ULFM Update (March to June 2014)

FTWG@MPI Forum meeting

Chicago, june 2014

ICL UT
INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE

Application Recovery Patterns

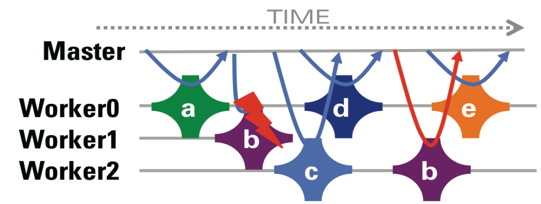
User Level Failure Mitigation: a set of MPI interface extensions to enable MPI programs to restore MPI communication capabilities disabled by failures

Coordinated Checkpoint/Restart, Automatic, Compiler Assisted, User-driven Checkpointing, etc.

In-place restart (i.e., without disposing of non-failed processes) accelerates recovery, permits in-memory checkpoint

Naturally Fault Tolerant Applications, Master-Worker, Domain Decomposition, etc.

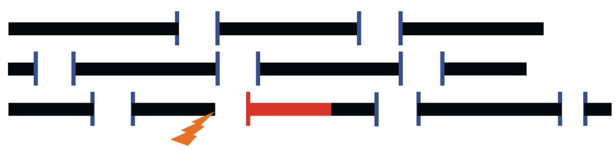
Application continues a simple communication pattern, ignoring failures



ULFM MPI Specification

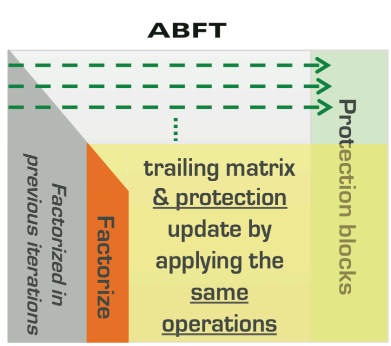
Uncoordinated Checkpoint/Restart, Transactional FT, Migration, Replication, etc.

ULFM makes these approaches portable across MPI implementations



Algorithm Fault Tolerance

ULFM allows for the deployment of ultra-scalable, algorithm specific FT techniques.



Minimal Feature Set for FT

- Failure Notification (MPI exceptions)
- Error Propagation (Revoke)
- Error Recovery (Shrink,Agree,Spawn)

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

Modified

- Ticket 0 (Bill’s comments)
- Finalize completes (removed “successfully”, due to attributes destructors)
- RMA rewrite
  - Relaxed memory consistency after failure (the entire window exposed memory may become undefined)
  - Relaxed error raising requirement (previous text overspecified our intention, now only synchronization function must raise exceptions)
  - Added advice on win_free (after raising PF, it becomes non-synchronizing, users should be careful).
  - Added advice to implementors “please, do not continue to deliver RMA operations from dead processes after win_free”
- Agree is now a bitwise AND (on integer)
- Examples use error classes and codes properly

Considered, but discarded

- MPI_Comm_ishrink
  - Performance benefit unclear at this point
  - Postponed until proven to serve a purpose (that is a better implementation than doing it all in wait is possible)
- MPI_Win_free synchronizes even with failures
  - Considered too costly
  - There is a way out for users, it can then be deployed only when FT is necessary

Coming next

- Upgrade from F77 to F08 interfaces
- Query of FT support
  - Predefined attribute on MPI_COMM_WORLD or info key in MPI_INFO_ENV
  - Alternative: using MPI_Init_with_info (future ticket from Hybrid group)
- Query status of revoked handles
  - MPI_Comm_is_revoked(comm)
- MPIT keys and better interaction with tools

Implementation progress

- Open MPI implementation
  - Failure free performance is satisfactory
  - Poor agreement algorithm in the current implementation results in poor post-failure performance
  - Work ongoing to provide better Agreement (ERA early august)
- MPICH implementation also well advanced

Some more applications

- Large number of papers at EuroMPI about ULFM (5+ submissions)

