One-sided Communication

Brian Barrett

MPI One-sided

- Talk about implementing the spec
- Shortcomings of the spec another topic
- Goals:
  - Expected performance
  - Framework for further research
- Pedigree
  - Synchronization ideas borrowed from MPICH-2 papers on one-sided implementation
  - Communication framework radically different

Synchronization

- Fence:
  - Global across window
  - Like a giant memory barrier
- Post/Wait/Start/Complete
  - Localized active synchronization
  - Uses MPI Groups (ugh)
- Lock/Unlock
  - Totally passive synchronization
- Goal: avoid explicit barriers everywhere

Epochs

- Access Epoch
  - Local window can be used as communication origin
  - Fence(), Start(), and Lock() start access epochs
- Exposure Epoch
  - Local window can be used as communication target
  - Fence(), Post(), and target of Lock() start exposure epochs

Communication

- To review: put, get, accumulate
- Put / Get
  - Fence: request started immediately
  - Others: request queued until synchronization
- Accumulate
  - Always queued until synchronization
- Communication over BTL (short) or PML (long)

Fence

- All-to-all or reduce - scatter at start to determine number incoming messages
- Start all outgoing requests
- Poll until number outgoing and incoming messages are zero
- Get ready for the next epoch

Post/Wait/Start/Complete

- Post() sends “epoch start” message to group
- Wait() waits for number of incoming messages, then incoming messages
- Start() is a no-op (just error checking)
- Complete() waits for post() messages, then sends group number outgoing messages, completes messages

Lock/Unlock

- Lock() sends “lock request” to target
- Target sends lock reply when available
- Unlock() waits for lock reply, then sends number of expected messages, then starts messages and polls for completion
- Target receives completion count and tries to complete messages. Releases lock at completion