Appendix

1. User directions

(To be appended)

2. Source Code and stock data formats of input files

(To be appended)

6. Conclusions

This work shows to us that:

- The importance to make use of existing(well-developed) visualization environment in the parallel programming environment;

- A distributed parallel computing environment with scientific data visualization can be achieved through the use of AVS;

- A highly portable interactive graphic tool for the option price modeling on parallel computers such as DECmpp-12000 can be implemented with relatively small programming effort, provided that AVS be the visualization environment and the parallel programming environment supports multi-paradigm programmings at compiler level.

Acknowledgments

The author is grateful to Kim Mills for his encouragement and continuous support during this study. I want to thank Marek Podgorny and Tomasz Haupt for the useful discussions about AVS. I also want to thank Peter Crockett for the system support.

This work is partially supported by a Corporate Partnership funding from Digital Eqipment Corporation.

The use of any trademark in this documentation is not intended in any way to infringe on the rights of the trademark holder.

References

[1]. T. Finucane, "Binomial Approximations of American Call Option Prices with Stochastic Volatilities," to be published in *Journal of Finance.* 1992.

[2] K. Mills, M. Vinson, G. Cheng, T. Finucane, "A Large Scale Comparison of Option pricing Models with Historical Market Data," to be appeared in *the 4th Symposium on the Frontiers of Massively Parallel Computation*, Oct. 19-21, 1992, McLean, Virginia.

[3]. Stardent Computer Inc. *Application Visualization System*, Volume 1 and 2, Release 3.0, April 1991.

[4]. K. Mills, G. Cheng, M. Vinson, etc. "Load Balancing, Communication, and Performance of a Stock option Pricing Model on the Connection machine-2 and DECmpp-12000,", *Syracuse Center for Computational Science-237*, 22 pps. 1992.

[5]. MasPar Computer Corporation. *MasPar Parallel Application Language(MPL) User Guide and Reference Manual*, Version 2.1, Revision A5, 1991.

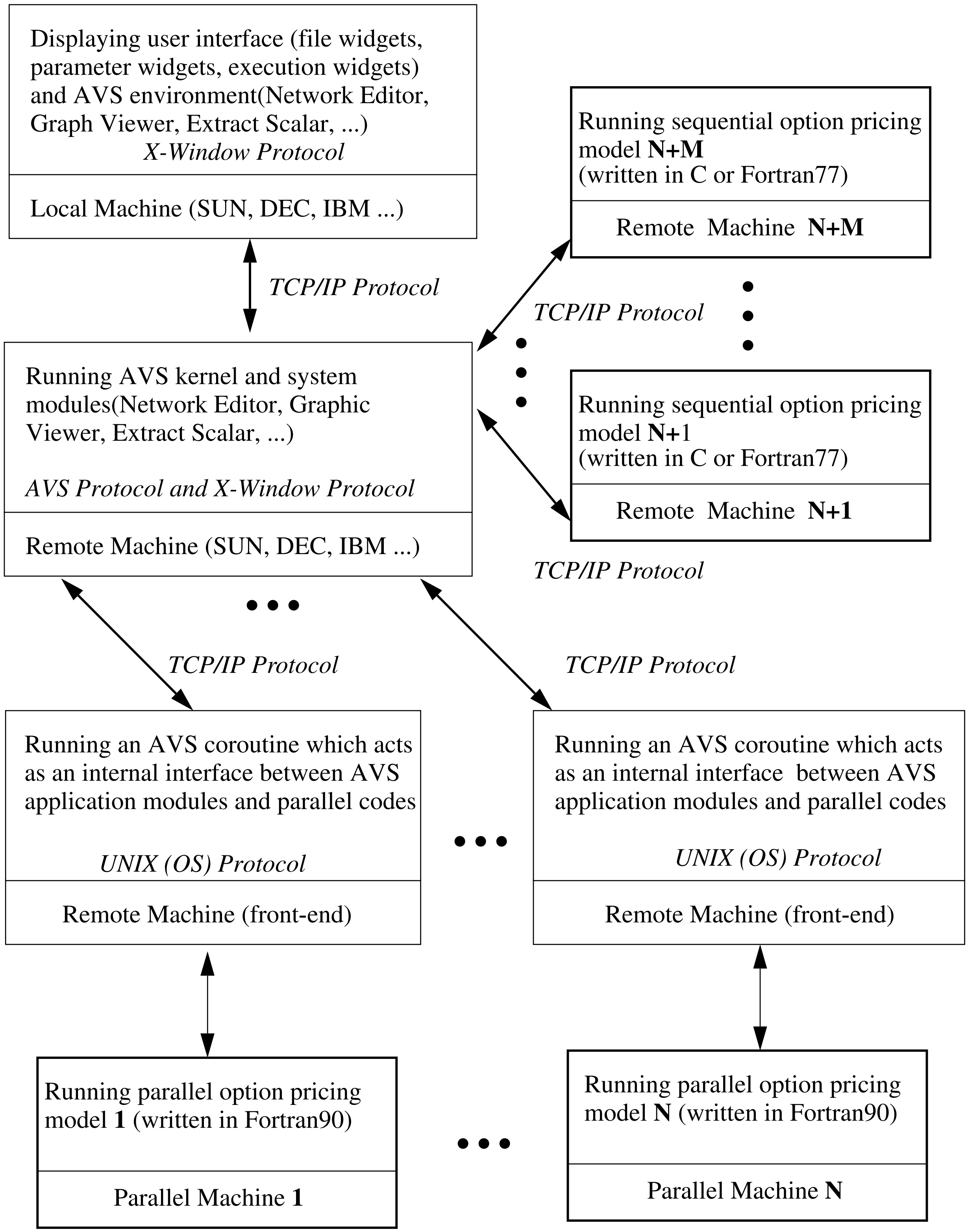

Figure 4 A complete distributed environment for a large application problem requiring scientific data visualization

5. Discussion and Further Work

In this study, we choose DECmpp-12000 to be the parallel machine running the option price models. As it becomes obvious from the above discussion, it will be very easy to port this tool to any other parallel machines. AVS and Fortran90 help us to achieve this goal. It is not impossible that using AVS we can achieve a complete high level heterogeneous distributed computing environment for a large application problem which requires data visualization, as illustrated in Figure 4, where each option pricing model can be thought as a relatively independent component in a large application problem domain.

But, as found in the practice of this study, the current parallel programming environment is far from flexible and complete to facilitate such a computing environment. Originally, our option price codes are implemented in Fortran-90 (MP-fortran)[4]. The DECmpp-12000 programming environment does not support a mechanism to call MPfortran subroutines from a front-end C program or Fortran77 program at the front-end. It does provide a way to call MPL functions from C. MPL is a C-like programming language with parallel extensions on DECmpp-12000[5] which is not portable to other parallel machines. To integrate our model into AVS modules which is written in C, we have to rewrite all parallel models in MPL. This reduces the portability of this tool, though our models have Fortran90 versions. This also brings us another parallel programming environment issue that because so far the parallel programing environments on parallel machines are far less rich than those on sequential machines as their front-ends, there should be a flexible mechanism to provide the multi-language mixed programming environment, at least on the compiler level such that subroutines(functions) written in different programming languages can be called each other.

Because three more protocols, namely, AVS kernel, TCP/IP, X-window Manager, and of course UNIX, are involved in the visualization, there is a performance bottleneck in the network communication. The remote module producing the source visualizing data has to wait until the data is visualized on the display window over the network, which proves to be far slower than the computing model on DECmpp-12000.

Another difficult comes from the AVS. AVS is designed mainly to visualize 3-D data. The Graph Viewer in AVS is too simple to perform complicated tasks for 2-D data. In our application, it would be good to visualize and animate more than one model's data in a single window or in different windows to have a graphic comparison of different model's performance. But the AVS failed to provide such functionality, even in the latest release version 4.0.

Our further work would port this tool to CM2 or/and CM5, and finally implement the distributed environment shown in Figure 4 for at least DECmpp-12000, CM2 and CM5. Adding more functionality as required by the model optimization in the user interface can be done in the near future.

inputs, invoke model's parallel (and/or sequential) subroutines to generate model's call prices, and produce animating data for output which is a vector for the two models.

• a system module "Extract Scalar" ---- extract from the vector data generated by the coroutine a specified model's data for output.

• a system module "Graph Viewer" ---- in a separate window plot the output data in XY linear plots where X represents for the number(and time) sequence of tradings and Y for model's call price minus market's call price.

The module diagram(namely in AVS, a network) is shown in Figure 2.

Figure 3 is a picture of the overall screen of this tool's user interface, which contains a parameters control panel(left), a display window(upper right), AVS network editor and two shell windows(-lower right).

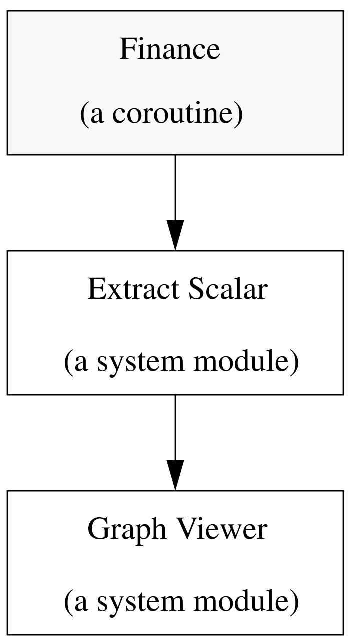

Figure 2 module diagram

As can be seen from Figure 1, AVS kernel acts as a coordinator to conduct data flows and control flows in the network. The main implementation tasks in the implementation are to have an internal interface between AVS modules and the model programs(subroutines) running on DECmpp-12000, and to design an user interface to define the interaction between tool user and the system.

In the user interface of the tool, two file browsers are used as graphic widgets for chossing file names of:

one file for storing primitive option pricing data, in which the basic input data of the models are contained and the other for initial values of variance of stock volatility and correlation between stock price and its volatility. They may be obtained through external ways or optimizations. The data formats of the two files can be found in Appendix.

Four dial widgets are designed for the 4 model parameters:

- variance of stock volatility, and
- correlation between stock price and its volatility

Their initial values are currently input from one of the files mentioned above. They can be changed anytime in the modeling.

- volatility for “Black Sholes Model”, and
- volatility for “Stochastic Volatility American Model”

In the first half-hour of a trading day, they are calculated by an optimization function and can be changed at anytime of the next 6 trading half hours of that day.

