# A Resilient Runtime Environment for HPC and Internet Core Router Systems


Ralph Castain <rcastain@cisco.com> Cisco Systems, Inc.
Joshua Hursey <jjhursey@osl.iu.edu> Indiana University
Timothy I. Mattox <timattox@cisco.com> Cisco Systems, Inc.
Chase Cotton <ccotton@udel.edu> University of Delaware
Robert M. Broberg <rbroberg@cisco.com> Cisco Systems, Inc.
Jonathan M. Smith <jms@cis.upenn.edu> University of Pennsylvania


## Abstract


Core routers, with aggregate I/O capabilities now approaching 100 terabits/second, are closely analogous to modern HPC systems (i.e., highly parallel with various types of processor interconnects). Maintaining or improving availability while continuing to scale demands integration of resiliency techniques into the supporting runtime environments (RTEs). Open MPI's Runtime Environment (ORTE) [7] is a modular open source RTE implementation which we have enhanced to provide resilience to both HPC and core router applications. These enhancements include proactive process migration and automatic process recovery services for applications, including unmodified MPI applications. We describe the distributed failure detection, prediction, notification, and recovery components required for resilient operations. During recovery, the fault topology aware remapping of processes on the machine (based on the Fault Group model) reduces the impact of cascading failures on applications. We present preliminary results and plans for future extensions.


## Open RTE's Architecture

The Open MPI Runtime Environment (ORTE) uses Open MPI's Modular Component Architecture (MCA) system [6] that partitions the software into frameworks, components, and modules. Each framework is dedicated to a specific set of related functionality, such as process launch or resource mapping. Each framework supplies a consistent API for invoking the desired functionality, while hiding the complexity of each implementation. Specific implementations are written as components which are then compiled into modules for a given framework. Which set of modules that are used can be selected at configure time, compile time, or run time as appropriate. The MCA design provides flexibility while supporting good software engineering practices, and allows the mixing of open source modules and closed source binary modules.

For the purposes of resilience on HPC and Core Router Systems, we have added or enhanced the following ORTE frameworks and components:

- Sensor Framework (process utilization, temperature, etc.)
- Recovery Service (RecoS) Framework
- Resilient Mapper Component
- ClusterManager Routed Component

## Fault Groups

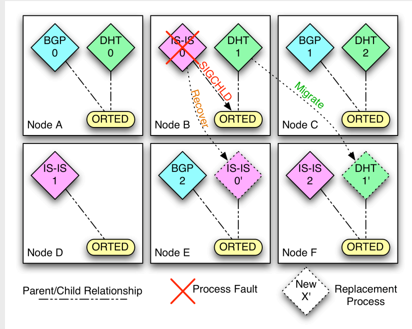
Fault Groups are user or system defined dependent nodes that are at greater risk of spatially/temporally correlated failures (e.g., because they share a common power supply). Upon recovery, processes are placed by the Resilient Mapper in the least affected fault group, thus decreasing the likelihood of a rolling or cascading failure. Fault groups affected by process failure may also be subject to higher levels of fault monitoring.

## Acknowledgments


Project support was provided by Cisco Systems, Inc., the United States Department of Energy, National Nuclear Security Administration's ASCI/PSE program and the Los Alamos Computer Science Institute; a grant from the Lilly Endowment and National Science Foundation grants NSF-0116050, EIA-0202048 and ANI-0330620.


## Fault Detection

There are many kinds of faults that can occur in an HPC or Internet Core Router system. Some faults will effect an individual process while others may disrupt an entire node or group of nodes. The ORTE code uses several different techniques for detecting faults of various kinds. For application processes that crash or otherwise unexpectedly exit, the local ORTE daemon will receive and respond to the POSIX SIGCHLD signal generated by that event. Several sensor modules can be configured to help detect other kinds of faults, such as an application process consuming more memory than allowed, or that a node's temperature is above a configurable threshold. Additional fault detection techniques will be added as the project progresses.




## Fault Prediction

At this point in the project, only preliminary fault prediction code has been written. However, it is clear that detecting temperature trends, supply voltage changes, ECC memory events, disk/SSD media errors, etc. can give some indication that a node is not healthy and applications running on it may soon experience faulty behavior. An example fault prediction module would mark a node as faulty when some number of application processes have crashed on that node. Once this threshold is reached, the ORTE could preemptively migrate any remaining processes to other node(s), as shown in the above figure.

## MPI Level Recovery Policy

The transparent process migration and automatic recovery of unmodified MPI applications is supported by the checkpoint/restart infrastructure currently available in the Open MPI Runtime Environment [4,5].

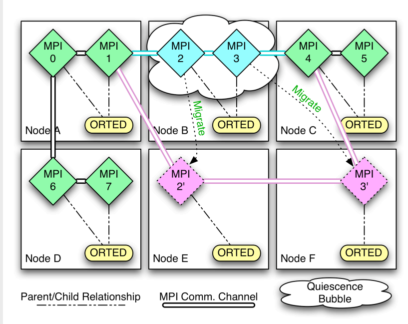
The Recovery Service (RecoS) framework also provides a foundation for implementing MPI application driven fault tolerance techniques. The MPI interface standardization forum has created a Fault Tolerance Working group [3] with the goal of defining the interfaces required for applications to dynamically adapt to process failures in HPC systems. A resilient runtime with flexible recovery policies is critical to supporting such an effort.

