Choosing MPI Alternatives

- MPI offers may ways to accomplish the same task
- Which is best?
  - » Just like everything else, it depends on the vendor, system architecture
  - » Like C and Fortran, MPI provides the programmer with the tools to achieve high performance without sacrificing portability
- Experiments with a Jacobi relaxation example

Tuning for MPI's Send/Receive Protocols

- Aggressive Eager
  - » Performance problem: extra copies
  - » Possible deadlock for inadequate eager buffering
  - » Ensure that receives are posted before sends
  - » MPI_Issend can be used to express "wait until receive is posted"
- Rendezvous with sender push
  - » Extra latency
  - » Possible delays while waiting for sender to begin
- Rendezvous with receiver pull
  - » Possible delays while waiting for receiver to begin

Rendezvous Blocking

- What happens once sender and receiver rendezvous?
  - » Sender (push) or receiver (pull) may complete operation
  - » May block other operations while completing
- Performance tradeoff
  - » If operation does *not* block (by checking for other requests), it adds latency or reduces bandwidth.
- Can reduce performance if a receiver, having acknowledged a send, must wait for the sender to complete a separate operation that it has started.

3

Tuning for Rendezvous with Sender Push

- Ensure receives posted before sends
  - » better, ensure receives match sends before computation starts; may be better to do sends before receives
- Ensure that sends have time to start transfers
- Can use short control messages
- Beware of the cost of extra messages
  - » Intel i860 encouraged use of control messages with ready send (force type)

4

Tuning for Rendezvous with Receiver Pull

- Place MPI_Isends before receives
- Use short control messages to ensure matches
- Beware of the cost of extra messages

5

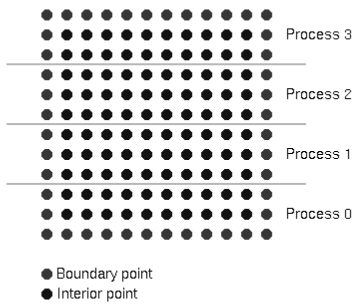
Experiments with MPI Implementations

- Multiparty data exchange
- Jacobi iteration in 2 dimensions
  - » Model for PDEs, Matrix-vector products
  - » Algorithms with surface/volume behavior
  - » Issues similar to unstructured grid problems (but harder to illustrate)
- Others at http://www.mcs.anl.gov/mpi/tutorials/perf

6

Jacobi Iteration

- Simple parallel data structure



- Processes exchange rows with neighbors

7

Background to Tests

- Goals
  - » Identify better performing idioms for the same communication operation
  - » Understand these by understanding the underlying MPI process
  - » Provide a starting point for evaluating additional options (there are many ways to write even simple codes)

8

Some Send/Receive Approaches

- Based on operation hypothesis. Most of these are for polling mode. Each of the following is a *hypothesis* that the experiments test
  - » Better to start receives first
  - » Ensure recvs posted before sends
  - » Ordered (no overlap)
  - » Nonblocking operations, overlap effective
  - » Use of Ssend, Rsend versions (EPCC/T3D can prefer Ssend over Send; uses Send for buffered send)
  - » Manually advance automaton
- Persistent operations

9

Scheduling Communications

- Is it better to use MPI_Waitall or to schedule/order the requests?
  - » Does the implementation complete a Waitall in any order or does it prefer requests as ordered in the array of requests?
- In principle, it should always be best to let MPI schedule the operations. In practice, it may be better to order either the short or long messages first, depending on how data is transferred.

10

Some Example Results

- Summarize some different approaches
- More details at http://www.mcs.anl.gov/mpi/tutorial/perf/mpiexmpl/src3/runs.html

Send and Recv

- Simplest use of send and recv

```
{
  MPI_Status status;
  MPI_Comm ring_comm = mesh->ring_comm;

  /* Send up, then receive from below */
  MPI_Send( xlocal + maxm * lrow, maxm, MPI_DOUBLE, up_nbr, 0,
          ring_comm );
  MPI_Recv( xlocal, maxm, MPI_DOUBLE, down_nbr, 0, ring_comm, &status );
  /* Send down, then receive from above */
  MPI_Send( xlocal + maxm, maxm, MPI_DOUBLE, down_nbr, 1, ring_comm );
  MPI_Recv( xlocal + maxm * (lrow + 1), maxm, MPI_DOUBLE, up_nbr, 1,
          ring_comm, &status );
}
```

- Very poor performance on SP2
  - » Rendezvous sequentializes sends/receives
- OK performance on T3D (implementation tends to buffer operations)

