A Steering and Visualization Toolkit for Distributed Applications

By Cara Stein[1], Daniel Bennett[1], Paul A. Farrell[2], and Arden Ruttan[2]

[1]Math & Computer Science Department
Edinboro University of PA
215 Meadville St.
Edinboro, PA 16444
Ph: 1-814-732-1184
Fax: 1-184-732-1170
{cstein, dbennett}@edinboro.edu

[2]Computer Science Department
Kent State University
P.O. Box 5190
Kent, OH 44242
Ph: 1-330-672-9980
Fax: 1- 330-672-7824
{farrell, ruttan}@cs.kent.edu

Submitted to the 2006 International Conference on Parallel and Distributed Processing Techniques and Applications
To be presented by Cara Stein (corresponding author)

Abstract—Parallel and high performance computing has enabled great strides to be made in advancing science and solving large problems. However, this progress is limited by the lack of needed tools and the difficulty of programming and running parallel applications. Specifically, there is a lack of needed steering and visualization tools, which can be easily integrated in existing applications. This paper presents the design of a steering and visualization toolkit to address this deficit.

Keywords: distributed computing, steering, visualization

1. Introduction

Parallel computing and the evolution of computer hardware over the past several decades have enabled great strides to be made in many fields. Problems that were once too big to attack are now within reach [4]. Areas that have been impacted by these changes include human genomics, climate modeling, jet propulsion, device and semiconductor simulation, and cosmology [1][4]. However, parallel programming and even simply running parallel programs can be difficult tasks. Tools such as CUMULVS [5] have been written to help manage parallel programs as they run. However, current tools are insufficient [4]. Even with these tools, there is very little facility for a scientist to monitor or interact with a computation as it is running. A large simulation may take days or weeks to run, even on a powerful collection of computers; during that time, it would be helpful to be able to intermittently check the progress of the computation and/or examine the results currently obtained. Also, upon seeing these results in progress, the scientist may desire to change one or more parameters to modify the simulation. Currently, there is no easy way to monitor the experiment as it runs or to steer it by changing parameters. According to Ahalt, the lack of truly useful high performance computing tools is the biggest barrier to widespread industrial use of high performance computing [1]. This paper presents a steering and visualization toolkit to assist in addressing this deficit.

2. Background

The toolkit we propose uses CUMULVS to interact with a distributed application. CUMULVS was created as middleware to provide an interface for data extraction, checkpointing, visualization and steering. Checkpointing allows archiving of the state of the program to permit resuming the application should it terminate prematurely, visualization involves extracting data from the application and displaying it in some way; steering allows the scientist to modify parameters of the application as it runs. The user interface of CUMULVS ascertains from the application what fields are available for monitoring and steering; it then collects that data from the various processors handling the computations as the application runs [6]. The user interface also permits viewers and steering tools to connect to and disconnect from the application as it is running [7]. The current version of CUMULVS provides an internal sequence number to ensure that data gathered from multiple processors are from the same iteration without having to add explicit barriers to the computation [5]. CUMULVS provides a text-only viewer, an AVS-compatible viewer, and a TCL/TK viewer design specifically for particle-based simulations, but the viewing and steering is done using function calls [5]. We feel that an application-driven user interface, using a GUI or a command-line client, will be much easier to use, and allow a larger range of application scientists to benefit from the improved interaction with their applications. Our tools build on the interface provided by CUMULVS to create such an interface which is more convenient, simpler, and easier to use for the user who wishes to monitor a computation or modify parameters as it runs.

3. Steering and Visualization Tools

To facilitate monitoring the status of a computation in progress and modifying its parameters, we propose a set of clients and tools to allow the scientist to select variables to monitor and/or modify. For many of these tools, it makes sense to include a text-based version as well as a GUI version. Although it is easier to provide a powerful and intuitive user interface for a GUI tool, a text-based tool is more certain to be accessible by remote connection. Then if a scientist is away from the office and wants to check the status of a computation, he can simply telnet in and use the text-based tool.

3.1. Simple Viewer

As the name suggests, the Simple Viewer allows the user to select fields to be monitored, and displays the values of those fields. In the GUI version, the fields will appear in a list with checkboxes. When fields are selected from the list, those fields are displayed with their values in a separate window (see Figure 1).

