Developing Interactive PVM-based Parallel Programs on Distributed Computing Systems within AVS Framework

Gang Cheng, Geoffrey C. Fox, Kim Mills, Marek Podgorny

gcheng@npac.syr.edu, (315)443-2083

Northeast Parallel Architectures Center
Syracuse University, Syracuse, NY 13244

Abstract

We discuss techniques in developing interactive Parallel Virtual Machine(PVM) concurrent programs on distributed systems under AVS framework. Using a stock option price modeling application as a case study, we demonstrate a simple, effective and modular approach to coupling network-based concurrent modules into an interactive remote visualization environment. A prototype simulation on-demand system is developed, in which parallel option pricing models locally implemented on two distributed systems, an Ethernet-connected IBM SP1 and a FDDI-based GIGswitch-connected DEC Alpha farm, are coupled with an interactive graphical user interface over a ATM-based wide area network. This integrated networking/visualization framework allows one to use a high level system software approach to combine AVS visualization modules with remote concurrent PVM tasks in fine-grain parallelism.

1 Introduction

Parallel computing environment of the future will likely be based on a heterogeneous networked computing system, and be required to solve applications with various requirements in scientific computation, communication, programming models, memory hierarchies, I/O, database management and visualization. Network-based concurrent computing and advanced data visualization are two important components in real-world applications of high performance computing and communication(HPCC), especially for industrial problems requiring real-time simulation/modeling, information processing and retrieval. System integration is expected to play a critical role in the development and applications of distributed and parallel software environment.

During the past two years, Northeast Parallel Architectures Center (NPAC) at Syracuse University has actively engaged in the research of software integration methodologies and environments on massively parallel computers and heterogeneous distributed systems. We use AVS as a system integrating tool in several research and application projects which require interactive data visualization and distributed and parallel processing. We have successfully developed real-time parallel numerical simulation/modeling applications in AVS on several parallel systems, including Thinking Machines Connection Machine 2 and 5, Maspar's DECmpp-12000 and networked heterogeneous workstation clusters[3, 4, 5]. We use AVS to combine sequential, data parallel, and message passing modules in a heterogenous environment[6]. Our latest work is on IBM SP1, IBM's newest parallel distributed-memory computer designed using powerful RISC technology combined with a high-speed switch, and DEC Alpha Farm, Digital's latest approach to distributed computing.

NPAC is initiating an new InfoMall[8] project which consists of a set of interactive distributed high performance information systems, collectively called InfoVision (INFOmation, Video, Imagery, and SImulation ON-demand). Targeted at the state-of-the-art demonstrations on NYNET, a regional ATM-based WAN network, and the utilization of supercomputing resource at NPAC, InfoVision [7] focuses on real-time information production, analysis, access and integration on distributed and parallel computing systems – allow geographically located regional end-user to interactively access multimedia information and to remotely perform parallel simulation demonstrations on supercomputers over a high-speed WAN network NYNET.

As a core enabling technology in a simulation on-demand application, interactive software integration environment plays a central role in coupling parallel and distributed computing, interactive remote user-interface, advanced data visualization and transparent networking to support the real-time simulation over a heterogeneous computing and networking environment.

In this paper, we report our initial work in using AVS as a parallel software integration tool to develop interactive parallel programs on distributed systems to allow real-time control of modeling system. Using a stock option price modeling application as a case study, and a portable message passing interface PVM(Parallel Virtual Machine) as the concurrent programming interface, we discuss parallel programming and system issues in AVS for a simulation on-demand system implemented on two distributed systems, an Ethernet-connected IBM SP1 and a GIGswitch-based FDDI-connected DEC Alpha Farm, and a heterogeneous networking environment.

Within AVS's dataflow-based visualization model, we demonstrate a simple, effective and modular approach to coupling network-based concurrent modules into an interactive remote visualization environment. This integrated networking/visualization framework allows one to use a high level system software approach to combine AVS visualization modules with remote concurrent PVM tasks in fine-grain parallelism.

2 Parallel Virtual Machine(PVM) and Application Visualization System (AVS)

PVM[13, 9] is a public-domain software package that supports utilization of heterogeneous networks of parallel and serial computers as a single computational resource. PVM provides a general-purpose message-passing interface to TCP/IP networking protocol, portable across multiple platforms.

