Prototyping Simple Access to Visualization Resources

JOSEPH A. INSLEY AND MICHAEL E. PAPKA

Visualization is an essential component in the process of analyzing data. This data may be generated by a simulation or collected from sensors; either way the act of visualization turns the numbers into something understandable. The task of analysis is often as computationally complex as the process of actually generating the data. The advantage of using clusters to address this problem is that they have lots of computation and storage capacity. Unfortunately, they can be difficult to use effectively. The tools that are required can also be intimidating to users. Scientists want to be able to visualize their data via familiar interfaces, such as Web browsers and Windows applications.

Often a user makes use of local resources: a desktop or local advanced visualization facility. If more advanced resources are needed, however, the barrier to gaining access to such resources discourages their use. The focus of this month's article is simplifying access to advanced visualization hardware and applications.

The TeraGrid is a multiyear effort sponsored by the National Science Foundation to build and deploy a collection of very powerful clusters and specialized resources connected by a high-speed network and dedicated to open scientific research. The TeraGrid resources include the University of Chicago's (UC) 96-node advanced visualization cluster with accelerated graphics hardware. The goal of the UC's TeraGrid visualization (TGviz) team is to provide straightforward approaches for the analysis of data, from initial access to resources to custom domain-specific solutions. The recently introduced UC TGviz portal is the first such offering. The aim of this portal is to provide simplified access in a manner similar in scope to running an application on the user's desktop.

Issues

The first issue that we address is launching a visualization application on the TGviz cluster. Advanced computing clusters typically use a scheduling application (LSF, PBS, LoadLeveler) to determine when and on which nodes jobs will run. To run a job on a particular resource, the user must first describe the job in a way that the scheduler on that resource understands. This task requires specific knowledge about both the application to be run and the resource itself. While a researcher generally knows what he would like to see from his data, he may not know the resource requirements that a particular application may have in order to produce the desired visualization. Further, he likely doesn't care about the specifics of the resources used to generate the results, as long as the results are generated in a timely fashion. Ideally, he shouldn't need to know these details.

Second is the issue of remote access. This is related, in part, to the removal of details about resources. Unless all required interactions with the resource can be done without actually logging into it, these details cannot be entirely hidden. Hence, remote access is not only desirable but also necessary for truly simplifying the interface for the user. Once the resource details can be entirely hidden from the user, the system can take advantage of the resources most appropriate for the given task, without requiring any action by the user.

The third issue relates more to policy than to technical details. To use many advanced resources, users must first submit a request for an allocation. Typically, they write a proposal describing how they intend to use the resource and how many cycles they think it will require to accomplish their stated goal. This can be a time-consuming process, and is a huge barrier to many researchers. We intend to address this issue in the UC TGviz portal as well.

ParaView

As was stated earlier, we aim to make it easy for users to gain access to advanced visualization resources. One way that we do this is through the use of ParaView, an application that supports distributed computation models for processing and

visualizing large data sets. Users can take advantage of advanced compute and visualization resources by running computation and rendering processes in parallel and connecting to them via a client application running on their local desktop. To make this visualization application easily accessible to TeraGrid users and to utilize the power of the UC TGviz resources, we have developed several scripts and implemented other new technology as part of the TGviz portal.

Launching the Application

To run an application on the desired resource, users must submit a job request to the resource scheduler. The globusrun command enables users to do so, using the Resource Specification Language (RSL) to describe the job that they want to run. Using the GRAM (Grid Resource Allocation and Management) component of the Globus® Toolkit, one submits a job request to a Globus gatekeeper, which starts a job manager for this request. The job manager communicates with the local scheduler to run the job and maintains information about the job, including the current state.

