PMIx Fault Events Detection and Dissemination use case: Fault-Tolerance in Open MPI 5 and more

Aurelien Bouteiller, Zhong Dong, George Bosilca



INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

PMIX Fault events use cases
ULFM available in Open MPI 5.0
SUNY OSHMEM, and more

MOTIVATION

MPI ULFM

- Detection and propagation in the scope of MPI_COMM_WORLD and MPI process
- Need capability to handle node failures and not limited to any MPI

OpenSHMEM

- No fault tolerance model in current code stack.
- Need failure detection and propagation service to trigger check-pointing and restart.

MPI EREINIT

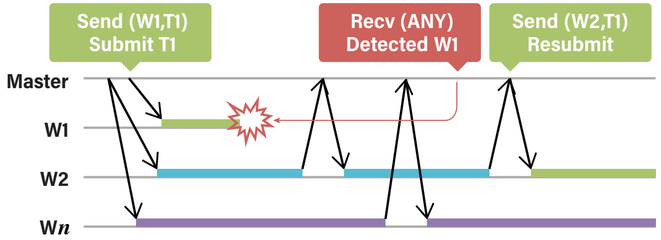
- Global-restart failure recovery by fast re-initialization of MPI
- In need of more efficient and generic service

DataSpaces and FTI

- Persistent data storage services with check-pointing feature.
- Need basic service for to detect and report failures of distributed storage service.

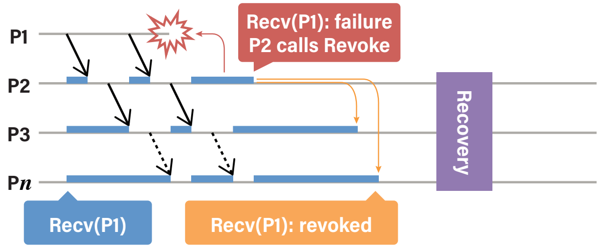
Continue across Errors

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.



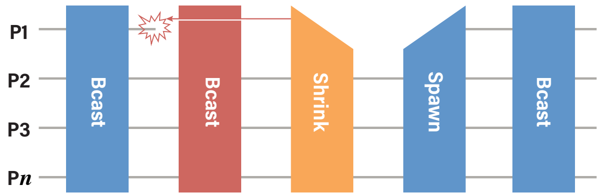
Exceptions in Contained Domains

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.



Example use cases: continue without repair, repair in domains, respawn missing processes with ULFM MPI

PMIx Fault Events Interface

Dong Zhong, Aurelien Bouteiller, Xi Luo, and George Bosilca. 2019. **Runtime Level Failure Detection and Propagation in HPC Systems.** In 26th European MPI Users' Group Meeting (EuroMPI 2019), September 11–13, 2019, Zürich, Switzerland. *https://doi.org/10.1145/3343211.3343225*

FRONT END: PMIx Interface

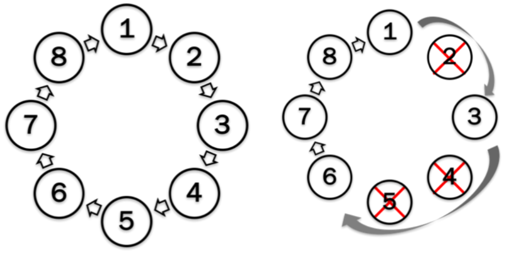
- Control of the detection service through PMIx interfaces
- **Fault Events are presented as normal PMIx Events**
- Very simple to hook into
- Compatible with non-ft builds (event is just not generated)

  - PMIX_Register_event_handler(..., PMIX_ERR_PROC_ABORTED, ...)

- Opens the gate for efficient management of failures for an emerging field of libraries, programming models, and runtime systems operating on large-scale systems.
- Well specified interface helps **usability by multiple client types**

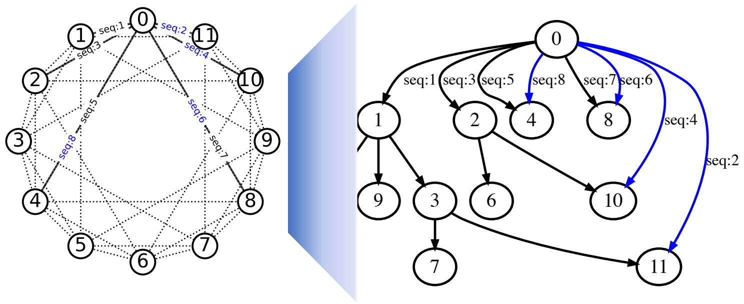
BACK END Capability integrated in **PRTE 2.0,** other LM/SMS can produce the same events using their own methodologies

