Process Failure Mitigation in the MPI 3.0 Standard

Wesley Bland

Friday Lunch Talk

3/9/12

INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE

MPI Forum Proposal

- Proposed new chapter for the MPI standard
- Presented to the MPI forum this week

Motivation

- Failures are common enough that we care but rare enough that failure free performance is most important.

- Want proposal to be
  - Lightweight
  - Efficient
  - Extensible

Why so lightweight?

```
for (i = 0; i < n; i++) {
  value = compute();
  rc = MPI_Bcast(comm, value);
  if (rc!=MPI_SUCCESS)
    recover();
}
```

Why so lightweight?

```
for (i = 0; i < n; i++) {
  value = compute();
  rc = MPI_Bcast(comm, value);
  if (rc!=MPI_SUCCESS)
    recover();
}
```

Bcast agrees on return code

$O(n^2)$

Why so lightweight?

```
for (i = 0; i < n; i++) {
  value = compute();
  rc = MPI_Bcast(comm, value);
  if (rc!=MPI_SUCCESS)
    recover();
}
```

```
while(morework) {
  rc = MPI_Recv(myrank-1);
  if (rc!=MPI_SUCCESS) recover();
  rc = MPI_Send(myrank+1);
  if (rc!=MPI_SUCCESS) recover();
}
```

Bcast agrees on return code

$O(n^2)$

Why so lightweight?

```
for (i = 0; i < n; i++) {
  value = compute();
  rc = MPI_Bcast(comm, value);
  if (rc!=MPI_SUCCESS)
    recover();
}
```

Bcast agrees on return code

$O(n^2)$

```
while(morework) {
  rc = MPI_Recv(myrank-1);
  if (rc!=MPI_SUCCESS) recover();
  rc = MPI_Send(myrank+1);
  if (rc!=MPI_SUCCESS) recover();
}
```

Failures must be reported during unrelated operations

Why so lightweight?

```
for (i = 0; i < n; i++) {
  value = compute();
  rc = MPI_Bcast(comm, value);
  if (rc!=MPI_SUCCESS)
    recover();
}
```

Bcast agrees on return code

$O(n^2)$

```
while(morework) {
  rc = MPI_Recv(myrank-1);
  if (rc!=MPI_SUCCESS) recover();
  rc = MPI_Send(myrank+1);
  if (rc!=MPI_SUCCESS) recover();
}
```

Failures must be reported during unrelated operations

EXPENSIVE

Two Guiding Principles

- A correct MPI program must not deadlock because of process failures.

- An MPI call that does not involve a failed process will complete normally.

Simple Master/Worker

- FT Method
  - Check Return Codes
- New MPI Constructs
  - MPI_ERR_PROC_FAILED

```
MPI_Irecv(/* all workers */);
while (work_available &&
       active_workers > 0
  rc = MPI_Wait_any(&worker);
  if (MPI_ERR_PROC_FAILED == rc)
    active_workers--;
    /* Recover Lost Work */
  else
    /* Send new work to worker */
    MPI_Irecv(worker);
```

Failure Acknowledgement

- MPI_COMM_FAILURE_ACK(comm)
- MPI_COMM_FAILURE_GET_ACKED(comm, &group)

- Unmatched MPI_ANY_SOURCE receptions will not return failure for the processes in **group**
- **group** is updated during the next call to ACK

Slightly harder…

```
MPI_Irecv(MPI_ANY_SOURCE);
while (work_available &&
        active_workers > 0)
    rc = MPI_Wait();
  if (MPI_ERR_PROC_FAILED == rc)
    active_workers--;
    /* Recover Lost Work */
    MPI_Irecv(MPI_ANY_SOURCE);
else if (MPI_ERR_PENDING == rc)
  MPI_Comm_failure_ack(comm);
  MPI_Comm_failure_get_acked(comm, &group);
  MPI_Group_size(group, &gsize);
  active_workers -= /* Size difference */
else
  /* Send new work */
  MPI_Irecv(MPI_ANY_SOURCE);
```

