Enabling Community Access to TeraGrid Visualization Resources

Justin Binns,[a,b] Jonathan DiCarlo,[a] Joseph A. Insley,[a,b] Ti Leggett,[a,b]
Cory Lueninghoener,[a,b] John-Paul Navarro,[a,b] and Michael E. Papka[a,b]
[a]University of Chicago, [b]Argonne National Laboratory,
{insley, papka}@ci.uchicago.edu

Abstract

Visualization is an important part of the data analysis process. Many researchers, however, do not have access to the resources required to do visualization effectively for large datasets. This problem is illustrated through several user scenarios. To remedy this problem, we propose a Visualization Gateway that provides simplified access to such resources to a broad population of users. The current implementation of this gateway is described, including technology used and services made available. In particular, a detailed description of a ParaView portlet is included. A proposed design for enabling access to community users is discussed. Technology as well as policy issues that were raised, including security and data management, are covered. Methods for providing additional services, scaling to include additional resources, and other areas of future development are conveyed. The paper concludes with a summary of the topics covered.

Introduction

Visualization is recognized as an essential component in the process of analyzing data. Whether the data is generated by simulation or collected from sensors, the act of visualization can turn the data into something understandable. Yet the task of visualization is often as computationally complex as the process of actually generating the data. It requires resources beyond those available on a typical workstation, including both advanced compute and graphics capabilities.

Unfortunately, the barrier to gaining access to such resources is often too high for many researchers. Usually, one must submit a proposal of the work to be accomplished, with justification of the need to use the particular resource. In addition to being a competitive process, it can also be a lengthy one. Many resources award new allocations only once or twice per year, with the review process itself taking up to several months. We further illustrate the need for simplified access to advanced visualization resources with several user scenarios.

Scenario 1: Visualization experts often collaborate with researchers from various scientific communities. Occasionally they may require capabilities beyond those of the resources at their local institution. It may be difficult to predict when this need might arise or for which of their collaborators. What the researchers need is an advanced visualization resource that they can use for a relatively short period of time, with little advanced notification. These requirements are too vague to acquire allocations under typical guidelines.

Scenario 2: A computational scientist submits a proposal for an allocation to run a large-scale simulation. His proposal is approved, and he is awarded his requested number of CPU cycles. He uses these cycles to run his simulation and produces several terabytes of data. As is often the case with many researchers, when he determined the number of cycles he needed, he did not account for those required to do visualization and analysis of this data. He must now seek alternative means for doing this important step of the process.

Scenario 3: A professor is teaching an undergraduate course in climate physics. She would like her students to investigate the results from various climate models that they have been studying, but her university has limited visualization capabilities. Enabling her thirty students to remotely access advanced visualization services to perform these investigations would be an invaluable teaching tool.

In each of these cases there is a lack of access to high-end visualization resources. In order to provide access to such resources, specifically the University of Chicago (UC) TeraGrid Visualization (TGviz) cluster, to a broad population of users, we aim to develop strategies for enabling "community access." Virtually anyone with a valid email address will be able to create an account on the TGviz gateway and start using the resources. In our proposed approach we allow for two distinct types of users: those who have existing TeraGrid accounts, and those who do not (referred to as *community users* in this paper).

The TeraGrid [1] is a multiyear effort sponsored by the National Science Foundation to build and deploy a collection of advanced compute clusters and specialized resources connected by high-speed networks and dedicated to open scientific research. The TeraGrid resources include the University of Chicago's 96-node advanced visualization cluster with accelerated graphics hardware. While advanced graphics hardware has traditionally been accessible only via a local console, it is also possible to utilize this hardware remotely. Both batch and interactive rendering can be done locally using the graphics hardware and the resulting images sent over the wide area and displayed remotely on a user's desktop. Many applications, ParaView being one such example, can also make use of multiple nodes to do rendering and data transformations in parallel. Another thing that makes these resources special is that they have high-speed access to large datasets stored on the TeraGrid. These datasets need not be moved to a user's local institution, where they are not likely to have the capacity to store them, nor the bandwidth to transfer them at reasonable rates.

The goal of the UC's 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 gateway, available at http://tg-portal.uc.teragrid.org, is the first such offering.

The remainder of the paper is in three sections. We first explain the work that has been done on the TGviz gateway (portal) thus far, the capabilities that it currently has, and how it was implemented. Next we discuss the vision for moving forward: capabilities that

are being enabled, issues that were raised, and description of the design currently under development. Finally, we discuss future advancements.