PVM consists of two parts: a daemon process on each host machine of the virtual machine, and a user library that contains primitive routines for initiating processes on other hosts, for communication and data transfer between processes and for configuration of the virtual machine. PVM assumes that each host in a virtual machine is directly connected to each other host via IP. The daemons communicate with one another using UDP sockets. TCP sockets are used between a daemon and the local PVM tasks, and also directly between tasks on the same host or on different hosts if necessary. The message-passing system is implemented on top of the UDP and TCP sockets.

The PVM software package has no build-in visualization capability. To develop a graphical user interface(GUI) for a networking application program written in PVM, one has to link to other graphics packages such as Xlib or X Toolkit. Most previous work with developing a GUI component in a PVM system was focused on performance monitoring, graphical debugging, and visual development tools for aiding the programmer in the tasks of designing, compiling, executing, debugging, and analyzing a parallel program[2, 10].

AVS[1, 14], a widely available commercial visualization environment, is based on a dataflow model for scientific data visualization and process control. While its main capacity focuses on data visualization and 3D graphics, the AVS software environment also incorporates a visual programming interface, modular process management and transparent networking into a comprehensive visualization application and development environment. Our previous experience using AVS in parallel programming and applications[3, 4, 5, 6] suggests that AVS can be an excellent software integration tool to couple various components together for a large application requiring heterogeneous computing services.

The networking capability in the AVS system is restricted to a data-flow paradigm. Message passing must be carried out at module level, and thus is limited to a coarse-grained fashion. Mechanisms in AVS to ensure synchronous execution of coroutine/subroutine modules are insufficient to facilitate a general purpose networking application with complex pattern of message passing in concurrent programs.

By integrating PVM software in the AVS framework, the resulting system offers an equally sophisticated networking and visualization functionality, with integrated networking and visualization programming interfaces.

3 Integration of PVM programs into the AVS environment

There can be two basic approaches to incorporate PVM programs into an AVS framework. They are based on the ways in which an AVS kernel interacts PVM daemons:

1. Our first approach embeds PVM programming model at lower level into AVS dataflow communications at the top inter-module level. As shown in Figure 1, there is a parallelism hierarchy in the system at two distinct levels: dataflow parallelism occurs among AVS visualization modules and between an AVS module and a remote PVM (host) computation module, and general message passing paradigm is used by the PVM node tasks. A PVM node task is spawned from the AVS/PVM host module when the module is registered in the Network Editor. The AVS kernel only communicates with one PVM host daemon through AVS input/output ports. All the AVS programming features, such as transparent networking, modular process management and event-driven dataflow, remain intact, as well as the PVM concurrent programming paradigm. We use this approach in our case study to be discussed in Section 4.

2. In the second approach, within the data-flow execution of AVS modules, PVM primitives are used by AVS remote modules to perform temporal synchronization of arbitrary pattern or even transfer data among subroutine or coroutine modules. The major purpose of PVM here is to enhance remote process concurrency control in AVS kernel. In this approach, a remote AVS/PVM module consists of an AVS/PVM host part and a PVM node part. The module can arbitrarily communicate with other AVS/PVM modules, a host part or a node part, or even both. This communication can occur anytime before or after AVS modules transfer data through normal dataflow input/output ports. Each AVS/PVM module executes as a relatively autonomous process (or a group of loosely coupled processes) similar to an AVS coroutine. The

AVS kernel can communicate with as many PVM daemons as required. This approach requires existing AVS kernel's interface to the Network Editor be extended to allow a spawned AVS/PVM module from a module program to be able to register in the Editor and to make ports connection.

4 Option Price Modeling on an IBM SP1 and a DEC Alpha Farm – A Case Study

In previous work, we implemented a set of stock option pricing models on a number of massively parallel processing systems. We used these high-performance pricing models to compare model prices and historical market data to evaluate pricing model performance, as well as the potential for optimization techniques to improve parameter estimates and model accuracy[11, 12].

Stock option pricing models are used to calculate a price for an option contract based on a set of market variables, (e.g. exercise price, risk-free rate, time to maturity) and a set of model parameters. Model price estimates are highly sensitive to parameter values for volatility of stock price, variance of the volatility, and correlation between volatility and stock price. These model parameters are not directly observable, and must be estimated from market data.

