Error Management Working Group Update

Aurélien Bouteiller

MPI Forum Bof @SC'17

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Summary of activities

- Default error handlers and error/abort behavior
- Non-catastrophic errors
- Integration between global C/R and scoped recovery models
- User Level Failure Mitigation

Default and Fatal Errors

> **MPI_ERRORS_ARE_FATAL** The handler, when called, causes the program to abort on all executing processes. This has the same effect as if MPI_ABORT was called by the process that invoked the handler.

- In Section 8.3, the above statement is self contradictory
  - It aborts "all" executing processes, but MPI_ABORT has a communicator argument
  - The later is more useful to contain errors in domains
- Proposed changes:
  - **MPI_ERRORS_ARE_FATAL will by default be attached to MPI_COMM_WORLD, MPI_COMM_SELF** and the communicator obtained from **MPI_COMM_GET_PARENT;**
  - It is fatal at all *connected processes*
  - New handler **MPI_ERRORS_ABORT** aborts (only) the communicator (window/file)
  - MPI errors during operations that are **not attached to a communicator/window/file** will be raised on **MPI_COMM_SELF** (instead of MPI_COMM_WORLD)
  - Clarification of the inheritance rules: after MPI_COMM_DUP(comm1, &comm2), comm2 has the same error handler as comm1
- More info on the MPI Forum ticket #1:
  - https://github.com/mpi-forum/mpi-issues/issues/1

(Non-)Catastrophic Errors

- After an error is detected, the state of MPI is undefined *if the error is catastrophic*, that is …
- MPI is in a correct, defined state after a “non-catastrophic” error
- `MPI_Get_state(OUT state)`
  - When state is **MPI_IS_OK,** the application may continue to use MPI (that is, communicating with MPI will yield correct results).
  - When state is **MPI_IS_CATASTROPHIC**, continued use of MPI interfaces may result in undefined behavior.
- Motivating examples
  - When an error is returned during MPI_WIN_ALLOCATE_SHARED, the user can try to use non-shared memory window, or resort to 2-sided MPI instead.
  - Posting multiple iRecv, creating multiple communicators, etc, running out of MPI resources
- More information on the MPI Forum ticket #28: https://github.com/mpi-forum/mpi-issues/issues/28

Interactions between multiple recovery models

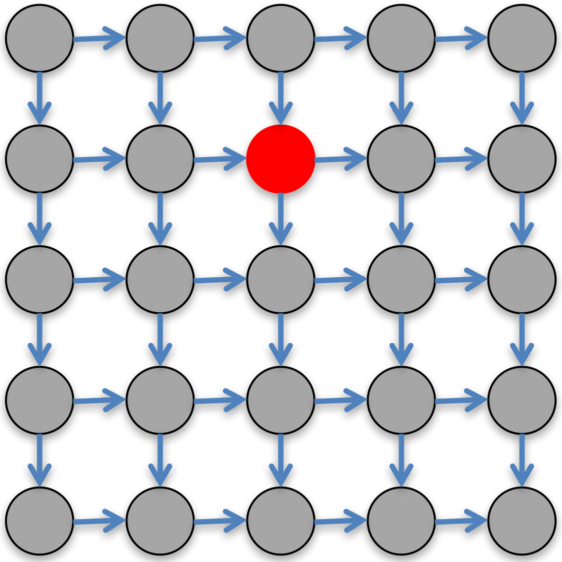
- Global C/R recovery proposed by I. Laguna & friends
  - Simpler to program and deploy
  - Limited to global C/R, no support for localized or scoped recovery
  - *Full text not produced yet* (devil is in the details ☺)

- ULFM
  - Expressive support for localized and communicator scoped recovery
  - Support for user CR and non-CR models
  - Implementing global recovery over ULFM is possible but requires more work from the user level

- WG tasked with evaluating if these models may coexist in the standard
  - WG confident that these may coexist and may be selected at runtime
  - WG still working to understand if/how an application may switch over time from one mode to the other and forth
  - WG investigating if an application may use simplified C/R on a subgroup of the processes, ULFM on another