Existing Implementation

The existing implementation of the TGviz gateway was developed over the course of about three months by a single developer with existing expertise in the area of Grid computing. Much of this time was spent becoming familiar with the different portal technologies that were available at the time. In particular OGCE2 [2] (Open Grid Computing Environment) and uPortal 2.4.1 [3], on which the portal is based, were investigated. While anyone is allowed to browse to the TGviz gateway (over SSL), only those with a valid X.509 proxy certificate are allowed to access resources. This policy, in effect, restricts access to existing TeraGrid users with current allocations. Like many Web portals the TGviz gateway uses a "tab" motif, similar to those in a filing cabinet, to group its content. Clicking on one of the labeled tabs displays a different set of services. The TGviz gateway has four sections, each contained on a separate tab. The first is the "TGviz Info" section, which is accessible by anyone visiting the portal and gives a brief overview of the TeraGrid project and the objectives of the TGviz gateway. The remaining three tabs contain portlets for accessing resources and are disabled until a valid proxy has been obtained. In order to facilitate the loading of such credentials, each tab includes a ProxyManager portlet, which is provided with OGCE2 and allows users to retrieve long-term proxies that have been stored in a MyProxy [4] server. In addition, in order to access the resource, the user's Distinguished Name (DN) from their certificate must also be in the local grid-mapfile on the UC TeraGrid resource. The ProxyManager portlet has thus been augmented to check for the user's DN in the local grid-mapfile and display the result in the portlet. If the DN is not found, a link to information on how to add it is also displayed.

Once a valid proxy has been loaded, the functionality of the remaining tabs is enabled. Two of these tabs, Data Management and Job Submission, contain additional OGCE2 portlets. The Data Management tab uses GridFTP [5] to enable users to transfer files between their local machine, where the Web browser that is displaying the portal is running, and a GridFTP server. While the user could enter the name of any GridFTP server, the default is that of the server on the UC TGviz cluster. In addition to local files, users can also use this portlet to transfer files between two GridFTP servers, provided that both servers accept their credential. The Job Submission tab uses the Java CoG Kit [6] to submit a job request to a Globus [7] resource manager, or gatekeeper. Again, the users could identify any gatekeeper that recognizes and accepts their credential, but the default is to use the gatekeeper on the UC TeraGrid cluster. The gatekeeper then starts a job manager, which is responsible for starting the job and maintaining its state. The user also specifies the job to be run, which could be any command entered at the shell prompt, along with any arguments that may be required.

The last tab is labeled ParaView. It contains a custom portlet developed to simplify the process of running ParaView on the TeraGrid. ParaView [8] is an application that supports distributed computation models for processing and visualizing large data sets. It

can be run in a client/server mode, enabling one to harness the advanced graphics capabilities of a visualization cluster, such as the UC TGviz nodes, from a local desktop client.

After the user has loaded his credentials, the portlet presents him with a simple interface where he can specify the number of nodes he would like to use, and for how long. Because the request to run the ParaView server will ultimately be submitted to the resource's scheduler, the user is required to specify a project ID, which maps to an allocation. Therefore, when the portlet is first loaded, it submits a job via a fork job manager, which does not require a project ID, to retrieve the list of projects that the user is associated with and presents the list in the portal for the user to choose from. The user can optionally set a default project ID through an interface in the portal. This will cause the default project to be preselected for them upon subsequent visits to the portal.

Starting the ParaView server on the TGviz nodes and then connecting to it from a local ParaView client is actually a complicated process, the details of which are described below and illustrated in Figure 1. From a Web browser on the client machine the user clicks the "Launch ParaView" button. This prompts the gateway to use Java CoG job submission to submit a job request to the Globus fork gatekeeper on the UC TeraGrid resource on the user's behalf, in much the same way as the Job Submission portlet. In order for the client to connect to the server, one must know the host and port where the server is listening. The server can be told what port to listen on, but the hosts that are used are determined by the resource's scheduler and cannot be determined ahead of time. Additionally, the server listens on the head node of the job, and this information is not currently available from the job manager that maintains the job.