We use a set of four option pricing models in this study. Simple models treat stock price volatility as a constant, and price only European (option exercised only at maturity of contract) options. More sophisticated models incorporate stochastic volatility processes, and price American contracts (option exercised at any time in life of contract). These models are computationally intensive and have significant communication requirements. The four pricing models are: BS – the Black-Scholes constant volatility, European model; AMC – the American binomial, constant volatility model; EUS – the European binomial, stochastic volatility model; and AMS – the American binomial, stochastic volatility model.

Figure 1 is the system configuration of a prototype interactive simulation-on-demand system for the option price modeling application, using the AVS/PVM framework we proposed in Section 3 and utilizing the network infrastructure and distributed computing facility at NPAC.

The AVS kernel runs on a SUN10 workstation which acts both as an AVS server to coordinate data-flow and top-level concurrent control among remote modules, and as a network gateway which links the NPAC in-house host machines locally networked by an Ethernet to the regional end-user through the ATM-based WAN (in the actual demonstration, a NYNET link to Rome Laboratory, Griffiss Airbase, NY). The ATM-based link is built around two Fore switches that operate at 155 Mbps (OC3c) while the wide area network portion of the network operates at OC48(2400 Mbps) speed.

The two parallel pricing models (EUS model and AMS model) are implemented in PVM and run respectively on a 8-node IBM SP1, networked by an Ethernet at the time of evaluation, and a 8-node DEC Alpha cluster inter-connected by a FDDI-based GIGAswitch. They are coupled under the proposed AVS environment with the other two sequential simple models(BS model and AMC model) running on a SUN4 and a DEC5000 workstation, respectively. The nodal processor of SP1 is IBM RISC/6000 processor running at 62.5 MHz and is one of the most powerful processors available. The DEC Alpha farm consists of 8 Alpha model 4000 workstations which are supported by a high performance networking backbone of a dedicated, switched FDDI segments. The GIGAswitch provides full FDDI bandwidth and low latency switching to every workstation in the farm.

