ULFM

USER LEVEL FAILURE MITIGATION

User Level Failure Mitigation is a set of MPI extensions to report errors, provide interfaces to stabilize the distributed state, and restore the communication capabilities in applications affected by process failures. Relevant communicators, RMA windows, and I/O files can be reconstructed online, without restarting the application, as required by the user recovery strategy.

ULFM's capability to restore communication after a fault is crucial infrastructure for supporting the design and deployment of production-grade recovery strategies. Multiple applications and programming frameworks are already taking advantage of ULFM constructs to deliver varied fault tolerance strategies–from run-through algorithms that continue without rejuvenating the lost processes, to methods that restore the lost processes and their dataset–either from checkpoints or from checkpoint-free forward recovery techniques.

OPEN MPI ULFM 2.1

SOFTWARE RELEASE

- Derived directly from the latest Open MPI master
- No measurable failure-free overhead on HPC networks
- Support for fault tolerance (FT) with
  - MPI_THREAD_MULTIPLE
  - Non-blocking collective
  - uGNI, Open IB, TCP, CMA Shared-memory
  - Slurm, ALPS, PBS launchers integration
  - Simplified build and runtime arguments
- Beta support for FT
  - RMA and FILES operations

ULFM

USER COMMUNITIES

- Programming languages
  - X10 over MPI with "DeadPlace" exception support
  - CoArrays Fortran with "FailedImages" extension
- Checkpointing Frameworks
  - Fenix, CRAFTS, LFLR, VELOC
- Applications
  - PDE solvers, FTLA
- Non-HPC workloads
  - SAP Databases, Hadoop over MPI

ULFM FEATURES

FLEXIBILITY

- No predefined recovery model is imposed or favored. Instead, a set of versatile APIs is included to provide support for different recovery styles (e.g., checkpoint, ABFT, iterative, Master-Worker).
- Application directs the recovery, and it only pays for the level of protection it needs.
- Recovery can be restricted to a subgroup, thereby preserving scalability and easing the composition of libraries.

PERFORMANCE

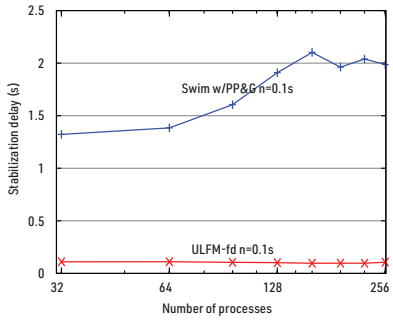
- Protective actions are outside of critical MPI routines.
- MPI implementors can uphold communication, collective, one-sided, and I/O management algorithms unmodified.
- Encourages programs to be reactive to failures and cost manifests only at recovery.

PRODUCTIVITY

- Backward compatible with legacy, fragile applications.
- Simple and familiar concepts to repair MPI.
- Provides key MPI concepts to enable FT support from library, runtime, and language extensions.

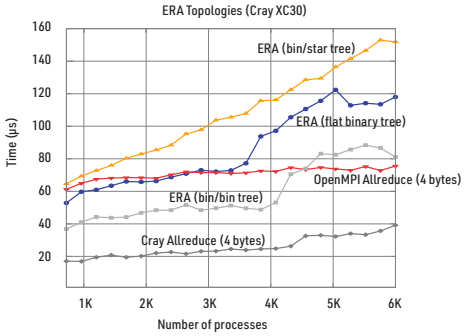
FAILURE DETECTION

Given the same heartbeat period (n), the deterministic failure detection algorithm improves scalability vs. state-of-the-art random probing detectors (Swim).



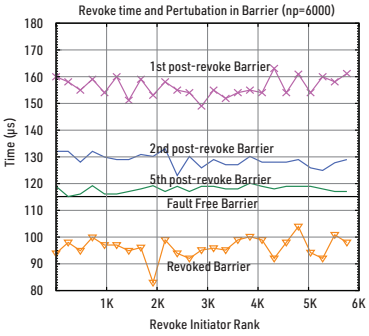
AGREEMENT

Users can stabilize the global state after a failure with this consensus operation. ERA (early returning agreement) latency is only double Cray's optimized, non-resilient Allreduce.



RELIABLE BROADCAST

Revoke permits disseminating fault information. It's latency is lower than a barrier. A reliable broadcast causes only a short burst of network activity (~700 µs).



ULFM SC18 Dallas, TX | hpc inspires.