Therefore, the job submitted by the portlet is a request to run a helper script. After generating a random port for the ParaView server to listen on, this script then submits a second job, this time to the PBS gatekeeper, that will actually start the ParaView server. It returns the JobID of this second request, which can be used to check the job's status or cancel it if so desired, to the gateway, which then returns control of the Web browser back to the user. Meanwhile, the helper script polls the scheduler directly in order to determine the head node of the job, where the server is listening. The executable of the second job, however, is not the ParaView server executable, but rather a wrapper script around it. Because ParaView requires a particular environment, one that the user is not likely to have by default, when the wrapper script is run on the TGviz nodes it first sets up the environment appropriately and then starts the ParaView server executable. Once the original helper script has confirmed that the server is up and running, it returns the hostname of the head node and the port that it generated to the gateway.

Luckily for the gateway user, all of these details are hidden behind the "Launch ParaView" button. Pressing the button in the portal will submit the job request that runs the helper script, as described above. The user can then check to see if the ParaView server has started by clicking the "Check Status" button in the portal. This causes the gateway to check whether it has been notified of the ParaView server's host and port. Once the gateway has this information, it creates a file with a MIME type of

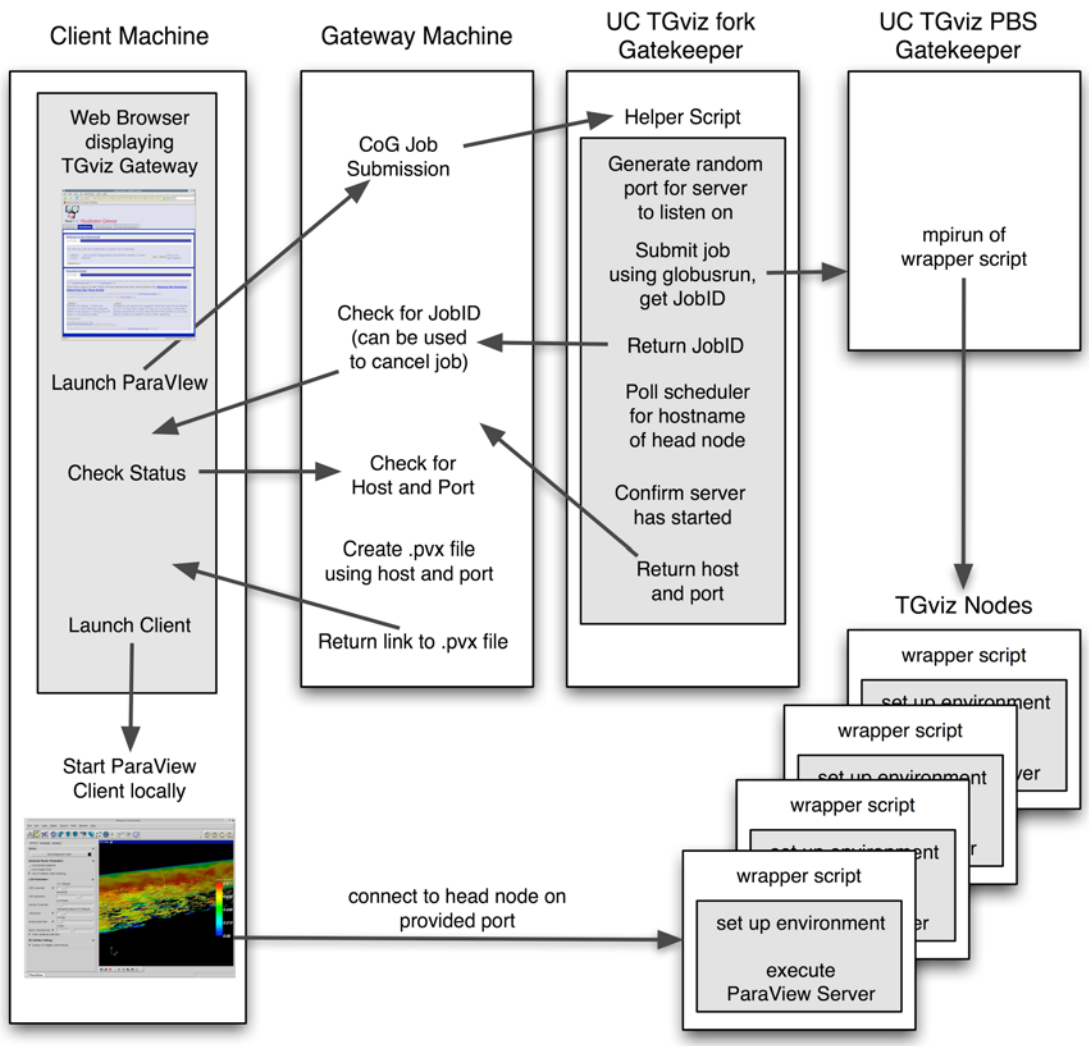

Figure 1: Diagram illustrating the details of the ParaView Portlet

application/paraview, an extension of .pvx, and provides the user with a link to it. Clicking on this link, labeled "Launch Client" in the user's Web browser, will cause the ParaView client to be launched on the local resource and connect to the ParaView server running on the TGviz nodes. Prior to launching the ParaView server from the portal, the ParaView client must be installed locally. The TGviz gateway provides links to binary versions of the client for Linux, Windows, and Mac OS, as well as information on how to install it and set up the application/paraview MIME type in the user's Web browser. Thus, after an initial one-time setup, the process for starting ParaView on the TeraGrid has been reduced to a few clicks of the mouse.

ParaView does not currently use authentication between client and server. This is recognized as a potential security risk, and should be noted as such to users of the portal. However, the server listens on a randomly generated port on one of the 96 nodes of the cluster, chosen by the scheduler at the time it is launched. Only one client can connect to the server, and the user that launched the server generally connects to it within a small number of seconds. In the unlikely event that the real client is unable to connect to the

server, because another client has already connected, the user can terminate the server from within the portal. Also, because the server is run as the user, if it is hijacked it could not be used to gain escalated privileges. It would be possible to add authentication to the client/server connection, and because the portal is used to launch both the client and the server, the credentials stored in the portal could be used for authentication. This may be considered for one of the areas of future work.

