Consistent Error Notification Within MPI

FT working group

This document summarizes the direction the MPI-3 FT working group is taking in defining consistent error notification across the MPI standard. The intent is to use an approach that is consistent with the MPI 2.x error handling, and which extends this to handle the failure scenarios from which the MPI library is expected to recover.

The current document covers failure scenarios associated with network and process failures. At this stage no work has been done in the area of failure as it relates to MPI-I/O.

We assume that from MPI's perspective actual process failure and complete network segmentation are indistinguishable. If network connection is re-established between network segments, it is the responsibility of the MPI implementation to ensure these "rediscovered" processes do not interfere with the running application. Also, we treat full network segmentation as a different use case than losing partial network connection. In the later case one can perhaps resort to a lower performing communication method, such as routing, to fully satisfy application communications requests.

Function return values

In addition to the standard set of return values, each MPI function may return two new error codes: `MPI_EVENT_READY_SUCCESS` and `MPI_EVENT_READY_FAIL`. `MPI_EVENT_READY_SUCCESS` is returned in cases where the function completes successfully (would normally have returned `MPI_SUCCESS`) but an event is ready to be picked up via `MPI_Get_event.` `MPI_EVENT_READY_FAIL` is returned in cases where the function itself fails but instead of returning a specific error code, places an event on this process' event queue.

I. Network partition

Description of event: a high level switch or major cable fails, causing a portion of the compute nodes to be inaccessible from the rest of the nodes for an unknown period of time.

Desired notification policy options:

Notification of failure

1. All processes are preemptively notified of failure before they try to communicate with processes in another partition.
2. Process only notified when it tries to communicate with node in another partition.

Notification of repair

1. Application never notified of repair that fixes the network partition. Application responsible for ensuring that different partitions don't interfere with each other. (Question: is this really what we want ?)
2. All processes are notified when partition is repaired.

3. Application must explicitly designate a process to be informed of the repair and is only notified if it has done so.

Possible user response:

Same as for process failure

II. Link Failure

Description of event: The failure of a network link or low-level switch prevents a subset of processes from communicating with another subset. For example, in a statically-routed torus topology the failure of one or more key links or nodes can prevent a specific pair of nodes from communicating while leaving other nodes free to communicate with each other.

Desired notification policy options:

Notification of failure (processes A and B can't communicate with each other)

1. Processes A and B preemptively notified of failure before they try to communicate with each other.
2. When process A tries to communicate with B or vice versa, it is informed of failure.
3. When process C tries to perform a collective operation that involves both A and B, it is informed of the failure.

Notification of repair (if this happens)

1. Both A and B notified when connection between them is repaired.
2. Application must explicitly designate a process to be informed of the repair and is only notified if it has done so.

Possible user response:

1. Continue with no action (degraded performance assumed, if repair does not happen)
2. Abort immediately
3. Finish current work, and then exit

III. Global or local network performance degradation

Description of event: performance is degraded because

- (global) Some replicated high-level switches have failed causing all cross-machine communication to pass through fewer switches.
- (global) The network is being shared with another application, which has entered a communication-heavy phase.
- (local) Some replicated low-level switches have failed causing all local communication to pass through fewer switches.
- (local) A given network wire has become frayed, causing its re-transmit rate to increase significantly.

Desired notification policy options:

Notification of degradation

1. All affected processes are informed before their next communication operation.
2. Each affected process informed at the time of its next communication operation.

3. Each process that participates in a collective call with an affected process is informed at the time of the call.
4. Application must explicitly designate a process to be informed of the degradation and is only notified if it has done so.

Notification of repair

1. All affected processes are informed before their next communication operation.
2. Each affected process informed at the time of its next communication operation.
3. Application must explicitly designate a process to be informed of the repair and is only notified if it has done so.

Possible user response:

1. Continue with no action (degraded performance assumed)
2. Abort immediately
3. Finish current work, and then exit

IV. Process failure

Use case: N process MPI job loses a process m. This can be due to process failure, or complete lack of contact with that process.

Need to decide if the error is one that should be recovered from or not. Failure due to hardware error may be one that should be recovered from, but one may not want to recover from a segv in the user application.

Lets assume that the failure is one that should be recovered from. The goals are:

Discard all traffic associated with process m, on the send and receive sides

Outstanding point-to-point communications with process m return an error code

Decide how to restore the affected communicators, if to reduce the size of the communicator, or try and regenerate the lost process

Implement the changes to the communicator

Restart the communicator

Question: How do we handle communicators that do not exist in the restored processes? What happens as these are recreated/destroyed as part of the recovery process?

In general, error notification is local, and occurs with MPI calls (push model). A process can register for to receive immediate notification (pull model). This is also true for collectives, by default error notification is local (and different MPI processes may get different return codes), with the ability for a process to register for global notification.

Possible user response:

1. Request that all failed processes be restored
2. Request that some of the failed processes be restored
3. Continue without restoring any of the failed processes