User Level Failure Mitigation Reading

Aurelien Bouteiller for the Error Management WG
Dec, 2016 MPI Forum, Dallas, TX
User Level Failure Mitigation: Specification Status

- Adds 3 error codes and 5 functions to manage process crash
  - **Error codes:** interrupt operations that may block due to process crash
  - **MPI_COMM_FAILURE_ACK/GET_ACKED:** continued operation with ANY-SOURCE RECV and observation known failures
  - **MPI_COMM_REVOKE** lets applications interrupt operations on a communicator
  - **MPI_COMM_AGREE:** synchronize failure knowledge in the application
  - **MPI_COMM_SHRINK:** create a communicator excluding failed processes
- Similar WIN/FILE_REVOKE
User Level Failure Mitigation: Implementations

• Implementation available in Open MPI and MPICH
  • Open MPI implementation updated in-sync with Open MPI 2.x
  • Scalable fault tolerant algorithms demonstrated in practice (SC’14, EuroMPI’15, SC’15, SC’16)

Point to point performance unchanged
With FT enabled

Fault Tolerant Agreement costs approximately 2x Allreduce
User Level Failure Mitigation: User Adoption

Fenix Framework

Fig. 3. Checkpoint time for different core counts (8.6 MB/core). The numbers above each test show the aggregated bandwidth (the total checkpoint size over the average checkpoint time).

Domain Decomposition PDE

(a) failure-free (b) few failures (c) many failures

Figure 5. Results of the FT-MLMC implementation for three different failure scenarios.

The performance improvement due to using ULFM v1.0 for running the LULESH proxy application [3] (a shock hydrodynamics stencil based simulation) running on 64 processes on 16 nodes with

- Fortran CoArrays “failed images” uses ULFM-RMA to support Fortran TS 18508
Changes since Sept. Reading

$ git log
5dec522 (HEAD -> ulfm/master) Merge branch 'mpi-3.x' into ulfm/master
Aurélien Bouteiller 13 days ago
10c8bdd (origin/ulfm/sept16/index, origin/ulfm/master, ulfm/sept16/index)
Adding index terms Aurélien Bouteiller 13 days ago
bf6ec8e reordering comm_join Bouteiller 2 weeks ago
403a67f (origin/ulfm/sept16/spawn, ulfm/sept16/spawn) Clarification of Spawn and hard/soft semantic...n Bouteiller 2 weeks ago
d84e6a1 more t0 Aurélien Bouteiller 2 weeks ago
d398ace (origin/ulfm/sept16/t0, ulfm/sept16/t0) Fix the changelog Aurélien Bouteiller 2 weeks ago
7ced80c raise exceptions--> raise an error of class Aurélien Bouteiller 2 weeks ago
a76b04b (origin/ulfm/sept16/shrinkv3, ulfm/sept16/shrinkv3) Reworking SHRINK according to Sept. 2016....Bouteiller 2 weeks ago
bd98b37 Fix incorrect use of MPI_ERR_CLASS Wesley Bland 3 weeks ago
f3d81b9 Add "at that process" also to qualify where it becomes local Aurélien Bouteiller 10 weeks ago
443acc1 (origin/ulfm/sept16/statusv2, ulfm/sept16/statusv2) Have only error codes remain defined in ....Bouteiller 10 weeks ago
Rolf reported we were missing general index terms
  • We had the C/Fotran indexes, but missing the ”terms” index
• Important Terms have been added to the index (pp738,739)

---
+++ b/chap-dynamic/dynamic-2.tex
@@ -1850,7 +1850,7 @@ connected processes is not defined.
 \end{itemize}

\begin{implementors}
-  An \MPI/ implementation that tolerates process failures (as defined
+  An \MPI/ implementation that tolerates process failures\mpitermindex{fault
tolerance!finalize}\mpitermindex{fault
tolerance!process failure} (as defined
  in Chapter~\ref{sec:ft-notification:init-
  finalize}) remains in a
defined state after a process has failed. In practice, it may be
difficult to distinguish between a process
failure and an
Changelog

- Changelog modified using macros and standard “look n feel”
- Put in page ordering as it should
Exceptions -> error codes

- From Bill during Sept 16 reading
  - The terminology “raise an exception of class ...” is unusual
  - Correct terminology is “raise the error code(s) ...”
- Multiple instances have been replaced (examples below)

```
\texttt{\textbackslash error\{MPI\_ERR\_REVOKED\}} at that process. Once a window has been revoked at a process, all subsequent RMA operations on that window are considered local and RMA synchronizations must complete by
\begin{itemize}
  \item raising an exception of class \texttt{\textbackslash error\{MPI\_ERR\_REVOKED\}} at that
  \item raising an error of class \texttt{\textbackslash error\{MPI\_ERR\_REVOKED\}} at that
\end{itemize}
process. In addition, the current epoch is closed and RMA operations originating from this process are interrupted and completed with undefined outputs.
```

The mechanisms for handling process failures are defined in Chapter~\ref{chap:ft}.
When a process failure happens, the \texttt{MPI/} implementation may raise one of the \texttt{MPI/}
-exceptions related to process failure as defined in that chapter.
+error classes related to process failure as defined in that chapter.
In this case, the \texttt{MPI/} implementation is still in a defined state and continues to operate.
Bug in Example

- Putting the error code from MPI_Error_class in the error class variable is wrong
  - Found by Geoffroy Vallee

```c
@@ -1021,7 +1020,7 @@
 while( gnorm > epsilon ) {
     /* Add a computation iteration to converge and compute local norm in lnorm */
     rc = MPI_Allreduce(&lnorm, &gnorm, 1, MPI_DOUBLE, MPI_MAX, comm);
-    ec = MPI_Error_class(rc, &ec);
+    MPI_Error_class(rc, &ec);
     if( (MPI_ERR_PROC_FAILED == ec) ||
         (MPI_ERR_REVOKED == ec) ||
```
Intro chapter missing a short descriptive of Chapt 15 (FT)