Moving Forward

While the existing TGviz gateway now provides simplified access to advanced visualization resources, it still requires users to have existing TeraGrid accounts and allocations. To provide such opportunities to a much broader set of users, we propose to develop a gateway that allows for access by community users. As stated in the introduction, our goal is to enable virtually anyone with a valid email address to create an account on the gateway and begin using the resources. The usage would then be charged against a community allocation, rather than that of an individual.

Issues. In developing a design for such a gateway we uncovered many issues that will need to be addressed. The first is that of policy. What restrictions, if any, should be made on who will be allowed to create an account on the gateway? Should it be based on the domain of the user's email address? For instance, perhaps only users with .edu and .gov addresses should be permitted to create accounts. How long should these accounts be valid? Again, perhaps this could be regulated based on email address. Rather than denying access by those with email addresses that are not .edu or .gov, gateway accounts could be granted, but with a shorter lifetime. There are also the issues of quotas and priority. What restrictions should be made on the amount of storage and CPU usage that users are allowed to utilize? Should jobs of users with TG accounts and allocations be given priority over those of community users? Most likely the answer to that question will be "yes," although this will raise questions about implementation, especially when expanding to include additional TG sites. Since different sites use different scheduling mechanisms, the priority policy and implementation issues will need to be addressed on a per-site basis.

Perhaps the most important issues to arise are those of security. It will be important to ensure the security of both the resources that are exposed, as well as the data of all of its users. These include not only those users who are accessing the resources via the gateway but also the more traditional users who log into the resource directly. What authentication and authorization requirements should be employed?

The next major issue to be identified is that of data management. How will these community users move data on and off the TeraGrid? As we describe in greater detail in the next section, community users will be issued credentials that are recognized only by the UC TeraGrid resources. Specifically, to transfer data to the UC TeraGrid GridFTP server from one outside the UC TeraGrid, or vice versa, users will need to use multiple credentials. They will need to specify one certificate for the source of the transfer, and a different one for the destination. To enable this capability we extended the GridFTP

portlet provided by OGCE2. The ProxyManager portlet already allowed multiple proxies to be loaded, but only one of them could be set as the default. The original interface of the GridFTP portlet then used this default proxy for both the source and destination of any transfers. In our augmented GridFTP portlet the user can choose from any of the loaded proxies to use for each of the servers involved in a transfer, independent of one another.