ULFM MPI Crash Recovery

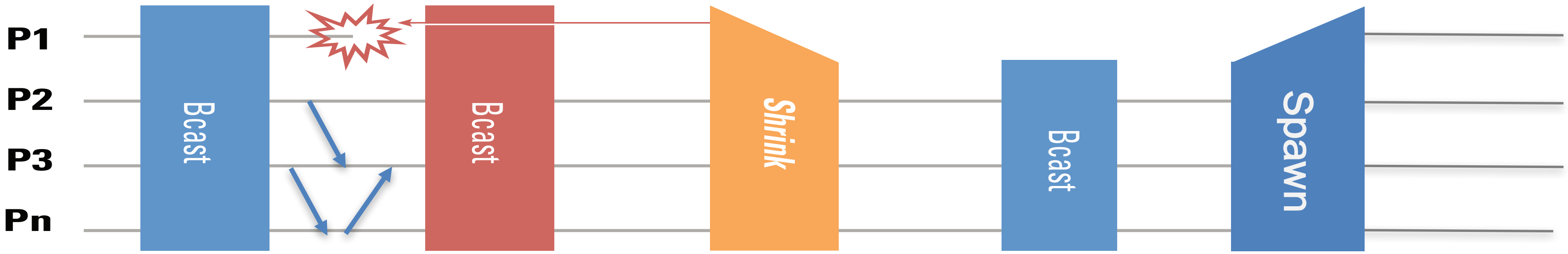
What is the scope of a failure?
Who should be notified about?
What actions should be taken?



- Failure Notification
- Error Propagation
- Error Recovery
- Respawn of nodes
- Dataset restoration

Not all recovery strategies require all of these features, that's why the interface should split notification, propagation and recovery.

- Some applications can continue w/o recovery
- Some applications are maleable
  - Shrink creates a new, smaller communicator on which collectives work
- Some applications are *not* maleable
  - Spawn can recreate a "same size" communicator
  - It is easy to reorder the ranks according to the original ordering
  - Pre-made code snippets available



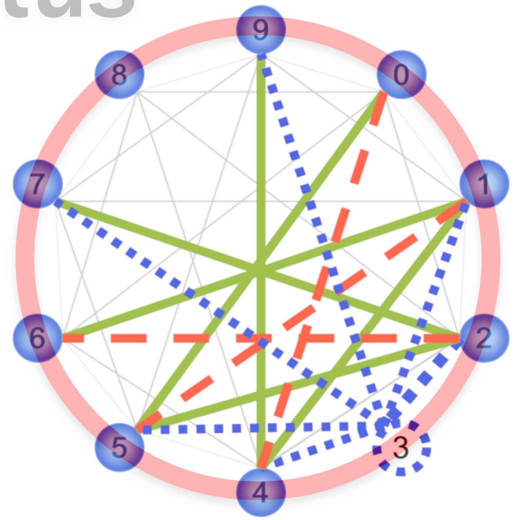
- Adds **3 error codes** and **5 functions to manage process crash**
  - **Error codes:** interrupt operations that may block due to process crash
  - **MPI_COMM_FAILURE_ACK / GET_ACKED:** continued operation with ANY-SOURCE RECV and observation known failures
  - **MPI_COMM_REVOKE** lets applications interrupt operations on a communicator
  - **MPI_COMM_AGREE**: synchronize failure knowledge in the application
  - **MPI_COMM_SHRINK**: create a communicator excluding failed processes
  - More info on the MPI Forum ticket #20: https://github.com/mpi-forum/mpi-issues/issues/20

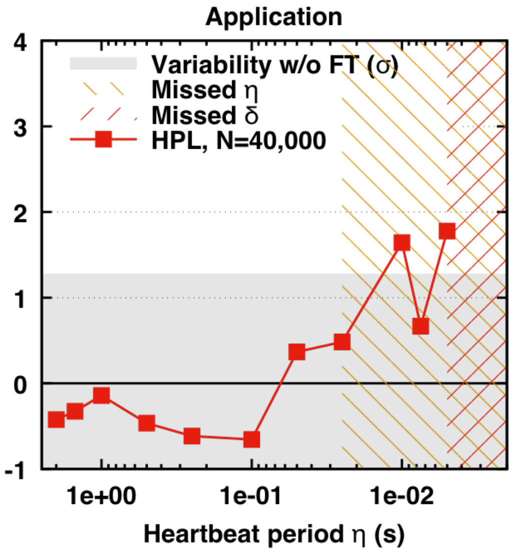
WG Researching ULFM Expansions

- Simplification of “global” recovery patterns
  - ULFM designed to provide “scoped” recovery
  - Addition of function “REVOKE_ALL” to revoke all communicators at once
- Automations
  - In many cases, one wants to discard failed communicators and requests
  - Addition of error handler “MPI_ERRORS_REVOKE, MPI_ERRORS_FREE” to automate these common usage patterns
- Run-through failures RMA
  - ULFM current design limited to ”stopping” RMA operations on a window impacted by a failure (the window may be rebuild from a communicator later)
  - Investigating more ambitious recovery models with continued operation on windows