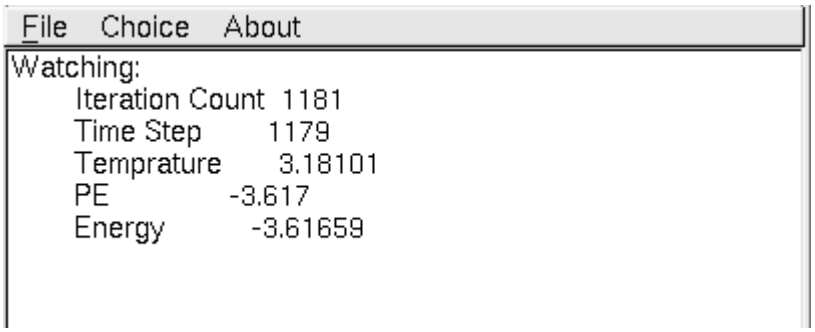

Figure 1: Simple Viewer – GUI version

In the text-based version, the fields to be monitored can be passed as command-line arguments. Then the fields and values are displayed in tabular form as plain text (see Figure 2).

```
[bennett@braveheart simp]$ scaview mdsd -p PE
Watching parameter named PE

CUMULVS Version 1.2.0 (Elwood)
Oak Ridge National Laboratory, Oak Ridge, TN
(C) 1996-1998 All Rights Reserved
CUMULVS was funded by the U.S. Department of Energy.

1732: PE -3.78096
1733: PE -3.78096
1734: PE -3.78096
1735: PE -3.78096
1736: PE -3.78096
1737: PE -3.78096
1738: PE -3.78096
1739: PE -3.78096
```

Figure 2: Simple Viewer – Text version

For both versions, the updating frequency can be controlled by the user, either in seconds or in iterations.

3.2. Graph Viewer

The graph viewer allows the user to see a graph of the value of one or more variables over time. For the GUI version, this will appear as a line graph showing the value at each specified number of iterations or amount of time. For the text-based version, asterisks will be used to draw bars of varying heights to represent the value of a variable at each specified number of iterations or amount of time. For both versions, the user will be able to specify how often to get a new value and how much history to show.

3.3. Stats Viewer

The stats viewer is an extension of the simple viewer. For the chosen field or fields, the stats viewer will show the user the current value, the maximum and minimum values seen so far for that field, and the iterations at which they occurred. It will also show the recent average value for the field, allowing the user to specify the number of iterations to use in the average calculation. This tool can be set to update automatically or upon request.

3.4. Iterations Per Second Viewer

This viewer displays the number of iterations being executed per second. In the text-based version, the viewer simply displays now many iterations were executed in the last second (or other time unit specified by the user). When the next second has passed, it displays how many iterations were executed in that second, and so on until the user quits.

In the GUI version, this information is displayed as a line graph showing the number of iterations executed in each second or other specified amount of time (see Figure 3). The number of iterations performed in the most recent time unit is also displayed numerically. The viewer has

controls to allow the user to specify the time unit to monitor, as well as a multiplier for applications that perform multiple updates per iteration.

