MPI-2

Sathish Vadhiyar

One Sided communications
Motivation

- Remote memory access (RMA)
- All communication parameters on one side (sender/receiver)
- For applications that have dynamic data access patterns
- For using hardware provided features
- Consists of communication (put, get, update) and synchronization functions
Allowing memory accesses

☐ MPI provides controls
  - Which parts of memory can be accessed by remote memory
  - During what time (synchronization – more later)

☐ Which parts of memory? – MPI helps creates window of memory access

☐ MPI_WIN_CREATE(base, size, disp_unit, info, comm, win)
Communication Calls

- 3 non-blocking calls:
  - MPI_PUT(origin_addr, origin_count, origin_datatype, target_rank, target_disp, target_count, target_datatype, win) for writing to remote memory
  - MPI_GET(origin_addr, origin_count, origin_datatype, target_rank, target_disp, target_count, target_datatype, win) for reading from remote memory
  - MPI_ACCUMULATE(origin_addr, origin_count, origin_datatype, target_rank, target_disp, target_count, target_datatype, op, win) for updating remote memory
Example - Get

- To compute $A = B(\text{map})$

```fortran
SUBROUTINE MAPVALS(A, B, map, m, comm, p)
USE MPI INTEGER m, map(m), comm, p
REAL A(m), B(m)
INTEGER sizeofreal, win, ierr

CALL MPI_TYPE_EXTENT(MPI_REAL, sizeofreal, ierr)
CALL MPI_WIN_CREATE(B, m*sizeofreal, sizeofreal, MPI_INFO_NULL, & comm, win, ierr)

CALL MPI_WIN_FENCE(0, win, ierr)
DO i=1,m
  j = map(i)/p
  k = MOD(map(i),p)
  CALL MPI_GET(A(i), 1, MPI_REAL, j, k, 1, MPI_REAL, win, ierr)
END DO CALL
MPI_WIN_FENCE(0, win, ierr)

CALL MPI_WIN_FREE(win, ierr)
RETURN END
```
Example - Accumulate

- To update \( B(j) = \sum_{\text{map}(i)=j} A(i) \)

```fortran
SUBROUTINE SUM(A, B, map, m, comm, p)
    CALL MPI_TYPE_EXTENT(MPI_REAL, sizeofreal, ierr)
    CALL MPI_WIN_CREATE(B, m*sizeofreal, sizeofreal, MPI_INFO_NULL, & comm, win, ierr)
    CALL MPI_WIN_FENCE(0, win, ierr)
    DO i=1,m
        j = map(i)/p
        k = MOD(map(i),p)
        CALL MPI_ACCUMULATE(A(i), 1, MPI_REAL, j, k, 1, MPI_REAL, & MPI_SUM, win, ierr)
    END DO
    CALL MPI_WIN_FENCE(0, win, ierr)
    CALL MPI_WIN_FREE(win, ierr)
RETURN END
```
Synchronization

- Active target communication - Both processes are explicitly involved in communication
- Passive target communication - Only origin process is involved
- Access epoch - Contains RMA calls in the origin. Starts and ends with synchronization calls.
- Exposure epoch – contains RMA calls in the active target
3 synchronization mechanisms:

- MPI_WIN_FENCE (at origin and target) (for active target)
- MPI_WIN_START, MPI_WIN_COMPLETE (origin)
- MPI_WIN_POST, MPI_WIN_WAIT (target) (for active target)
- MPI_WIN_LOCK, MPI_WIN_UNLOCK (only at origin) (passive target)
Active synchronization

ORIGIN
PROCESS

TARGET
PROCESS

wait
load
store
post

Local window accesses

put
executed
in origin
memory

Window is exposed to RMA accesses

put
executed
in target
memory

start

complete

Local window accesses
Passive synchronization
Illustration – start, complete, post, wait
Extended Collectives

Collective communications with inter-communicators
Intercommunicator Collectives

- Result of Collective operations on one group is seen on the other group
- Applies to following:
  1. MPI_BCAST,
  2. MPI_GATHER, MPI_GATHERV,
  3. MPI_SCATTER, MPI_SCATTERV,
  4. MPI_ALLGATHER, MPI_ALLGATHERV,
  5. MPI_ALLTOALL, MPI_ALLTOALLV, MPI_ALLTOALLW
  6. MPI_REDUCE, MPI_ALLREDUCE,
  7. MPI_REDUCE_SCATTER,
  8. MPI_BARRIER.
Illustration – Intercommunicator all-gather