MPI_Op or MPI_Rma_op handle?
  - Pro MPI_Rma_op: less confusion/user-error
  - Con MPI_Rma_op: no new datatype

Fetch and Add

- MPI_Get_accumulate(o_addr, o_cnt, o_type, rank, t_displ, t_cnt, t_type, op, win)
  - fetches value into o_addr and accumulates o_addr into t_displs
  - needs buffering to do so ☹
  - Do we want a third buffer to return value in?
  - Pro: very flexible, no buffering required (we should talk about an MPI_IN_PLACE option)
  - Con: at least one more argument (if we use same type as origin layout) -- straw vote!

Accumulate - Get

- MPI_Accumulate_get(o_addr, o_cnt, o_type, rank, t_displ, t_cnt, t_type, op, win)
  - less mighty than get_accumulate
  - sufficient for invertible bijective functions
  - potentially much faster (no buffering or additional arg.)

New predefined operations?

- we want fast compare&swap semantics
  - e.g.,
    - MPI_CAS_IF_LARGER
    - MPI_CAS_IF_SMALLER
    - … that for all MPI types ☹
  - Enables hardware optimizations (e.g., IB)
  - How to handle count >1 (or forbid it?)
  - Any input? Should we pursue (Straw Vote)

Relaxed Correctness Requirements

- every “erroneous” behavior becomes “undefined” behavior
  - allows for programs who really know what they are doing
  - Put/get is still **not** atomic!
  - Accumulate is (need to reconsider)
    - and even allows put emulation (MPI_REPLACE ☺)

Passive target multiple epochs!

- Allow multiple simultaneous access epochs in passive target mode
  - Basically, allow multiple locks per rank
  - Current model top update two ranks:
    - lock(1), update, unlock(1), lock(2), update, unlock(2)
  - Potential deadlocks (of course)
    - Not worse than p2p though!
    - Seems to be a tool issue

Flush the … window

- MPI_Win_flush(rank, win)
  - New point-to-point synchronization
    - replaces p2p windows ☺
  - all operations completed at the target (public window) when call returns!
- MPI_Win_flush_all(win)
  - well, flush all the … ranks

MPI_Alloc_mem()?

- Lift restriction that allows implementers to only allow passive target mode in memory returned by MPI_Alloc_mem()
- Could not find an example where this is necessary
- Limits portability of codes unnecessary

MPI_Win_unlock()

- Should be allowed to not flush() implicitly!
  - Breaks backwards compatibility!
  - Or do we want a new call (MPI_Win_unlock_noflush())
  - Straw Vote!

Supporting Cache Coherency

- Be quick, it won't be there for long ☺
- MPI_Rma_query(optype, win, model)
  - returns memory model for (win, op)
  - MPI_RMA_ONE
    - Public and private windows are the same (cache coherent)
  - MPI_RMA_SEPARATE
    - Public and private windows are separate! Current MPI-2.2 model.

Advice on p. 341

- Advice to implementers should be dropped!
  - "A high-quality implementation will attempt to prevent remote accesses to memory outside the window that was exposed by the process"
  - scalability problems!

Questions?