Another two “oneshot” widgets are designed to allow user:

- to temporarily pause the modeling process to have a detailed view of the animated pictures or change model parameters;
- to execute the modeling in the way of single-step.

The basic data of the option price models are simple 2-D data. We use the Graph Viewer in AVS as the display tool. A simple (high-level) animation technique is used to show the dynamic evolution of the model price vs. market price over the successive trades and trading time. In the current implementation, two models, “Black-Scholes Model” and “Stochastic Volatility American Model”, are contained in this tool.

An AVS coroutine is used as the internal interface between AVS modules and the model programs on DECmpp-12000. This coroutine controls the data flow between the DECmpp-12000’s front-end (a DECstation 5000) and the DPU, and also the data flow between the user interface and the system components which are composed of:

- the same coroutine called “Finance” on the front-end ---- read the primitive stock data, calculate initial volatilities for the first half hour of a trading day, gather the runtime user model parameter

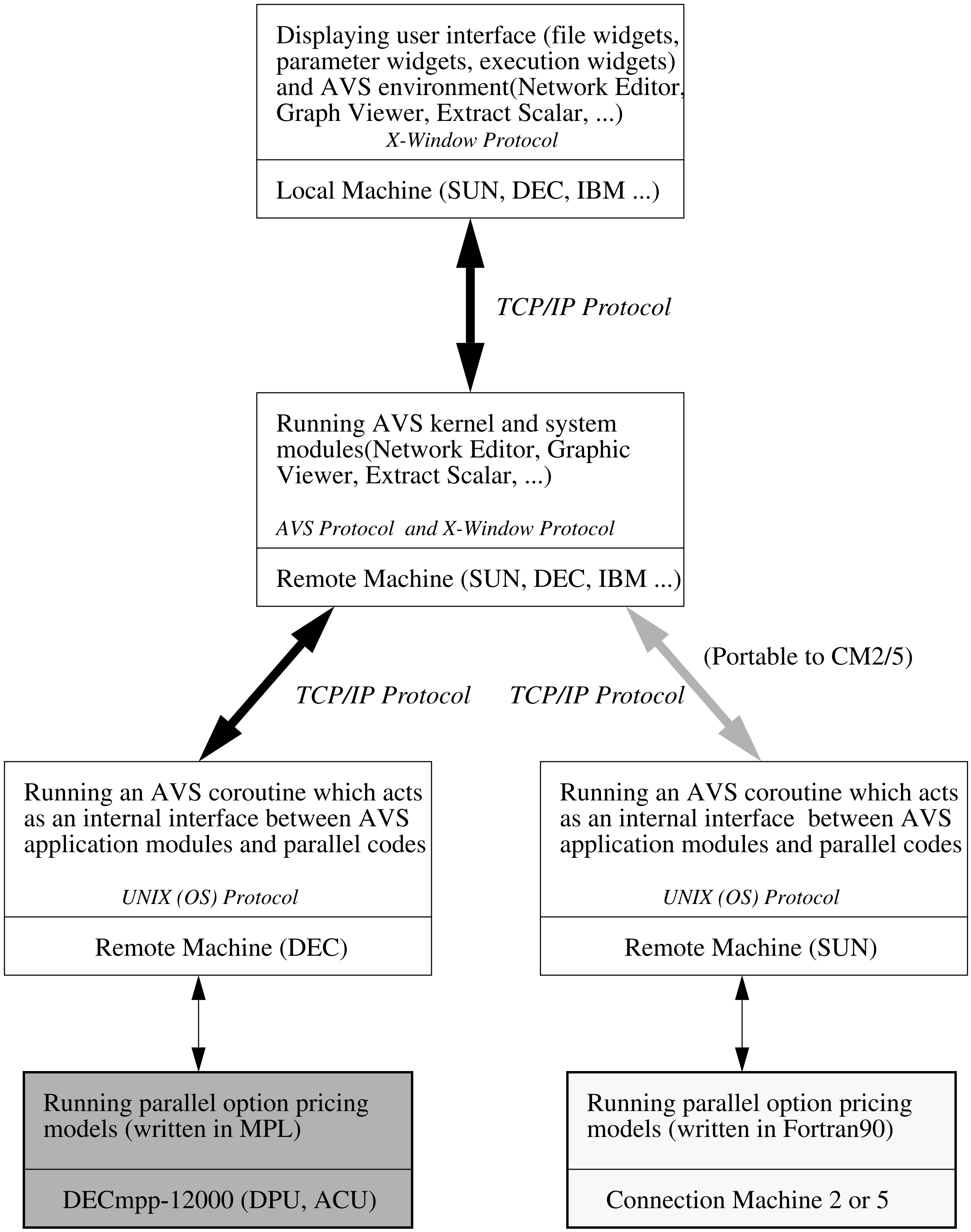

Figure 1 A distributed environment executing the visualization tool

Fortran) which performs a specific visualization function or set of functions. More than 110 (system) modules are provided with standard AVS. External programs or subroutines written in C or Fortran cab be easily converted into AVS modules.

AVS was designed from the outset to operate in distribute, multi-vendor, heterogeneous computing environments. The support for remote module execution allows modules to execute on computational servers in the network. X-terminal support allows AVS to display on 8-bit color X-terminals or workstations running the X-Window system. These distributed capabilities enable the user to incorporate AVS into the networked, distributed computing model already established today. This makes our models running on DECmpp-12000 easily to be combined into the AVS environment.

For detailed functionality about AVS, refer to [3].

4. Integrating the Option Price Modeling on DECmpp-12000 into AVS

There are two alternatives for data visualizations on parallel machines. One is to use the display or image processing library provided by the parallel machine's programming environment, such as MPDDL on DECmpp-12000, CMX11 on CM5. This can be very efficient as they fetch the data to be visualized directly from each processor element but it generally requires particular hardware like framebuffer which can not be shared by other users. The other way is to make use of the well-developed conventional graphics environment such as X-window Toolkit, OSF/Motif, AVS on sequential machines which are usually connected to the parallel machines as front ends.

Based on the following observations, we choose AVS as front-end visualization environment to implement our tool:

The characteristics of the option price modeling on DECmpp-12000 requires not only scientific data visualization but also dynamic interaction with the modeling itself. The MPDDL on DECmpp-12000 provides very limited facilities to support the graphic interaction with the running program.

Our tool is designed to be portable to any parallel machines which can run the option price models. AVS is now available on the major UNIX-based workstation and visualization systems. These include computer systems from Convex Computer Corporation, Cray Research Inc., Digital Equipment Corporation, Evans and Sutherland, Hewlett-Packard, IBM Corporation, Kubota Pacific Computer Inc., Set Technology Corporation, Sun Microsystems, and Wavetracer.

Our data visualization does not necessarily require fetching data to be visualized directly from PE nodes. So there would be no significant performance problem to display the scalar (2D) data from the front-end.

In order to illustrate the portability and extensibility of this tool, and also to overcome an unknown AVS system problem (probably a memory problem) that the repeated invocations of the Graph Viewer on the same machine as the invoking module will lead to system crash, we use a distributed environment to implement the tool. Figure 1 shows the configuration of this environment.

while allowing model parameters either estimated by some optimization method(function) or given by interactively parameter input at run-time, or by both.

Through AVS(Application Visualization System), this tool integrates the parallel option pricing models into an interactive, user-friendly graphic interface and provide the model users and researchers with a dynamic visualization for model verification and optimization.

This work also investigates the feasibility to use AVS as a data visualization tool in parallel programming environment. It further suggests that a complete, high-level distributed computing environment with a comprehensive visualization environment can be achieved through AVS, in which heterogeneous parallel machines and workstations may be integrated all together to accomplish a large application problem which requires scientific data visualization.

This paper discusses the design issues and implementation details of this tool.

2. Option Pricing Models on DECmpp-12000

The parallel option pricing models in our study are binomial approximation models which incorporates stochastic volatility and American call[1]. Another two sequential models assuming constant volatility are used for the purpose of comparison. After a large scale comparison of these models with historical market data[2], we found that a few key model parameters have significant impact on the model performance of (parallel) stochastic models. These parameters are:

- stock initial volatility
- variance of stock volatility
- correlation between stock price and its volatility

The determinations of the parameters are based on many factors in real stock market situations. They are generally unable to be observed directly but must be estimated either from historical data or by financial expertise. They are the major concerns in the model optimization. The model optimization requires more computing time than the model themselves, even on parallel machines. In the comparison of the models running on DECMPP-12000, we use some criteria, most of them are plain data, to evaluate the model performance of a whole stock over a 1-month or 6-month time. In the run of the comparison, we are unable to interact with the modeling process and it will take relatively long time to visualize a parameter’s impact on the model performance, which is done statically, due to the lack of an interactive visualization tool.

3. The AVS environment

Advanced Visual System’s Application Visualization System(AVS) is a comprehensive visualization environment which incorporates the latest advances in visualization software, graphics, networking, high performance systems, and industry standards.

Modules are the building blocks of AVS. A module is an independent computing element (C or

An Interactive Graphic Tool For The Option Price Modeling On DECmpp-12000 Using AVS

Gang Cheng Geoffrey Fox

Northeast Parallel Architrctures Center and School of Computer and Information Science

Syracuse University
Syracuse, New York 13244-4100
315-443-4889
gcheng@nova.npac.syr.edu

Abstract

An interactive graphic tool has been implemented for the option price modeling on DECmpp-12000. Through the use of AVS, it integrates the parallel option pricing models on DECmpp-12000 into an interactive, user-friendly graphic interface which visualizes the modeling process dynamically while allowing model parameters input at run-time by the user.

This paper discusses design issues and implementation of this tool. The feasibility of a distributed parallel computing environment with a comprehensive data visualization environment is also investigated.

1. Introduction

A set of (parallel) option pricing models have been implemented, which can run efficiently on parallel machines such as DECmpp-12000, CM2 and CM5. After a large-scale comparison of the option price models on DECmpp-12000 using historical market data[2], we evaluated that a group of models, namely, “stochastic volatility model”, provide more accurate option price estimates than another group of “conventional” models. We also found that the “stochastic volatility models” are highly sensitive to a couple of model parameters, such as volatility of stock price and variance of the volatility. Those parameters can not be observed directly and can only be estimated either from historical data through some optimization method or from financial expertise.