Unfortunately, writing the RSL can often be a cumbersome task, in particular because it still requires knowledge about both the application and the resource. To simplify this task, we developed a helper script that takes as arguments parameters necessary for running the computation and rendering components (i.e., the server) of ParaView on the TGviz nodes. These parameters include the number of nodes and the duration that the application should run, which have been given default values. The script then generates the proper RSL. Since appropriate credentials are required in order to submit a job to the gatekeeper, the script also ensures that the user has a valid proxy certificate. If not, the script prompts the user to create a proxy, by executing the grid-proxy-init command, before proceeding. Next the script executes globusrun to submit the job request.

Many of the particulars for launching the ParaView server are hidden from the user. For example, the user's runtime environment must be configured properly, to include paths to necessary executables and dynamic libraries. The process for actually starting the executable in parallel on multiple TGviz nodes can also be somewhat complex. It requires information about which resources are being used, in addition to application-specific parameters. These details are all hidden from the user.

In order for the local ParaView client to connect to the ParaView server running on the TGviz nodes, it needs to know the host and port where the server is listening. This host information is determined by the local scheduler and cannot be resolved ahead of time. Therefore, after the script executes globusrun, it inspects information produced by the scheduler to determine when ParaView has actually started and the host where it is listening for the client to connect. It then returns to the user the command that can be used to start the client and connect to the server running on the TGviz nodes.

Remote Access via the TGviz Portal

While this script is a step in the right direction, in terms of making access to remote advanced visualization resources easier, it must be executed directly on the TGviz resource. The reason is that the scheduler information is not accessible remotely and requires the user to be logged into the TGviz resource.

This is where the TGviz portal comes in. To eliminate this requirement and further the ease of use, we used OGCE2 (Open Grid Computing Environment) technology to create a user portal, which is accessible via a Web browser. OGCE2 provides a MyProxy portlet for managing user credentials and a GridFTP portlet for managing data. It also enabled us to create a ParaView portlet for launching the ParaView server on the TGviz nodes. Once the user of the portal obtains his credentials, the ParaView portlet presents him with a simple interface. The user can specify how many nodes to run on and for how long. All of the other details, such as how to contact the gatekeeper or how to cancel the job request, are maintained by the portlet. The user simply presses the button labeled "Launch ParaView."

When this button is pressed, the portlet submits a job request to the appropriate gatekeeper to run the helper script. The script is then executed on the TGviz resource. As before, it executes globusrun on behalf of the user, determines the host information produced by the scheduler, and returns this information, this time to the portlet. Since the request to run ParaView still has to go through the scheduler, it may take some time before it actually starts. Once the request has been successfully submitted, control of the portal is returned to the user, where he can check the status of a request or cancel it if he chooses. When the ParaView server has started and the host information has been determined, the portlet will display the command that can be used to launch the ParaView client on the local resource. Thus, the TGviz portal provides remote access to the advanced compute and visualization resources of the TeraGrid, all from the comfort of a Web browser on the user's local machine. Ultimately the display of the ParaView client command will be replaced with a button. Pressing this button will confirm that ParaView is installed on the local machine or will install it if it is not already there, and start it with the appropriate information, connecting to the server running on the TGviz nodes.