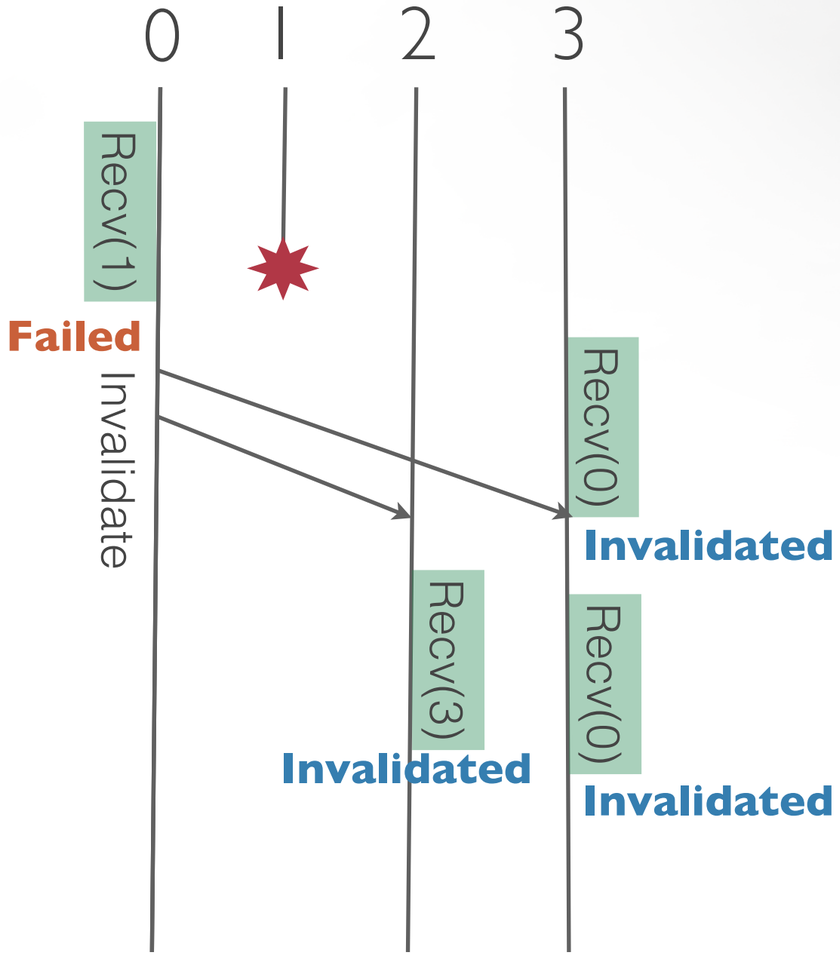
Slightly harder…

```
MPI_Irecv(MPI_ANY_SOURCE);
while (work_available &&
       active_workers > 0)
  rc = MPI_Wait();
 if (MPI_ERR_PROC_FAILED == rc)
  active_workers--;
  /* Recover Lost Work */
  MPI_Irecv(MPI_ANY_SOURCE);
```

```
else if (MPI_ERR_PENDING == rc)
  MPI_Comm_failure_ack(comm);
  MPI_Comm_failure_get_acked(comm, &group);
  MPI_Group_size(group, &gsize);
  active_workers -= /* Size difference */
else
  /* Send new work */
  MPI_Irecv(MPI_ANY_SOURCE);
```

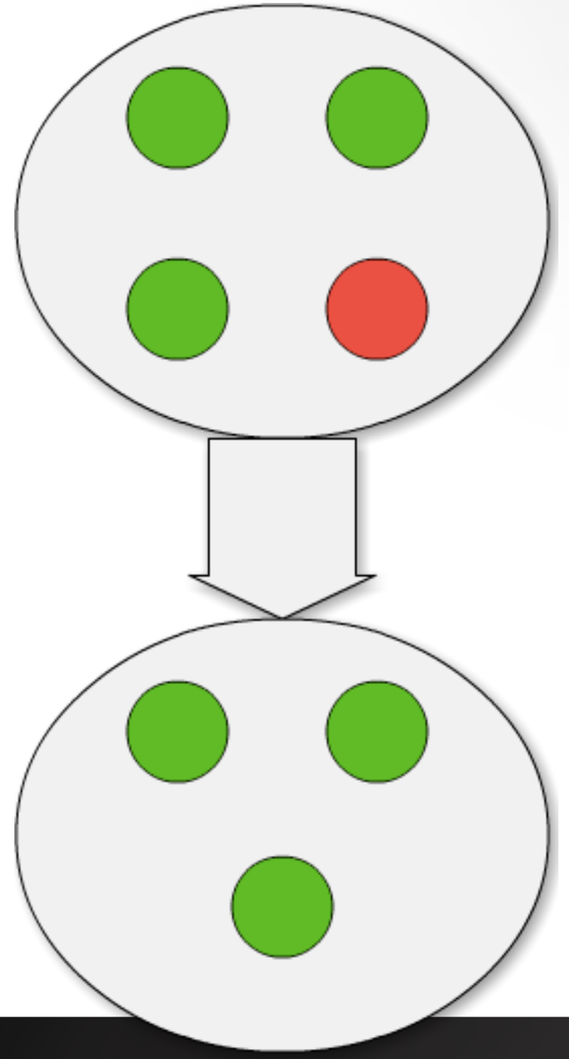
MPI_COMM_INVALIDATE (comm)

