Parallel Processing in ROSA II

Sara Siegal and Glenn Schrader
MIT Lincoln Laboratory
ssiegal@ll.mit.edu, gschrad@ll.mit.edu

Introduction

ROSA II (Realtime Open Systems Architecture II) is a component architecture designed for implementing real-time sensor systems. The sensors that use ROSA II include radar and optical sensors. The underlying concepts of ROSA II are generally applicable to both types of sensor systems. The architecture allows a given system to be either distributed across a set of processors or run on a single processor, depending on the needs of that system. While radar signal processing is often parallelized, for our application we needed ROSA II components and PVL to coexist and communicate in the same system. Further, the ROSA II framework is based on the Unix operating system, while our signal processing is based on Mercury's MCOE operating environment. No common communications middleware existed for both these operating systems. Thus we needed to combine PVL with our ROSA II application and RTCL.

We will discuss the challenges faced while combining these libraries, various aspects of system performance, and the lessons learned during implementation.

ROSA II Framework

The ROSA II framework is a common, open sensor infrastructure that enables a wide variety of DoD future capabilities. It allows a developer to "compose" a sensor by selecting and adapting reusable components, both software and hardware, and is designed to support a wide range of device applications. It provides a common open framework for sensor and device back end computing; a solid basis for rapid prototyping as well as development based on modern, open, component-based architecture. It is platform independent, flexible, scalable, and compatible with net-centric approaches.

ROSA II Component Architecture

All ROSA II components share a generic base class, "`ROSA II_Component`", which contains common interfaces (such as the control Parameters subscriber, and status message publisher), common methods (such as state machine logic), component attributes (such as a component's name, process ID, and process state), and environment attributes (such as a the host's computer class, host ID, processor type, OS name and version). Figure 1 shows a typical component diagram. Each component implements the ROSA II state machine as shown in Figure 2.

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and not necessarily endorsed by the United States Government.

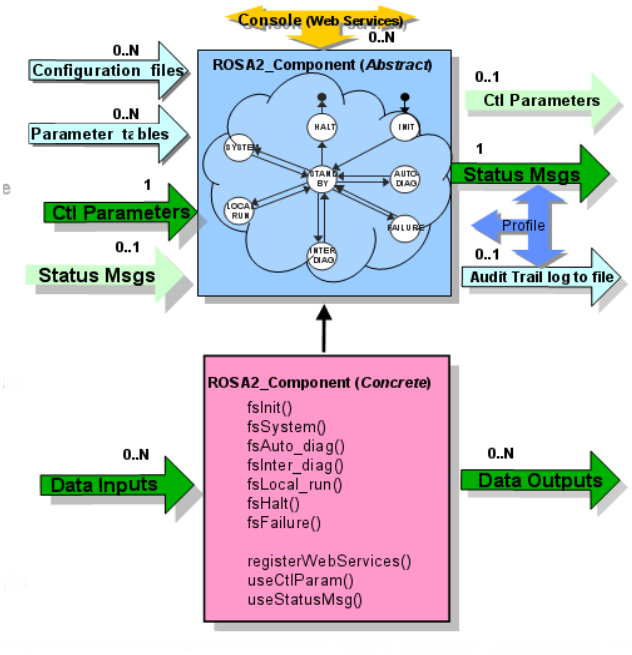

Figure 1: ROSA II Component Diagram[1]

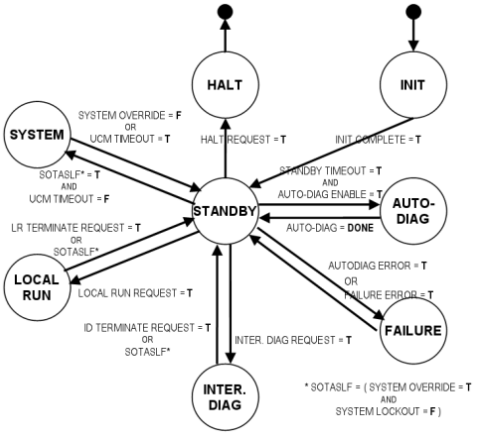

Figure 2: ROSA II Component State Transition Diagram[2]

RTCL

RTCL (Radar Thin Communications Layer) is a communications middleware with publish/subscribe semantics for use in radar applications. It is built on top of other communications middlewares as shown in Figure 3 and provides a consistent communications abstraction to software engineers while allowing them to use different underlying middlewares as needed.

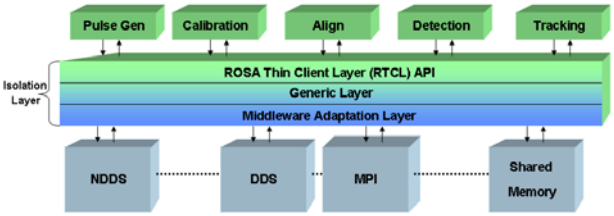

Figure 3: RTCL layered architecture

Hardware

We used a heterogeneous development system, comprised of a 6U VME/RACE++ Mercury chassis containing a 6U 300 Mhz Sparc CPU host running Solaris 5.8, and four 6U dual-port Mercury motherboards, each containing a dual PPC G4 daughterboard.

The Challenge

The ROSA II framework is based on the Unix operating system, while our signal processing is based on Mercury's MCOE operating environment. Some middlewares available for Unix platforms include RTI-DDS and shared memory. The middleware available for MCOE is MPI. No common communications middleware existed for both these operating systems. Thus we needed to combine PVL with our ROSA II application and RTCL.

PVL

PVL (Parallel Vector Library) is an object-oriented software library for parallel signal processing implemented in C++. It allows signal processing algorithms to be written with high-level mathematical constructs that are independent of the underlying parallel mapping. Programs written using PVL can be ported to a wide range of parallel computers without sacrificing performance. Furthermore, the mapping concepts in PVL provide the infrastructure for enabling new capabilities such as fault tolerance and self-optimization.[3]

Solution

We implemented a new middleware adaptation layer, McBridge, for the Mercury CEs (compute elements) running MCOE on the Mercury motherboards. This allowed us to use RTCL to communicate among the CEs. The Solaris host was responsible for booting the CEs and loading our application onto them. Each CE ran a copy of the application in parallel (and independent of the other CEs). The CEs used RTCL and McBridge to communicate with the outside world, and Mercury conduits to communicate with each other. We used PVL to distribute the workload among the CEs.

The combination of these technologies (ROSA II application, PVL, and RTCL using the McBridge middleware), enabled us to develop a fully parallelized system within the ROSA II framework.

References

[1] P. Jurgensen, “ROSA II Generic Component Base Class”, 2008

[2] G. Schrader, H. Nguyen, M. Eskowisz, “ROSA II Airborne Radar PDR”, 2007

[3] J. Kepner, J. Lebak, “Software Technologies for High-Performance Parallel Signal Processing”, 2003