To further evaluate and optimize the models, a visualizing tool for interacting with the models running on parallel machines would be very useful. An interactive graphic tool for the option price modeling is then implemented on DECmpp-12000 in this study. It visualizes the modeling process

Appendix

1. User directions

(To be appended)

2. Source Code and stock data formats of input files

(To be appended)

6. Conclusions

This work shows to us that:

- The importance to make use of existing(well-developed) visualization environment in the parallel programming environment;

- A distributed parallel computing environment with scientific data visualization can be achieved through the use of AVS;

- A highly portable interactive graphic tool for the option price modeling on parallel computers such as DECmpp-12000 can be implemented with relatively small programming effort, provided that AVS be the visualization environment and the parallel programming environment supports multi-paradigm programmings at compiler level.

Acknowledgments

The author is grateful to Kim Mills for his encouragement and continuous support during this study. I want to thank Marek Podgorny and Tomasz Haupt for the useful discussions about AVS. I also want to thank Peter Crockett for the system support.

This work is partially supported by a Corporate Partnership funding from Digital Eqipment Corporation.

References

[1]. T. Finucane, “Binomial Approximations of American Call Option Prices with Stochastic Volatilities,” to be published in *Journal of Finance.* 1992.

[2] K. Mills, M. Vinson, G. Cheng, T. Finucane, “A Large Scale Comparison of Option pricing Models with Historical Market Data,” to be appeared in *the 4th Symposium on the Frontiers of Massively Parallel Computation*, Oct. 19-21, 1992, McLean, Virginia.

[3]. Stardent Computer Inc. *Application Visualization System*, Volume 1 and 2, Release 3.0, April 1991.

[4]. K. Mills, G. Cheng, M. Vinson, etc. “Load Balancing, Communication, and Performance of a Stock option Pricing Model on the Connection machine-2 and DECmpp-12000,”, *Syracuse Center for Computational Science-237*, 22 pps. 1992.

[5]. MasPar Computer Corporation. *MasPar Parallel Application Language(MPL) User Guide and Reference Manual*, Version 2.1, Revision A5, 1991.

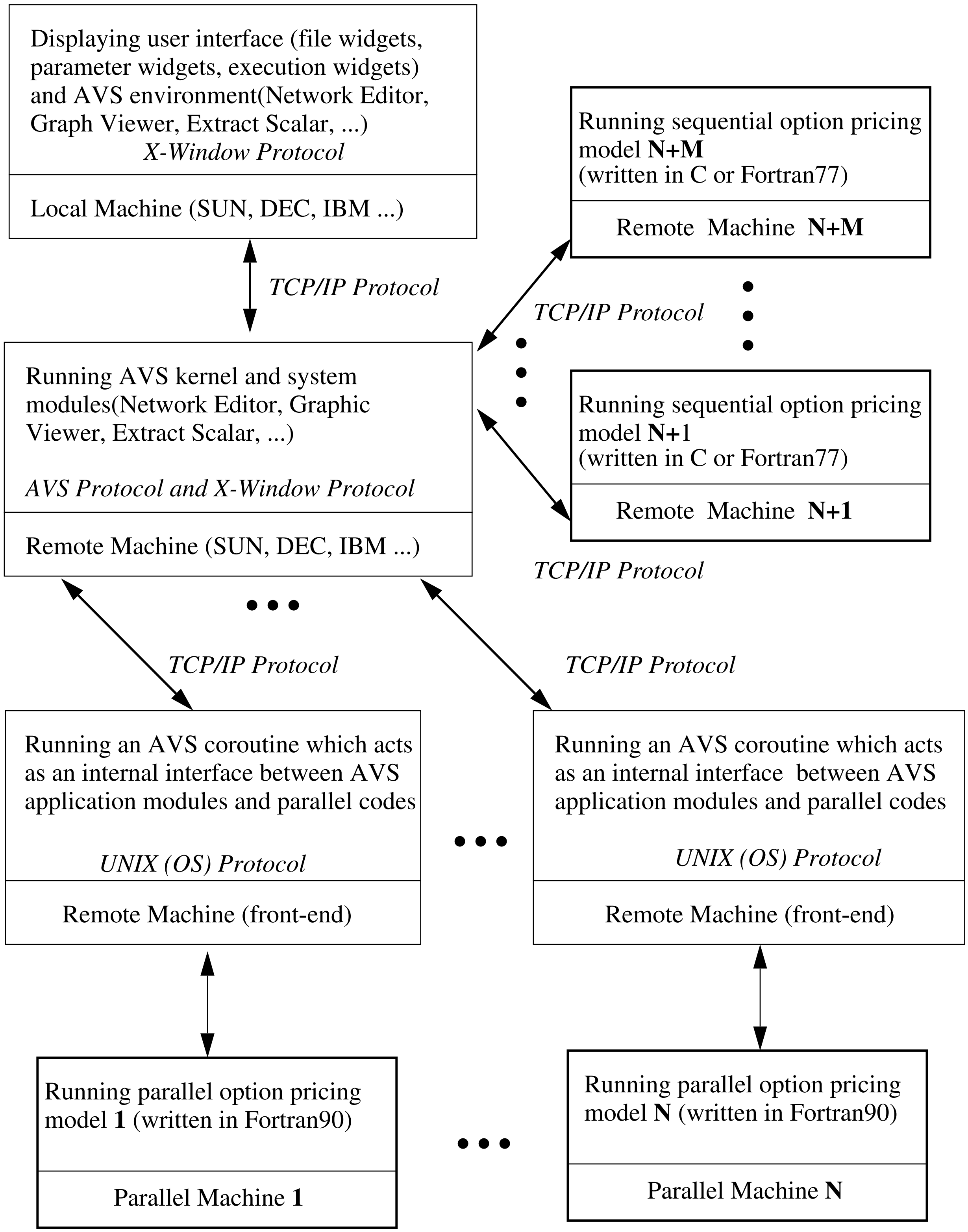

Figure 4 A complete distributed environment for a large application problem requiring scientific data visualization

5. Discussion and Further Work

In this study, we choose DECmpp-12000 to be the parallel machine running the option price models. As it becomes obvious from the above discussion, it will be very easy to port this tool to any other parallel machines. AVS and Fortran90 help us to achieve this goal. It is not impossible that using AVS we can achieve a complete high level heterogeneous distributed computing environment for a large application problem which requires data visualization, as illustrated in Figure 4, where each option pricing model can be thought as a relatively independent component in a large application problem domain.

But, as found in the practice of this study, the current parallel programming environment is far from flexible and complete to facilitate such a computing environment. Originally, our option price codes are implemented in Fortran-90 (MP-fortran)[4]. The DECmpp-12000 programming environment does not support a mechanism to call MPfortran subroutines from a front-end C program or Fortran77 program at the front-end. It does provide a way to call MPL functions from C. MPL is a C-like programming language with parallel extensions on DECmpp-12000[5] which is not portable to other parallel machines. To integrate our model into AVS modules which is written in C, we have to rewrite all parallel models in MPL. This reduces the portability of this tool, though our models have Fortran90 versions. This also brings us another parallel programming environment issue that because so far the parallel programing environments on parallel machines are far less rich than those on sequential machines as their front-ends, there should be a flexible mechanism to provide the multi-language mixed programming environment, at least on the compiler level such that subroutines(functions) written in different programming languages can be called each other.

Because three more protocols, namely, AVS kernel, TCP/IP, X-window Manager, and of course UNIX, are involved in the visualization, there is a performance bottleneck in the network communication. The remote module producing the source visualizing data has to wait until the data is visualized on the display window over the network, which proves to be far slower than the computing model on DECmpp-12000.

Another difficult comes from the AVS. AVS is designed mainly to visualize 3-D data. The Graph Viewer in AVS is too simple to perform complicated tasks for 2-D data. In our application, it would be good to visualize and animate more than one model's data in a single window or in different windows to have a graphic comparison of different model's performance. But the AVS failed to provide such functionality, even in the latest release version 4.0.

Our further work would port this tool to CM2 or/and CM5, and finally implement the distributed environment shown in Figure 4 for at least DECmpp-12000, CM2 and CM5. Adding more functionality as required by the model optimization in the user interface can be done in the near future.

inputs, invoke model's parallel (and/or sequential) subroutines to generate model's call prices, and produce animating data for output which is a vector for the two models.

• a system module "Extract Scalar" ---- extract from the vector data generated by the coroutine a specified model's data for output.

• a system module "Graph Viewer" ---- in a separate window plot the output data in XY linear plots where X represents for the number(and time) sequence of tradings and Y for model's call price minus market's call price.

The module diagram(namely in AVS, a network) is shown in Figure 2.

Figure 3 is a picture of the overall screen of this tool's user interface, which contains a parameters control panel(left), a display window(upper right), AVS network editor and two shell windows(-lower right).

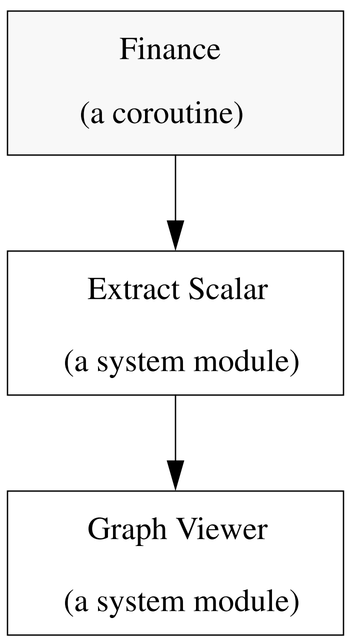

Figure 2 module diagram

As can be seen from Figure 1, AVS kernel acts as a coordinator to conduct data flows and control flows in the network. The main implementation tasks in the implementation are to have an internal interface between AVS modules and the model programs(subroutines) running on DECmpp-12000, and to design an user interface to define the interaction between tool user and the system.

In the user interface of the tool, two file browsers are used as graphic widgets for chossing file names of:

one file for storing primitive option pricing data, in which the basic input data of the models are contained and the other for initial values of variance of stock volatility and correlation between stock price and its volatility. They may be obtained through external ways or optimizations. The data formats of the two files can be found in Appendix.

Four dial widgets are designed for the 4 model parameters:

- variance of stock volatility, and
- correlation between stock price and its volatility

Their initial values are currently input from one of the files mentioned above. They can be changed anytime in the modeling.

- volatility for "Black Sholes Model", and
- volatility for "Stochastic Volatility American Model"

In the first half-hour of a trading day, they are calculated by an optimization function and can be changed at anytime of the next 6 trading half hours of that day.