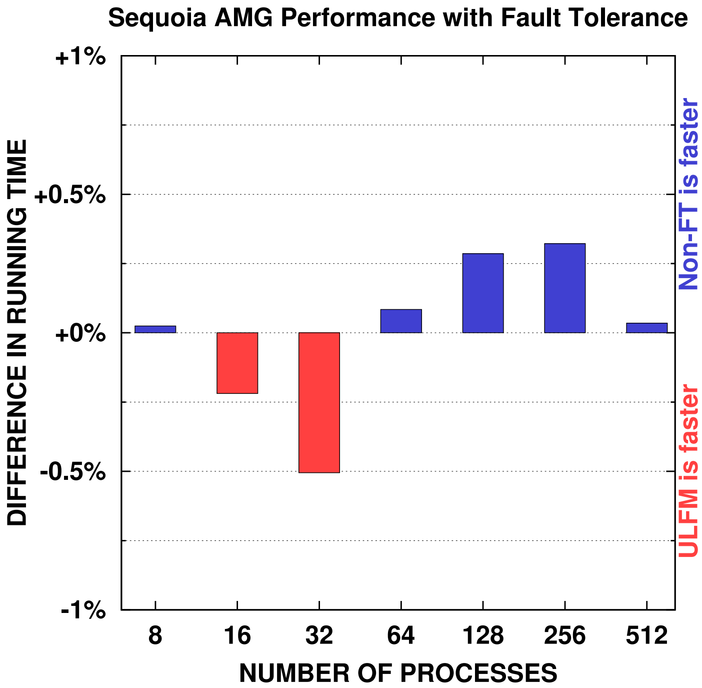
- IPDPS: ANU presented a sparse PDE code (deployed in GENE application)

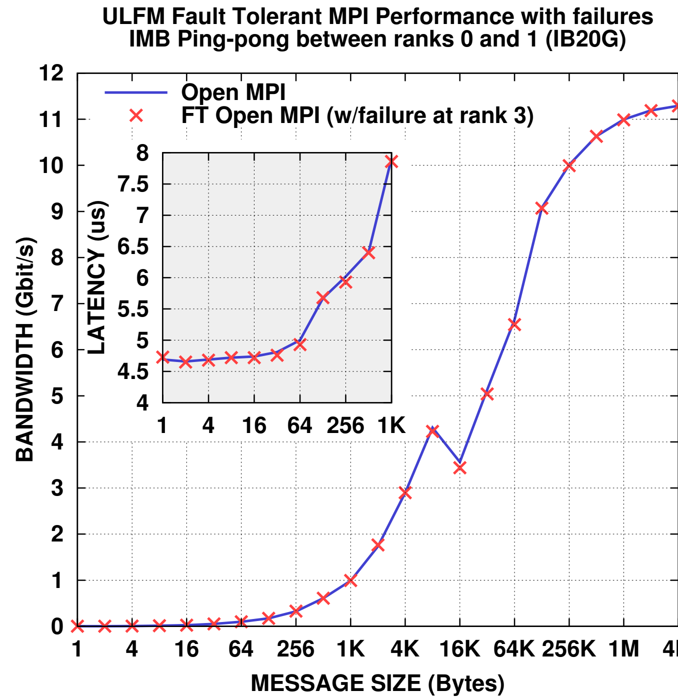
Getting more info

- Ticket wiki with standard text, links
  - https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/323
- Users and applications using ULFM
  - http://fault-tolerance.org/2014/05/27/anu-presents-pde-solver-with-ulfm-at-ipdps/20140527-ulfm-users/
- Example codes and modular snippets
  - http://fault-tolerance.org/2014/02/04/slides-with-ulfm-examples/ulfm-mpi-dec13forum/
- Implementations
  - Open MPI: https://bitbucket.org/icldistcomp/ulfm
  - MPICH
- Publications
  - Bland, W., Bouteiller, A., Herault, T., Bosilca, G., Dongarra, J.J. "Post-failure recovery of MPI communication capability: Design and rationale," *International Journal of High Performance Computing Applications August 2013 27: 244-254, doi:10.1177/1094342013488238*
  - Bland, W., Bouteiller, A., Herault, T., Hursey, J., Bosilca, G., Dongarra, J.J. "An Evaluation of User-Level Failure Mitigation Support in MPI," Computing, Springer, 2013, issn 0010-4885X, http://dx.doi.org/10.1007/s00607-013-0331-3

Implementation in Open MPI

- It works! Performance is good!





