ULFM Process Fault Tolerance reading

Aurelien Bouteiller
FT WG

MPI Forum, June 2016
Bellevue, WA

INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Info, resources, participate

- Issue Ticket (w/ links to PRs)
  - https://github.com/mpi-forum/mpi-issues/issues/20

- Implementation available
  - Version 1.1 based on Open MPI 1.6 released early November 2015
    https://bitbucket.org/icldistcomp/ulfm
  - Full communicator-based (point-to-point and all flavors of collectives) support
  - Network support IB, uGNI, TCP, SM
  - Runs with ALPS, PBS, etc…
  - RMA, I/O in progress

http://fault-tolerance.org/

Use cases: Fenix+S3D

- Fenix is a framework to provide scoped user level checkpoint/restart
  - Provides some of the same services provided by the “MPI_Reinit” idea floated around by T. Gamblin and I. Laguna
  - Recover failed processes with revoke-shrink-spawn-reoder sequence
  - Revovered and surviving processes jump back to the start (longjump in Fenix_init)
  - Fenix has helpers to perform user directed “in-memory” or “buddy” checkpointing (and reload)
  - Injection of FT layer: PMPI based
- **Fenix_Checkpoint_Allocate** mark a memory segment (baseptr,size) as part of the checkpoint.
- **Fenix_Init** Initialize Fenix, and restart point after a recovery, status contains info about the restart mode
- **Fenix_Comm_Add** can be used to notify Fenix about the creation of user communicators
- **Fenix_Checkpoint** performs a checkpoint of marked segments

```
allocate(yspc(nx,ny,nz,nslvs))
allocate(other_arrays)
call MPI_Init()
[...] ! Initialize non-conflicting modules
call Fenix_Checkpoint_Allocate(C_LOC(yspc),
     sizeof(yspc),ckpt_yspc)
call Fenix_Init(Fenix_Neighbors,PEER_NODE_SIZE,
     Fenix_resume_to_init, status, C_LOC(world))

if(status.eq.Fenix_st_survivor) then
     [...] ! Finalize conflicting modules
endif
[...] ! Initialize conflicting modules
if(status.eq.Fenix_st_new)
     call initialize_yspc()
endif

do ! Main loop
     [...]  ! Iterate and update yspc array
     if(mod(step-1,CHECKPOINT_PERIOD).eq.0) then
          call Fenix_Checkpoint(ckpt_yspc);
     endif
enddo

call Fenix_Finalize()
call MPI_Finalize()
```

GAMELL, Marc, KATZ, Daniel S., KOLLA, Hemanth, *et al.* Exploring automatic, online failure recovery for scientific applications at extreme scales. In : *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014. p. 895-906.

Use cases: Fenix+S3D

- S3D is a production, highly parallel method-of-lines solver for PDEs
  - used to perform first-principles-based direct numerical simulations of turbulent combustion
- S3D rendered fault tolerant using Fenix/ULFM
- 35 lines of code modified in S3D in total!
- Order of magnitude performance improvement in failure scenarios
  - thanks to online recovery and in-memory checkpoint advantage over I/O based checkpointing
- Injection of FT layer: addition of a couple of Fenix calls

```
call MPI_Comm_split(gcomm, py+1000*pz, r, xcomm)
call MPI_Comm_split(gcomm, px+1000*pz, r, ycomm)
call MPI_Comm_split(gcomm, px+1000*py, r, zcomm)
call Fenix_Comm_Add(xcomm);
call Fenix_Comm_Add(ycomm);
call Fenix_Comm_Add(zcomm);
[...]
call MPI_Comm_split(gcomm, xid, r, yz_comm)
call MPI_Comm_split(gcomm, yid, r, xz_comm)
call MPI_Comm_split(gcomm, zid, r, xy_comm)
call Fenix_Comm_Add(yz_comm);
call Fenix_Comm_Add(xz_comm);
call Fenix_Comm_Add(xy_comm);
```

S3D Code snippet to declare to Fenix the communicators to recover

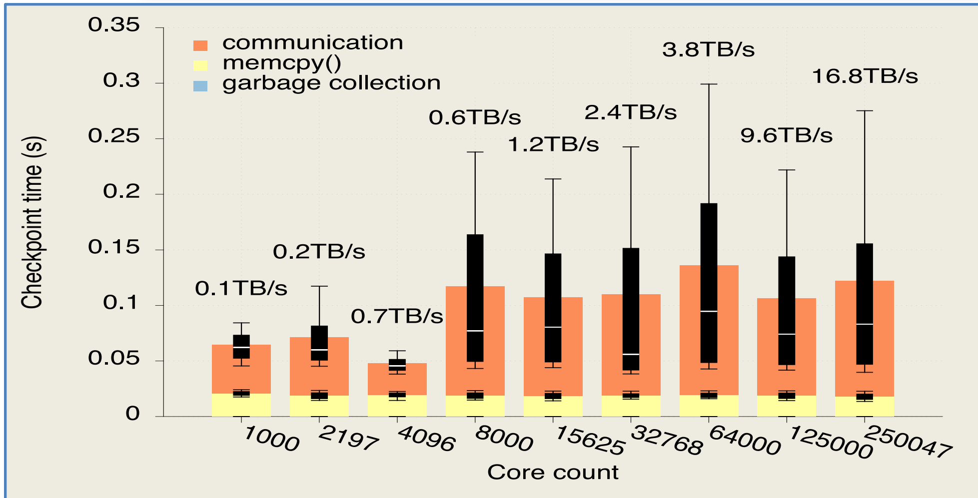


Fig. 3. Checkpoint time for different core counts (8.6 MB/core). The numbers above each test show the aggregated bandwidth (the total checkpoint size over the average checkpoint time).