Another two "oneshot" widgets are designed to allow user:

- to temporarily pause the modeling process to have a detailed view of the animated pictures or change model parameters;
- to execute the modeling in the way of single-step.

The basic data of the option price models are simple 2-D data. We use the Graph Viewer in AVS as the display tool. A simple (high-level) animation technique is used to show the dynamic evolution of the model price vs. market price over the successive trades and trading time. In the current implementation, two models, "Black-Scholes Model" and "Stochastic Volatility American Model", are contained in this tool.

An AVS coroutine is used as the internal interface between AVS modules and the model programs on DECmpp-12000. This coroutine controls the data flow between the DECmpp-12000's front-end (a DECstation 5000) and the DPU, and also the data flow between the user interface and the system components which are composed of:

- the same coroutine called "Finance" on the front-end ---- read the primitive stock data, calculate initial volatilities for the first half hour of a trading day, gather the runtime user model parameter

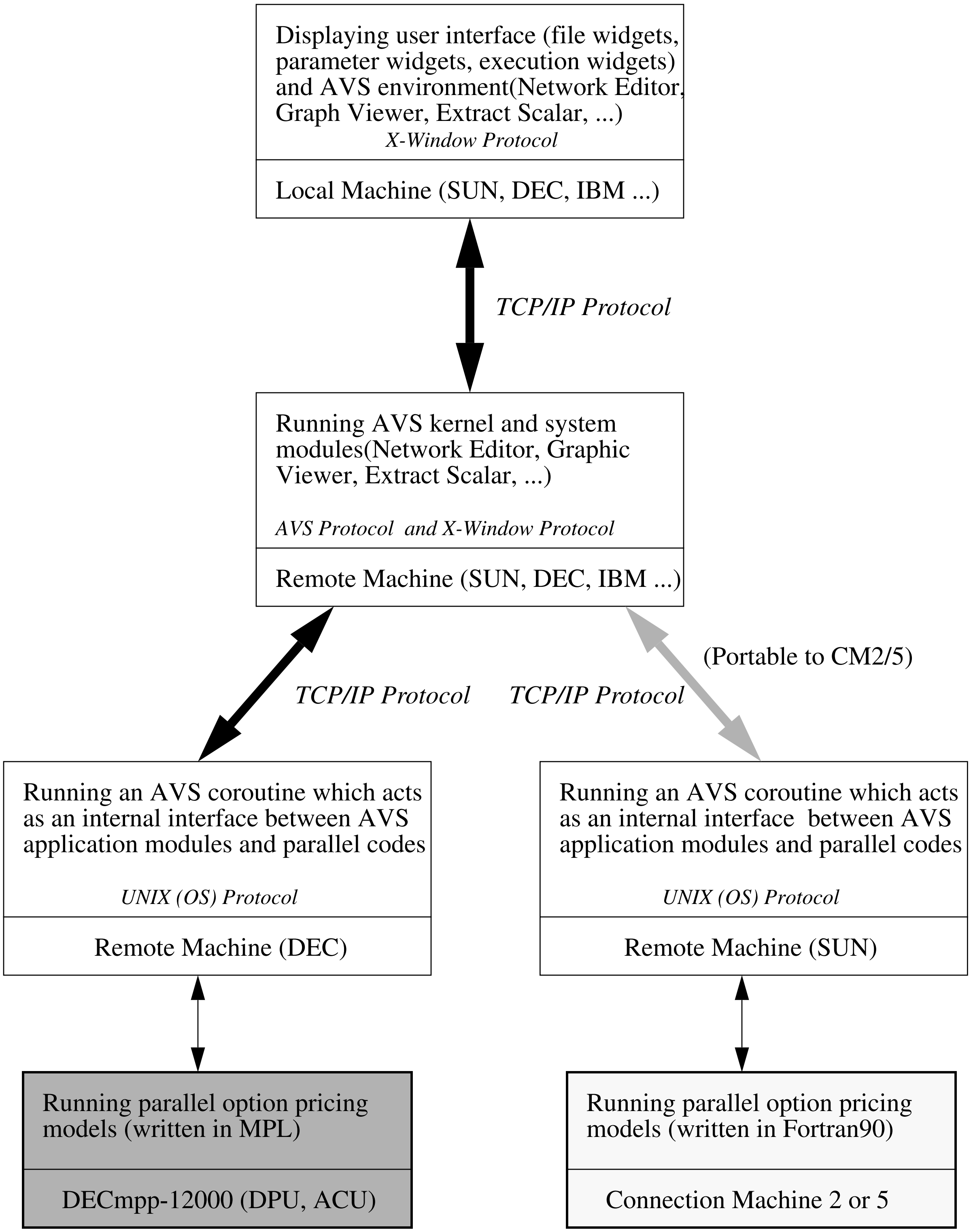

Figure 1 A distributed environment executing the visualization tool

Fortran) which performs a specific visualization function or set of functions. More than 110 (system) modules are provided with standard AVS. External programs or subroutines written in C or Fortran cab be easily converted into AVS modules.

AVS was designed from the outset to operate in distribute, multi-vendor, heterogeneous computing environments. The support for remote module execution allows modules to execute on computational servers in the network. X-terminal support allows AVS to display on 8-bit color X-terminals or workstations running the X-Window system. These distributed capabilities enable the user to incorporate AVS into the networked, distributed computing model already established today. This makes our models running on DECmpp-12000 easily to be combined into the AVS environment.

For detailed functionality about AVS, refer to [3].

4. Integrating the Option Price Modeling on DECmpp-12000 into AVS

There are two alternatives for data visualizations on parallel machines. One is to use the display or image processing library provided by the parallel machine's programming environment, such as MPDDL on DECmpp-12000, CMX11 on CM5. This can be very efficient as they fetch the data to be visualized directly from each processor element but it generally requires particular hardware like framebuffer which can not be shared by other users. The other way is to make use of the well-developed conventional graphics environment such as X-window Toolkit, OSF/Motif, AVS on sequential machines which are usually connected to the parallel machines as front ends.

Based on the following observations, we choose AVS as front-end visualization environment to implement our tool:

The characteristics of the option price modeling on DECmpp-12000 requires not only scientific data visualization but also dynamic interaction with the modeling itself. The MPDDL on DECmpp-12000 provides very limited facilities to support the graphic interaction with the running program.

Our tool is designed to be portable to any parallel machines which can run the option price models. AVS is now available on the major UNIX-based workstation and visualization systems. These include computer systems from Convex Computer Corporation, Cray Research Inc., Digital Equipment Corporation, Evans and Sutherland, Hewlett-Packard, IBM Corporation, Kubota Pacific Computer Inc., Set Technology Corporation, Sun Microsystems, and Wavetracer.

Our data visualization does not necessarily require fetching data to be visualized directly from PE nodes. So there would be no significant performance problem to display the scalar (2D) data from the front-end.

In order to illustrate the portability and extensibility of this tool, and also to overcome an unknown AVS system problem (probably a memory problem) that the repeated invocations of the Graph Viewer on the same machine as the invoking module will lead to system crash, we use a distributed environment to implement the tool. Figure 1 shows the configuration of this environment.

while allowing model parameters either estimated by some optimization method(function) or given by interactively parameter input at run-time, or by both.

Through AVS(Application Visualization System), this tool integrates the parallel option pricing models into an interactive, user-friendly graphic interface and provide the model users and researchers with a dynamic visualization for model verification and optimization.

This work also investigates the feasibility to use AVS as a data visualization tool in parallel programming environment. It further suggests that a complete, high-level distributed computing environment with a comprehensive visualization environment can be achieved through AVS, in which heterogeneous parallel machines and workstations may be integrated all together to accomplish a large application problem which requires scientific data visualization.

This paper discusses the design issues and implementation details of this tool.

2. Option Pricing Models on DECmpp-12000

The parallel option pricing models in our study are binomial approximation models which incorporates stochastic volatility and American call[1]. Another two sequential models assuming constant volatility are used for the purpose of comparison. After a large scale comparison of these models with historical market data[2], we found that a few key model parameters have significant impact on the model performance of (parallel) stochastic models. These parameters are:

- stock initial volatility
- variance of stock volatility
- correlation between stock price and its volatility

The determinations of the parameters are based on many factors in real stock market situations. They are generally unable to be observed directly but must be estimated either from historical data or by financial expertise. They are the major concerns in the model optimization. The model optimization requires more computing time than the model themselves, even on parallel machines. In the comparison of the models running on DECMPP-12000, we use some criteria, most of them are plain data, to evaluate the model performance of a whole stock over a 1-month or 6-month time. In the run of the comparison, we are unable to interact with the modeling process and it will take relatively long time to visualize a parameter's impact on the model performance, which is done statically, due to the lack of an interactive visualization tool.

3. The AVS environment

Advanced Visual System's Application Visualization System(AVS) is a comprehensive visualization environment which incorporates the latest advances in visualization software, graphics, networking, high performance systems, and industry standards.

Modules are the building blocks of AVS. A module is an independent computing element (C or

An Interactive Graphic Tool For The Option Price Modeling On DECmpp-12000 Using AVS

Gang Cheng Geoffrey Fox

Northeast Parallel Architrctures Center and School of Computer and Information Science

Syracuse University
Syracuse, New York 13244-4100
315-443-4889
gcheng@nova.npac.syr.edu

Abstract

An interactive graphic tool has been implemented for the option price modeling on DECmpp-12000. Through the use of AVS, it integrates the parallel option pricing models on DECmpp-12000 into an interactive, user-friendly graphic interface which visualizes the modeling process dynamically while allowing model parameters input at run-time by the user.

This paper discusses design issues and implementation of this tool. The feasibility of a distributed parallel computing environment with a comprehensive data visualization environment is also investigated.

1. Introduction

A set of (parallel) option pricing models have been implemented, which can run efficiently on parallel machines such as DECmpp-12000, CM2 and CM5. After a large-scale comparison of the option price models on DECmpp-12000 using historical market data[2], we evaluated that a group of models, namely, "stochastic volatility model", provide more accurate option price estimates than another group of "conventional" models. We also found that the "stochastic volatility models" are highly sensitive to a couple of model parameters, such as volatility of stock price and variance of the volatility. Those parameters can not be observed directly and can only be estimated either from historical data through some optimization method or from financial expertise.