SUNDAY, NOVEMBER 11
8:30 a.m. to 5:00 p.m.
Fault Tolerance Tutorial
ROOM C148

TUESDAY, NOVEMBER 13
12:15 p.m. to 1:15 p.m.
Open MPI State of the Union BoF
ROOM C146

WEDNESDAY, NOVEMBER 14
12:15 p.m. to 1:15 p.m.
MPI Forum BoF
C141/143/149

WEDNESDAY, NOVEMBER 14
5:15 p.m. to 6:45 p.m.
PMIx BoF
ROOM C144

WITH SUPPORT FROM

U.S. DEPARTMENT OF ENERGY

National Science Foundation

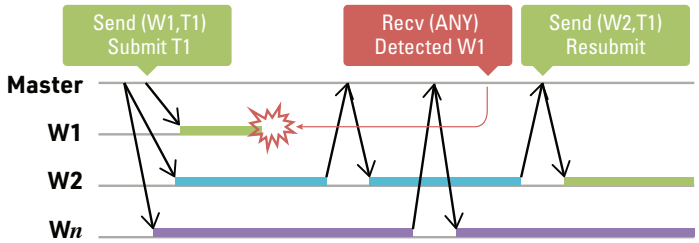
FIND OUT MORE AT
http://fault-tolerance.org/

RESILIENCE EXTENSIONS FOR MPI: ULFM

ULFM provides targeted interfaces to empower recovery strategies with adequate options to restore communication capabilities and global consistency, at the necessary levels only.

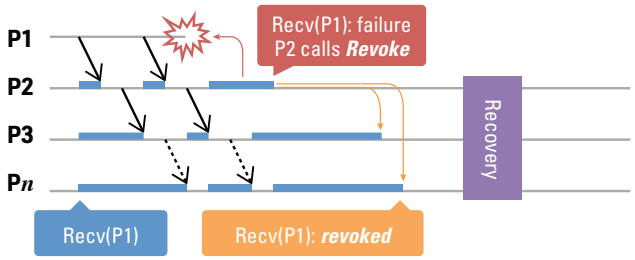
CONTINUE ACROSS ERRORS

In ULFM, failures do not alter the state of MPI communicators. Point-to-point operations can continue undisturbed between non-faulty processes. ULFM imposes no recovery cost on simple communication patterns that can proceed despite failures.



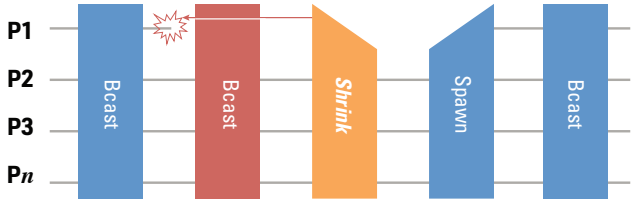
EXCEPTIONS IN CONTAINED DOMAINS

A process can use MPI_[Comm,Win,File]_revoke to propagate an error notification on the entire group, and could, for example, interrupt other ranks to join a coordinated recovery.



FULL-CAPABILITY RECOVERY

Allowing collective operations to operate on damaged MPI objects (communicators, RMA windows, or files) would incur unacceptable overhead. The MPI_Comm_shrink routine builds a replacement communicator–excluding failed processes–that can be used to resume collective operations in malleable applications, spawn replacement processes in non-moldable applications, and rebuild RMA windows and files.



ONGOING RESEARCH: EVALUATE THE COST AND EXPRESSIVITY OF ASYNCHRONOUS RECOVERY

ERROR SCOPING

Adding per-communicator (window/file) control knobs for the application to control the scope of error reporting: set Info key **mpix_error_scope** on a communicator to control which errors interrupt MPI calls.

- **"local"**: current ULFM behavior: report an error only **when communicating with a failed peer** (e.g., recv from failed process, collective communication) **default, current ULFM**
- **"group"**: report errors (i.e., REVOKE) for a **failure at any process with a rank in the comm**/win/file (e.g., in recv from an alive process in comm)
- **"global"**: report errors (i.e., REVOKE) for a **failure anywhere in "universe"**

ERROR UNIFORMITY

All processes partake in a collective operation, should they return an error in unison? Use sets info key **mpix_error_uniform** on a communicator to control if error reports need to be uniform.

- **"local"**: errors reported as needed to **inform of invalid outputs** (buffers/comms) at the reporting rank (i.e., other ranks may report success); **default, current ULFM**
- **"create"**: if communicator/win/file creation operations (e.g., comm_split, file_open, win_create, comm_spawn) reports at a rank, it has reported the same ERR_PROC_FAILED/REVOKED **at all ranks**
- **"coll"**: same as above, for all collective calls (including creates)

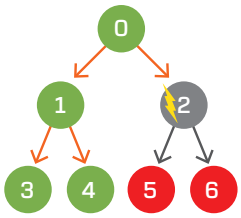
ASYNCHRONOUS ERROR RECOVERY

Error recovery is difficult to overlap, because MPI currently misses asynchronous dynamic processes constructs.

- Adding **MPI_COMM_ISHRINK** to enable asynchronous failed processes exclusion
- Adding **MPI_COMM_ISPAWN** (and ICONNECT/IACCEPT) to enable asynchronous spare respawn (as well as many other non-ft application use cases)

SCOPING EXAMPLE:
Only rank4 reports as it is communicating with failed rank 5.

UNIFORMITY EXAMPLE:
An error is reported only at some leaf node in a broadcast topology with a failure.



DOWNLOAD THE LATEST RELEASE
http://fault-tolerance.org/