Shared Memory Windows

Hybrid+RMA Working Groups

(Editors: James Dinan, Torsten Hoefler)

Presented to the MPI Forum

July 2011

Motivation

- Need to share data within a node
  - Memory/core ratio is not increasing
  - Replication is expensive
  - Applications adopting hybrid programming models
    - Added complexity of multi-model programming
    - Primary feature needed: shared memory
- Solution: MPI processes plus shared segment

MPI Shared Segments

1. New shared segments API
   - Requires many additions to make feature-complete (consistency, atomics, …)

2. Extension to RMA interface
   - Leverage existing RMA functionality and semantics
     - Consistency, synchronization, atomics
   - Small extension to the RMA API

Shared Segment Creation

MPI_WIN_ALLOCATE_SHARED(size, info, comm, baseptr, win)

IN	size	size of local window segment in bytes
IN	info	info argument
IN	comm	intra-communicator
OUT	baseptr	address of local allocated window segment
OUT	win	window object returned by the call

- Allocates a contiguous memory of the sum of sizes at all processes
  - Predefined info argument can specify alloc_shared_noncontig

Query Segment Addresses

- MPI_WIN_SHARED_QUERY(win, rank, size, baseptr)

IN	win	shared memory window
IN	rank	rank in the group of window win
OUT	size	size of the window segment
OUT	baseptr	address for load/store access to window segment

- baseptr returns the address of the local segment
  - If a rank specified size=0, baseptr is NULL
- Must be queried in case of noncontiguous allocation!
- Returns error if win is not of flavor MPI_WIN_FLAVOR_SHM

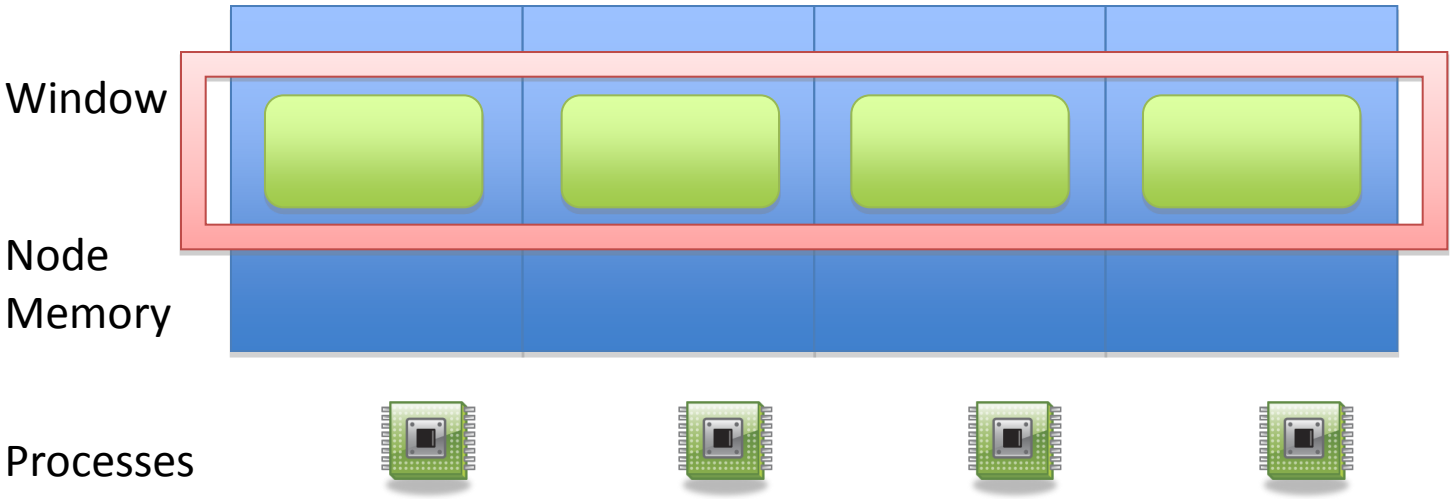
Changes to MPI_Win_lock_all

- Will accept MPI_LOCK_EXCLUSIVE
  - Locks the whole shared memory region (on all processes)
  - Behaves as mutex (implied by rules about load/store – same for MPI_Win_lock)
- May use MPI_Win_lock for fine-grained process locking
- NO locks for fine-grained data locking within window (may use mechanisms outside MPI or MPI-3 CAS)
  - This is already true in MPI-2 (no change)

Shared Segment Semantics

- Behaves exactly like an MPI Window
  - Get, put, accumulate
  - Active, passive transfer modes
  - Separate, unified models
- Only one added capability
  - Load/store access from all processes in group of win
  - Must follow RMA data access rules
    - Conflicting accesses, consistency, …

Logical Mapping



- Window segment on process *X* is logically local to process *X*
  - May be physically local or not! This is outside the scope of MPI, implementers should document (advice to implementers)

Simple Example

```
Get_node_comm(&node_comm);
size = 0;
if (rank == 0) size = SHARED_SIZE;
MPI_Win_allocate_shared(size, MPI_INFO_NULL, node_comm, &baseptr, &shr_win);
MPI_Win_shared_query(shr_win, 0, &size, &baseptr);

if(rank==0) {
  MPI_Win_lock(MPI_LOCK_EXCLUSIVE, 0, MPI_MODE_NOCHECK, shr_win);
  Initialize_shared_data(baseptr); // rank 0 initializes global data
  MPI_Win_unlock(shr_win);
}
MPI_Barrier(node_comm);

MPI_Win_lock(MPI_LOCK_SHARED, 0, MPI_MODE_NOCHECK, shr_win);
Compute(baseptr);  // Only loads performed during computation
MPI_Win_unlock(shr_win);
```