To further evaluate and optimize the models, a visualizing tool for interacting with the models running on parallel machines would be very useful. An interactive graphic tool for the option price modeling is then implemented on DECmpp-12000 in this study. It visualizes the modeling process

Appendix

1. User directions

(To be appended)

2. Source Code and stock data formats of input files

(To be appended)

6. Conclusions

This work shows to us that:

- The importance to make use of existing(well-developed) visualization environment in the parallel programming environment;

- A distributed parallel computing environment with scientific data visualization can be achieved through the use of AVS;

- A highly portable interactive graphic tool for the option price modeling on parallel computers such as DECmpp-12000 can be implemented with relatively small programming effort, provided that AVS be the visualization environment and the parallel programming environment supports multi-paradigm programmings at compiler level.

Acknowledgments

The author is grateful to Kim Mills for his encouragement and continuous support during this study. I want to thank Marek Podgorny and Tomasz Haupt for the useful discussions about AVS. I also want to thank Peter Crockett for the system support.

This work is partially supported by a Corporate Partnership funding from Digital Eqipment Corporation.

References

[1]. T. Finucane, "Binomial Approximations of American Call Option Prices with Stochastic Volatilities," to be published in *Journal of Finance.* 1992.

[2] K. Mills, M. Vinson, G. Cheng, T. Finucane, "A Large Scale Comparison of Option pricing Models with Historical Market Data," to be appeared in *the 4th Symposium on the Frontiers of Massively Parallel Computation*, Oct. 19-21, 1992, McLean, Virginia.

[3]. Stardent Computer Inc. *Application Visualization System*, Volume 1 and 2, Release 3.0, April 1991.

[4]. K. Mills, G. Cheng, M. Vinson, etc. "Load Balancing, Communication, and Performance of a Stock option Pricing Model on the Connection machine-2 and DECmpp-12000,", *Syracuse Center for Computational Science-237*, 22 pps. 1992.

[5]. MasPar Computer Corporation. *MasPar Parallel Application Language(MPL) User Guide and Reference Manual*, Version 2.1, Revision A5, 1991.

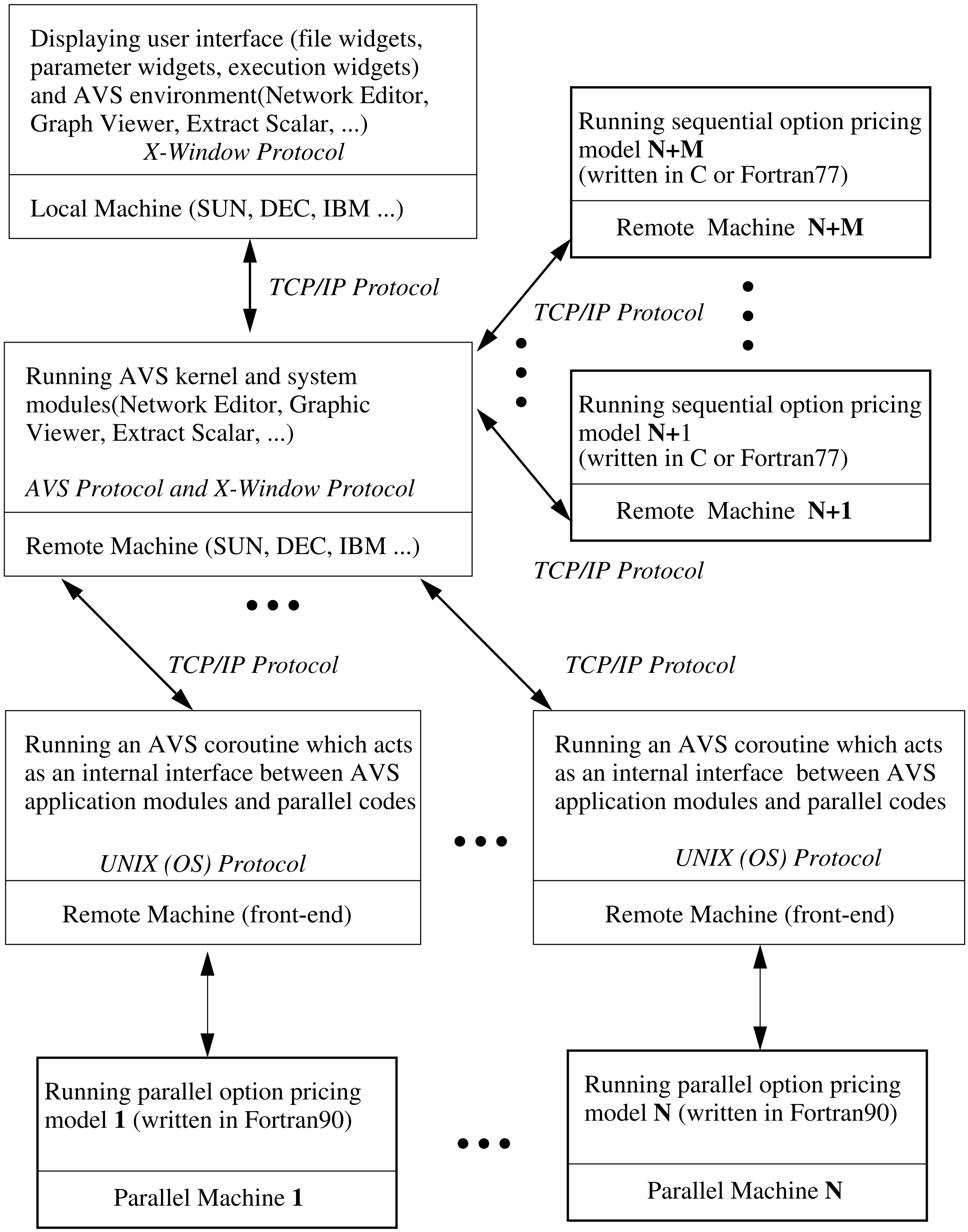

Figure 4 A complete distributed environment for a large application problem requiring scientific data visualization

5. Discussion and Further Work

In this study, we choose DECmpp-12000 to be the parallel machine running the option price models. As it becomes obvious from the above discussion, it will be very easy to port this tool to any other parallel machines. AVS and Fortran90 help us to achieve this goal. It is not impossible that using AVS we can achieve a complete high level heterogeneous distributed computing environment for a large application problem which requires data visualization, as illustrated in Figure 4, where each option pricing model can be thought as a relatively independent component in a large application problem domain.

But, as found in the practice of this study, the current parallel programming environment is far from flexible and complete to facilitate such a computing environment. Originally, our option price codes are implemented in Fortran-90 (MP-fortran)[4]. The DECmpp-12000 programming environment does not support a mechanism to call MPfortran subroutines from a front-end C program or Fortran77 program at the front-end. It does provide a way to call MPL functions from C. MPL is a C-like programming language with parallel extensions on DECmpp-12000[5] which is not portable to other parallel machines. To integrate our model into AVS modules which is written in C, we have to rewrite all parallel models in MPL. This reduces the portability of this tool, though our models have Fortran90 versions. This also brings us another parallel programming environment issue that because so far the parallel programing environments on parallel machines are far less rich than those on sequential machines as their front-ends, there should be a flexible mechanism to provide the multi-language mixed programming environment, at least on the compiler level such that subroutines(functions) written in different programming languages can be called each other.

Because three more protocols, namely, AVS kernel, TCP/IP, X-window Manager, and of course UNIX, are involved in the visualization, there is a performance bottleneck in the network communication. The remote module producing the source visualizing data has to wait until the data is visualized on the display window over the network, which proves to be far slower than the computing model on DECmpp-12000.

Another difficult comes from the AVS. AVS is designed mainly to visualize 3-D data. The Graph Viewer in AVS is too simple to perform complicated tasks for 2-D data. In our application, it would be good to visualize and animate more than one model's data in a single window or in different windows to have a graphic comparison of different model's performance. But the AVS failed to provide such functionality, even in the latest release version 4.0.

Our further work would port this tool to CM2 or/and CM5, and finally implement the distributed environment shown in Figure 4 for at least DECmpp-12000, CM2 and CM5. Adding more functionality as required by the model optimization in the user interface can be done in the near future.

inputs, invoke model's parallel (and/or sequential) subroutines to generate model's call prices, and produce animating data for output which is a vector for the two models.

• a system module "Extract Scalar" ---- extract from the vector data generated by the coroutine a specified model's data for output.

• a system module "Graph Viewer" ---- in a separate window plot the output data in XY linear plots where X represents for the number(and time) sequence of tradings and Y for model's call price minus market's call price.

The module diagram(namely in AVS, a network) is shown in Figure 2.

Figure 3 is a picture of the overall screen of this tool's user interface, which contains a parameters control panel(left), a display window(upper right), AVS network editor and two shell windows(-lower right).

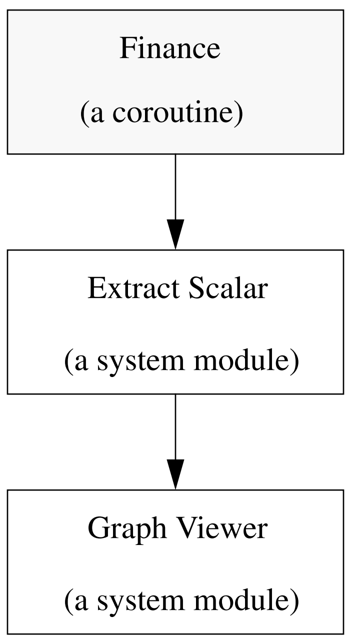

Figure 2 module diagram

As can be seen from Figure 1, AVS kernel acts as a coordinator to conduct data flows and control flows in the network. The main implementation tasks in the implementation are to have an internal interface between AVS modules and the model programs(subroutines) running on DECmpp-12000, and to design an user interface to define the interaction between tool user and the system.

In the user interface of the tool, two file browsers are used as graphic widgets for chossing file names of:

one file for storing primitive option pricing data, in which the basic input data of the models are contained and the other for initial values of variance of stock volatility and correlation between stock price and its volatility. They may be obtained through external ways or optimizations. The data formats of the two files can be found in Appendix.

Four dial widgets are designed for the 4 model parameters:

- variance of stock volatility, and
- correlation between stock price and its volatility

Their initial values are currently input from one of the files mentioned above. They can be changed anytime in the modeling.

- volatility for "Black Sholes Model", and
- volatility for "Stochastic Volatility American Model"

In the first half-hour of a trading day, they are calculated by an optimization function and can be changed at anytime of the next 6 trading half hours of that day.