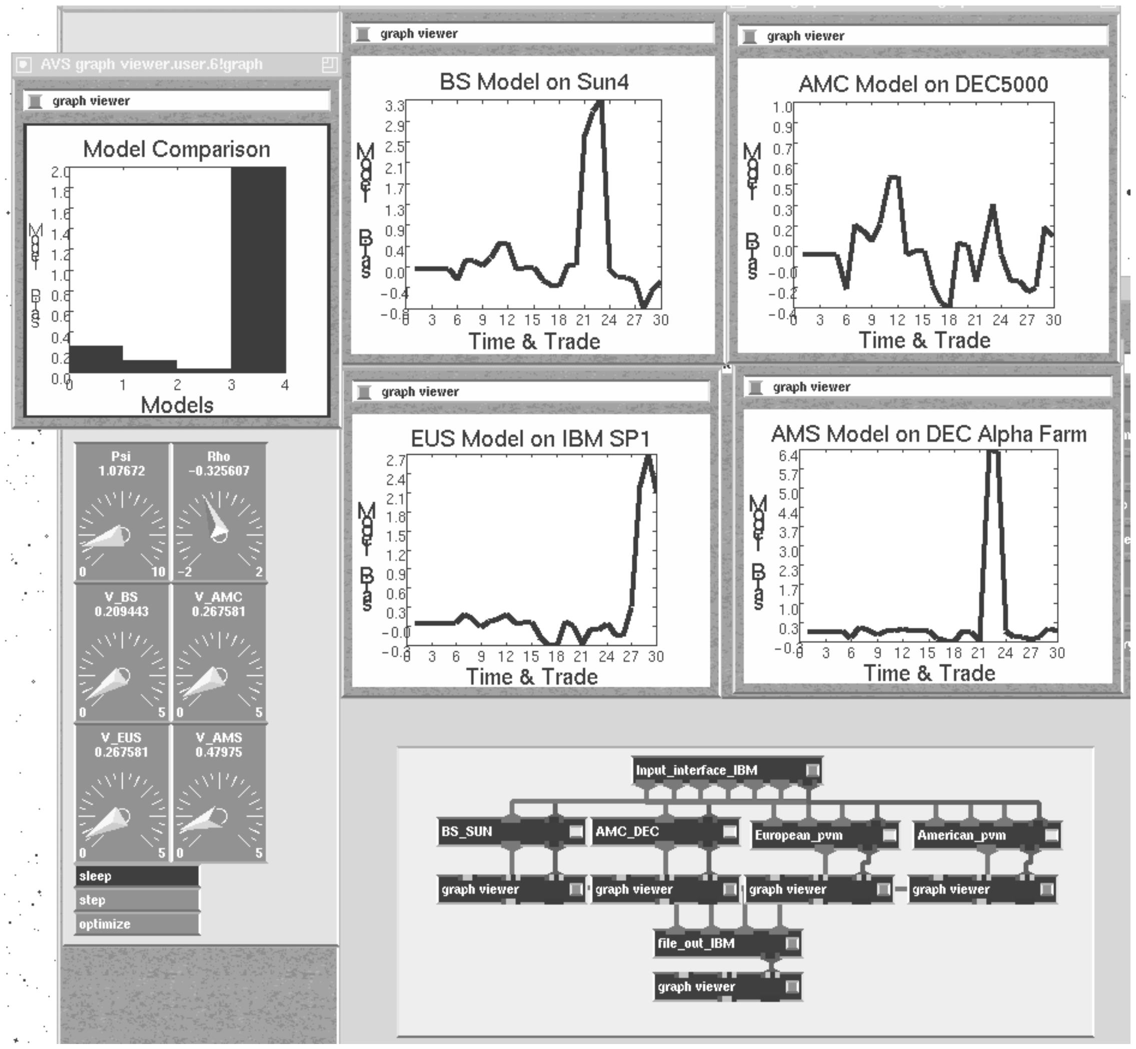

Fig. 4. *The Graphical User Interface on the Home Machine*

While displayed on the end-user's home machine, a user interface actually runs on a remote SUN4 which combines user runtime input (model parameters, network configuration) with historical market databases stored on disk, and broadcasts this data to remote compute nodes. Top-level system synchronization occurs with each broadcast.

An IBM RS/6000 is used as a file server for non-graphical output of model data. In this application, model prices calculated at remote compute nodes and corresponding market data are written to databases for later analysis.

All models output are graphically displayed on the end-user's home machine(a SUN10) in AVS graph viewers. Figure 4 gives the user interface showing the simulation control panel(left), model output windows(top) and the flow network(bottom).

Notice that with the use of PVM software, only one AVS license, i.e., the one on Sun10, is needed for this heterogeneous distributed system.

5 Performance Consideration

The major performance issue in the discussed integration context is a centralized AVS versus a decentralized parallel system. AVS is a centralized system in terms of facilitating networking, i.e. a single AVS kernel supervises every networking activity. But most message passing systems such as PVM, P4, Express, Fortran-M are decentralized (or distributed) system in which there are multiple daemons (one on each processor) taking care of data/control transfers among processes. Because of this centralized nature there is one critical requirement in AVS programming environment for integrating parallel programming: a host-node programming model must be used for any parallel computational module in AVS.

For any AVS (remote) module to be implemented as a parallel module on a particular parallel platform (such as a PVM program, a CMfortran module or a CMMD program), there must be an explicit sequential host component which acts like a daemon for data/control transfer from/to other AVS remote modules.

On the top AVS data-flow level, starting from AVS 4.0, data can flow directly between any two remote (host) modules, bypassing the kernel and only necessary control communication needs to be between the kernel and the remote modules. In this paper, we didn't discuss performance gain of the integrated system, due to thet fact that our models run on high performance systems.

At the bottom level, assuming there are two parallel AVS modules: one consists of a host process p_1(runs on a processor P_1) and a couple of node processes in which a node process p_2 runs on P_2, the other has a host process p_3 on P_3 and a node process p_4 on P_4, p_2 needs to send data to p_4. As shown in Figure 3, under current AVS framework, the data transfer path has to be $P_2 \Rightarrow P_1 \Rightarrow P_4 \Rightarrow P_3$. But the most efficient path (or parallel i/o channel) is directly $P_2 \Rightarrow P_3$, bypassing both host processors(and of course the AVS kernel for both data and control communication). In this sense, current AVS kernel serializes parallel communication.