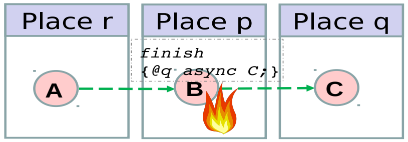
Pictures and examples from M. Gamel&al SC14' paper

Use cases: Resilient X10

- X10 is a PGAS programming language
  - Legacy resilient X10 TCP based

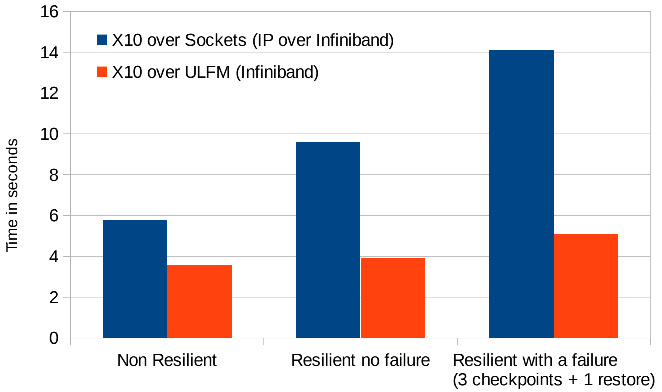
Happens Before Invariance Principle (HBI): *Failure of a place should not alter the happens before relationship between statements at the remaining places.*

```
try{ /*Task A*/
 at (p) { /*Task B*/
  finish { at (q) async { /*Task C*/ } }
 }
} catch(dpe:DeadPlaceException){ /*recovery steps*/}
D;
```



By applying the HBI principle, Resilient X10 will ensure that statement D executes after Task C finishes, despite the loss of the synchronization construct (finish) at place p

- MPI operations in resilient X10 runtime
  - Progress loop does MPI_Iprobe, post needed recv according to probes
  - Asynchronous background collective operations (on multiple different comms to form 2d grids, etc).
- Recovery
  - Upon failure, all communicators recreated (from shrinking a large communicator with spares, or using MPI_COMM_SPAWN to get new ones)
  - Ranks reassigned identically to rebuild the same X10 “teams”
- Injection of FT layer
  - Unnecessary, x10 has a runtime that hides all MPI from the application and handles failures internally



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.* In : poster session SC’15, Austin, TX, 2015.

Use cases: CoArrays "failed images" WIP to support Fortran TS 18508

```
if (num_images(failed=.true.) > 0 ) then
  form subteam(1, recover)
  sync all (stat=st) ! Will return stat_failed_image
  change team (recover)
    : ! Execute as a subteam
  end team
end if
```

Additional Coarray Features in Fortran, John Reid (JKR Associates), 7th international conference on PGAS programmning models, 2013

- Implementation effort in progress using ULFM
  - Failure detection/propagation is communicator-based (service communicator)
  - RMA based communications (win_revoke interrupted)
  - Team repair based on comm_shrink – team windows recreated from service communicator

Use cases: Monte-Carlo PDE solver

- ALSVID-UQ algorithm solving the 2-dimensional stochastic Euler equations of gas dynamics.
  - Multi-level Monte-carlo expressed as a telescopic sum

$$E[X_{h_L}] = E_{M_0}[X_{h_0}] + \sum_{\ell=1}^{L} E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}].$$

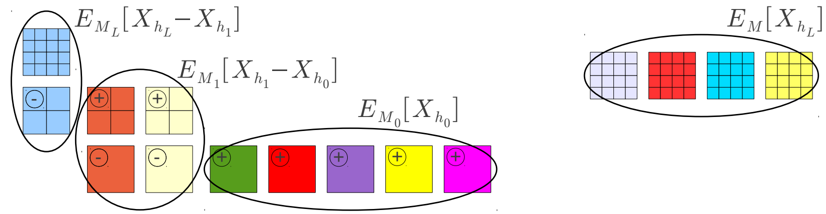


Figure 2. The idea of MLMC is illustrated on the left and compared to the MC method on the right.

- Communication pattern:
  - P2p Halo exchange between decomposed domains
  - Collective allreduce inside levels (between domains)
  - Collective aggregation between levels
- FT pattern:
  - Fine levels domain decomposed, with halo exchange between domains and in-memory checkpoints on neighbors processes, work redistributed after failure
  - Coarse domains replicated (failure ignored)
  - Failure of all processes holding a domain looses the results for that domain
  - Massive failure will degrade the solution

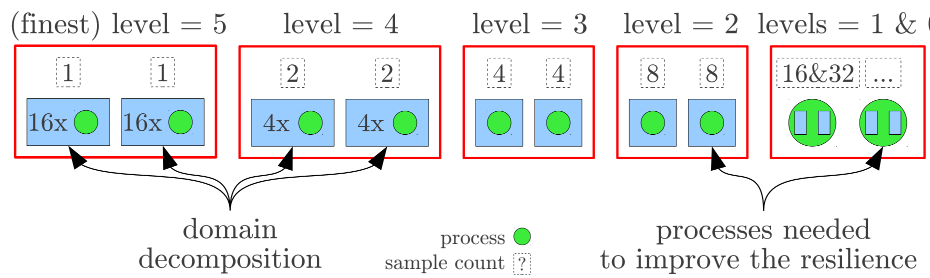


Figure 4. Parallel distribution of work in FT-MLMC with improved failure resilience.

