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 4.0.2**
- Derived from the latest stable Open MPI 4.0.2.
- All new features of OpenMPI with resilience support.
- Same simplified build and runtime arguments as Open MPI.
- Resilience support with most networks and job schedulers:
  - Networks: UCX, uGNI, Open IB, TCP, and CMA.
  - Launchers: Slurm, ALPS, and PBS.
- No measurable failure-free overhead on HPC networks.
- Beta resilience support for RMA and FILE operations.

**ULFM User Communities**
- Programming languages:
  - X10 over MPI with ‘DeadPlace’ exception support.
  - CoArrays Fortran with ‘FailedImages’ extension.
- Checkpointing Frameworks:
  - Fenix, CRAFTS, FLFR, and VELOC.
- Applications:
  - PDE solvers and FTLA.
- Non-HPC workloads:
  - SAP Databases and Hadoop over MPI.

**Fault Detection**
The Failure Detection and notification service of ULFM is available as a PMix component (called RDaemon) and can report failures in a matter of milliseconds at 4k ranks.

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

**Reliable Broadcast**
Revoke permits disseminating fault information. Its latency is lower than a barrier. A reliable broadcast causes only a short burst of network activity (~700 µs).

---

**ULFM at SC19**
- **Monday, November 18**
  - 8:30 a.m. to 5:00 p.m.
  - Fault Tolerance for High Performance and Big Data Applications: Theory and Practice
  - Room 403
- **Tuesday, November 19**
  - 12:15 p.m. to 1:15 p.m.
  - MPI 4.0 Is Coming—What Is In It and What Is Next?
  - Rooms 201-203
- **Wednesday, November 20**
  - 12:15 p.m. to 1:15 p.m.
  - Open MPI State of the Union
  - Rooms 201-203

---

**SPONSORED BY**
- The U.S. Department of Energy Office of Science
- 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, and 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 participate in a collective operation, should they return an error in unison? The user sets the 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, and comm_spawn) reports at a rank, it has reported the same ERR_PROC_FAILED/REVOKE at all ranks
- "call": 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/ACCEPT) to enable asynchronous spare respawn (as well as many other non-fault-tolerant application use cases)