Another two "oneshot" widgets are designed to allow user:

- to temporarily pause the modeling process to have a detailed view of the animated pictures or change model parameters;
- to execute the modeling in the way of single-step.

The basic data of the option price models are simple 2-D data. We use the Graph Viewer in AVS as the display tool. A simple (high-level) animation technique is used to show the dynamic evolution of the model price vs. market price over the successive trades and trading time. In the current implementation, two models, "Black-Scholes Model" and "Stochastic Volatility American Model", are contained in this tool.

An AVS coroutine is used as the internal interface between AVS modules and the model programs on DECmpp-12000. This coroutine controls the data flow between the DECmpp-12000's front-end (a DECstation 5000) and the DPU, and also the data flow between the user interface and the system components which are composed of:

- the same coroutine called "Finance" on the front-end ---- read the primitive stock data, calculate initial volatilities for the first half hour of a trading day, gather the runtime user model parameter

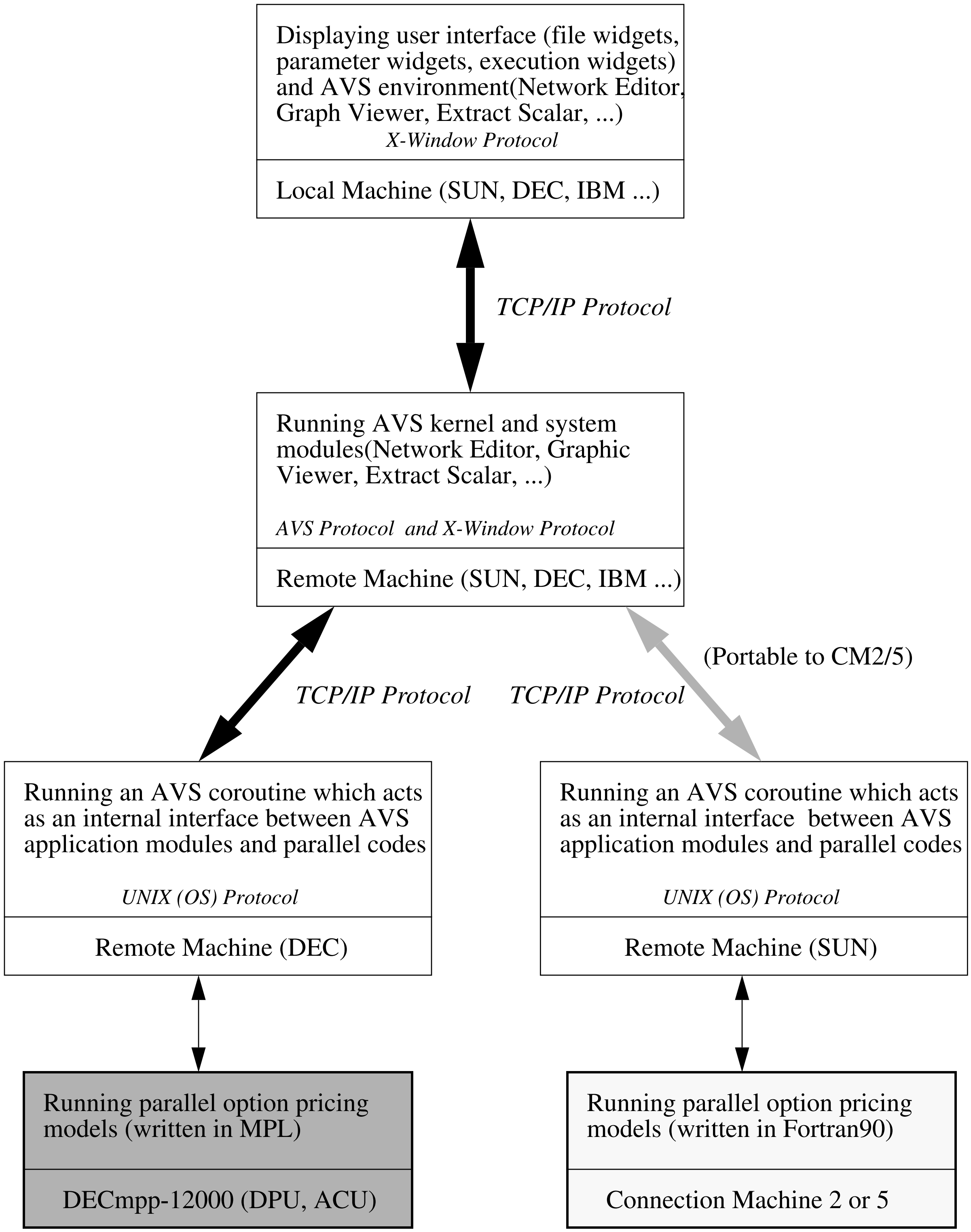

Figure 1 A distributed environment executing the visualization tool

Fortran) which performs a specific visualization function or set of functions. More than 110 (system) modules are provided with standard AVS. External programs or subroutines written in C or Fortran cab be easily converted into AVS modules.

AVS was designed from the outset to operate in distribute, multi-vendor, heterogeneous computing environments. The support for remote module execution allows modules to execute on computational servers in the network. X-terminal support allows AVS to display on 8-bit color X-terminals or workstations running the X-Window system. These distributed capabilities enable the user to incorporate AVS into the networked, distributed computing model already established today. This makes our models running on DECmpp-12000 easily to be combined into the AVS environment.

For detailed functionality about AVS, refer to [3].

4. Integrating the Option Price Modeling on DECmpp-12000 into AVS

There are two alternatives for data visualizations on parallel machines. One is to use the display or image processing library provided by the parallel machine's programming environment, such as MPDDL on DECmpp-12000, CMX11 on CM5. This can be very efficient as they fetch the data to be visualized directly from each processor element but it generally requires particular hardware like framebuffer which can not be shared by other users. The other way is to make use of the well-developed conventional graphics environment such as X-window Toolkit, OSF/Motif, AVS on sequential machines which are usually connected to the parallel machines as front ends.

Based on the following observations, we choose AVS as front-end visualization environment to implement our tool:

The characteristics of the option price modeling on DECmpp-12000 requires not only scientific data visualization but also dynamic interaction with the modeling itself. The MPDDL on DECmpp-12000 provides very limited facilities to support the graphic interaction with the running program.

Our tool is designed to be portable to any parallel machines which can run the option price models. AVS is now available on the major UNIX-based workstation and visualization systems. These include computer systems from Convex Computer Corporation, Cray Research Inc., Digital Equipment Corporation, Evans and Sutherland, Hewlett-Packard, IBM Corporation, Kubota Pacific Computer Inc., Set Technology Corporation, Sun Microsystems, and Wavetracer.

Our data visualization does not necessarily require fetching data to be visualized directly from PE nodes. So there would be no significant performance problem to display the scalar (2D) data from the front-end.

In order to illustrate the portability and extensibility of this tool, and also to overcome an unknown AVS system problem (probably a memory problem) that the repeated invocations of the Graph Viewer on the same machine as the invoking module will lead to system crash, we use a distributed environment to implement the tool. Figure 1 shows the configuration of this environment.

while allowing model parameters either estimated by some optimization method(function) or given by interactively parameter input at run-time, or by both.

Through AVS(Application Visualization System), this tool integrates the parallel option pricing models into an interactive, user-friendly graphic interface and provide the model users and researchers with a dynamic visualization for model verification and optimization.

This work also investigates the feasibility to use AVS as a data visualization tool in parallel programming environment. It further suggests that a complete, high-level distributed computing environment with a comprehensive visualization environment can be achieved through AVS, in which heterogeneous parallel machines and workstations may be integrated all together to accomplish a large application problem which requires scientific data visualization.

This paper discusses the design issues and implementation details of this tool.

2. Option Pricing Models on DECmpp-12000

The parallel option pricing models in our study are binomial approximation models which incorporates stochastic volatility and American call[1]. Another two sequential models assuming constant volatility are used for the purpose of comparison. After a large scale comparison of these models with historical market data[2], we found that a few key model parameters have significant impact on the model performance of (parallel) stochastic models. These parameters are:

- stock initial volatility
- variance of stock volatility
- correlation between stock price and its volatility

The determinations of the parameters are based on many factors in real stock market situations. They are generally unable to be observed directly but must be estimated either from historical data or by financial expertise. They are the major concerns in the model optimization. The model optimization requires more computing time than the model themselves, even on parallel machines. In the comparison of the models running on DECMPP-12000, we use some criteria, most of them are plain data, to evaluate the model performance of a whole stock over a 1-month or 6-month time. In the run of the comparison, we are unable to interact with the modeling process and it will take relatively long time to visualize a parameter’s impact on the model performance, which is done statically, due to the lack of an interactive visualization tool.

3. The AVS environment

Advanced Visual System’s Application Visualization System(AVS) is a comprehensive visualization environment which incorporates the latest advances in visualization software, graphics, networking, high performance systems, and industry standards.

Modules are the building blocks of AVS. A module is an independent computing element (C or

An Interactive Graphic Tool For The Option Price Modeling On DECmpp-12000 Using AVS

Gang Cheng Geoffrey Fox

Northeast Parallel Architrctures Center and School of Computer and Information Science

Syracuse University
Syracuse, New York 13244-4100
315-443-4889
gcheng@nova.npac.syr.edu

Abstract

An interactive graphic tool has been implemented for the option price modeling on DECmpp-12000. Through the use of AVS, it integrates the parallel option pricing models on DECmpp-12000 into an interactive, user-friendly graphic interface which visualizes the modeling process dynamically while allowing model parameters input at run-time by the user.

This paper discusses design issues and implementation of this tool. The feasibility of a distributed parallel computing environment with a comprehensive data visualization environment is also investigated.

1. Introduction

A set of (parallel) option pricing models have been implemented, which can run efficiently on parallel machines such as DECmpp-12000, CM2 and CM5. After a large-scale comparison of the option price models on DECmpp-12000 using historical market data[2], we evaluated that a group of models, namely, "stochastic volatility model", provide more accurate option price estimates than another group of "conventional" models. We also found that the "stochastic volatility models" are highly sensitive to a couple of model parameters, such as volatility of stock price and variance of the volatility. Those parameters can not be observed directly and can only be estimated either from historical data through some optimization method or from financial expertise.