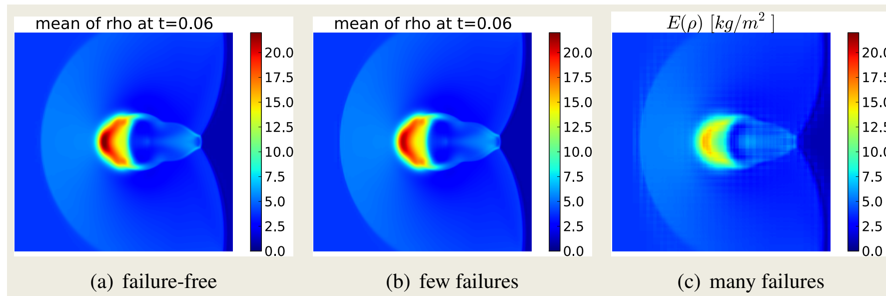


Figure 5. Results of the FT-MLMC implementation for three different failure scenarios.

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

Use cases: Hadoop over MPI

- Non-HPC workflow usually do not consider MPI because it lacks FT

  Judicael A. Zounmevo, Dries Kimpe, Robert Ross, and Ahmad Afsahi. 2013. Using MPI in high-performance computing services. In *Proceedings of the 20th European MPI Users' Group Meeting* (EuroMPI '13). ACM, New York, NY, USA, 43-48.*SE), 2013 IEEE 16th International Conference on*. IEEE, 2013. p. 58-65.

  - ULFM permits high performance exchange in non-HPC runtimes (like Hadoop)

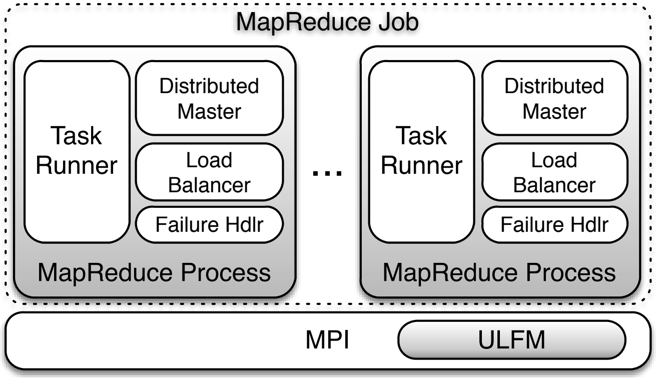


Figure 2: The architecture of FT-MRMPI.

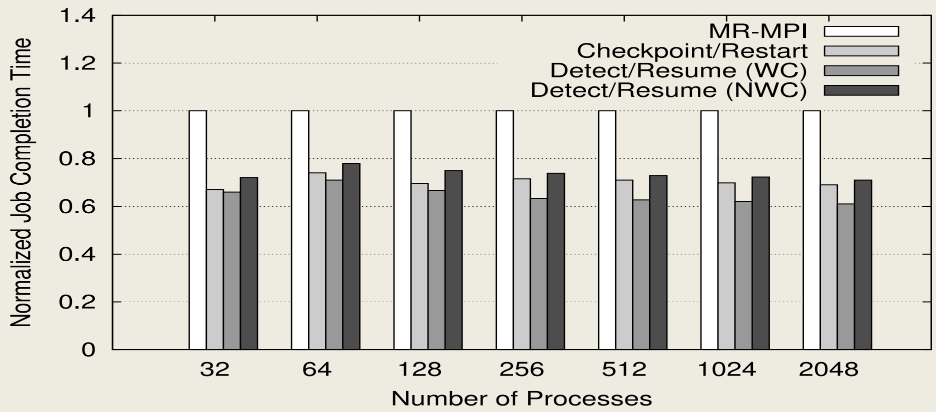
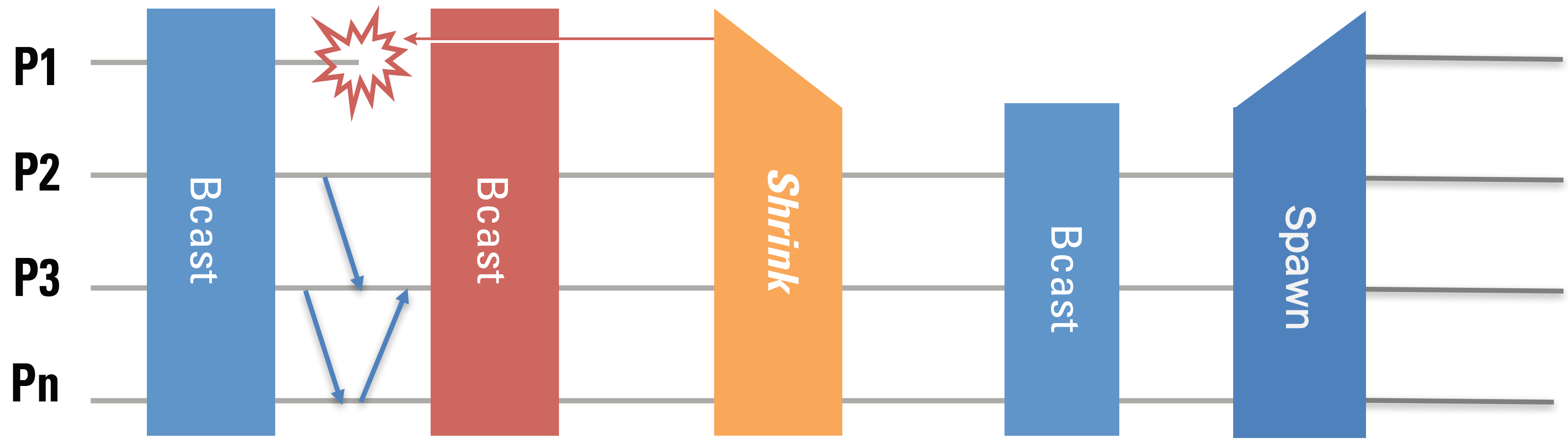