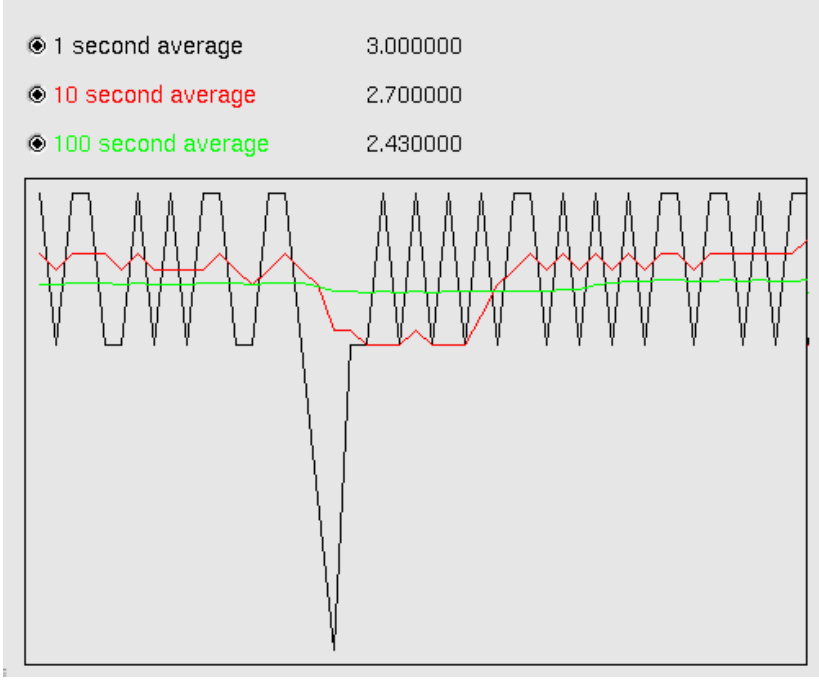

Figure 3: Iterations per Second Viewer

Furthermore, the viewer could also allow the user to watch a specific variable and count based on it instead of actual iterations. For example, if an application has three main parts and a state variable to track which part is currently executing, the user could see how many times per second the state variable changed, or how many times it was equal to a specified value.

The iterations per second viewer is useful for tracking the progress of a computation. For instance, for a molecular dynamics simulation code, using this tool we observed changes in the number of iterations per second depending on the number of molecules involved in the simulation, and also depending on the number processors involved in visualizing the simulation. This helps the user make intelligent decisions about the meta-parameters of how the computation will run. For example, the user can determine that the computation will take two extra days if processor time is spent displaying the simulation as it is run. The user can use this information to decide whether to display, or whether to leave the display off to get the final results two days sooner.

3.5. Alert Tool

The alert tool's purpose is to watch one or more variables and take action based on their values. Actions that can occur include flashing a messing on the screen, sending an email or text message, pausing the computation, halting the computation, or forcing a checkpoint. This way, if the user does not wish to continuously monitor the system during a long running computation, he or she can still be alerted if a significant event occurs, such as the computation entering an invalid range of values or when an algorithmically or scientifically important development

occurs. Examples of these include such developments as an iterative algorithm stalling, that is the error ceasing to decrease or decreasing at an unacceptably slow rate, or the nature of the solution changing, such as at a bifurcation point.

The most basic version of this tool could simply perform a predefined action when a particular variable is equal to a specific value. An improvement would be to add the ability for the user to enter any logical expression based on the values of fields in the computation, so that the system would perform the specified alert whenever the logical expression became true. Also, rules could be established so that different actions are taken based on the time of day. For example, the user could specify that if the alert condition occurs during normal business hours, the tool should pop up an alert on the screen, but if the condition occurs after hours, pause the computation.

3.6. Standard Steering Tool

The standard steering tool builds on the other tools' ability to display data, allowing the user to modify the values of fields used as parameters while the computation is running. For instance, for a weather simulation, the user might want to adjust the temperature and see what impact that would have on the simulation. In the standard steering tool, the field names and their current values are displayed, along with a text box or slider to allow the user to enter a new value. When the user enters a new value and clicks the set button for that field, the new value is entered into the simulation. Multiple fields could be changed at once using a "Set all" button.

3.7. Delta Tool

Although it is valuable to simply change the value of a variable, there may be instances in which it would be better to gradually increase or decrease the value of the variable to a specified point instead of changing it suddenly. For instance, in a weather simulation, it might be more realistic to gradually increase the temperature over many simulated hours, rather than instantly changing the temperature by 50 degrees. The delta tool allows the user to specify which variable to change, and how much to change it by per specified time period. This tool could include an interface similar to the standard steering tool's, showing a list of fields and their current values and allowing the user to fill in the amount to change by and the unit of time, with a set button for each field to make the changes take effect. A text-based version could simply take the field name, amount to change by, and time unit as command-line arguments.

3.8. VCR Tool

Whereas the standard steering tool allows the user to change the values of variables within the computation, the VCR controls the running of the computation. Like a VCR, it will have a pause button to allow the user to pause the computation and a play button to allow the user to run the computation. It will also have buttons to advance one iteration in the computation, or to move forward multiple iterations in the computation. One extension to this tool could be to add a rewind button – if checkpointing and backtracking are implemented, the rewind button could take the user back to the last checkpoint.

