MPI-3 RMA Proposal 1

Torsten Hoefler and William Gropp

On behalf of the RMA working group

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

New Features at a Glance

- MPI_Win_allocate
- MPI_Win_create_allmem
  - MPI_Win_register
  - MPI_Win_deregister
- MPI_Get_accumulate
  - MPI_Accumulate_get
- MPI_Compare_and_swap

New Features Continued

- MPI_Win_lock_all
  - MPI_Win_unlock_all
- MPI_Win_flush
  - MPI_Win_flush_all
- MPI_Win_flush_local
  - MPI_Win_flush_local_all
- MPI_RMA_query
- erroneous->undefined

Win Allocate

- Collective Window Allocation
  - enables symmetric allocation for simple offset calculation

- MPI_WIN_ALLOCATE(size, disp_unit, info, comm, base, win)
  - int MPI_Win_allocate(MPI_Aint size, int disp_unit, MPI_Info info, MPI_Comm comm, void *base, MPI_Win *win)

Win Create Allmem

- Creates a window that includes all the process' memory
  - Memory needs to be registered before being accessed remotely
- int MPI_Win_create_allmem(MPI_Info info, MPI_Comm comm, MPI_Win *win)
  - int MPI_Win_register(MPI_Win win, void *base, MPI_Aint size)
  - int MPI_Win_deregister(MPI_Win win, void *base)

Get Accumulate

- Similar to fetch&add

- int MPI_Get_accumulate(void *origin_addr, void *result_addr, MPI_Datatype datatype, int target_rank, MPI_Aint target_disp, MPI_Op op, MPI_Win win)
  - Supports MPI_NO_OP

Accumulate Get

- Reverse of Get Accumulate
  - Could easily be emulated by applying the op locally
  - Straw-vote for/against inclusion

- int MPI_Accumulate_get(void *origin_addr, void *result_addr, MPI_Datatype datatype, int target_rank, MPI_Aint target_disp, MPI_Op op, MPI_Win win)

Compare and Swap

- int MPI_Compare_and_swap(void *origin_addr, void *compare_addr, void *result_addr, int target_rank, MPI_Aint target_disp, MPI_Win win)

- No datatype and op argument
  - Have special datatype and only bitwise equal as op ☹
  - Limited by today’s hardware implementations

- Needs query function for size of the type
  - Maybe create and free too?

Win Lock All

- Locks all target processes in a window
  - Optimization for a simple loop

- int MPI_Win_lock_all(int assert, MPI_Win win)
  - Needs int MPI_Win_unlock_all(MPI_Win win)