To further evaluate and optimize the models, a visualizing tool for interacting with the models running on parallel machines would be very useful. An interactive graphic tool for the option price modeling is then implemented on DECmpp-12000 in this study. It visualizes the modeling process

Appendix

1. User directions

(To be appended)

2. Source Code and stock data formats of input files

(To be appended)

6. Conclusions

This work shows to us that:

• The importance to make use of existing(well-developed) visualization environment in the parallel programming environment;

• A distributed parallel computing environment with scientific data visualization can be achieved through the use of AVS;

• A highly portable interactive graphic tool for the option price modeling on parallel computers such as DECmpp-12000 can be implemented with relatively small programming effort, provided that AVS be the visualization environment and the parallel programming environment supports multi-paradigm programmings at compiler level.

Acknowledgments

The author is grateful to Kim Mills for his encouragement and continuous support during this study. I want to thank Marek Podgorny and Tomasz Haupt for the useful discussions about AVS. I also want to thank Peter Crockett for the system support.

This work is partially supported by a Corporate Partnership funding from Digital Eqipment Corporation.

References

[1]. T. Finucane, “Binomial Approximations of American Call Option Prices with Stochastic Volatilities,” to be published in *Journal of Finance.* 1992.

[2] K. Mills, M. Vinson, G. Cheng, T. Finucane, “A Large Scale Comparison of Option pricing Models with Historical Market Data,” to be appeared in *the 4th Symposium on the Frontiers of Massively Parallel Computation*, Oct. 19-21, 1992, McLean, Virginia.

[3]. Stardent Computer Inc. *Application Visualization System*, Volume 1 and 2, Release 3.0, April 1991.

[4]. K. Mills, G. Cheng, M. Vinson, etc. “Load Balancing, Communication, and Performance of a Stock option Pricing Model on the Connection machine-2 and DECmpp-12000,”, *Syracuse Center for Computational Science-237*, 22 pps. 1992.

[5]. MasPar Computer Corporation. *MasPar Parallel Application Language(MPL) User Guide and Reference Manual*, Version 2.1, Revision A5, 1991.

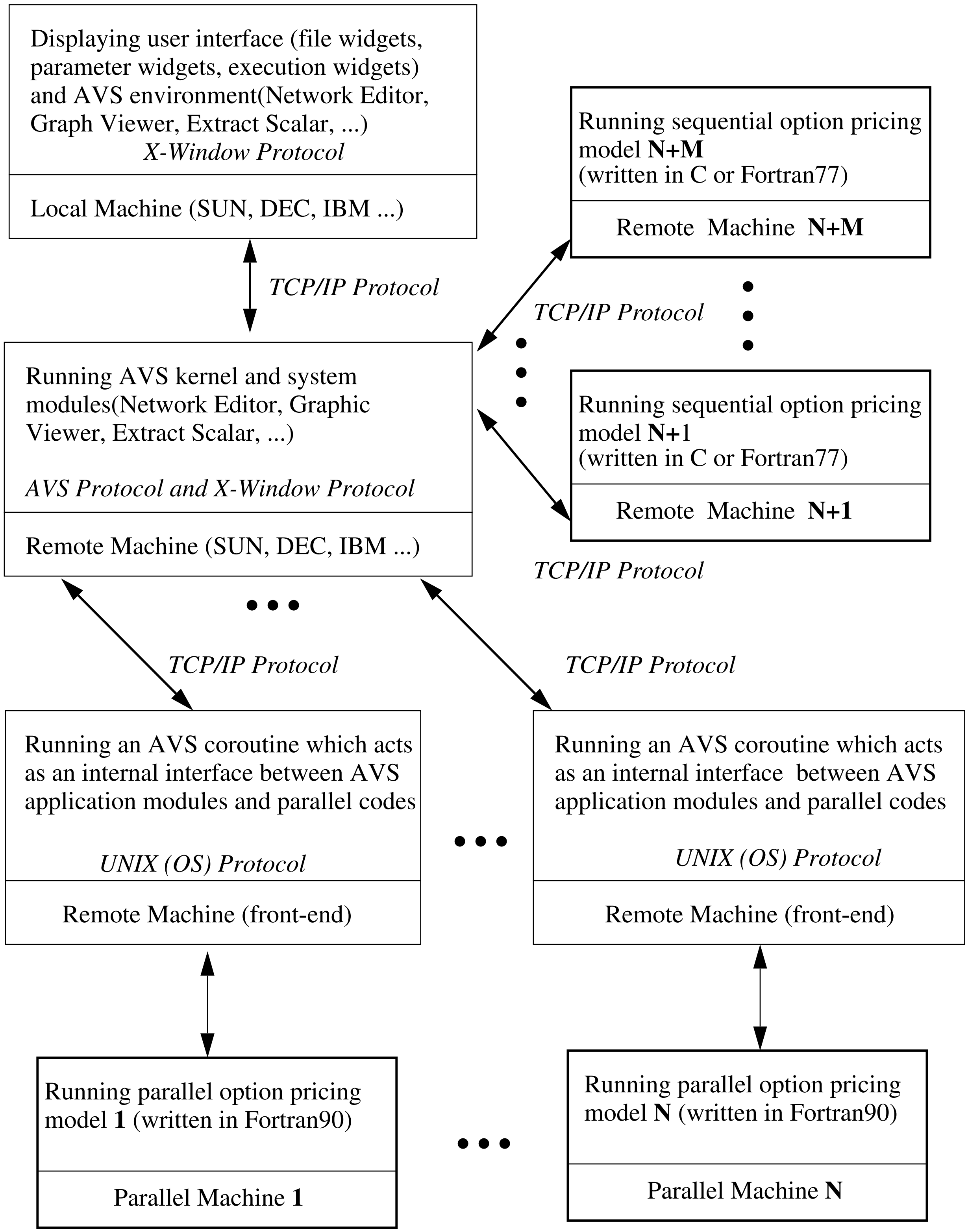

Figure 4 A complete distributed environment for a large application problem requiring scientific data visualization

5. Discussion and Further Work

In this study, we choose DECmpp-12000 to be the parallel machine running the option price models. As it becomes obvious from the above discussion, it will be very easy to port this tool to any other parallel machines. AVS and Fortran90 help us to achieve this goal. It is not impossible that using AVS we can achieve a complete high level heterogeneous distributed computing environment for a large application problem which requires data visualization, as illustrated in Figure 4, where each option pricing model can be thought as a relatively independent component in a large application problem domain.

But, as found in the practice of this study, the current parallel programming environment is far from flexible and complete to facilitate such a computing environment. Originally, our option price codes are implemented in Fortran-90 (MP-fortran)[4]. The DECmpp-12000 programming environment does not support a mechanism to call MPfortran subroutines from a front-end C program or Fortran77 program at the front-end. It does provide a way to call MPL functions from C. MPL is a C-like programming language with parallel extensions on DECmpp-12000[5] which is not portable to other parallel machines. To integrate our model into AVS modules which is written in C, we have to rewrite all parallel models in MPL. This reduces the portability of this tool, though our models have Fortran90 versions. This also brings us another parallel programming environment issue that because so far the parallel programing environments on parallel machines are far less rich than those on sequential machines as their front-ends, there should be a flexible mechanism to provide the multi-language mixed programming environment, at least on the compiler level such that subroutines(functions) written in different programming languages can be called each other.

Because three more protocols, namely, AVS kernel, TCP/IP, X-window Manager, and of course UNIX, are involved in the visualization, there is a performance bottleneck in the network communication. The remote module producing the source visualizing data has to wait until the data is visualized on the display window over the network, which proves to be far slower than the computing model on DECmpp-12000.

Another difficult comes from the AVS. AVS is designed mainly to visualize 3-D data. The Graph Viewer in AVS is too simple to perform complicated tasks for 2-D data. In our application, it would be good to visualize and animate more than one model's data in a single window or in different windows to have a graphic comparison of different model's performance. But the AVS failed to provide such functionality, even in the latest release version 4.0.

Our further work would port this tool to CM2 or/and CM5, and finally implement the distributed environment shown in Figure 4 for at least DECmpp-12000, CM2 and CM5. Adding more functionality as required by the model optimization in the user interface can be done in the near future.

inputs, invoke model's parallel (and/or sequential) subroutines to generate model's call prices, and produce animating data for output which is a vector for the two models.

• a system module "Extract Scalar" ---- extract from the vector data generated by the coroutine a specified model's data for output.

• a system module "Graph Viewer" ---- in a separate window plot the output data in XY linear plots where X represents for the number(and time) sequence of tradings and Y for model's call price minus market's call price.

The module diagram(namely in AVS, a network) is shown in Figure 2.

Figure 3 is a picture of the overall screen of this tool's user interface, which contains a parameters control panel(left), a display window(upper right), AVS network editor and two shell windows(-lower right).

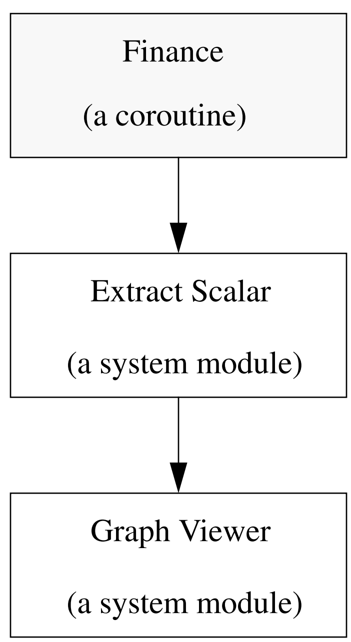

Figure 2 module diagram

As can be seen from Figure 1, AVS kernel acts as a coordinator to conduct data flows and control flows in the network. The main implementation tasks in the implementation are to have an internal interface between AVS modules and the model programs(subroutines) running on DECmpp-12000, and to design an user interface to define the interaction between tool user and the system.

In the user interface of the tool, two file browsers are used as graphic widgets for chossing file names of:

one file for storing primitive option pricing data, in which the basic input data of the models are contained and the other for initial values of variance of stock volatility and correlation between stock price and its volatility. They may be obtained through external ways or optimizations. The data formats of the two files can be found in Appendix.

Four dial widgets are designed for the 4 model parameters:

- variance of stock volatility, and
- correlation between stock price and its volatility

Their initial values are currently input from one of the files mentioned above. They can be changed anytime in the modeling.

- volatility for “Black Sholes Model”, and
- volatility for “Stochastic Volatility American Model”

In the first half-hour of a trading day, they are calculated by an optimization function and can be changed at anytime of the next 6 trading half hours of that day.

