HARNESS: Heterogeneous Adaptable Reconfigurable NEtworked SystemS *

Jack Dongarra – Oak Ridge National Laboratory and
University of Tennessee, Knoxville
Graham Fagg – University of Tennessee, Knoxville
Al Geist – Oak Ridge National Laboratory
James Arthur Kohl – Oak Ridge National Laboratory
Philip M. Papadopoulos – Oak Ridge National Laboratory †
Stephen L. Scott – Oak Ridge National Laboratory
Vaidy Sunderam – Emory University
Mauro Magliardi – Emory University

Abstract

This poster presentation describes our vision, goals and plans for HARNESS, a distributed, reconfigurable and heterogeneous computing environment that supports dynamically adaptable parallel applications. HARNESS builds on the core concept of the personal virtual machine as an abstraction for distributed parallel programming, but fundamentally extends this idea, greatly enhancing dynamic capabilities. HARNESS is being designed to embrace dynamics at every level through a pluggable model that allows multiple distributed virtual machines (DVMs) to merge, split and interact with each other. It provides mechanisms for new and legacy applications to collaborate with each other using the HARNESS infrastructure, and defines and implements new plug-in interfaces and modules so that applications can dynamically customize their virtual environment.

HARNESS fits well within the larger picture of computational grids as a dynamic mechanism to hide the heterogeneity and complexity of the nationally distributed infrastructure. HARNESS DVMs allow programmers and users to construct personal subsets of an existing computational grid and treat them as unified network computers, providing a familiar and comfortable environment that provides easy-to-understand scoping. Similarly, a particular site could use HARNESS to construct a virtual machine that is presented and utilized as a single resource for scheduling within the grid.

Our research focuses on understanding and developing three key capabilities within the framework of a heterogeneous computing environment: 1) Techniques and methods for creating an environment where multiple distributed virtual machines can collaborate, merge or split; 2) Specification and design of plug-in interfaces to allow dynamic extensions to services and functionality within a distributed virtual machine; and 3) Methodologies for distinct parallel applications to discover each other, dynamically attach, collaborate, and cleanly detach.

1 Overview

Current software systems, like PVM and MPI, provide a utilitarian model for personally managing a collection of computing resources into a single virtual machine (VM). Users are comfortable with the virtual machine abstraction and the encapsulation of resources that it provides. However, current VM implementations have limited degrees of support for handling faults and failures, utilizing heterogeneous and dynamically changing communication substrates, and enabling migrating or cooperative applications. Virtual machines have the benefit of providing information scoping that is independent of an application and insulates a program from the entire universe. We see the VM approach as a practical mechanism for a wide variety of, but certainly not all, applications that can utilize an Internet or computational grid environment. For example, The VM approach is not appropriate for anonymous communication services like web servers, because the VM builds a wall where none is needed. In the space of dynamically adaptable (migrating, cooperating, or fault-tolerant) applications, monitoring

* Research supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy, under contract DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation

† Corresponding author: Philip M. Papadopoulos, (423) 241-3972, Oak Ridge National Laboratory, 1 Bethel Valley Rd. Oak Ridge, TN 37831-6367, phil@msr.epm.ornl.gov

agents must exist to watch for events that affect a particular application. VMs are particularly compelling in this application domain because they allow one to design several types of monitoring agents. These agents are part of the virtual construction and can be used by all applications as a guaranteed service. The distributed virtual machine (DVM) scoping of resources allows these agents to narrow their focus (and thereby reduce complexity) from the "entire" Internet to a manageable subset.

While the encapsulation provided by a virtual machine is very compelling, it becomes clear that sometimes entities from two different virtual machines would like to communicate across VM boundaries and share resources. Processes within one virtual machine cannot communicate using messages to tasks within another virtual machine without some sort of sharing. While, TCP sockets allow physical machines to inter-communicate, we believe a more powerful notion is to enable merging of distinct VMs into a single VM (plug together) and, just as easily, to split a single VM into multiple independent VMs (pull apart). HARNESS will design and implement mechanisms to accomplish these goals.

Our research agenda is to explore the dynamic needs and capabilities of the extended virtual machine environment. HARNESS is not about reinventing message passing APIs (initially, both PVM and MPI semantics will be supported), but rather how to construct a virtual environment that can dynamically change (almost) anything at runtime. To this end, our focus is on

- Extension of the present network and cluster computing model to include multiple distributed virtual machines, each with multiple users, where the virtual machines can dynamically merge, split and in general collaborate with each other. Today's cluster computing frameworks typically involve the static configuration of a single distributed virtual machine on which a user executes multiple applications.

- Development of a generalized plug-in paradigm for distributed virtual machines that allows users or applications to dynamically customize, adapt and extend the distributed computing environment's features to match their needs. This is analogous to the plug-in interfaces in use today for serial applications (e.g. web browsers, graphics editors, multimedia players), but extended to distributed systems.

- Creation of a collaborative computing framework that allows multiple parallel applications running in a heterogeneous distributed environment to dynamically attach, interact and detach from one another. One existing example of such capabilities is the CUMULVS [8] system, which allows collaborative computational steering. We will provide a framework that supports a wide class of parallel tools and applications that would benefit from being able to attach and detach from each other. The framework will integrate discovery services with an API that defines baseline attachment and detachment protocols between heterogeneous, distributed applications.

2 Where HARNESS Fits and Why Pluggability?

Although distributed computing frameworks continue to be expanded and improved, the growing need in terms of functionality, paradigms, and performance quite simply are increasing faster than the pace of these improvements. By developing a distributed computing framework that supports plug-ins, it will be possible to extend or modify the capabilities available to parallel applications without requiring immediate changes in the standards, or endless iterations of ever-larger software packages. For example, a distributed virtual machine could plug in modules for distributed shared memory programming support along with message passing support, allowing two legacy applications, each written in their own paradigm, to interoperate in the same DVM. This type of plug-in enables efficient adaptation to many new capabilities, such as new advanced communication protocols or networks, programming models, and encryption methods, without the need for extensive reengineering of the computing framework. To derive full benefit, HARNESS plug-ins will be dynamic in nature, or "hot-pluggable." Certain features or functionality will plug in temporarily, only while needed by an application, and then unplug to free up system resources. Distributed applications no longer will need to adjust to fit the capabilities of the distributed computing environment. Instead, the environment can be dynamically adapted to the changing needs of the application.

Beyond just plugging small functional components into the distributed environment, this concept of pluggability can be extended to encompass the merging and splitting of DVMs, as well as the attachment of tools and applications in collaborative scenarios. Analysis tools can plug into applications on-the-fly to collect information or steer computations. Peer applications will be able to "dock" with each other to exchange intermediate results or even active objects (e.g., Java bytecode) thereby facilitating collaborative computing at the software level.

The poster presentation will highlight these issues and our plans for building such an environment.