- Did not include MPI_Win_lock_group
  - No use-case

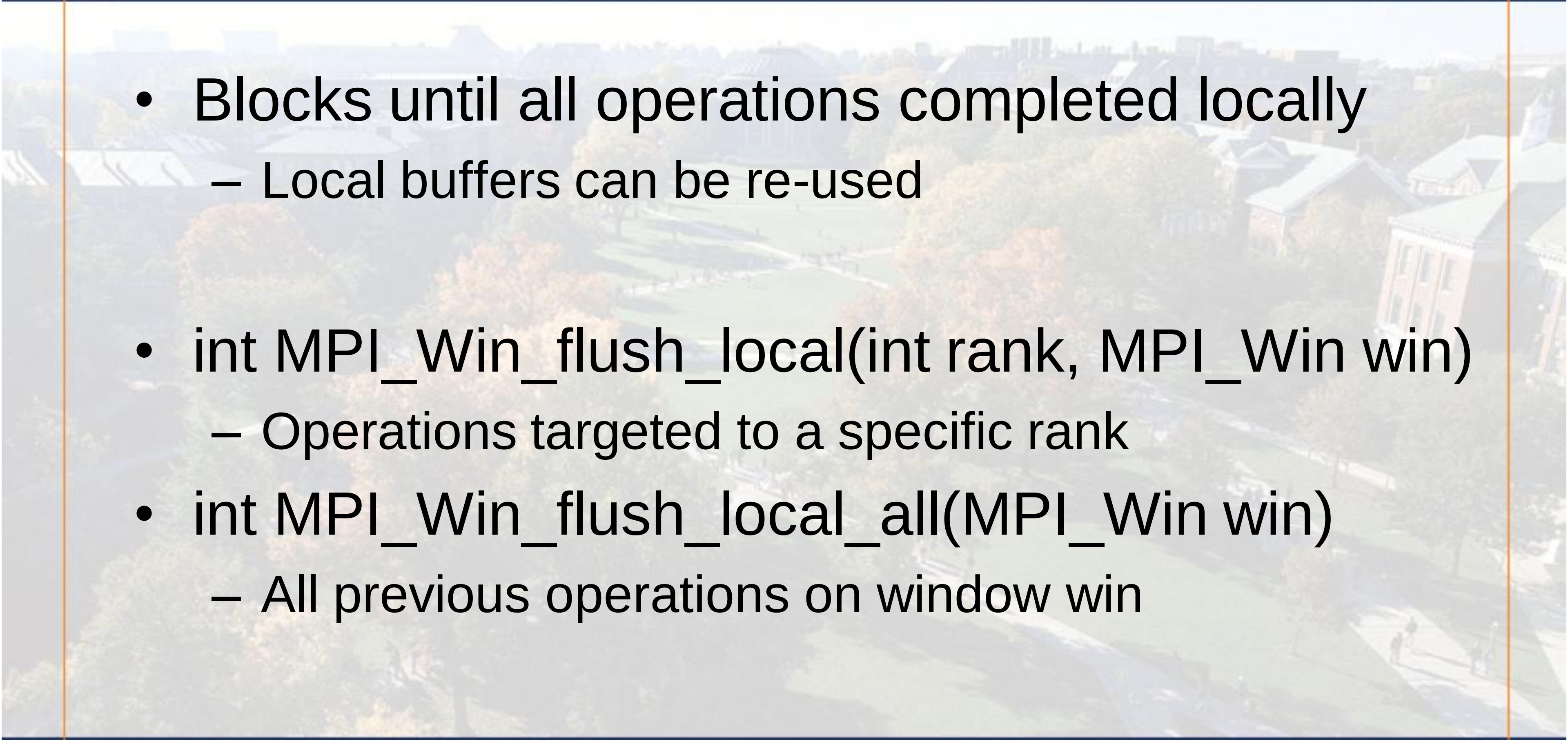
Win Flush

- Blocks until all operations completed remotely

- int MPI_Win_flush(int rank, MPI_Win win)
  - Flush a specific rank

- int MPI_Win_flush_all(MPI_Win win)
  - Flush all ranks

Win Flush Local



- Blocks until all operations completed locally
  - Local buffers can be re-used

- int MPI_Win_flush_local(int rank, MPI_Win win)
  - Operations targeted to a specific rank
- int MPI_Win_flush_local_all(MPI_Win win)
  - All previous operations on window win

RMA Query

- Query remote memory consistency
  - Separate for each operation

- int MPI_RMA_query(int optype, MPI_Win win, int *model)
  - Returns either MPI_RMA_SEPARATE (MPI-2) or MPI_RMA_ONE (public window = private window)
  - MPI_RMA_ONE changes semantic rules 5+6

RMA Query Continued

```
 Process A:                         Process B:
                                   window location X
                                   MPI_Win_lock(EXCLUSIVE,B)
                                   store X /* local update to copy of B */
                                   MPI_Win_unlock(B)
MPI_Barrier                        MPI_Barrier
MPI_Win_lock(EXCLUSIVE,B)
MPI_Get(X) /* ok, read from window */
MPI_Win_unlock(B)
```

RMA Query Continued

```
Process A:                          Process B:
                                    window location X
                                    MPI_Win_lock(EXCLUSIVE,B)
                                    store X /* local update to copy of B */
                                    MPI_Win_unlock(B)
                                    MPI_Win_flush(B)
MPI_Barrier                         MPI_Barrier
MPI_Win_lock(EXCLUSIVE,B)
MPI_Get(X) /* ok, read from window */
MPI_Win_unlock(B)
```