Better to start receives first

- Irecv, Isend, Waitall

```
MPI_Status statuses[4];
MPI_Comm ring_comm;
MPI_Request r[4];

   /* Send up, then receive from below */
   MPI_Irecv( xlocal, maxm, MPI_DOUBLE, down_nbr, 0, ring_comm, &r[1] );
   MPI_Irecv( xlocal + maxm * (lrow + 1), maxm, MPI_DOUBLE, up_nbr, 1,
              ring_comm, &r[3] );
   MPI_Isend( xlocal + maxm * lrow, maxm, MPI_DOUBLE, up_nbr, 0,
              ring_comm, &r[0] );
   /* Send down, then receive from above */
   MPI_Isend( xlocal + maxm, maxm, MPI_DOUBLE, down_nbr, 1, ring_comm,
              &r[2] );
   MPI_Waitall( 4, r, statuses );
}
```

- Ok performance

13

Ensure recvs posted before sends

- Irecv, Sendrecv/Barrier, Rsend, Waitall

```
void ExchangeInit( mesh )
Mesh *mesh;
{
   MPI_Irecv( xlocal, maxm, MPI_DOUBLE, down_nbr, 0, ring_comm,
              &mesh->rq[0] );
   MPI_Irecv( xlocal + maxm * (lrow + 1), maxm, MPI_DOUBLE, up_nbr, 1,
              ring_comm, &mesh->rq[1] );
}

void Exchange( mesh )
Mesh *mesh;
{
   MPI_Status statuses[2];

   /* Send up and down, then receive */
   MPI_Rsend( xlocal + maxm * lrow, maxm, MPI_DOUBLE, up_nbr, 0,
              ring_comm );
   MPI_Rsend( xlocal + maxm, maxm, MPI_DOUBLE, down_nbr, 1, ring_comm );

   MPI_Waitall( 2, mesh->rq, statuses );
}

void ExchangeEnd( mesh )
Mesh *mesh;
{
   MPI_Cancel( &mesh->rq[0] );
   MPI_Cancel( &mesh->rq[1] );
}
```

14

7

Use of Ssend versions

- Ssend allows send to wait until receive ready
  - » At least one implementation (T3D) gives better performance for Ssend than for Send

```
void Exchange( mesh )
Mesh *mesh;
{
   MPI_Status  status;

   /* Send up, then receive from below */
   MPI_Irecv( xlocal, maxm, MPI_DOUBLE, down_nbr, 0, ring_comm, &rq );
   MPI_Ssend( xlocal + maxm * lrow, maxm, MPI_DOUBLE, up_nbr, 0,
            ring_comm );
   MPI_Wait( &rq, &status );
   /* Send down, then receive from above */
   MPI_Irecv( xlocal + maxm * (lrow + 1), maxm, MPI_DOUBLE, up_nbr, 1,
            ring_comm, &rq );
   MPI_Ssend( xlocal + maxm, maxm, MPI_DOUBLE, down_nbr, 1, ring_comm );
   MPI_Wait( &rq, &status );
}
```

15

Nonblocking Operations, Overlap Effective

- Isend, Irecv, Waitall
- A variant uses Waitsome with computation

```
void ExchangeStart( mesh )
Mesh *mesh;
{
    /* Send up, then receive from below */
   MPI_Irecv( xlocal, maxm, MPI_DOUBLE, down_nbr, 0, ring_comm,
            &mesh->rq[0] );
   MPI_Irecv( xlocal + maxm * (lrow + 1), maxm, MPI_DOUBLE, up_nbr, 1,
            ring_comm, &mesh->rq[1] );
   MPI_Isend( xlocal + maxm * lrow, maxm, MPI_DOUBLE, up_nbr, 0,
            ring_comm, &mesh->rq[2] );
   MPI_Isend( xlocal + maxm, maxm, MPI_DOUBLE, down_nbr, 1, ring_comm,
            &mesh->rq[3] );
}

void ExchangeEnd( mesh )
Mesh *mesh;
{
   MPI_Status statuses[4];
   MPI_Waitall( 4, mesh->rq, statuses );
}
```

16

Persistent Operations

- Potential saving
  - » Allocation of MPI_Request
  - » Validating and storing arguments
  - » Fewer interactions with “polling” engine
- Variations of example
  - » sendinit, recvinit, startall, waitall
  - » startall(recvs), sendrecv/barrier, startall(rsends), waitall
- Some vendor implementations are buggy
- Persistent operations may be slightly *slower*
  - » if vendor optimizes for non-persistent operations

17

Polling Engine

- Some implementations poll for incoming control message on “important” MPI calls
  - » Each polling operation costs time
  - » HP/Convex is one vendor implementation that does this
- Operations that *create* persistent operations don’t poll

18

Summary of Results

- Better to start sends before receives
  - » Most implementations use rendezvous protocols for long messages (Cray, IBM, SGI)
- Synchronous sends better on T3D
  - » otherwise system buffers
- MPI_Rsend can offer some performance gain on IBM SP
  - » as long as receives can be guaranteed *without* extra messages