Ready-Mode Receive:
An Optimized Receive Function for MPI

Ron Brightwell

Sandia National Labs

Scalable Computing Systems Department

bright@cs.sandia.gov

Sandia is a multiprogram laboratory operated by Sandia Coporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC-94AL85000.

Outline

- Ready-mode receive
- Potential benefits
- Drawbacks
- Current motivations
- PERUSE
- Future work

MPI Send Modes

- Standard
  - No buffering or semantic guarantees to application
  - Non-local completion
- Buffered
  - User explicitly provides buffer
  - Local completion
- Synchronous
  - Completes when receive operation has been initiated
  - Non-local completion
- Ready
  - Guarantee from application that receive has been pre-posted
  - Avoids handshaking protocols
  - Avoid buffering complications

MPI Receive Modes

- Semantics of a receive operation are determined by the matching send operation
  - Synchronous-mode send must send an acknowledgment
- Standard, buffered, and synchronous modes are defined by completion
- Ready mode is defined by initiation
  - Opportunity for optimization exists only when the receive is posted

Ready-Mode Receive

- Since the programmer guarantees that a matching receive is posted on the send side, the matching receive operation is guaranteed that no matching message has already arrived
- This eliminates any need to search the unexpected message queue before posting the receive
- First proposed informally to the MPI-2 Forum via email in early 1996

Potential Benefits

- Avoid the time needed to search the unexpected queue
  - Is this time significant?
  - No real data to answer this question
- Avoid bringing unnecessary stuff into user space
  - Could be significant for NIC-based implementations like Portals 3.x in terms of memory usage and time spent holding on to events
- Avoid trying to maintain atomicity
  - Searching unexpected queue and posting must be atomic
  - Network must be quiescent to post a receive
- Performance should be deterministic
  - Possible benefit for soft real-time-like applications
  - Might be good for finding performance anomalies
- It's a basic, primitive operation

Technical Drawbacks

- Opportunity for performance is not easily quantifiable
  - No data that supports the claim for optimization
  - Applications that are worried about saving microseconds searching a queue should not have long unexpected queues in the first place
- Ready-mode sends are only for large messages
  - Saving a few microseconds on searching should not matter at all for large messages
- Applications that would benefit from deterministic performance likely do not have long unexpected queues

Social Drawbacks

- Ready-mode sends squeaked by in the first place
  - Most MPI Forum members did not like it or want it
  - No clear-cut performance benefit
- Users are not sophisticated enough to use multiple receive functions
  - MPI is hard enough and ready-mode receive would just confuse them more
  - Incorrect use would cause non-compliant programs and cause MPI implementors to have to find the problem and explain it to users

Current Motivations

- Still need empirical evidence
- User-level networking technologies have increased the need for an analysis
- Hardware with NIC-based descriptor queues may benefit
  - Reduces the number of PCI crossings
- More MPI protocol processing is being moved to the NIC
  - May just exacerbate the problem
- We see a large number of unexpected messages in large-scale applications

Current Motivations (cont'd)

- May be useful for layering
  - MPI-2 one-sided get operation
    - May decrease latency
  - MPI Collective operations
    - Scatter and gather operations when a reply is posted before the local contribution is sent
- Motivate users to use ready-mode
  - Avoid buffering problems at large-scale
- Ready-mode receive can default to a normal receive for implementations that don't support it
- Users are more sophisticated(?)

Part of the Problem

- No good idea of what applications do in terms of message passing
- No good idea about expected versus unexpected messages until there's a problem
- No good idea how long unexpected or posted queues really get
- No good idea how they change as jobs scale
- Might be able to make some decisions about the effectiveness of ready-mode receive if we knew some of this
- (Most application developers don't know any of this either)

PERUSE

- Portable interface for exposing low-level MPI implementation data
  - Unexpected/expected messages
  - Length of MPI queues
- Started as a DOE ASCI Software PathForward project between LANL, LLNL, Sandia, and MPI Software Technologies, Inc., and Pallas
- Currently involves several MPI implementors and tool vendors
- Current version of the API specification is 1.6
- Birds-Of-a-Feather at SC2002

Future Work

- Use the PERUSE infrastructure to gather data on the behavior of applications with respect to unexpected messages
- Implement ready-mode receive on ASCI/Red (Portals 2.0), Cplant™ (Portals 3.x), possibly ASCI/Red Storm (Portals 3.x)
- Use microbenchmarks to measure performance
- Investigate potential benefit for real applications