3.9. Manger Utility

The manager utility provides a convenient interface to allow the user to connect to an application and launch any of the other tools to monitor or steer that application. Most of the other tools in the toolkit use common information such as the application to be monitored or steered, the update frequency, and the names of all the fields. By having the manager oversee all of the other tools, the user doesn't have to specify the same information again for each tool he or she wants to use. For instance, the manager utility connects to the application, so the user doesn't have to specify the application information for each smaller tool within the toolkit. Similarly, the user can use the manager utility to set the update frequency for all of the other tools, so that they all run consistently without having to specify over and over again the time unit to be monitored. Also, the manager utility provides a convenient user interface from which to launch all of the other tools, so that the user can simply push a button and launch a tool, rather than having to try to remember what tools there are and how to launch them. It can also provide extensibility to the toolkit by providing a simple interface for adding new tools. An easy way to do this would be to keep a list, either in a flat file or in a database, of the tools available and how to launch them. When a new tool is created, its information could be added to the list, and then it would be available through the manager utility. This way, if a programmer needs to create an application-specific tool, he or she could easily customize the toolkit to include the necessary functionality.

In order for the manager utility to be capable of extension in order to enable the launching of new tools, it needs to know the interface for those tools. The cleanest way to accomplish this is to use a standard interface for all tools in the toolkit. In general, all tools could be launched by command line as follows:
[1]

tool_name application --frequency integer --paramcount integer --parameters name1 name2 …

Here, the tool_name is the name of the tool to be executed. The application is the system to be monitored or steered. The frequency is how often CUMULVS sends new information to the tool. The paramcount is the number of fields to monitor, and the parameters are the names of the fields to monitor. The frequency and parameters to monitor can also be set by configuration file or environment variable. These values can be set specifically for each tool, or they can use the values set in the manager utility as default values. Also, all GUI tools will start in a disconnected status to allow the user to modify parameters before connecting. Then the user will be able to connect to the application simply by pressing a "Connect" button. Alternately, any GUI tool can be launched with the --go command-line argument to connect immediately to the application.

4. Discussion

Some of the tools described have already been implemented; others are still in process. We have already received positive feedback for the iterations per second viewer, which has allowed us to quantify the difference in speed when using one machine to display our molecular dynamics simulation as opposed to using Chromium to distribute display tasks over several machines. We have done performance studies and found that the impact of the tools developed to date is negligible on the performance of simulations [2]. We feel that all of these tools will be useful for monitoring and steering distributed applications, no matter what the domain. We are

[1] This work has been supported in part by NSF ITR 0081324

also certain that using these tools will inspire the creation of more tools, some general-purpose and some application-specific. One possible extension to this work would be tools to deal with distributed data fields as opposed to scalar values.

We believe the flexibility of the manager utility provides added value by increasing the ease of use of the existing tools and by providing a convenient facility for the addition of new ones. As the proposed tools are created, they will be made available to the community using an open source repository. We hope that they will improve the ease of use involved in interacting with parallel applications, and thus help to lower the barriers to widespread use of high performance computing.

5. References

[1] Ahalt, S. "Towards a High Performance Computing Economy: Blue Collar Computing." Supercomputing 2004.

[2] Bennett, D., and P. Farrell. "Experiences instrumenting a distributed molecular dynamics program." Technical Report TR-KSU-CS-2006-04, Kent State University, Department of Computer Science, 2006. Submitted to *PACISE-06.*

[3] Chin, J., J. Harding, S. Jha, and P. V. Coveney. "Steering in computational science: Mesoscale modeling and simulation." *Contemporary Physics,* 44, 417-434, 2003.

[4] Dongarra, J., I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, A. White. *Sourcebook of Parallel Computing.* Morgan Kaufmann Publishers, New York, 2003.

[5] Geist, G. A. II, J. Kohl, and P. Papadopoulos. "CUMULVS: Providing fault tolerance, visualization, and steering of parallel applications." *The International Journal of High Performance Computing Applications,* 11(3): 24-235, 1997.

[6] Mulder, J., J. van Wijk, and R. van Liere. "A survey of computational steering environments." *Future Generation Computer Systems,* 13(6): 119-129, 1999.

[7] Papadopoulos, P., J. Kohl, and B. Semeraro. "CUMULVS: Extending a generic steering and visualization middleware for application fault-tolerance." *Proceedings of the 31st Hawaii International Conference on System Sciences,* Kona, Hawaii, January 1998.