Sequoia AMG is an unstructured physics mesh application with a complex communication pattern that employs both point-to-point and collective operations. Its failure free performance is unchanged whether it is deployed with ULFM or normal Open MPI.

The failure of rank 3 is detected and managed by rank 2 during the 512 bytes message test. The connectivity and bandwidth between rank 0 and rank 1 are unaffected by failure handling activities at rank 2.

ICL UT

Thanks for CREST, Riken support

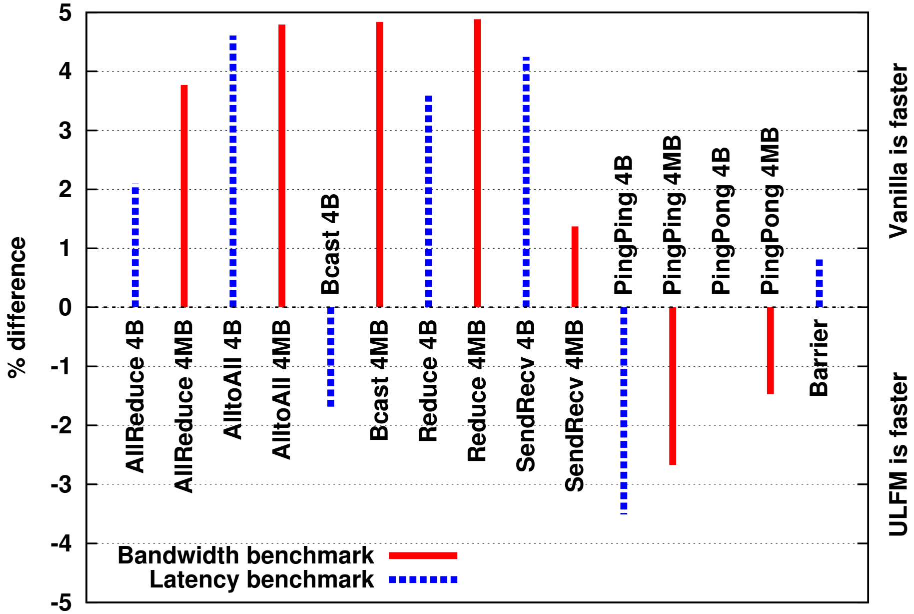
More performance: synthetic benchmarks

1-byte Latency (microseconds) (cache hot)

Interconnect	Vanilla	Std. Dev.	Enabled	Std. Dev.	Difference
Shared Memory	0.8008	0.0093	0.8016	0.0161	0.0008
TCP	10.2564	0.0946	10.2776	0.1065	0.0212
OpenIB	4.9637	0.0018	4.9650	0.0022	0.0013

Bandwidth (Mbps) (cache hot)

Interconnect	Vanilla	Std. Dev.	Enabled	Std. Dev.	Difference
Shared Memory	10,625.92	23.46	10,602.68	30.73	-23.24
TCP	6,311.38	14.42	6,302.75	10.72	-8.63
OpenIB	9,688.85	3.29	9,689.13	3.77	0.28

