MPI-3 fault handling = error handling

MPI-3 FT WG

Alexander Supalov, Intel Corporation

April 29, 2008

Contents

Idea

Basic mechanism

Notification delivery

Notification control

Notification granularity

Fault codes and classes

Repercussions

Conclusions

Idea

- Extend MPI-2 error handler mechanism to cover faults
  - Applications may need to be notified of the faults to take appropriate action.
  - Faults are essentially errors that occur outside of the application control. Still, they can be dealt with as conventional MPI errors.
- Hence, a slightly existing error handling mechanism may also be adequate for fault handling
  - Extend error code/class list
  - Extend existing error handlers (MPI_ERRORS_ABORT, MPI_ERRORS_RETURN, MPI::ERRORS_THROW_EXCEPTIONS)
  - Define point and moment of error handler activation on faults
    - Communicators, windows, files as earlier
    - Invoke synchronously – inside MPI calls
    - Invoke asynchronously (optional) – anywhere
  - Possibly add a couple of predefined fault-specific error handlers
  - Provide the means to react to faults in the rest of the FT chapter

Basic mechanism

- The existing MPI error handling mechanism can be extended to return fault conditions as usual MPI errors thru the MPI error handlers. Indeed, the existing standard error handlers (MPI_ERRORS_RETURN, MPI_ERRORS_ARE_FATAL, and additionally, in C++, MPI::ERRORS_THROW_EXCEPTIONS) appear to be fully applicable to the fault handling, and backward compatible with the existing MPI implementations.
  - Since the MPI_ERRORS_ARE_FATAL is the default, an application that is not prepared to deal with the faults will fail.
  - An application that is prepared to deal with faults as well as errors may use the MPI_ERRORS_RETURN or MPI::ERRORS_THROW_EXCEPTIONS.
  - Finally, an application that elects to deal with the faults more selectively may register its own error handler using the existing mechanisms.
- This will match the spirit of providing MPI users with the means for building their fault tolerance solutions, rather than building solutions into the MPI itself. Compare this to the file I/O and the decision to use datatypes instead of parallel file system modes.

Notification delivery

- The fault notifications can be delivered:
  - Synchronously. This may happen in two ways:
    - From the MPI call immediately affected by the fault. Think MPI_Send that discovers that the destination process is dead. This MPI_Send will fail with the respective fault code.
    - From an MPI call after the fault. Think an MPI_Comm_size returning an error because one of the processes failed to reply to an internal MPI heartbeat.
  - Asynchronously. This resembles a long jump, signal, or asynchronous exception.
    - In this case the MPI_ERRORS_RETURN will basically ignore faults.
    - The MPI::ERRORS_THROW_EXCEPTIONS will, er, throw an exception and hope it's caught.
    - The default MPI_ERRORS_ARE_FATAL will terminate the application as usual.
  - An application defined error handler will try to correct the situation, or at least let the job die gracefully, depending on the will of the MPI user.
- The MPI standard may also provide additional standard fault/error handlers with predefined reactions.
- TODO. Need to specify what happens in threaded environment: will a fault cross the thread boundary?

Notification control

- Apart from defining the error handler, the application may want to request a certain level of fault handling support from the underlying implementation. Just like in the case of thread support, the required levels may not be provided, and the application may elect to terminate itself rather than adapt to the provided level of fault support.
- TODO. It remains to be seen whether control should be available at the job startup or may be made individually for each communicator, windows, or file object (see granularity below).

Notification granularity

- Since error handlers are attached to communicators, windows, and files, one can flexibly control what will be happening inside any part of the MPI job.
- As usual, the error handler attached to the MPI_COMM_WORLD will control the job default, while the one attached to the MPI_COMM_SELF may control process' reaction to its own internal problems.
- Analogously, the error handler attached to the MPI_WIN_NULL and MPI_FILE_NULL will control the default reaction of the respective MPI objects.

Fault codes and classes

- TODO. Need to define fault classes analogously to the error classes. They might exist within the same error list and possibly have MPI_ERROR_... names for consistency, or make a separate list. The latter would be more compatible with the MPI subsetting and optional nature of the fault tolerance.

Repercussions

- The reaction to the detected fault is out of the scope of this proposal, but one can imagine a set of communicator attributes and extra MPI calls to handle communicator contraction, out-of-band communication, remote process signaling, etc.
- Due to additional implementation complexity and possible performance repercussions, the fault tolerance support should probably be optional. This connects this activity with the MPI subsetting.
- Ideally, all MPIs should be ABI compatible. This will require similarity of the MPI constants, including the values of the fault classes.

Conclusions

It appears possible to extend the existing MPI error handling mechanism almost transparently to handle faults. This approach may become a part of the bigger FT chapter.

intel®

Leap ahead™