ONE-SIDED COMMUNICATION IN MPI

Meiling Traeger & Damian Franco

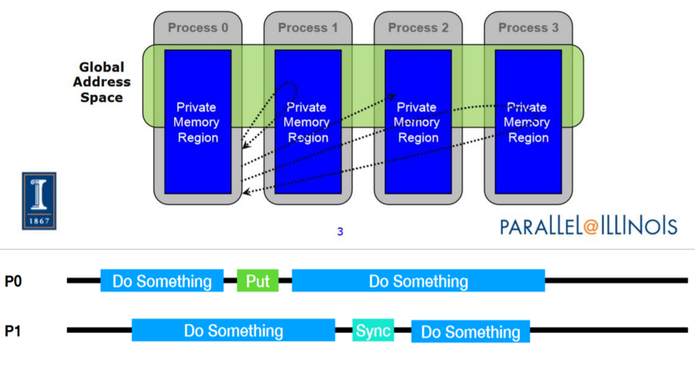
CS 542 – Introduction to Parallel Processing

THE UNIVERSITY OF NEW MEXICO

Background: In the context of the Message Passing Interface (MPI), one-sided communication refers to a set of communication routines that allow a process to access the memory of another process without the need for explicit communication or synchronization between both processes, hence one-sided. This can be useful for optimizing communication patterns in parallel programs. One-sided communication routines in MPI include routines for putting data into a remote memory location, getting data from a remote memory location, and performing remote memory operations such as atomic updates and memory fence operations.

Basic Components

- Remote memory access (RMA) operations, which allow a process to access the memory of another process without the need for explicit communication or synchronization.
- Window objects, which are data structures that define the memory region that can be accessed by other processes using RMA operations.



One Sided Timeline Example

- RMA communication calls, which are the routines that processes use to perform RMA operations on the memory of other processes.
- Memory synchronization modes, which are used to ensure that RMA operations are performed in a specified order.

Notable API

MPI Window

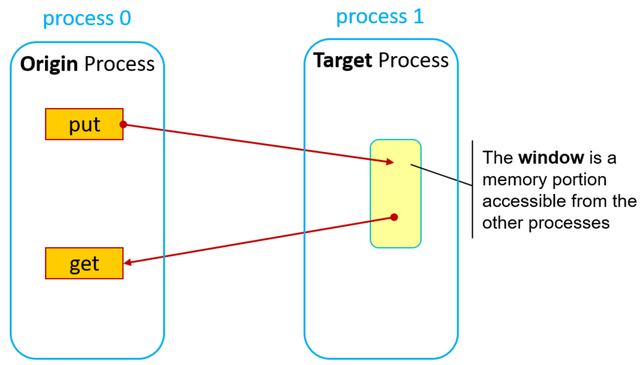
An MPI Window, MPI_Win_create, is a memory region that is shared by multiple MPI processes and can be accessed directly by those processes without sending and receiving messages.

MPI_Put

MPI_Put allows one process directly accessing the memory of another process without sending and receiving message. The process can write data directly to the memory of another process, which can be more efficient than sending and receiving messages.

MPI_Get

MPI_Get involves one process directly accessing the memory of another process without sending and receiving messages. The process can read data directly from the memory of another process, which can be more efficient than sending and receiving messages.

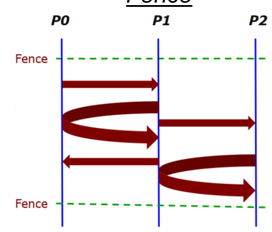


Synchronization

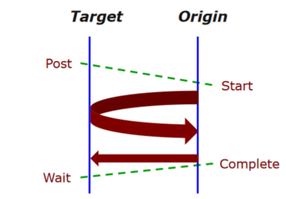
Three modes of synchronization with one-sided communication in MPI:

- Fence synchronization pattern has a "fence" is inserted into the communication flow between the processes. This fence acts as a barrier that prevents any processes from continuing until all the processes have reached the fence and signaled that they are ready to continue.
- Post-start-complete-wait (PSCW) refers to a pattern of communication in which one process sends a message to another process, and then waits for a response before continuing.
- Lock/unlock synchronization states that each process must first "lock" the shared resource before it can access it. Once the process has finished using the resource, it must then "unlock" it, which allows other processes to access the resource.