Another two “oneshot” widgets are designed to allow user:

- to temporarily pause the modeling process to have a detailed view of the animated pictures or change model parameters;
- to execute the modeling in the way of single-step.

The basic data of the option price models are simple 2-D data. We use the Graph Viewer in AVS as the display tool. A simple (high-level) animation technique is used to show the dynamic evolution of the model price vs. market price over the successive trades and trading time. In the current implementation, two models, “Black-Scholes Model” and “Stochastic Volatility American Model”, are contained in this tool.

An AVS coroutine is used as the internal interface between AVS modules and the model programs on DECmpp-12000. This coroutine controls the data flow between the DECmpp-12000’s front-end (a DECstation 5000) and the DPU, and also the data flow between the user interface and the system components which are composed of:

- the same coroutine called “Finance” on the front-end ---- read the primitive stock data, calculate initial volatilities for the first half hour of a trading day, gather the runtime user model parameter

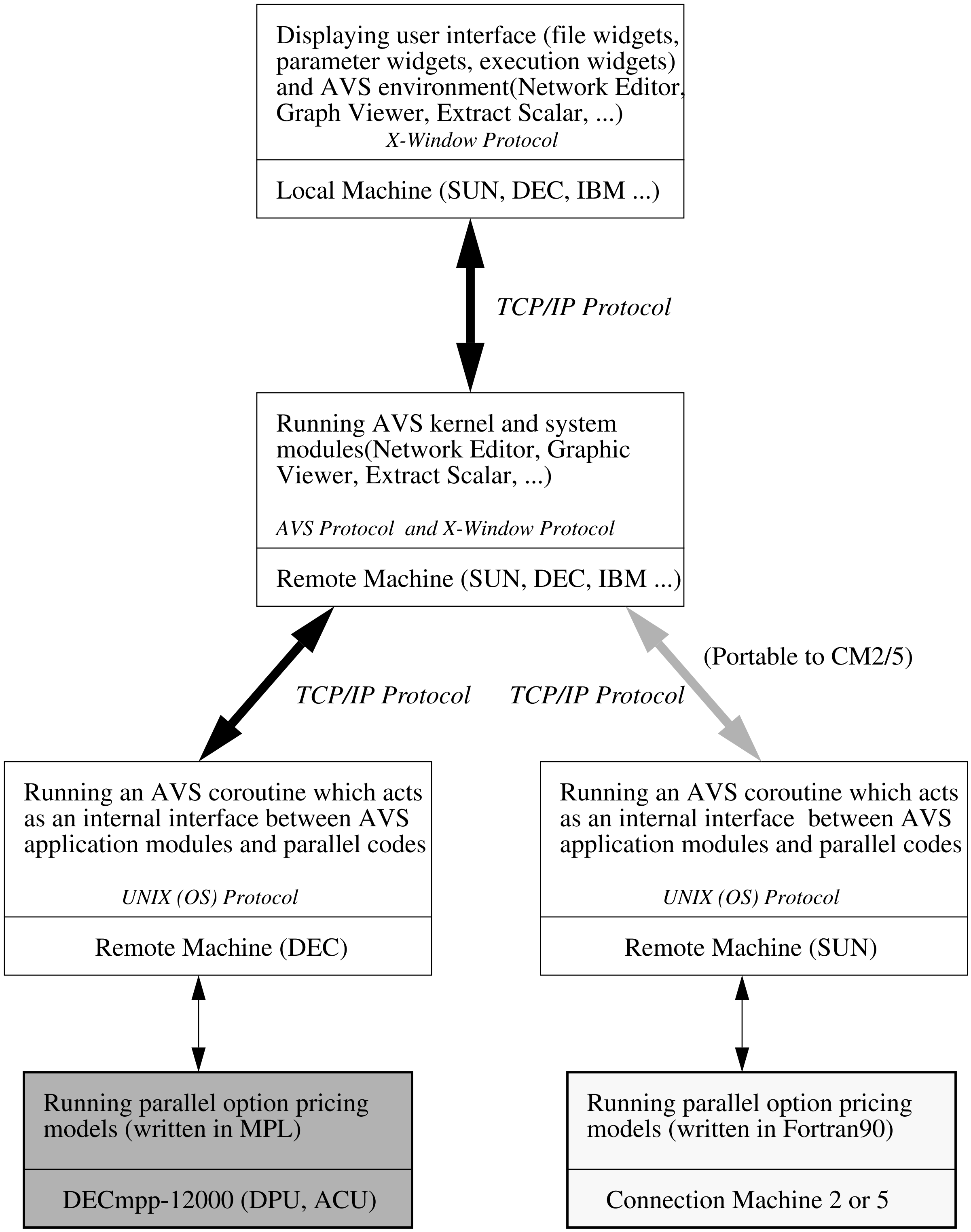

Figure 1 A distributed environment executing the visualization tool

Fortran) which performs a specific visualization function or set of functions. More than 110 (system) modules are provided with standard AVS. External programs or subroutines written in C or Fortran cab be easily converted into AVS modules.

AVS was designed from the outset to operate in distribute, multi-vendor, heterogeneous computing environments. The support for remote module execution allows modules to execute on computational servers in the network. X-terminal support allows AVS to display on 8-bit color X-terminals or workstations running the X-Window system. These distributed capabilities enable the user to incorporate AVS into the networked, distributed computing model already established today. This makes our models running on DECmpp-12000 easily to be combined into the AVS environment.

For detailed functionality about AVS, refer to [3].

4. Integrating the Option Price Modeling on DECmpp-12000 into AVS

There are two alternatives for data visualizations on parallel machines. One is to use the display or image processing library provided by the parallel machine's programming environment, such as MPDDL on DECmpp-12000, CMX11 on CM5. This can be very efficient as they fetch the data to be visualized directly from each processor element but it generally requires particular hardware like framebuffer which can not be shared by other users. The other way is to make use of the well-developed conventional graphics environment such as X-window Toolkit, OSF/Motif, AVS on sequential machines which are usually connected to the parallel machines as front ends.

Based on the following observations, we choose AVS as front-end visualization environment to implement our tool:

The characteristics of the option price modeling on DECmpp-12000 requires not only scientific data visualization but also dynamic interaction with the modeling itself. The MPDDL on DECmpp-12000 provides very limited facilities to support the graphic interaction with the running program.

Our tool is designed to be portable to any parallel machines which can run the option price models. AVS is now available on the major UNIX-based workstation and visualization systems. These include computer systems from Convex Computer Corporation, Cray Research Inc., Digital Equipment Corporation, Evans and Sutherland, Hewlett-Packard, IBM Corporation, Kubota Pacific Computer Inc., Set Technology Corporation, Sun Microsystems, and Wavetracer.

Our data visualization does not necessarily require fetching data to be visualized directly from PE nodes. So there would be no significant performance problem to display the scalar (2D) data from the front-end.

In order to illustrate the portability and extensibility of this tool, and also to overcome an unknown AVS system problem (probably a memory problem) that the repeated invocations of the Graph Viewer on the same machine as the invoking module will lead to system crash, we use a distributed environment to implement the tool. Figure 1 shows the configuration of this environment.

while allowing model parameters either estimated by some optimization method(function) or given by interactively parameter input at run-time, or by both.

Through AVS(Application Visualization System), this tool integrates the parallel option pricing models into an interactive, user-friendly graphic interface and provide the model users and researchers with a dynamic visualization for model verification and optimization.

This work also investigates the feasibility to use AVS as a data visualization tool in parallel programming environment. It further suggests that a complete, high-level distributed computing environment with a comprehensive visualization environment can be achieved through AVS, in which heterogeneous parallel machines and workstations may be integrated all together to accomplish a large application problem which requires scientific data visualization.

This paper discusses the design issues and implementation details of this tool.

2. Option Pricing Models on DECmpp-12000

The parallel option pricing models in our study are binomial approximation models which incorporates stochastic volatility and American call[1]. Another two sequential models assuming constant volatility are used for the purpose of comparison. After a large scale comparison of these models with historical market data[2], we found that a few key model parameters have significant impact on the model performance of (parallel) stochastic models. These parameters are:

- stock initial volatility
- variance of stock volatility
- correlation between stock price and its volatility

The determinations of the parameters are based on many factors in real stock market situations. They are generally unable to be observed directly but must be estimated either from historical data or by financial expertise. They are the major concerns in the model optimization. The model optimization requires more computing time than the model themselves, even on parallel machines. In the comparison of the models running on DECMPP-12000, we use some criteria, most of them are plain data, to evaluate the model performance of a whole stock over a 1-month or 6-month time. In the run of the comparison, we are unable to interact with the modeling process and it will take relatively long time to visualize a parameter's impact on the model performance, which is done statically, due to the lack of an interactive visualization tool.

3. The AVS environment

Advanced Visual System's Application Visualization System(AVS) is a comprehensive visualization environment which incorporates the latest advances in visualization software, graphics, networking, high performance systems, and industry standards.

Modules are the building blocks of AVS. A module is an independent computing element (C or

An Interactive Graphic Tool For The Option Price Modeling On DECmpp-12000 Using AVS

Gang Cheng Geoffrey Fox

Northeast Parallel Architrctures Center and School of Computer and Information Science

Syracuse University
Syracuse, New York 13244-4100
315-443-4889
gcheng@nova.npac.syr.edu

Abstract

An interactive graphic tool has been implemented for the option price modeling on DECmpp-12000. Through the use of AVS, it integrates the parallel option pricing models on DECmpp-12000 into an interactive, user-friendly graphic interface which visualizes the modeling process dynamically while allowing model parameters input at run-time by the user.

This paper discusses design issues and implementation of this tool. The feasibility of a distributed parallel computing environment with a comprehensive data visualization environment is also investigated.

1. Introduction

A set of (parallel) option pricing models have been implemented, which can run efficiently on parallel machines such as DECmpp-12000, CM2 and CM5. After a large-scale comparison of the option price models on DECmpp-12000 using historical market data[2], we evaluated that a group of models, namely, "stochastic volatility model", provide more accurate option price estimates than another group of "conventional" models. We also found that the "stochastic volatility models" are highly sensitive to a couple of model parameters, such as volatility of stock price and variance of the volatility. Those parameters can not be observed directly and can only be estimated either from historical data through some optimization method or from financial expertise.

To further evaluate and optimize the models, a visualizing tool for interacting with the models running on parallel machines would be very useful. An interactive graphic tool for the option price modeling is then implemented on DECmpp-12000 in this study. It visualizes the modeling process