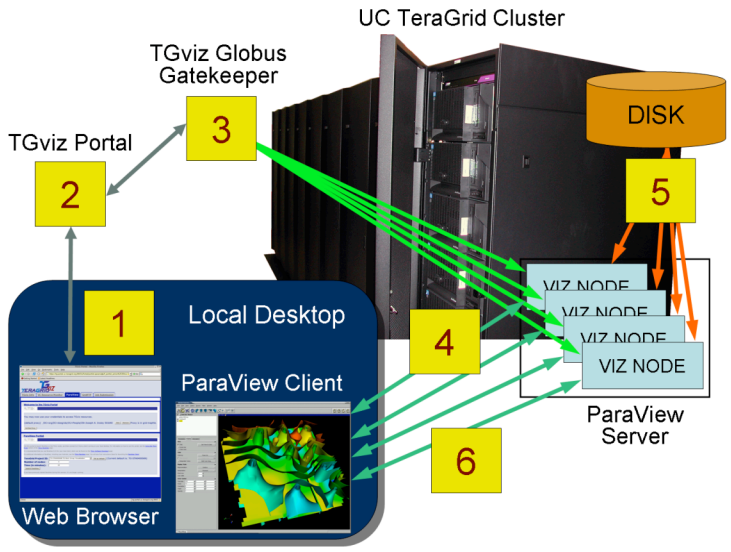

FIGURE 1 (1) From a Web browser on the local desktop a user loads his credentials into the TGviz portal. From there he submits a request to launch ParaView in server mode. (2) On behalf of the user, the portal submits the job request to the Globus gatekeeper on the UC TGviz resource. (3) The gatekeeper starts a job manager, which talks to the local scheduler to start ParaView on the TGviz nodes. Once ParaView has started, the appropriate information is relayed back through the portal to the user's Web browser. (4) Using this information, the ParaView client can be launched on the local desktop, connecting to the remote ParaView servers on TGviz nodes. (5) The ParaView servers read data from a parallel file system. (6) The local ParaView client is used to control computation and rendering done on the TGviz nodes, and displayed locally.

Future Development

TGviz portal is under active development, so some of the features proposed here may already be implemented by the time this article is published. For instance, the previous section mentions adding the ability to launch the ParaView client on the local resource directly from the portal. We hope to provide this capability rather quickly, using a combination of CoG Kit and Java Web Start technologies.

The current implementation of the TGviz portal relies on Pre-Web Services-based components of the Globus Toolkit. We plan to migrate to Web Services-based solutions in the near future.

Ideally, the host information, and potentially other information produced by the scheduler, would be included in the data that gets reported as part of a job. One could then obtain this information remotely by querying the GRAM service. This approach would remove the requirement of being physically on the TeraGrid resource to determine the host and would eliminate the need for the helper script altogether. Everything could then be handled in the portal. We anticipate that this capability will be made available in a future release of the Globus Toolkit, potentially GT 4.2.

Resources

TeraGrid
- www.teragrid.org

ParaView
- www.paraview.org

Resource Specification Language (RSL)
- www.globus.org/gram/rsl_spec1.html

The Globus Toolkit
- www.globus.org/toolkit/

Grid Resource and Allocation Management (GRAM)
- www-unix.globus.org/toolkit/docs/3.2/gram/prews/index.html

OGCE2
- www.collab-ogce.org

MyProxy
- grid.ncsa.uiuc.edu/myproxy/

GridFTP
- www.globus.org/datagrid/gridftp.html

CoG Kit
- www.cogkit.org

Java Web Start
- java.sun.com/products/javawebstart/

As it is, the TGviz portal provides access to specialized resources for existing TeraGrid users. However, as pointed out earlier, the user still must obtain an allocation on the resource. What about users who need short-term access but do not have the necessary time or resources needed to write a proposal to get an official allocation? For such users we envision providing access through the use of a "community" allocation. Users would be able to come to the portal, create an account for themselves, stage in their data, visualize it using TGviz resources, and save the results, staging the data back to their local resource. The account creation process would need to be automated, requiring no human in the loop, and would need a mechanism for ensuring that the system was not abused. It would include the dynamic creation of a user account on the TeraGrid resource, with a credential that gets mapped to the community allocation. Once the user logged into the portal, he would have only limited access to resources, but he would be able to accomplish real work.

How do we plan to provide all of these capabilities? That will have to be the topic of a future article.

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38, and in part by NSF under Grant No. SCI-0525310.

Joseph A. Insley (insley@mcs.anl.gov) is a senior software developer at Argonne National Laboratory, where he focuses on Grid computing, particularly in the area of visualization, and helps to lead the University of Chicago's TeraGrid effort.

Michael E. Papka (papka@mcs.anl.gov) is research manager for the Futures Laboratory at the University of Chicago/Argonne National Laboratory, where he focuses on large-scale visualization problems.