Figure 8: Normalized job completion time of failed and recovery run.

Yanfei Guo, Wesley Bland, Pavan Balaji, and Xiaobo Zhou. 2015. Fault tolerant MapReduce-MPI for HPC clusters. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (SC '15). ACM, New York, NY, USA, , Article 34 , 12 pages.

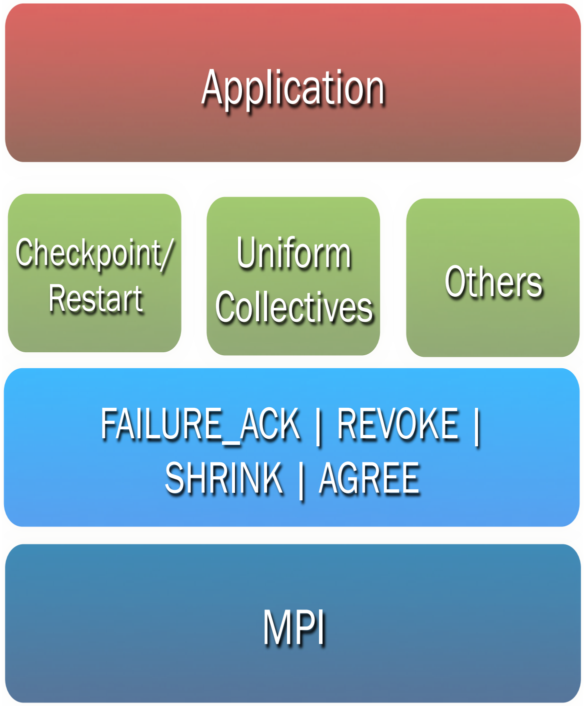
Capabilities Recovery



- 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

Minimal Feature Set for a Resilient MPI

1. Failure Notification
2. Error Propagation
3. Error Recovery



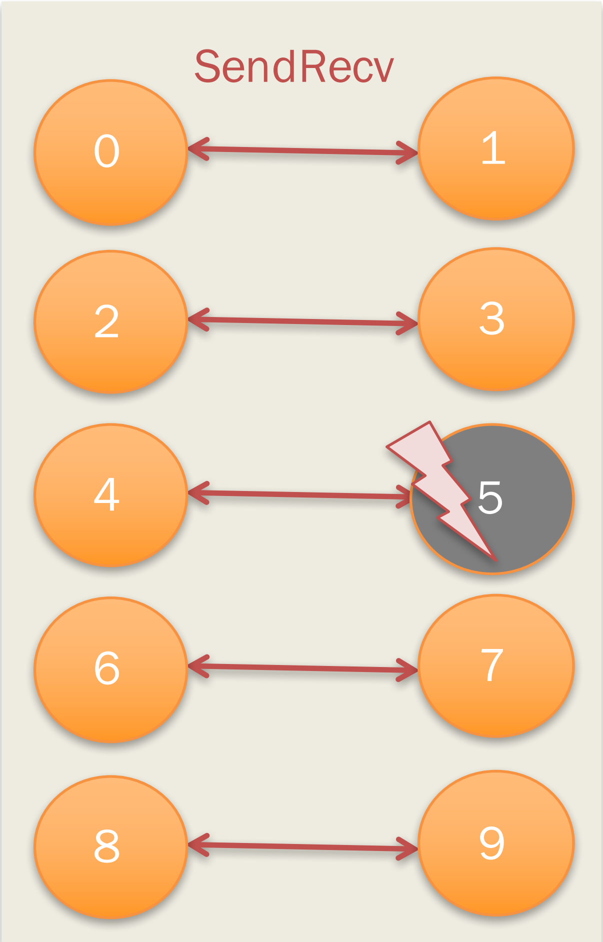
Not all recovery strategies require all of these features, that's why the interface splits notification, propagation and recovery.*

ULFM is not a recovery strategy, but a minimalistic set of building blocks for more complex recovery strategies.