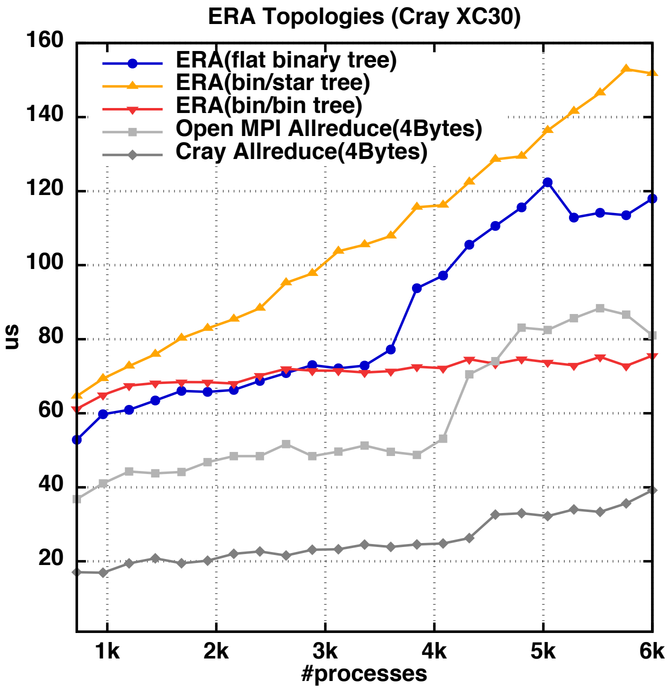
User Level Failure Mitigation: Implementation status

- ULFM available in Open MPI and MPICH
  - ULFM in MPICH release
  - Open MPI ULFM implementation updated in-sync with Open MPI master
- Scalable fault tolerant algorithms
  - Research on algorithms dedicated to HPC resilience bearing fruits
  - New algorithms demonstrated in practice (SC’14, EuroMPI’15, SC’15, SC’16)

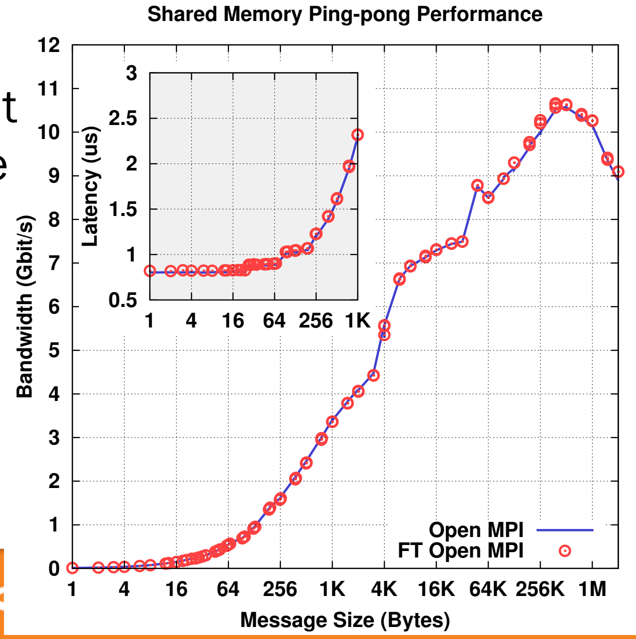




Fault Tolerant Agreement costs approximately 2x Allreduce



Point to point performance unchanged With FT enabled



User Level Failure Mitigation: User Adoption

Judicael A. Zounmevo, Dries Kimpe, Robert Ross, and Ahmad Afsahi. 2013. **Using MPI in high-performance computing services.**

MapReduce

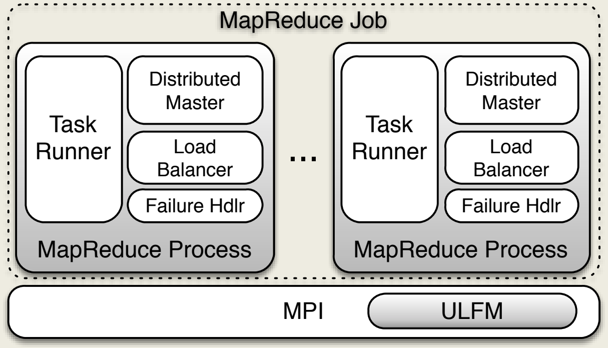


Figure 2: The architecture of FT-MRMPI.