- Non-collective operation
- All non-local operations complete with error and new operations will always return an error
  - With two exceptions
- Communicator is invalid when local process calls INVALIDATE or returns MPI_ERR_INVALIDATED because another process called it



MPI_COMM_SHRINK(comm, comm2)

- Creates a new communicator from an invalidated one
- Does not include failed processes
- Will not fail due to process failure



MPI_COMM_AGREEMENT(comm, flag)

- All living processes perform logical AND over **flag**
- Failed processes do not contribute
- Will not return error due to process failure
- Works on invalidated communicators

Error Codes

- MPI_ERR_PROC_FAILED
  - Operation cannot complete because involved process is dead
- MPI_ERR_INVALIDATED
  - Communication object is invalidated and will not work again
- MPI_ERR_PENDING
  - Non-blocking completion **may** not complete because involved process is dead

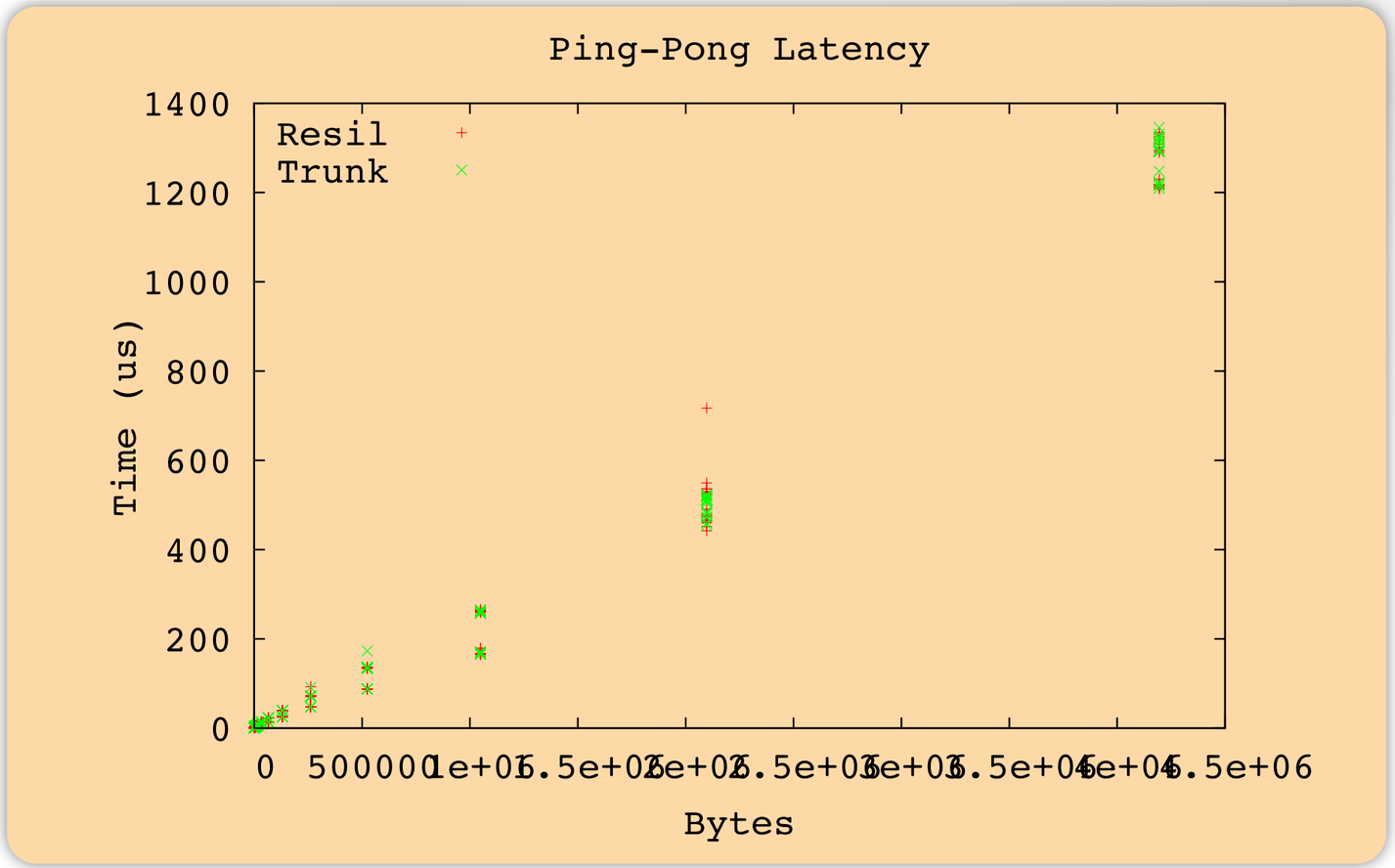
Iterative Refinement

```
while( gnorm > epsilon ) {
  rc = MPI_Allreduce( &lnorm, &gnorm, 1, MPI_DOUBLE, MPI_MAX,
  comm);