**: some machines are stable, supporting post-failure semantic is optional*

Notification integrating with existing error handling features

- Use existing error handlers
  - MPI_COMM_SET_ERRHANDLER
  - conveniently capture and manage the new survivable error codes
- New error codes to deal with failures
  - MPI_ERROR_PROC_FAILED: report that the operation discovered a newly dead process. Returned from all blocking function, and all completion functions.
  - MPI_ERROR_PROC_FAILED_PENDING: report that a non-blocking MPI_ANY_SOURCE potential sender has been discovered dead.
  - MPI_ERROR_REVOKED: a communicator has been declared improper for further communications. All future communications on this communicator will raise the same error code, with the exception of a handful of recovery functions
- Operations that **can't complete** return **ERR_PROC_FAILED**
  - State of MPI objects unchanged (communicators, etc)
  - Repeating the same operation has the same outcome
- Operations that **can be completed** return **MPI_SUCCESS**
  - Pt-2-pt operations between non failed ranks can continue



Example: only rank4 should report the failure of rank 5

Summary of new functions

- MPI_Comm_failure_ack(comm)
  - Resumes matching for MPI_ANY_SOURCE
- MPI_Comm_failure_get_acked(comm, &group)
  - Returns to the user the group of processes acknowledged to have failed

Notification

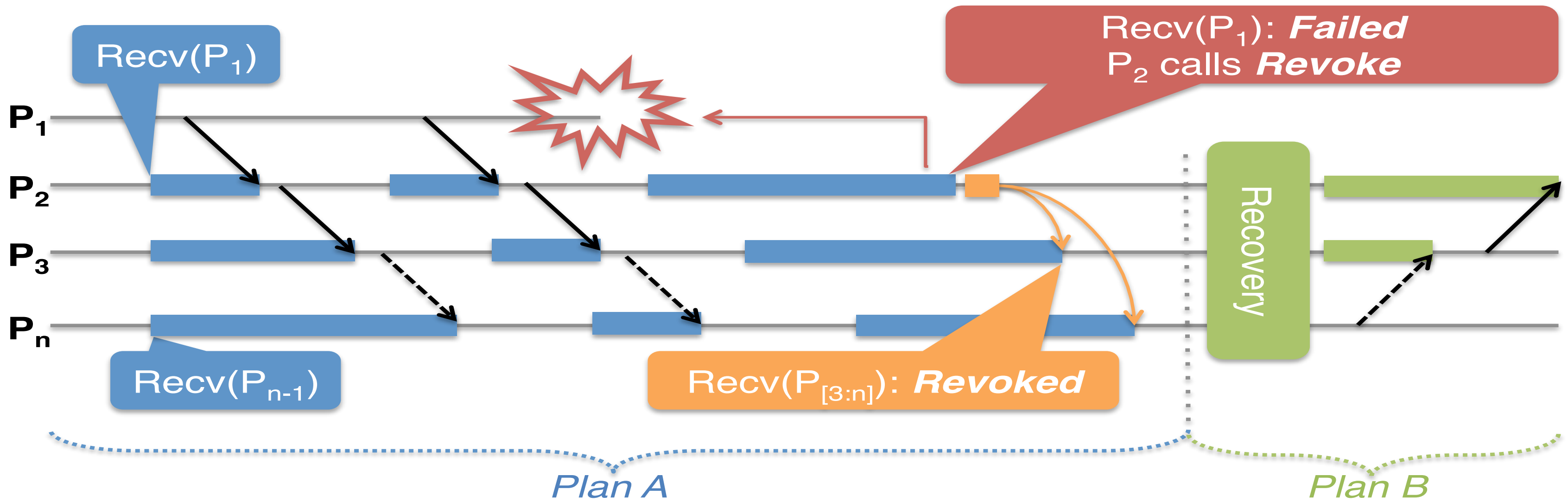
- MPI_Comm_revoke(comm)
  - Non-collective, interrupts all operations on comm (future or active, at all ranks) by raising MPI_ERR_REVOKED

Propagation

- MPI_Comm_shrink(comm, &newcomm)
  - Collective, creates a new communicator without failed processes (identical at all ranks)
- MPI_Comm_agree(comm, &mask)
  - Collective, agrees on the AND value on binary mask, ignoring failed processes (reliable AllReduce), and the return code