RMA Query Continued

Process A:	Process B:
	window location X
MPI_Win_lock(EXCLUSIVE,B)	
MPI_Put(X) /* update to window */	
MPI_Win_unlock(B)	
MPI_Win_flush(B)	
MPI_Barrier	MPI_Barrier
	MPI_Win_lock(EXCLUSIVE,B)
	load X
	MPI_Win_unlock(B)

Bonachea's and Duell's criticism

- Window creation is collective
  - hinders efficient exposure for local objects
  - no "sparse" communication

- MPI_Win_create_allmem
  - Well suited for automatic memory management
  - Good as compilation target

Bonachea’s and Duell’s criticism

- Exposed memory must be MPI_Alloc_mem()’d
  - no exposure of static memory or stack-variables
  - alloc_mem might be limited by the implementation

- Also addressed in MPI_Win_create_allmem
  - Can register stack
  - Dangerous though! Undefined results if stack is out of focus.

Bonachea's and Duell's criticism

- Forbids conflicting get/put (or local load/store) accesses to same memory
  - really hard to track for compilers (halting problem?)
  - Easy source of bugs in user codes

- Addressed in the proposal 1
  - outcome is undefined

Bonachea's and Duell's criticism

- Window's memory may not be updated by remote gets and local stores concurrently
  - simplifies MPI implementation significantly
  - seems very artificial and suboptimal from user's perspective

- Addressed in the proposal 1
  - outcome is undefined

Bonachea’s and Duell’s criticism

- Overlapping memory regions of multiple windows can be created but not be used
  - “concurrent communications may lead to erroneous results”

- now also “undefined”
  - [is this much better than erroneous?]
  - But these applications are likely to use a single “allmem” window, so problem avoided

Bonachea's and Duell's criticism

- Passive target RMA ops only lock a single process during an epoch
  - ops from one source to different targets are serialized
  - one window for each target to enable concurrent access?
    - scalability limitation
- MPI_Win_lockall

Comparison to ARMCI

- Similar to MPI_RMA_ONE (if available)
  - ARMCI does not support non-CC systems without an additional activity (thread/process/callback)
- Local completion is different
  - Either blocking, implicit handles or Test/Wait(all)
  - Proposal 1 is similar to “implicit handles”
- Remote completion similar
  - (all)Fence == Flush(_all)
- Ordering with collectives
  - ARMCI_Barrier combines barrier + allfence

Comparison to ARMCI cont.

- ARMCI_Malloc for exposed memory
  - Collective and local; no register (in an undocumented version)
- ARMCI_RMW
  - Similar to Get_accumulate
  - No compare-and-swap
- ARMCI_Lock/Unlock
  - Lock special synch. objects
  - Proposal 1 doesn’t include user-defined locks
- Limited set of collectives
  - Supports MPI collectives

Co-Array Fortran

- RMA through co-arrays
  - Explicit collective allocation
- Execution divided in segments (delimited by “image control statements”)
  - (non-volatile) operations in a segment are unordered
  - Exceptions are values with “atom” or volatile argument
- Image control statements:
  - Sync all, sync images, sync memory, lock/unlock
- Can be implemented with proposal 1
  - Active messages might be necessary

Unified Parallel C

- Explicit collective allocation
- Relaxed and strict access
  - – Relaxed accesses are unordered
    - Exception are conflicting accesses to the same memory which appear in program order
  - – Strict accesses are ordered (program order)
- Synch operations:
  - – upc_fence, upc_(notify,wait), upc_barrier, upc_(un)lock
  - – flush, (nonblocking) barrier, barrier, <no user locks>

Comparison Summary

- Proposal 1 covers most features from existing RMA systems
  - No remote completion of individual operations
    - Can often be emulated with flush
  - No user-defined locks
    - Can be emulated with compare and swap
- Proposal 1 can efficiently simulate other RMA systems
  - AMs seem to be missing but they can be simulated with point-to-point

Executive Summary

- Proposal 1 fixes known deficits of MPI-2 RMA and can be implemented on non-CC architectures
        - Part of the complexity is pushed to the user
    - Enables the use of MPI-RMA for new important application domains
        - E.g., Graph computation, well, AMs are missing ☺
- Proposal 2 has further extensions
    - Individual operation completions among others

Thanks!

Please review the proposal!

Questions?

https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/RmaWikiPage