• All chapters have a short introduction
  • Additional chapter 15 found (myself proof reading) to not have one
  • Short intro added for uniformity

+++ b/chap-intro/intro.tex
@@ -459,6 +459,12 @@ analyzers, and other tools to obtain data about the operation of \MPI/
 processes. This chapter includes Section~\ref{sec:prof} (\nameref{sec:prof}), which was a chapter in previous versions of \MPI/.
\item
+Chapter~\ref{chap:ft}, \nameref{chap:ft},
+covers interfaces that allow developers to design applications tolerant to process failures. The interfaces presented in this chapter define the state of the \MPI/ library after a process crash, and provide supplementary interfaces to restore the communication capabilities of \MPI/.
+\item
 Chapter~\ref{chap:deprecated}, \nameref{chap:deprecated}, describes routines that are kept for reference. However usage of these functions is discouraged, as they may be deleted in future versions of the standard.
Error codes remain defined but the remainder of status remains undefined,

- During the Sept 16. plenary, it was decided that the status should remain an undefined output parameter (although as noted during the June 16 plenary, the ERROR field must remain defined, duh).

A synchronizing operation may not have synchronized) and the content of the output buffers, targeted memory, or output parameters (except for status objects and error return codes) is \textit{undefined}. Exceptions to this rule are explicitly stated in the remainder of this chapter.

+ Error codes returned from a function, output in arrays of error codes, or + in status objects remain defined after an operation raised a
Shrink

• "failed processes" contributing implicitly found confusing
  • Definition now explicit the content of the groups of the produced communicator

@@ -502,23 +517,22 @@ This collective operation creates a new intra- or intercommunicator respectively, by excluding the group of failed processes as agreed upon during the operation.

- The group of \texttt{mpiarg\{newcomm\}} must include (at least) every process that returns from \texttt{-mpifunc\{MPI\_COMM\_SHRINK\}}, and it must exclude (at least)
+ The groups of \texttt{mpiarg\{newcomm\}} must include every process that returns from \texttt{+mpifunc\{MPI\_COMM\_SHRINK\}}, and it must exclude
  every process whose failure caused an operation on \texttt{mpiarg\{comm\}} to raise an
  \texttt{+MPI/ error of class \error\{MPI\_ERR\_PROC\_FAILED\} or \error\{MPI\_ERR\_PROC\_FAILED\_PENDING\}}
- at a member of the group of \texttt{mpiarg\{newcomm\}}, before that member initiated
+ at a member of the groups of \texttt{mpiarg\{newcomm\}}, before that member initiated
- \texttt{mpifunc\{MPI\_COMM\_SHRINK\}}.
+ \texttt{mpifunc\{MPI\_COMM\_SHRINK\}}.

This call is semantically equivalent to an \texttt{mpifunc\{MPI\_COMM\_SPLIT\}} operation that would succeed despite
-failures, where processes participate with the same
-color and a key equal to their rank in \texttt{mpiarg\{comm\}}, except failed
-processes, which implicitly contribute the color \texttt{const\{MPI\_UNDEFINED\}}.
+failures, where members of the groups of \texttt{mpiarg\{newcomm\}} participate with the same
+color and a key equal to their rank in \texttt{mpiarg\{comm\}}.
MPI_Comm_Disconnect semantic

- The semantic (and text) is identical to MPI_Comm_free.
MPI_Comm_spawn soft/hard

• Text found too oblique/unclear during Sept 16 reading
• Text verified for correctness (found correct) and clarified

+++ b/chap-ft/ft.tex
@@ -270,18 +270,15 @@ process during \mpifunc{MPI\_INIT} when it cannot setup an
intercommunicator with the root process of the spawn operation
because of a process failure.

-An implementation may report it spawned all the requested processes
-in \mpifunc{MPI\_COMM\_SPAWN} or \mpifunc{MPI\_COMM\_SPAWN\_MULTIPLE}
-and instead raise a process failure error when these processes
-are later involved in a communication. \end{implementors}

+An implementation may report it spawned all the requested processes even
+when a process created from \mpifunc{MPI\_COMM\_SPAWN} or \mpifunc{MPI\_COMM\_SPAWN\_MULTIPLE}
failed, and instead delay raising a process failure error to a later communication involving this
process. \end{implementors}

\begin{users} To determine how many new processes have effectively
been spawned, the normal semantic for hard and soft spawn applies: if
-a failure has prevented spawning the requested number of
-processes in a hard spawn, an error of class
-\error{MPI\_ERR\_SPAWN} is raised (leaving \MPI/ in an
-undefined state). In a soft spawn, an appropriate error code is set
-in the \mpiarg{array\_of\_errcodes} parameter. \end{users}

+the requested number of processes is unavailable for a hard spawn, an error
+of class \error{MPI\_ERR\_SPAWN} is raised (possibly leaving \MPI/ in an
+undefined state), and an appropriate error code is set
+in the \mpiarg{array\_of\_errcodes} parameter. Note however that an implementation may report that
it has spawned the requested number of processes even when some processes have failed before exiting
\mpifunc{MPI\_INIT}. This condition can be detected by communicating over the created
intercommunicator with these processes.\end{users}

\par After a process failure, \mpifunc{MPI\_COMM\_DISCONNECT} (as with all
other collective operations) may not complete successfully at all processes. For