Recovery

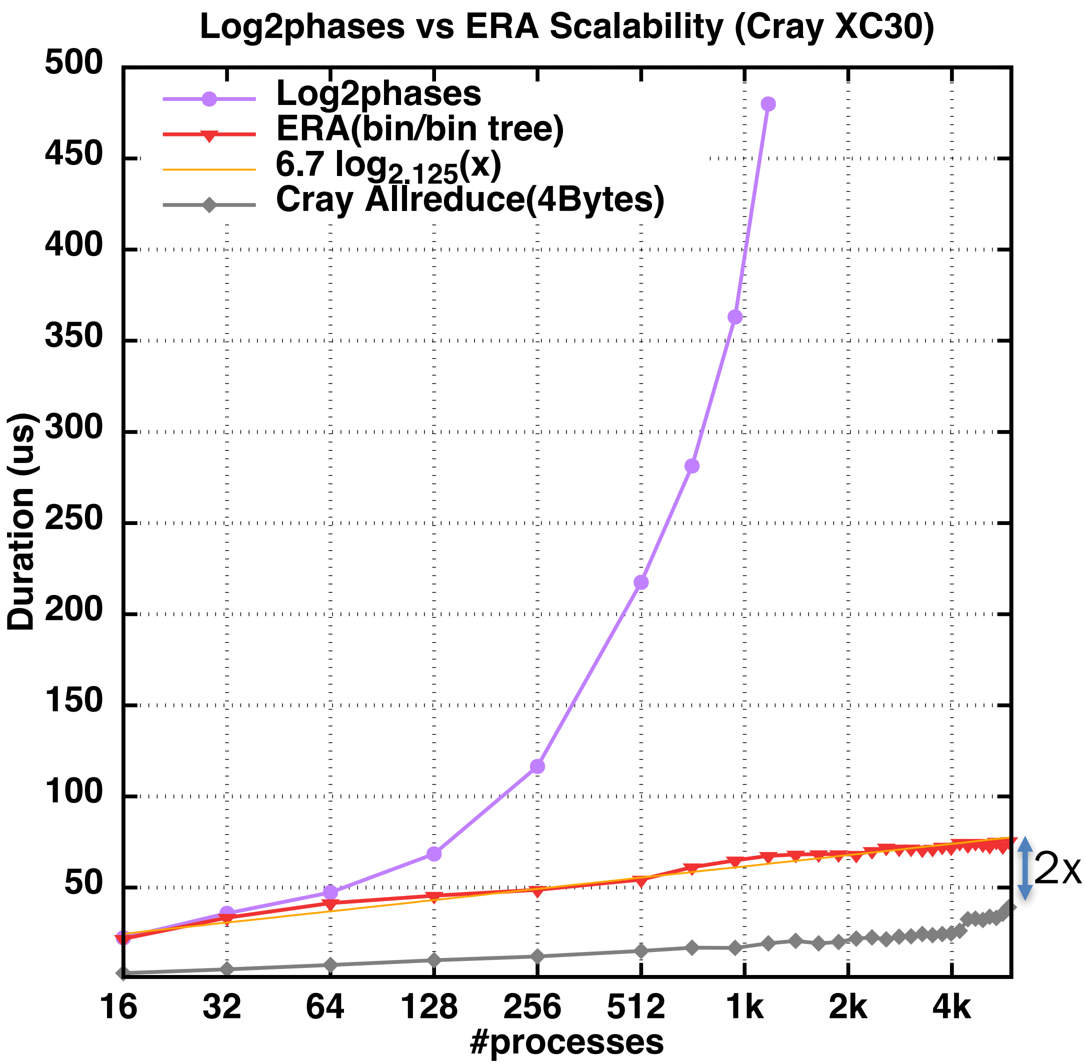
Resolving transitive dependencies




```
proc_failed_err_handler(MPI_Comm comm, int err, ...) {
   if(err == MPI_ERR_PROC_FAILED ||
      err == MPI_ERR_REVOKED ) {
     if(err == MPI_ERR_PROC_FAILED) MPI_Comm_revoke(comm);
     recovery(comm);
   }
}
ft_transitive_deps(void) {
   for(i=0; i<nbrecv; i++) {
      if(myrank>0) MPI_Irecv(buff, count, datatype,
                             myrank-1, tag, comm, &req);
      if(myrank<n) MPI_Send(buff2, count, datatype,
                             myrank+1, tag, comm, &req); }
}
```

- 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 can unlock them with Revoke
  - P3..Pn join P2 in the recovery

Scalable Agreement/Shrink



- Novel Early Returning Agreement algorithm*
- Logarithmic topology & logarithmic computation: scalable
- 2x the Cray AllReduce latency at 6k processors!

* Herault, T., Bouteiller, A., Bosilca, G., Gamell, M., Teranishi, K., Parashar, M., Dongarra, J. "Practical Scalable Consensus for Pseudo-Synchronous Distributed Systems," SuperComputing, Austin, TX, November, 2015

Summary

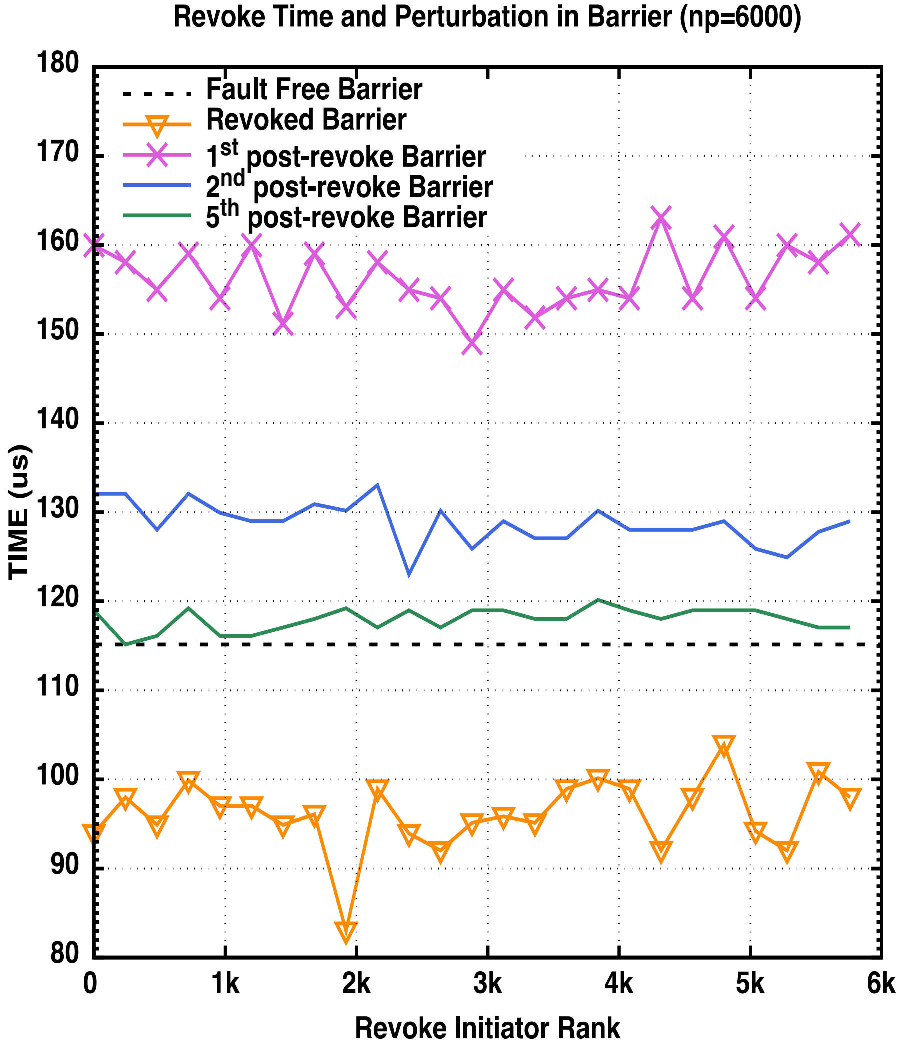
- Issue Ticket (w/ links to PRs)
  - https://github.com/mpi-forum/mpi-issues/issues/20