  if( (MPI_ERR_PROC_FAILED == rc) ||
      (MPI_ERR_COMM_INVALIDATED == rc) ||
      (gnorm <= epsilon) ) {
    if( MPI_ERR_PROC_FAILED == rc)
      MPI_Comm_invalidate(comm);

    allsuceeded = (rc == MPI_SUCCESS);
```

```
    if( MPI_ERR_PROC_FAILED == rc)
      MPI_Comm_invalidate(comm);

    allsuceeded = (rc == MPI_SUCCESS);
    MPI_Comm_agreement(comm, &allsuceeded);
    if( !allsucceeded ) {
      MPI_Comm_invalidate(comm);
      MPI_Comm_shrink(comm, &comm2);
      MPI_Comm_free(comm);
      comm = comm2;
      gnorm = epsilon + 1.0;
    }
  }
}
```

Performance

- Expected to have similar performance to MPI_ERRORS_RETURN
- Implementation in progress

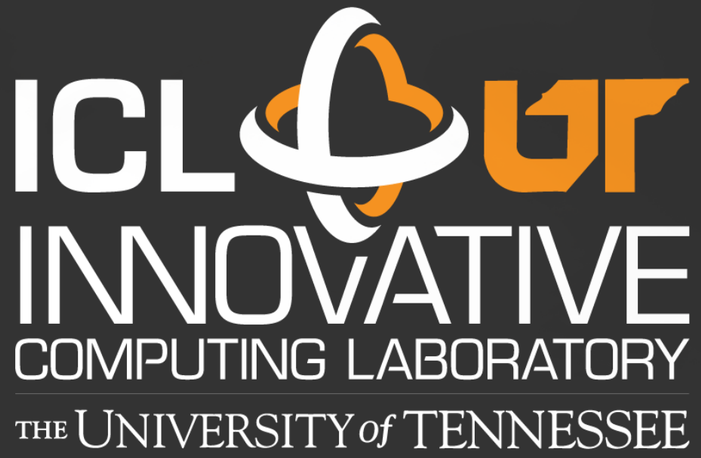


This seems weak...

- Stronger consistency models can sit on top of MPI inside a library
- Most flexibility for the user
- No “silver bullet”

Status of MPI Forum

- Backing of Fault Tolerance Working Group
- Support in the full forum
- Completed first reading
- Will present revised version with implementation at meeting in late May
- Final decision in May or July



http://icl.eecs.utk.edu/