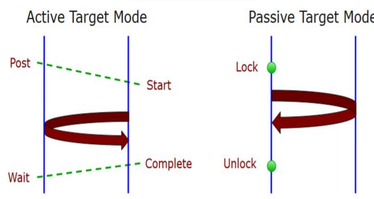
Fence



Post-start-complete-wait



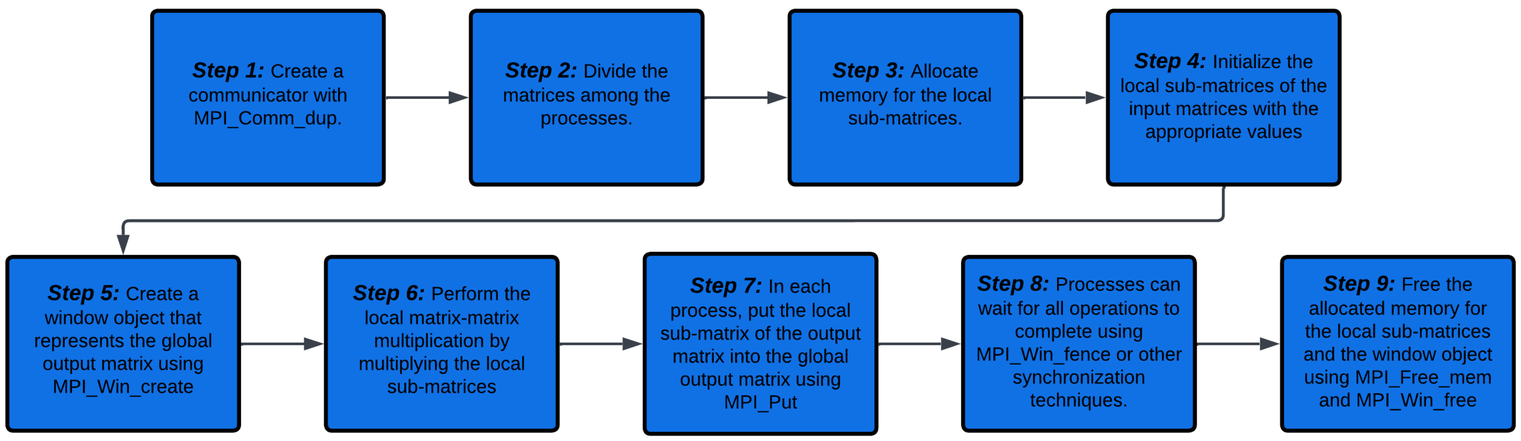
Lock/Unlock



Implementation

Our team developed an implementation of a dense matrix-matrix multiplication example using only one-sided communication. After implementation we then wanted to compare performances between dense matrix-matrix multiplication with one-sided communication and two-sided communication. The steps are shown below in the flow chart.

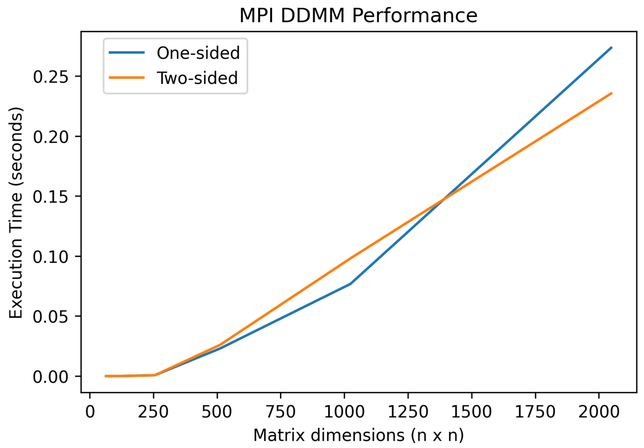
Dense matrix-matrix multiplication with one-sided communication steps:



The implementation above used the synchronization technique fence. This model above is a simple approach to dense matrix-matrix multiplication in the sense of one-sided and two-sided communication. One-sided communication is similar when compared to two-sided communication.

Dense matrix-matrix multiplication performance:

One-sided communication vs. two-sided communication



The plot above are timings averaged from over 15 runs of one-sided and two-sided communication for dense matrix-matrix multiplication. Each run had the same initialization of hardware use, as well as stripped any advantages away that one implementation had over the other (i.e, both approaches were the simplest, most naive approach to implementing matrix-matrix multiplication).

Conclusion

- One-sided communication can be useful in situations where a process needs to access data that is stored in the memory of another process but does not need to synchronize with that process.
- Two-sided communication could have some possible synchronization overhead and one-sided communication can reduce synchronization and improve performance.
- One-sided communication allows for more efficient data transfer and can improve the performance of the overall multiplication operation but does not show significant performance improvements for dense matrix-matrix multiplication.
- Performance always depends on the implementation of the operations one is trying to perform, not based on the various MPI communication options.

References

- Lecture 34 &35: One-sided Communication in MPI, William Gropp, University of Illinois Urbana-Champaign, https://wgropp.cs.illinois.edu/courses/cs598-s15/lectures/lecture34.pdf
- One-Sided Interface for Matrix Operations using MPI-3 RMA: A Case Study with Elemental. Washington State University School of Electrical Engineering and Computer Science