- Implementation available
  - It is actually fast, now. http://fault-tolerance.org/
- User base has grown quickly
  - Filling a need
  - outlined best practice
  - Varied use cases exercise all capabilities

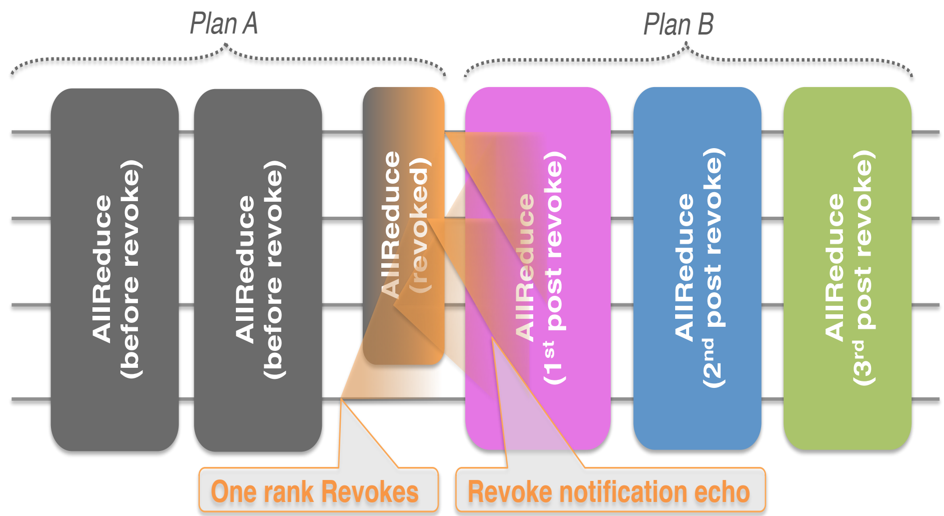
- FAQ
  - *I don't care, my machines are stable*
    - Fair enough, your implementation does not have to support FT (just provide stub interfaces so that FT programs compile and run w/o faults)
  - *I want to do only Checkpoint/Restart*
    - ULFM opens up faster, better C/R than before (that can use NVRAM effectively, etc)
  - *This is too complicated*
    - It doesn't have to be: high level frameworks and code snippets for common tasks are available and help tremendously for quick prototyping

Scalable Resilient Constructs: Revoke

Darter, ugni network, 6000 processes



- BMG* Revoke propagation in less than 100μs
- First post-Revoke collective operation sustains some performance degradation resulting from the network jitter associated with the circulation of revoke tokens
- After the fifth Barrier (approximately 700μs), the Revoke reliable broadcast has completely terminated, therefore leaving the application free from observable jitter.



* Bouteiller, A., Bosilca, G., Dongarra, J.J. "Plan B: Interruption of Ongoing MPI Operations to Support Failure Recovery," In *Proceedings of the 22nd European MPI Users' Group Meeting* (EuroMPI '15). ACM