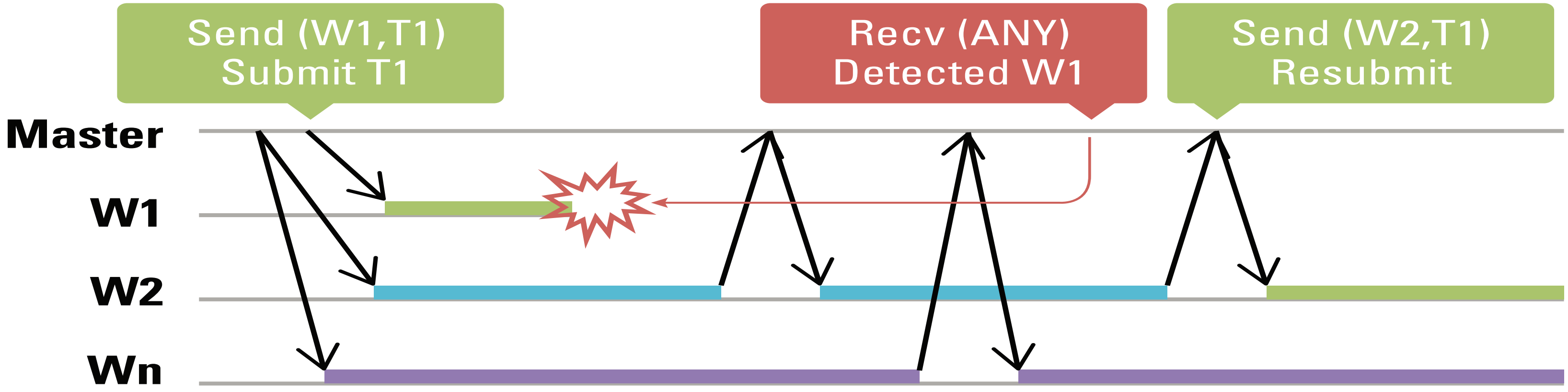


Collective communications:
48 core shared memory (very stressful)
Performance difference is less than std-deviation

Failure Notification

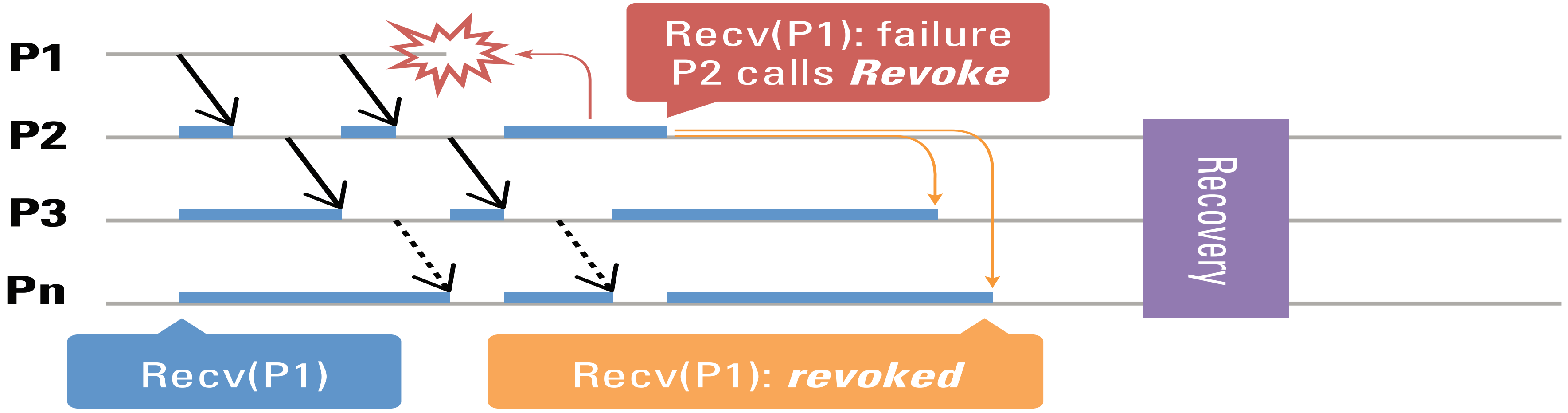
- Notification of failures is local only
  - New error MPI_ERR_PROC_FAILED Raised when a communication with a targeted process fails
- In an operation (collective), some process may succeed while other raise an error
  - Bcast might succeed for the top of the tree, but fail for some subtree rooted on a failed process
- ANY_SOURCE must raise an exception
  - the dead could be the expected sender
  - Raise error MPI_ERR_PROC_FAILED_PENDING, preserve matching order
  - The application can complete the recv later (MPI_COMM_FAILURE_ACK())
- Exceptions indicate an operation failed
  - To know what process failed, apps call MPI_COMM_FAILURE_ACK(), MPI_COMM_FAILURE_GET_ACKED()

App using notification only



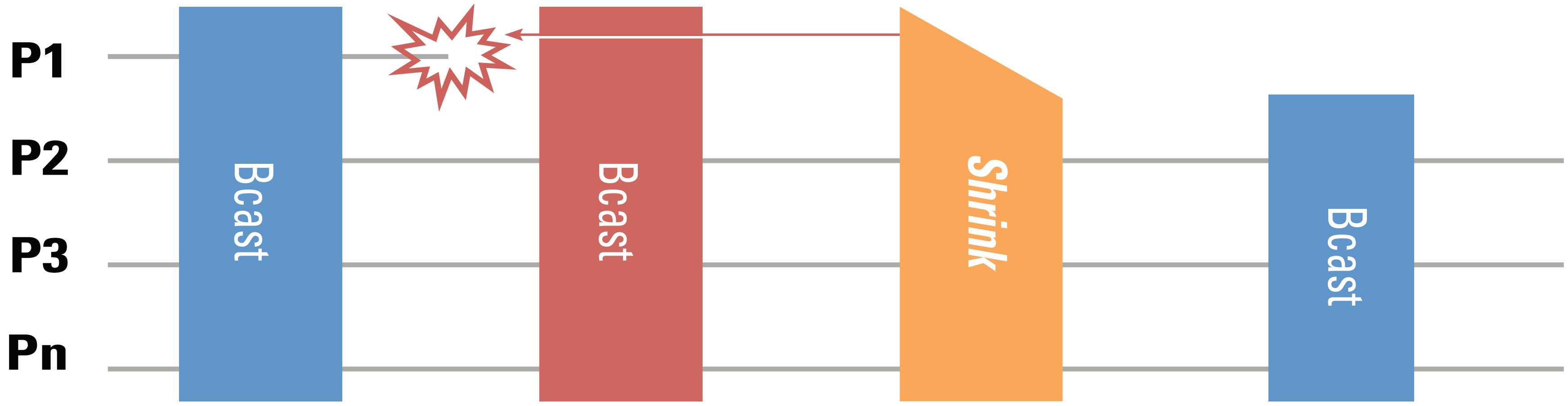
- Error notifications do not break MPI
  - App can continue to communicate on the communicator
  - More errors may be raised if the op cannot complete (typically, most collective ops are expected to fail), but p2p between non-failed processes works
- In this Master-Worker example, we can continue w/o recovery!
  - Master sees a worker failed
  - Resubmit the lost work unit onto another worker
  - Quietly continue

App using propagation only



- Application does only p2p communications
- P1 fails, P2 raises an error and wants to change comm pattern to do application recovery
- but P3..Pn are stuck in their posted recv
- P2 unlocks them with Revoke
- P3..Pn join P2 in the new recovery p2p communication pattern