## Fault Recovery

A runtime environment should provide a range of recovery policy choices to support higher level libraries and applications. The more flexible the runtime policy the less restricted higher level implementations will be in the types of resilient behavior options provided to the end users.

The Recovery Service (RecoS) framework implements a set of runtime policy choices for how to recover the runtime environment in the event of the loss of one or more processes. The four core policy components are:

- **Abort**: Upon failure, terminate the runtime environment and all dependent processes. This is the default to support MPI implementations in HPC.
- **Ignore**: Upon failure, stabilize the runtime and continue operating.
- **Recover**: Upon failure, automatically recover the lost process(es) from either the beginning of execution or from the last checkpoint, if available.
- **Migrate**: In anticipation of a failure (indicated by a fault prediction service or an end user), transparently move a set of processes from one set of nodes onto another.




## Preliminary Results

Although the project is still in the early stages of development, we have done some preliminary performance tests as we get parts of the system functioning. For a non-MPI application that is responsible for restoring its own state, our system can restart an application process in approximately 6 milliseconds, which includes the time taken by the fault group node selection logic run remotely from the application's node. In comparison, a simple shell script loop takes 3 milliseconds to restart a process locally on the same node, and an ssh based restart takes about 80 milliseconds. These times were measured on an 18 node Linux PC cluster connected with Fast Ethernet.

For MPI task migration, we performed measurements on a Linux PC cluster at Indiana University with an InfiniBand network. A 128 process LAMMPS [1] application (metallic solid benchmark) simulating 13.5 million atoms was run with an aggregate state of 6 GB distributed on 32 nodes. With preliminary non-optimized code, we measured a factor of five reduction in overhead when migrating four processes from a single node to another node, versus checkpointing and restarting the entire 128 process application on 32 nodes.

## Future Extensions

We are actively investigating a variety of future extensions, such as:

- Add more sensor components.
- Add fault prediction algorithms.
- Reduce memory footprint of ORTE daemons.
- Implement support for the application driven fault tolerance standards that come out of the MPI Forum Fault Tolerance Working group [3].
- Add a watchdog API to ORTE so that faults that result in an application hanging instead of exiting can be detected.
- Investigate methods for applications to tell ORTE that one or more of its siblings seem to be faulty, allowing ORTE to kill off rogue processes in an orderly fashion.
- Add fault notifications to more external systems, such as the CIFTS Fault Tolerant Backplane (FTB) [2].

## Conclusions

Recovering from faults in HPC environments becomes critical when the application has real-time demands or the problem size and run time grow large enough to intersect with the combined MTBF of the computing and storage elements used. In either case, restarting such an application is an ineffective response to faults. The required non-stop behavior of a large core router has similar needs for recovery from process and element faults so as to prevent service disruptions which would be caused if a router restart was required.

The software architecture described here, based on the Open MPI Runtime Environment (ORTE), and augmented with support for fault management allows for the development of redundant software applications which provide additional system resilience for both HPC and core router systems.

## References


[1] http://lammps.sandia.gov/

[2] R. Gupta, P. Beckman, H. Park, E. Lusk, P. Hargrove, A. Geist, D. K. Panda, A. Lumsdaine and J. Dongarra. "CIFTS: A Coordinated infrastructure for Fault-Tolerant Systems," International Conference on Parallel Processing (ICPP), 2009.

[3] MPI Forum Fault Tolerance Working Group. http://meetings.mpi-forum.org/mpi3.0_ft.php

[4] J. Hursey, J. M. Squyres, T. I. Mattox, and A. Lumsdaine. "The Design and Implementation of Checkpoint/Restart Process Fault Tolerance for Open MPI." DPDNS 07: 12th IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems. March 2007.

[5] J. Hursey, T. I. Mattox, and A. Lumsdaine. "Interconnect agnostic checkpoint/restart in Open MPI." In HPDC '09: Proceedings of the 18th ACM international symposium on High Performance Distributed Computing, pages 49–58, New York, NY, USA, 2009. ACM.

[6] B. Barrett, J. M. Squyres, A. Lumsdaine, R. L. Graham, and G. Bosilca. "Analysis of the Component Architecture Overhead in Open MPI." In Proceedings, 12th European PVM/MPI Users' Group Meeting, Sorrento, Italy, September 2005.

[7] R. H. Castain, T. S. Woodall, D. J. Daniel, J. M. Squyres, B. Barrett, and G. E. Fagg. "The Open Run-Time Environment (OpenRTE): A Transparent Multi-Cluster Environment for High-Performance Computing." In Proceedings, 12th European PVM/MPI Users' Group Meeting, Sorrento, Italy, September 2005.