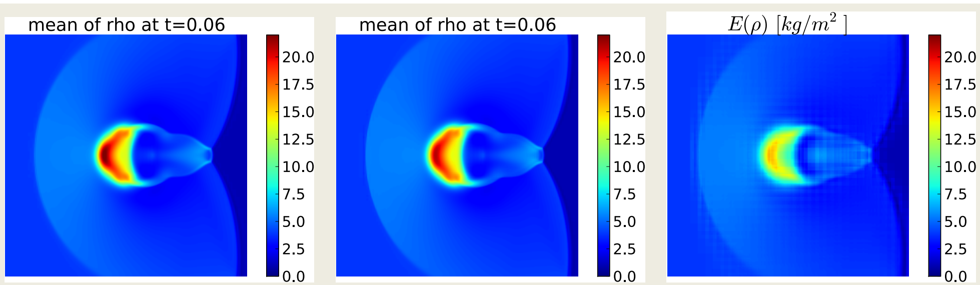
Bibliography of users' activity

These works use ULFM

FRAMEWORKS USING ULFM

LFLR, FENIX, FTLA, Falanx, X10

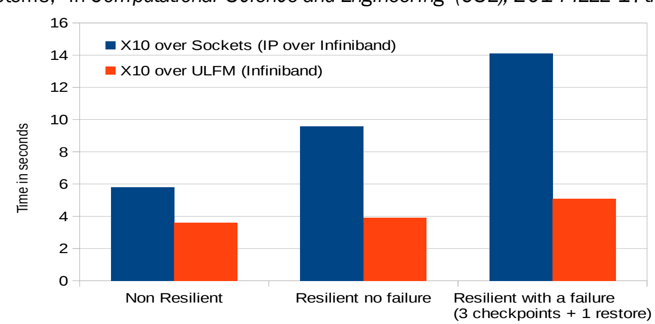
- HAMOUDA, Sara S., MILTHORPE, Josh, STRAZDINS, Peter E., *et al.* A Resilient Framework for Iterative Linear Algebra Applications in X10. In : *16th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC 2015)*. 2015.
- ST PAULI, P. Arbenz et SCHWAB, Ch. Intrinsic fault tolerance of multi level Monte Carlo methods. *ETH Zurich, Computer Science Department, Tech. Rep*, 2012.
- PAULI, Stefan, KOHLER, Manuel, et ARBENZ, Peter. A fault tolerant implementation of Multi-Level Monte Carlo methods. In : *PARCO*. 2013. p. 471-480.
- BLAND, Wesley, DU, Peng, BOUTEILLER, Aurelien, *et al.* Extending the scope of the Checkpoint-on-Failure protocol for forward recovery in standard MPI. *Concurrency and computation: Practice and experience*, 2013, vol. 25, no 17, p. 2381-2393.
- ALI, Md Mortuza, SOUTHERN, James, STRAZDINS, Peter, *et al.* Application Level Fault Recovery: Using Fault-Tolerant Open MPI in a PDE Solver. In : *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*. IEEE, 2014. p. 1169-1178.
- NAUGHTON, Thomas, ENGELMANN, Christian, VALLÉE, Geoffroy, *et al.*Supporting the development of resilient message passing applications using simulation. In : *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*. IEEE, 2014. p. 271-278.
- ENGELMANN, Christian et NAUGHTON, Thomas. Improving the Performance of the Extreme-scale Simulator. In : *Proceedings of the 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2014. p. 198-207.
- TERANISHI, Keita et HEROUX, Michael A. Toward Local Failure Local Recovery Resilience Model using MPI-ULFM. In : *Proceedings of the 21st European MPI Users' Group Meeting*. ACM, 2014. p. 51.
- ALI, Md Mohsin, STRAZDINS, Peter E., HARDING, Brendan, *et al.* A fault-tolerant gyrokinetic plasma application using the sparse grid combination technique. In : *High Performance Computing & Simulation (HPCS), 2015 International Conference on*. IEEE, 2015. p. 499-507.
- VALLÉE, Geoffroy, NAUGHTON, Thomas, BOHM, Swen, *et al.* A runtime environment for supporting research in resilient HPC system software & tools. In : *Computing and Networking (CANDAR), 2013 First International Symposium on*. IEEE, 2013. p. 213-219.
- ZOUNMEVO, Judicael A., KIMPE, Dries, ROSS, Robert, *et al.* Extreme-scale computing services over MPI: Experiences, observations and features proposal for next-generation message passing interface. *International Journal of High Performance Computing Applications*, 2014, vol. 28, no 4, p. 435-449.
- NAUGHTON, Thomas, BÖHM, Swen, ENGELMANN, Christian, *et al.* Using Performance Tools to Support Experiments in HPC Resilience. In : *Euro-Par 2013: Parallel Processing Workshops*. Springer Berlin Heidelberg, 2014. p. 727-736.
- ENGELMANN, Christian et NAUGHTON, Thomas. A NETWORK CONTENTION MODEL FOR THE EXTREME-SCALE SIMULATOR.
- GAMELL, Marc, KATZ, Daniel S., KOLLA, Hemanth, *et al.* Exploring automatic, online failure recovery for scientific applications at extreme scales. In : *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014. p. 895-906.
- XIAOGUANG, Ren, XINHAI, Xu, YUHUA, Tang, *et al.* An Application-Level Synchronous Checkpoint-Recover Method for Parallel CFD Simulation. In : *Computational Science and Engineering (C*
- Judicael A. Zounmevo, Dries Kimpe, Robert Ross, and Ahmad Afsahi. 2013. Using MPI in high-performance computing services. In *Proceedings of the 20th European MPI Users' Group Meeting* (EuroMPI '13). ACM, New York, NY, USA, 43-48.*SE), 2013 IEEE 16th International Conference on*. IEEE, 2013. p. 58-65.
- Jinho Ahn, "N Fault-Tolerant Sender-Based Message Logging for Group Communication-Based Message Passing Systems," in *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on* , vol., no., pp.1296-1301, 19-21 Dec. 2014.



(a) failure-free (b) few failures (c) many failures

Credits: ETH Zurich

Figure 5. Results of the FT-MLMC implementation for three different failure scenarios.



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