Error Recovery



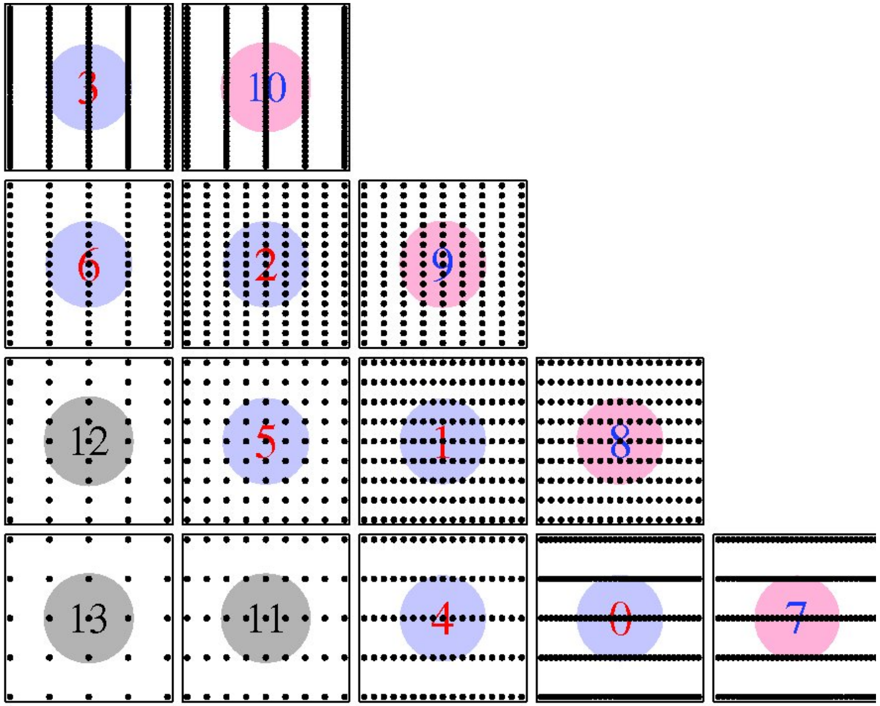
- Restores full communication capability (all collective ops, etc).
- MPI_COMM_SHRINK(comm, newcomm)
  - Creates a new communicator excluding failed processes
  - New failures are absorbed during the operation
  - The communicator can be restored to full size with MPI_COMM_SPAWN

Error Agreement

- When in need to decide if there is a failure and if the condition is recoverable (collectively)
  - MPI_COMM_AGREE(comm, flag)
    - Fault tolerant agreement over boolean flag
    - Unexpected failures (not acknowledged before the call) raise MPI_ERR_PROC_FAILED
    - The flag can be used to compute a user condition, even when there are failures in comm
- Can be used as a global failure detector

ANU: Sparse PDE

4 Two-dimension PDE Solver: Recovery Methods



- replication/re-sampling:

  recover grids 0–3 from duplicate grids 7–10;

  recover grids 4–6 via resampling from grid 0–3

- alternate combination:

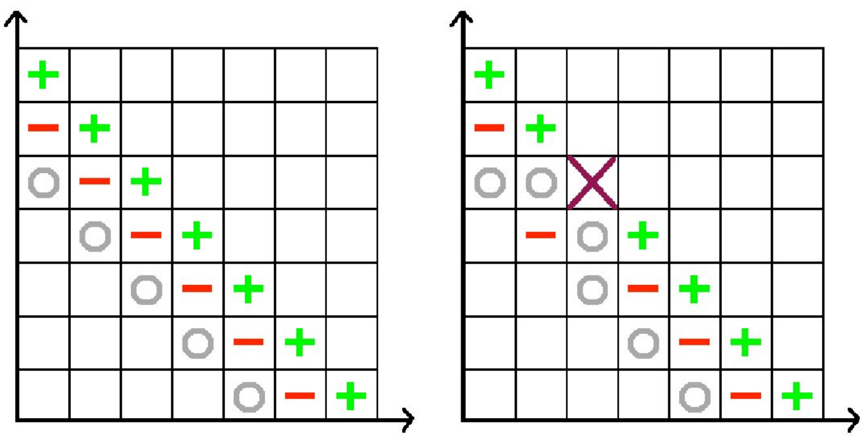
  lost grid $g \in \{0..6\}$ is ignored; final result (sparse grid) is constructed via a subset of $\{0..6, 11..13\} - \{g\}$