FAILURE DETECTION COMPONENT



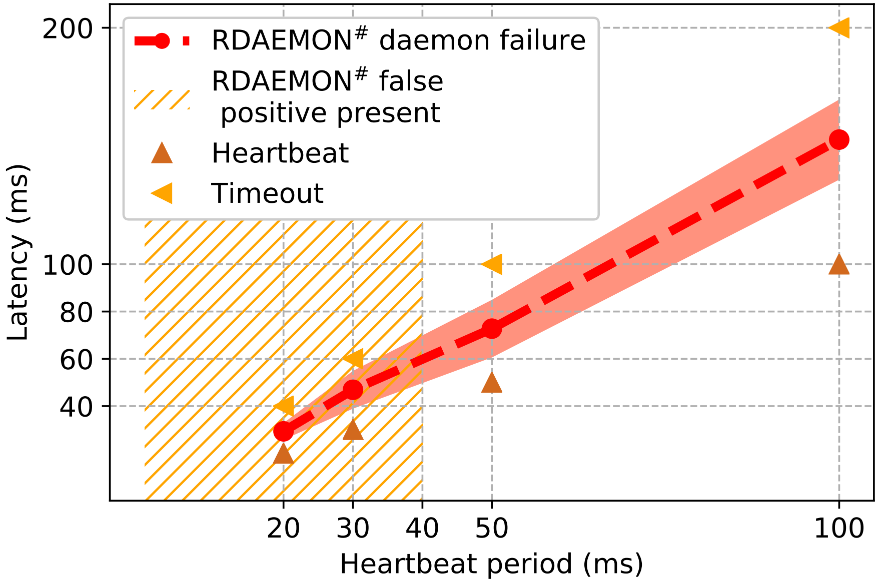
Node/PRTE servers failure detected with **ping on a mendable ring**
Minimal ping noise

RELIABLE EVENT PROPAGATION COMPONENT



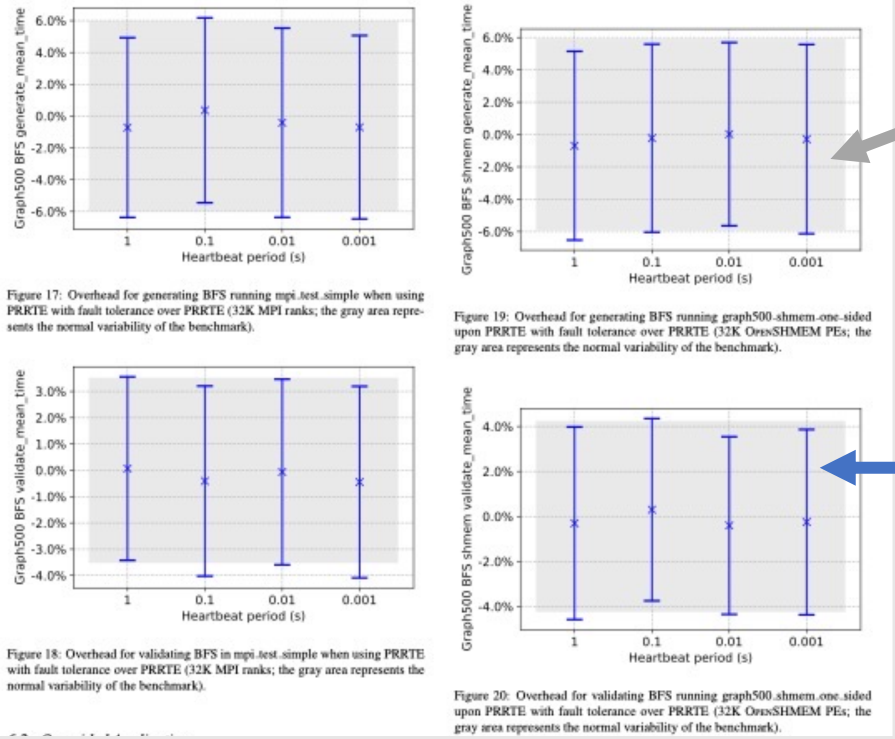
Event Propagation along **multiple binomial trees** extracted from a circulant graph: **fixed degree (log), fast and resilient**

Advantages in delegating detection to the PMIx runtime



- Accuracy of detection is very good (in the order of 100ms can be achieved in practice at scale)
- False detection rate **independent of the application communication pattern**
  - *Prior MPI-based detector would produce false positive when application does not call MPI procedures*
- *Reusable in different programming models*

Performance variability in **GRAPH500 with an active PMIx-PRRTE Failure detector**
Left: **MPI** Right: **OpenSHMEM**



Gray area represents normal benchmark variability

Blue error bars show the variability as measured with detection ON

Experiments performed on NERSC's Cori: Cray XC40 supercomputer with Intel Xeon "Haswell" processors and the Cray "Aries" high speed inter-node network, 32 cores per node, 32K processes total.

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Fault Management in PMIx What works, and Future Directions

- PMIx Event Handling proved to be a natural fit to disseminate fault events
- PMIx abstracts the ‘detector’, which lets application write portable fault-tolerant code that can operate on a variety of PMIx servers (including non-FT ones)
- PMIx servers have freedom of implementation behind the API curtain

Future Extensions

- Fault-events implemented only in PRTE at the moment (PRTE can be launched under Slurm/Jsrun/ALPS, etc., in managed mode, but not all have native support yet).
- Current enabling of fault-detection/management is command-line based
- Can we move to a programmatic way to turn-on/turn-off resilient features?
- PRTE: Resilient overlay communication for commands/modex (reduce vulnerability to startup faults)
- Could user have fine-grain control over what operation (e.g., PMIx modex or Fence) are resilient, or not (performance optimization)?

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

ICL
INNOVATIVE
COMPUTING LABORATORY

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE