

Progress towards Petascale Virtual Machines*

Al Geist

Oak Ridge National Laboratory,

PO Box 2008,

Oak Ridge, TN 37831-6367

gst@ornl.gov

<http://www.csm.ornl.gov/geist>

Abstract. Petascale Virtual Machines (PVM) continues to be a popular software package both for creating personal grids and for building adaptable, fault tolerant applications. We will illustrate this by describing a computational biology environment built on top of PVM that is used by researchers around the world. We will then describe or recent progress in building an even more adaptable distributed virtual machine package called Harness. The Harness project includes research on a scalable, self-adapting core called H2O, and research on fault tolerant Message Passing Interface (MPI). The H2O core can be configured to support distributed web computing like SETI@home, parallel virtual machines like PVM, and OGSA compliant grid environments. The Harness software is being designed to scale to petascale virtual machines. We will describe work at Oak Ridge National Laboratory (ORNL) on developing algorithms for such petascale virtual machines and the development of a simulator to test these algorithms on simulated 100,000 processor systems.

1 Background

In 21st century scientific research programs, computation is becoming a third arm in scientific discovery along with theory and experiment. Computing was the key to unlocking the human genome and now is being used to decode the genomes of hundreds of other organisms. The next stage of genomics is encompassed by the U.S. Department of Energy's (DOE) Genomes to Life project [1] where all the molecular machines in a microbe and their regulation will be determined. This 10 year effort will involve enormous data analysis and computational challenges.

Nanotechnology is another area where major advances are being made through advanced computer simulations. Over the last decade computational materials science has improved by three orders of magnitude through increased computing power and another eight orders of magnitude through advanced algorithms. With such improvements, computing has become a key method to push forward the nanotechnology frontier.

PVM and MPI are the key software packages used by scientific applications today and will continue to be the programming paradigm of choice for the foreseeable future. But as computers reach the 100 TF and upwards to a petaflop in the next 5 years, fault tolerance and adaptability may become as important as latency and bandwidth to both computational biology and nanotechnology applications. In anticipation of these needs, the developers of PVM started a new project, called Harness [2] that has all the features of PVM and MPI, but also has peer-to-peer distributed control, self adaptability and the ability to survive multiple points of failure. These features are important to be able to scale to petascale virtual machines.

*The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

This paper describes the Genomes to Life project and uses it to illustrate PVM use today. Next we describe the Harness project and the features it has to help scale to petascale virtual machines. The paper discusses H2O, which is the self adapting core of Harness. Finally, we describe the latest developments in research going on at ORNL to develop multigrid and finite element algorithms that can self-adapt to failures on a petascale virtual machine.

2 Using PVM in Genomes to Life

DOE initiated and ran the Human Genome project, which decoded all 3 billion bases in the human genome. That project finished this year and DOE has embarked on an even more ambitious computational biology program called Genomes to Life. The plan for the 10-year program is to use DNA sequences from microbes and higher organisms, including humans, as starting points for systematically tackling questions about the essential processes of living systems. Advanced technological and computational resources will help to identify and understand the underlying mechanisms that enable organisms to develop, and survive under myriad environmental conditions. This approach ultimately will foster an integrated and predictive understanding of biological systems and offer insights into how both microbial and human cells respond to environmental changes. The applications of this level of knowledge will be extraordinary and will help DOE fulfill its broad missions in energy research, environmental remediation, and the protection of human health.

PVM plays a role in the Genomes to Life program through the Genome Integrated Supercomputer Toolkit (GIST). GIST is a middleware layer used to create the Genome Channel [3]. The Genome Channel is a computational biology workbench that allows biologists to transparently run a wide range of genome analysis and comparison studies on supercomputers at ORNL. When a request comes in to the Genome Channel, PVM is used to track the request, create a parallel virtual machine combining database servers, Linux clusters, and supercomputer nodes tailored to the nature of the request, and then spawning the appropriate analysis code on the virtual machine.

A key feature of PVM that is exploited in Genome Channel is fault tolerance. The creators of GIST require that their system be available 24/7 and that analyses that are running when a failure occurs are reconfigured around the problem and automatically restarted. PVM's dynamic programming model is ideal for this use. The Genome Channel has been cited in "Science" and used by thousands of researchers from around the world.

This is just one of many illustrations of PVM use today.

3 Harness - the next generation of PVM

Harness is the code name for the next generation heterogeneous distributed computing package being developed by the PVM team at ORNL, the University of Tennessee (UT), and Emory University. Harness is being designed to address the future needs of PVM and MPI application developers. Harness version 1.9 was released last year and Harness 2 is scheduled to be released in November 2003.

The basic idea behind Harness is to allow users to dynamically customize, adapt, and extend a virtual machine's features to more closely match the needs of their application and to optimize the virtual machine for the underlying computer resources. For example, taking advantage of a high-speed communication channel. At the same time the software is being designed to be able to scale to petascale virtual machines through the use of distributed control and minimized global state.

As part of the Harness project, UT is developing a fault tolerant MPI called FT-MPI. This package which includes all the MPI 1.2 functions allows applications to be developed that can dynamically

recover from task failures within an MPI job. FT-MPI supplies the means to notify the application that it has had a failure and needs to recover. In the simplest case recovery involves killing the application and restarting it from the last checkpoint. FT-MPI also allows an application to “run through” failure by recreating a new MPI COMM WORLD communicator on the fly and continue with the computation. MPI COMM SPAWN is included in FT-MPI to provide a means to replace failed tasks where appropriate. More fault tolerant MPI implementations and the use of them in super scalable algorithms is expected to increase significantly in the next decade.

