## RMA Issues and Updates

1. For shared memory, additions or changes are needed to indicate when sync routines must ensure that load/stores see the “correct” data (as opposed to put/get/accumulate). See # 456 and the earlier [#437](https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/437). The key issues here are (a) to define behavior rather than implementation (which was a flaw in the early proposal), (b) only describe the behavior of MPI, not of the surrounding C or Fortran code or the processor, and (c) describe the behavior of the RMA synchronization and flush routines with respect to load/store operations on shared memory windows.

The key problem is illustrated by this example:

|  |  |
| --- | --- |
| Process 0 | Process 1 |
| Store into A |  |
| Win\_fence | Win\_fence |
|  | Load from A |

Here, “A” is a location in the shared memory window. The question is: Does the load of A in process 1 retrieve the value stored by process 0? There are two parts to this question:

1. Does the Win\_fence act as a barrier, i.e., is it known that process 1 cannot exit the fence until process 0 enters it?
2. By the time process 1 exits the fence, has the store into A on process 0 become visible to process 1?

We know that Win\_fence need not be a barrier, and in fact this can be important for performance in implementations, particularly for situations demanding low latency. Similar rules apply to PSCW and to lock/unlock.
However, this situation is most similar to the case for lock/unlock when the target process is the calling process (e.g., a process accessing itself). In this case, the MPI standard already requires that the MPI\_Win\_lock act as if the call blocks, so that local loads and stores (local refers to the process issuing the operations) may be used instead of RMA Put/Get/Accumulate and friends. Thus, one approach is to apply the same thinking to shared memory windows; since loads and stores can be performed by other processes, the RMA synchronization routines will need to act *as if* they are blocking *when applied to a shared memory window*.

The new flush routines also need defined behavior with respect to loads and stores in shared memory windows. These are likely to be memory barriers that do not return until the store operations have “completed”. Note that only ensures that the values are visible to other processes that access the shared memory. We should define this in terms of the behavior, not an ill-defined term such as “memory barrier”. The flush routines provide no guarantees as to synchronization with other processes ((a) in the above).

On specifying behavior, not implementation, note that fancy hardware might trap loads or stores to the memory rather than forcing the code to block (consider a distributed shared memory operation).
In addition, the standard shouldn’t be a place that describes all of the user issues in using shared memory; the issues are subtle and difficult, and its better to leave those to other documents rather than try to add something short (and likely incorrect) to the MPI document.

See an update from Rolf, [#456](https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/456) , which replaces #437, 413, 436, 435, 429, 434.

1. Make clear that “public and private copies are identical” does not mean that Win\_sync etc. are not required. See [#436](https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/436) and #456. The suggested text may be ok. My concern is that in the unified model, the public and private copies are the same physical memory, even though updates may not immediately appear. It might be best to remove the “are identical” and use the suggested text, but possibly add another note that the expected implementation is that the physical memory is the same location but that synchronization operations such as Win\_sync may be needed to ensure that updates are visible to other processes.

Open issues, by ticket:

1. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/458> . A minor clarification to undefined “outcome” to indicate that the resulting data is undefined, not other unspecified behavior. A reasonable fix. Status: MPI-3 Errata, scheduled for vote.
2. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/456> . An omnibus ticket, replacing #437, 413,436, 435, 429, and 434. This is the “how are loads and stores to shared memory synchronized” ticket. The description of the ticket is confusing and mixes shared memory issues with other RMA issues. The goal of the ticket was to be an MPI-3 errata, and to require small changes to the text. This may not be the best solution.
3. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/439> . Adds versions of flush and unlock (for the passive target mode) that provide a notification counter that can be queried at the target; there is a wait to block until notification. This can be done with existing message passing calls, but the claim is that this can be done with lower overhead. Adds 6 functions.
4. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/437> . Obsolete. Replaced by 456.
5. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/436> . Makes it clear that in the unified model, the public and private copies *eventually* become the same (e.g., the physical memory is really the same, but the updates from different sources aren’t necessarily immediately visible to other accessors). Status: MPI-3 errata; included in #456.
6. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/435> . Obsolete. Replaced by 456.
7. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/434> . Obsolete. Replaced by 456.
8. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/433> . Make it clear that MPI\_Win\_attach in Fortran only permits simply contiguous buffers. Note that the Fortran statement about “one can pass the first element of a memory region or a whole array” assumes, incorrectly, that in Fortran, the address of such an element is passed (it may be possible to guarantee this with the correct Fortran binding). Thus, the text needs to be corrected. Otherwise, a reasonable errata. Status: MPI-3 Errata.
9. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/429> . A small clarification on the meaning of assertions such as MPI\_MODE\_NOCHECK when a shared memory window is involved. Included in #456.
10. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/416> . Change the default accumulate\_ops from same\_op\_no\_op to none; add new values to give more control. Note that the statement of the ticket has an error, detected by Rajeev. Status: MPI-4, feedback (and corrections) needed.
11. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/413> . This is a passed ticket (MPI-3 Errata). It says that MPI\_WIN\_SYNC can be used to synchronize loads and stores to a shared memory window. It also includes an example (Danger! The example assumes that the statements are executed in the order given, which is probably true from the use of MPI\_F\_SYNC\_REG, so this is a Fortran-only example). See #456.
12. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/412> . Adds support for RMA on intercommunicators. Status: not ready, but a 5-1-1 straw vote in favor of a more detailed proposal.
13. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/401> . Add MPI\_IN\_PLACE for Get\_accumulate. Status: not ready.
14. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/398> . Add fine-grain remote completion. The ticket has no embedded description or even summary; you have to read the full PDF proposal. The ticket was discussed at length and the WG was not in favor. However, it suggests that either a nonblocking Flush (Iwin\_flush) or something called “Jim’s WIN\_TEST” be considered. Status: awaiting specific proposals for those last two.
15. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/397> . Permit any of the WIN create routines (except dynamic) to return shared memory, and list WIN\_SHARED\_QUERY tell you about that memory. Status: For MPI-4; feedback requested. Specific text proposed.
16. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/396> . Add or extend RMA assertions. Ticket confuses info keys and assertions; from the text, it looks like it meant assertion values, not info keys. Status: For MPI-4; not ready.
17. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/372> . Add additional predefined keys for Comm\_split\_type. Status: MPI-4, feedback requested.
18. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/369> . Add same\_disp\_unit info key for RMA window creation. Status: Appears to have been dropped, but is still active as a ticket.
19. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/367> . Fix typo in name. Status: passed but not closed.
20. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/366> . Use datatypes to replace displacement unit. Status: Appears to have been dropped, but is still active as a ticket.
21. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/363> . Discusses fairness. Status: Appears to have been dropped, but it is still active as a ticket.
22. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/362> . Clarify same size info key. Status: passed but not closed.
23. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/359> . Update RMA progress rationale. Status: MPI errata; passed but not closed.
24. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/350> . Ignore origin with MPI\_NO\_OP. Status: MPI errata; passed but not closed.
25. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/348> . Fix Example 11.21. Status: MPI errata; passed but not closed.
26. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/347> . Fix info in appendices for RMA. Status: MPI errata; passed but not closed.
27. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/338> . Generalize the accepted datatypes for accumulate operations. See also #34. Status: Feedback requested
28. <https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/334> . Fixes to examples. Status: Unclear. Was this already done in the MPI 3.0 text?