Fenix Framework: user-level C/R With scoped recovery

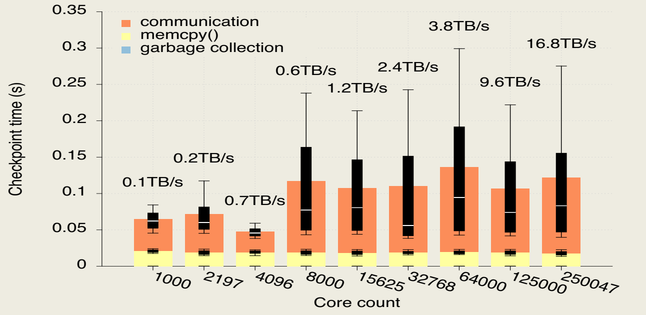


Fig. 3. Checkpoint time for different core counts (8.6 MB/core). The numbers above each test show the aggregated bandwidth (the total checkpoint size over the average checkpoint time).

SAP: Resilient Databases over MPI

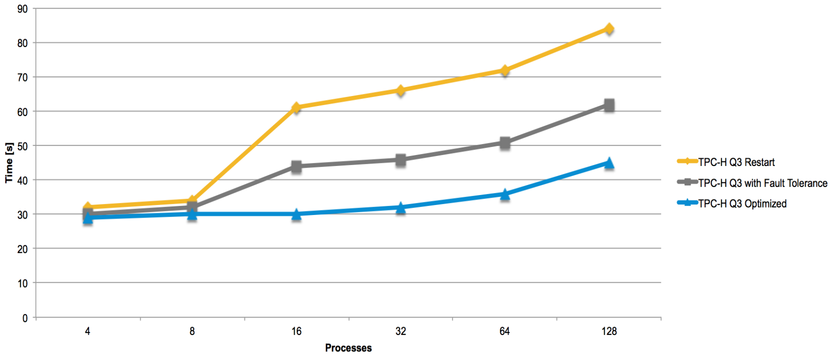
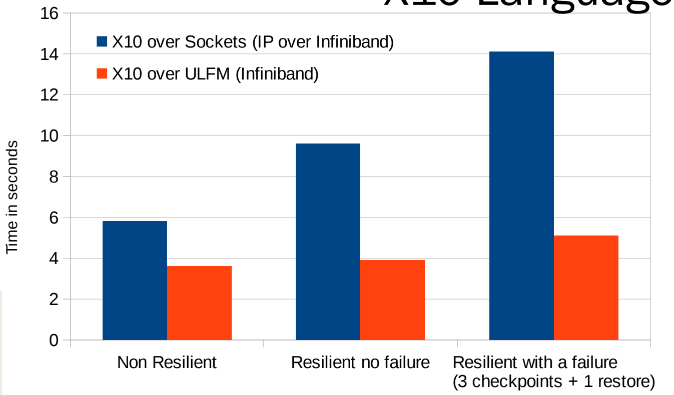


Figure 5.24: Optimization: Runtime of TPC-H Benchmark Query 3 with Failure in Phase 4 (1GB Data per Process)

Master-Thesis von Jan Stengler aus Mainz April 2017

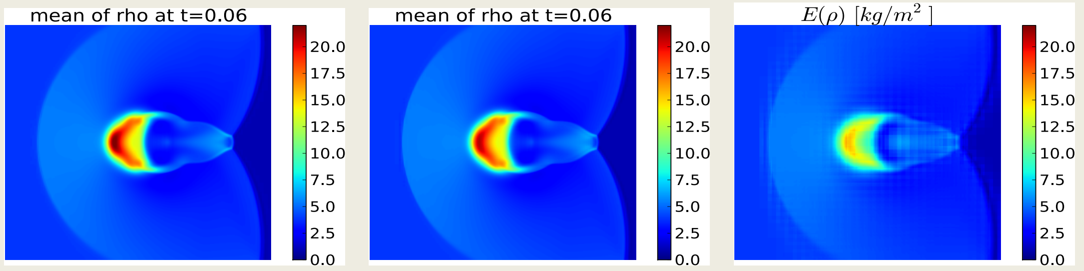
X10 Language



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

Source: Sara Hamouda, Benjamin Herta, Josh Milthorpe, David Grove, Olivier Tardieu. *Resilient X10 over Fault Tolerant MPI.*

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.

Stefan Pauli, Manuel Kohler, Peter Arbenz: ***A fault tolerant implementation of Multi-Level Monte Carlo methods.*** *PARCO 2013: 471-480*

And many more…

- Fortran CoArrays "failed images" uses ULFM-RMA to support Fortran TS 18508 in gcc-7.2

Thanks

Participate!

- WG mailing list
  - https://lists.mpi-forum.org/mailman/listinfo/mpiwg-ft
- WG issue tracker
  - https://github.com/mpiwg-ft/ft-issues
- WG meeting notes, documents, and telecon info
  - https://github.com/mpiwg-ft/ft-issues/wiki