6 Conclusion and Future Work

A software framework using AVS and PVM is described in this paper for a simulation-on-demand prototype system of a financial modeling application on a distributed system. Based on our initial work with PVM and AVS, we conclude that a general heterogeneous distributed computing/visualization environment can be developed with the integrated and complementary networking and graphics programming capabilities of both software packages.

By using portable PVM software, we expect to build similar integrated parallel computing/interactive visualization applications on other MPP architectures where PVM is supported while AVS is not available currently on the host processor, such as Intel Paragon and Cray T3D.

We are exploring new software framework in this area and plan to apply the integration technique described in this paper to other InfoMall applications. We plan to add on top of the AVS framework a network user interface, Mosaic, a distributed hypermedia software from NCSA, to support InfoVision simulation-on-demand projects over the ATM-based wide area network. We believe that methodologies and tools for information integration will play a more and more important role with the adoption of HPCC technologies in industry.

Acknowledgment: Gang Cheng would like to thank Peter Crockett, Mark Levinson and

Steve Leonard for systems support, Roman Markowski for useful discussion about ATM switch and Mosaic.

References

[1] Advanced Visual Systems Inc. *AVS 4.0 Developer's Guide and User's Guide*, May 1992.

[2] A. Beguelin, J. Dongarra, G. A. Geist, R. Manckek, V. S. Sunderam, *Graphical Development Tools for Network-Based Concurrent Supercomputing*, Proc. of Supercomputing '91, IEEE, pp. 435-445, 1991.

[3] G. Cheng, K. Mills and G. Fox, *An Interactive Visualization Environment for Financial Modeling on Heterogeneous Computing Systems*, Proc. of the 6th SIAM Conference on Parallel Processing for Scientific Computing, R. F. Sincovec, eds., SIAM, Norfolk, VA, March 1993.

[4] G. Cheng, Y. Lu, G.C. Fox, K. Mills and T. Haupt, *An Interactive Remote Visualization Environment for an Electromagnetic Scattering Simulation on a High Performance Computing System*, Proc. of Supercomputing '93, Portland, OR, November, 1993.

[5] G. Cheng, C. Faigle, G. C. Fox, W. Fumanski, B. Li, and K. Mills, *Exploring AVS for HPDC Software Integration: Case Studies Towards Parallel Support for GIS*, Proc. of the 2nd AVS Conference AVS'93, Lake Buena Vista, FL, May 1993.

[6] G. Cheng, G. C. Fox, and K. Mills, *Integrating Multiple Programming Paradigms on Connection Machine CM5 in a Dataflow-based Software Environment*, Technical Report, Syracuse Center for Computational Science SCCS-548, 15 pps, August, 1993.

[7] G. C. Fox, et al, *InfoVision: Information, Video, Imagery and Simulation on Demand*, Technical Report, Syracuse Center for Computational Science SCCS-575, December, 1993.

[8] G. C. Fox, et al, *InfoMall: A Scalable Organization for the Decelopment of HPCC Software and Systems*, Technical Report, Syracuse Center for Computational Science SCCS-531, October, 1993.

[9] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM 3.0 User's Guide and Reference Manual,* Oak Ridge National Laboratory, ORNL/TM-12187, February, 1993.

[10] G. A. Geist, V. S. Sunderam, *Network-Based Concurrent Computing on the PVM System*, Concurrency: Practice and Experience, Vol. 4(4), 293-311 (June 1992).

[11] K. Mills, M. Vinson, and G. Cheng, *A Large Scale Comparison of Option Pricing Models with Historical Market Data,* in the 4th Symposium on the Frontiers of Massively Parallel Computation, Oct. 19-21, 1992, McLean, Virginia.

[12] K. Mills, G. Cheng, M. Vinson, S. Ranka, and G. Fox, *Software Issues and Performance of a Parallel Model for Stock Option Pricing,* in the Fifth Australian Supercomputing Conference, Dec. 6-7, 1992, Melbourne, Australia.

[13] V. Sunderam, *PVM: A Framework for Parallel Distributed Computing*, Concurrency: practice and experience, 2(4), Dec. 1990.

[14] C. Upson, T. Faulhaber, Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz and A. van Dam, *The Application Visualization System: A Computational Environment for Scientific Visualization*, IEEE Computer Graphics and Applications, July, 1989.