Full slides deck available from http://cs.anu.edu.au/Peter.Strazdins/seminars

ANU: Sparse PDE

5 Recovery Methods: Alternate Combination Formula

- uses extra set of smaller sub-grids on a 3rd (next lower) diagonal (modest amount of extra overhead)
- for a single failure on a fine sub-grid, can find a new combination with an inclusion/exclusion principle avoiding the failed sub-grid
- also works for many (but not all) cases of multiple failures

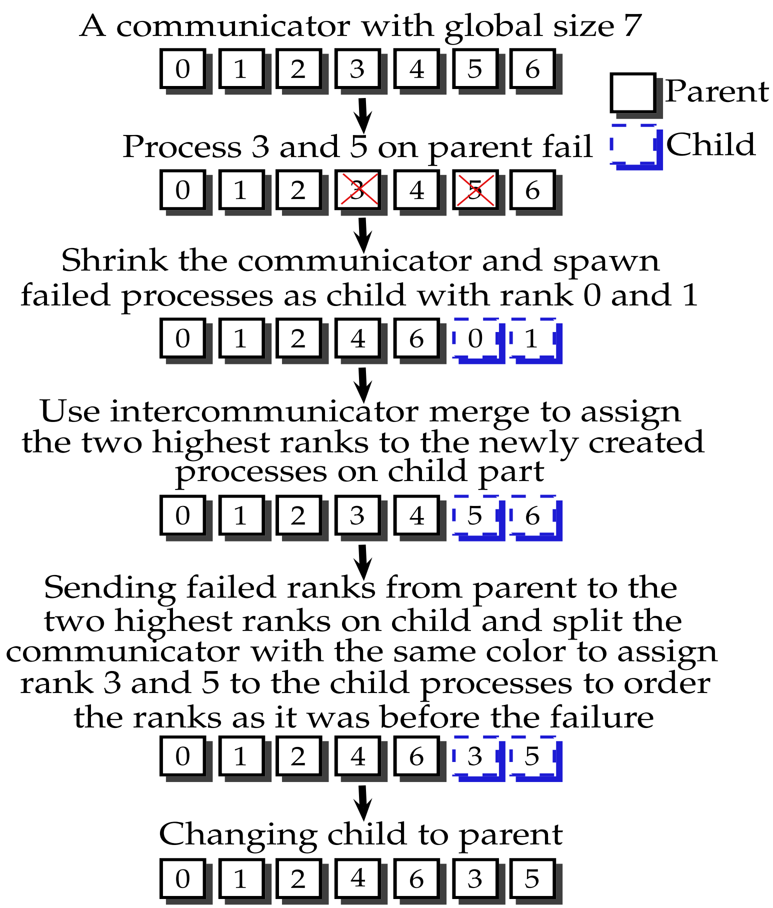


- if the failure is on 2nd diagonal, can similarly use a 4th (lower) diagonal to avoid this