ORNL's part in the Harness project is to develop a distributed peer-to-peer control system that can survive and adapt to multiple points of failure. The development of such a control system is necessary to support petascale virtual machines. Fast updates of state and the ability to recreate state lost to failures are important attributes. Global state is minimized, replicated, and distributed across the virtual machine. The Harness peer-to-peer control system is felt to be much more scalable than the client-server system used in PVM. The control system is presently in the process of being incorporated into Harness 2.

Emory has taken the lead in the architectural design of Harness and development of the H2O core [4], which will be described in the next section. In developing Harness 1.9, it was realized that the distributed computing architecture could be generalized by creating a lower software level called H2O. A petascale virtual machine could then be built by adding distributed virtual machine plug-ins to the H2O pluggable daemon.

4 H2O - the Harness core

In Harness, the role now played by the PVM daemon is replaced with an H2O daemon. The H2O daemon starts out knowing little more than how to load and unload plug-ins from itself. It doesn't know anything about other daemons or the concept of a virtual machine. These are capabilities that the plug-ins provides. For example, a high-speed communication plug-in and a PVM plug-in can be loaded into the H2O daemon and it would have all the capabilities of a PVM daemon. In fact in the Harness 1.9 release, the PVM plug-in can run legacy PVM applications without recompiling. Load in an MPI plug-in and Harness is able to run MPI applications. With Harness, a distributed virtual machine is no longer restricted to a given set of capabilities - new scalable features can be incorporated at any time.

But H2O is a general framework [4]. In the H2O model, a dynamic collection of providers make their networked resources available to a defined set of clients. With this general model, H2O is able to provide a framework for several different forms of distributed computing today.

To provide a PVM framework, the provider specifies only himself as a client, and he loads a PVM plug-in.

To provide a Grid framework, the provider specifies that any clients with a grid certificate can run on his resource. In addition, he would most likely load a few OGSA plug-ins to provide a Grid interface for his grid clients.

To provide a web computing framework like SETI@HOME, the provider specifies one external client who can use his machine while it is idle. This client would then load and run a distributed application on thousands of such providers across the Internet.

Other frameworks can be supported by the H2O model. For example, a provider may sell use of his resource to one client. This client may in turn provide a popular software application (a game, or scientific program) that they then sell use of to other clients. Thus users of the resource may be a superset of the provider's client list. Here is a hypothetical scenario. A client comes up with the next great "Google" software that instead of running on 6,000 node clusters, runs on idle Internet desktops. The client goes out and convinces many providers to supply their idle

resources to him. This client in turn sets up his software and provides this service to many other people. These people, not the original client, are the ones who execute the software on the provider's resources.

While H2O is very flexible, our main focus in this paper is in using it to create Harness petascale virtual machines. To build such large systems distributed control and distributed state plug-ins are loaded into the daemon. These along with an MPI or PVM plug-in provide the basis for the next generation of distributed computing environment.

5 Algorithms for Petascale Virtual Machines

Exploiting PVMs is going to take more than the Harness software. Fundamentally new approaches to application development will be required to allow programs to run across petascale systems with as many as 100,000 processors. With such large-scale systems the mean time to failure is less than the run time of most scientific jobs and at the same time checkpoint-restart is not a viable method to achieve fault tolerance due to IO requirements. Thus the algorithms in the next generation scientific applications will have to be both super scalable and fault tolerant.

ORNL has developed a meshless finite difference algorithm that is naturally fault tolerant. Parallel algorithms with "natural fault tolerance" have the property that they get the right answer even if one or more of the tasks involved in the calculations fail without warning or notification to the other tasks [5]. On a simulated 100,000 processor system the finite element algorithm gets the correct answer even if up to a hundred random parallel tasks are killed during the solution. Using a hierarchical implementation of the fault tolerant finite element algorithm, a "gridless multigrid" algorithm has now being developed and its performance and fault tolerance results will be reported at the conference.

Global information is often needed in science applications to determine convergence, minimum energy, and other optimization values. ORNL has been able to create a naturally fault tolerant global maximum algorithm. Despite failures, all tasks (that are still alive) know the largest value when the algorithm finishes. The global max algorithm runs in logarithmic time, but unlike a typical binary tree, it is robust no matter which tasks die during the solution.

These examples are just the beginning of a new area of naturally fault tolerant algorithm development. Much research is still required to develop algorithms that can run effectively on petascale virtual machines.

6 Conclusion

Sciences such as genomics and nanotechnology are being driven by computation and PVM is helping to make that happen today. But tomorrow's discoveries are predicted to require computing capabilities 1,000 times greater than today's computers. Utilizing such Petascale Virtual Machines will require new approaches to developing application software that incorporate fault tolerance and adaptability. Our ongoing research in fault tolerant MPI and Harness are just beginning to address some of these needs. The new finite difference, global max, and multigrid algorithm results are encouraging but much research remains in order to understand the scalability and fault tolerance required for 100,000 processor systems.

7 References

1. G. Heffelfinger, et al, "Carbon Sequestration in Synechococcus Sp.: From Molecular Machines to Hierarchical Modeling," OMICS Journal of Integrative Biology. November 2002 (www.genomes-to-life.org).
2. G. A. Geist, et al, "Harness," (www.csm.ornl.gov/harness).
3. M. Land, et al, "Genome Channel," (<http://compbio.ornl.gov/channel>).

4. D. Kurzyniec, et al, "Towards Self-Organizing Distributed Computing Frameworks: The H2O Approach," International Journal of High Performance Computing (2003).
5. G. A. Geist and C. Engelmann, "Development of Naturally Fault Tolerant Algorithms for Computing on 100,000 Processors," Parallel Computing (submitted) 2003. (PDF at www.csm.ornl.gov/geist/Lyon2002-geist.pdf)