A Resilient Runtime Environment for HPC and Internet Core Router Systems

Ralph Castain <rcastain@cisco.com> Cisco Systems, Inc.
Joshua Hursey <jhursey@osl.edu> Indiana University
Timothy I. Mattox <timattox@cisco.com> Cisco Systems, Inc.

Chase Cotton <cccotton@udel.edu> University of Delaware
Robert M. Broberg <broberg@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 terabit/second, are closely analogous to modern HPC systems (i.e., highly parallel with various types of process interconnections). 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 that provides an architecture to address these needs in both HPC and core router systems. This architecture includes the ability to provide resiliency to both the HPC and core router applications. These enhancements include proactive process migration and automatic process recovery services for applications, including unmodified MPI applications. We described the distributed failure detector, prediction, notification, and recovery components required for resilient operations. During recovery, the fault topology aware re-mapping 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 ORTE's Architecture

The Open MPI Runtime Environment (ORTE) uses Open MPI's Modular Component Architecture (MCA) system [8] 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 management. Each module is a minimal component for a specific functionality, while hiding the complexity of each implementation. Specific implementations of these components are then defined into the module for a given framework. Which set of modules that are used can be selected at compile time, runtime, 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 (FScoped Framework)
- Resilient Mapper Component
- ClusterManager Routed Component

Fault Groups

Fault Groups are an architectural design dependent on the node that are at risk of having any one of several types of process interconnections. Understanding these flaws and/or inaccuracies can help in improving the overall resiliency of the system.

Fault Detection

There are many kinds of faults that can occur in an HPC or Internet Core Router system. Some faults will affect 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 PDSX SIGCHLD signal generated by that event. Several sensor modules are 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 Recovery

A runtime environment should provide a range of recovery policies to choose from higher levels (i.e., applications and/or users). 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 (ResoS) framework implements a set of runtime policies for how to recover the runtime environment, in case 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 processes from either the beginning of execution or from the last checkpoint, if available.
- MPI: in anticipation of a failure (indicated by a fault detection service or an end user), transparently move a set of processes from one set of nodes onto another.

Fault Groups

Fault Groups are a user-defined dependent on the node that are at risk of having any one of several types of process interconnections. Understanding these flaws and/or inaccuracies can help in improving the overall resiliency of the system.

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 errors, disk/ISO media errors, etc. can give some indication of problems occurring on it or that the operator 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/rollback infrastructure currently available in the Open MPI Runtime Environment [4,5]. The ORTE's Recovery Service (ResoS) 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 [9] 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 environment.

Acknowledgments

Project support was provided by Cisco Systems, Inc., the United States Department of Energy (DOE)'s Argonne Leadership Computing Facility (ALCF) program and the Los Alamos Computer Science Institute, a grant from the Lilly Endowment and National Science Foundation grants NSF-0119300, EIA-0202048 and ANI-0330620.

References


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 tolerate standards that come out of the MPI Forum Fault Tolerance Working group [5].
- Add a facility to ORTE to let an application mark an application process that is hanged instead of offering it as can be detected.
- Investigate methods for applications to tell ORTE that one or more of its sibs 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 Baseline (FTB) [5].

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 interest the combined MRF of the computing and memory elements used. For either reason, restarting such an application is an inefﬁcient response to faults. The required non-stop behavior of a large core router has similar needs for recovery from process and element faults as so as to prevent service degradations which would cause a nodal router expected service. 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 resilient software applications which provide additional system resilience for both HPC and core router systems.