THE AUSTRALIAN NATIONAL UNIVERSITY

ANU: Sparse PDE

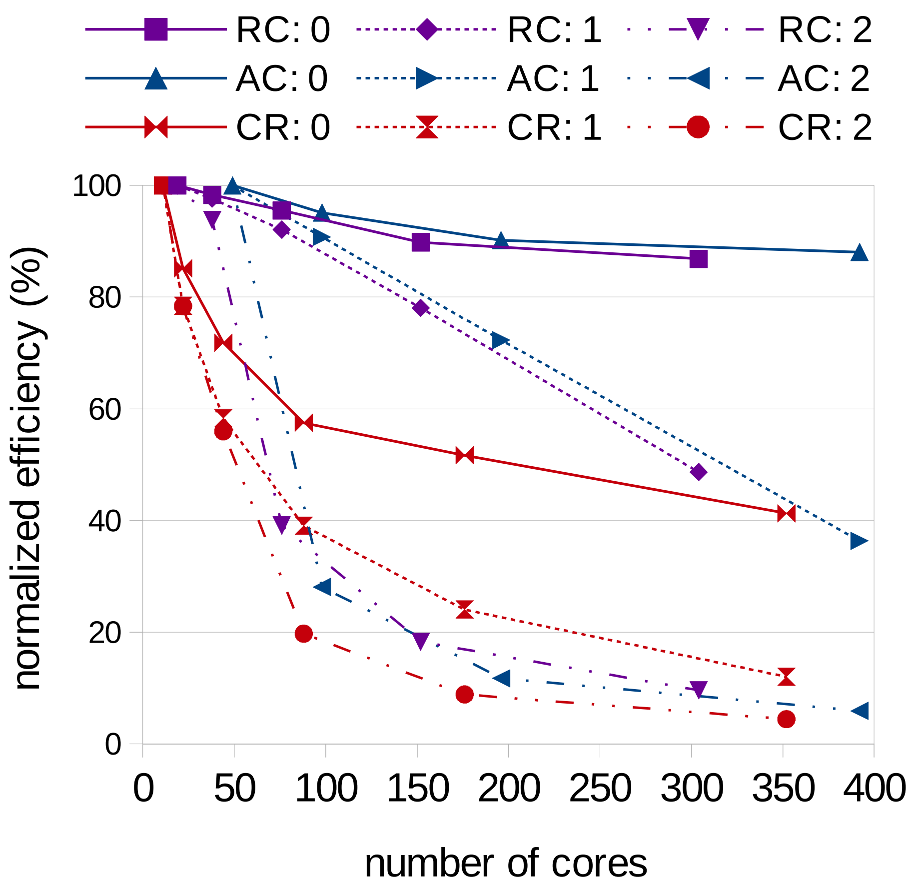
7 Fault Recovery Procedure: Detect Failed Process



- can detect failed processes as follows:
  - attach an error handler ensuring failures get acknowledged on (original) communicator `comm`
  - call `MPI_Barrier(comm)`; if fails:
  - revoke it via `MPI_Comm_revoke(comm)` and create shrunken communicator via `OMPI_Comm_shrink(comm, &scomm)`
  - use `MPI_Group_difference(..., &fg)` to make a globally consistent list of failed processes

ANU: Sparse PDE

12 Results: Scalability



RC=Replication/resampling
AC=Alternate recombination
CR=Checkpoint/Restart

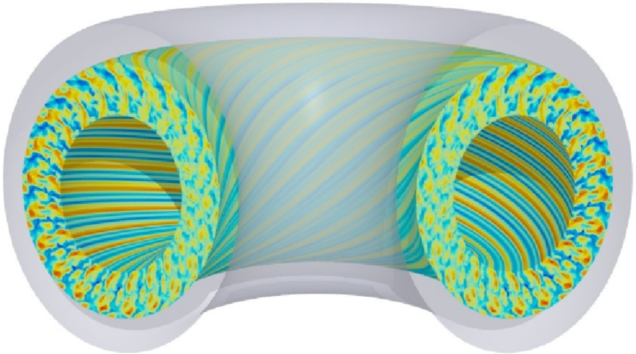
- results on OPL cluster, max. resolution of 2^{13}
- in terms of absolute time, CR is always more longer (however, uses fewer processes)
- RC and AC also show best scalability
- plots for 2 failures erratic due to high overheads in β version of ULFM MPI

OPL cluster node: 2x6 cores Xeon5670, QDR IB

ANU: GENE application

13 Fault Recovery of a Real Application - GENE

- GENE: Gyrokinetic Electromagnetic Numerical Experiment
  - plasma microturbulence code
  - multidimensional solver of Vlasov equation
  - fixed grid in five-dimensional phase space $(x_{||}, x_\perp, x_r, v_{||}, v_\perp)$



- computes gyroradius-scale fluctuations and transport coefficients
  - these fields are the main output of GENE
- hybrid MPI/OpenMP parallelization – high scalability to 2K cores

cores	t_g	t_c	Δt_f	t_G
49	48.9	3.4	1.0	107.6
98	36.8	3.8	7.4	65.3
196	63.2	11.5	19.9	98.7

times: t_g for GENE instance
t_c for comb. alg.
Δt_f extra for one failure
t_G for full-grid GENE instance