Another issue that was identified is rooted in the implementation details and concerns user management within the portal. The current TGviz gateway was implemented using uPortal 2.4.1. This version of uPortal had limited capabilities for creating and managing user accounts, and the capabilities that it did have were neither documented nor supported. We looked into the possibility of upgrading to use uPortal 2.5 or 3.0. The user management capabilities in these versions were considerably improved and appeared to be well documented. Unfortunately, at the time of this writing, neither of these versions of uPortal was supported by OGCE2. At this point we took a closer look at using GridSphere [9] as our portlet hosting environment. GridSphere 2.0 also seemed to have sufficient user management capabilities, with ample documentation. GridSphere 2.0, however, is supported by OGCE2, which is why we chose to switch to it for the implementation of the community user gateway, which is currently under development.

As we move forward with the design and implementation of our community user gateway, we recognize that these, and other, issues will need to be addressed. We attempt to provide solutions for some, while others remain open questions. Further investigation and practical experience with such paradigms will be required in order to draw sensible conclusions.

Current Design and Implementation. Next we will describe the proposed design for our community gateway, the different components of which are currently in various stages of implementation. Because the goal is to provide access to a large volume of users, efforts will be made to automate the account creation procedure as much as possible. Ideally the entire process will complete without the need for a person to get involved at all. When a new user first arrives at the portal and requests to create an account, he will be presented with a form to fill out and submit. The form will request information such as name, email address, and preferred password. The user will also be asked about the science community to which he belongs. Since there are specific scientific communities that the gateway has proposed to enable, the user will be presented with a list of these, and potentially other, categories to choose from. Thus, the portal can track resource usage based on the community enabled. While the categories indicated by existing TG users could potentially be checked against those in their TG project descriptions, it is unlikely that there is a way to validate this information in an automated way for community users. This should be recognized when considering the accuracy of these usage statistics. The size of the data sets that the user expects to use will also be requested. This information may be used to determine resource requirements, which may be useful in configuring the user's environment or establishing quotas. Users will also be asked if they have an existing TeraGrid account; if so, they will be asked to provide their login name on the UC TeraGrid resource. This will be used to verify their TeraGrid

account, as these users will have increased functionality through the gateway. As the process is refined, additional information is also likely to be requested.

Once all of this information is collected, various account creation policies will be applied, such as requiring a .edu or .gov email address. If the requirements for account creation have been successfully met, then all of the user's information will be entered into a table in a MySQL [10] database. Additionally, the user's password is written to a temporary file and the location of this file and the user's email address are entered into a second table in the database, used specifically for account generation. This temporary file is secured by making it accessible only by the root user on a host that requires a one-time password. Once the account creation process is complete, typically within an hour, the password file is removed. The user's email address will then be verified before the gateway account creation will proceed. An email will be sent to the address that was supplied on the account request form, with instructions on how to reply to this email. Once the user's response has been received, the gateway account creation process will continue. If after some predetermined amount of time, on the order of three to seven days, the user has not yet responded, the information will be removed from the database, and the user will need to start the process over.

This account generation process is a way for us to get the gateway up and working for scientists to start using quickly. It is not an ideal method for managing identities and automatic certificate and proxy creation. Nor does it easily scale to running at other TeraGrid sites. Concurrently we are investigating the account management solution proposed by the TeraGrid's User Portal Working Group, namely, standing up our own Kerberos CA (KCA) [11] and leveraging the TeraGrid User Portal central KCA. Using this technology will make our system more secure and allow easier execution of jobs on remote TeraGrid sites. We expand on these possibilities in the section on Future Work.

As mentioned previously, we allow for two distinct types of users: those who have existing TeraGrid accounts and those who don't. When the user's gateway account is created, the layout of the portal is based on whether the user has an existing TeraGrid account, as users with existing accounts will have increased functionality through the gateway. Specifically, existing TeraGrid users will have the ability to run arbitrary commands on TeraGrid resources, through the use of the Job Submission portlet. Meanwhile, community users will have access only to those services that are predefined and exposed through the gateway, such as the ParaView portlet. Once the gateway account is created, if it is for an existing TeraGrid user, the process is complete, and the user can start using the portal in much the same way as in the existing implementation described earlier. If, on the other hand, the account is for a community user, several more steps must first be completed. These steps are illustrated in Figure 2.

A backend process running on one of the management nodes of the UC TeraGrid cluster periodically checks the MySQL database for newly created gateway accounts that belong to community users. Specifically, it checks for an email address that has no Unix username associated with it, since accounts for existing TG users will have their usernames listed. This is shown in step 3 in Figure 2. When such an account is

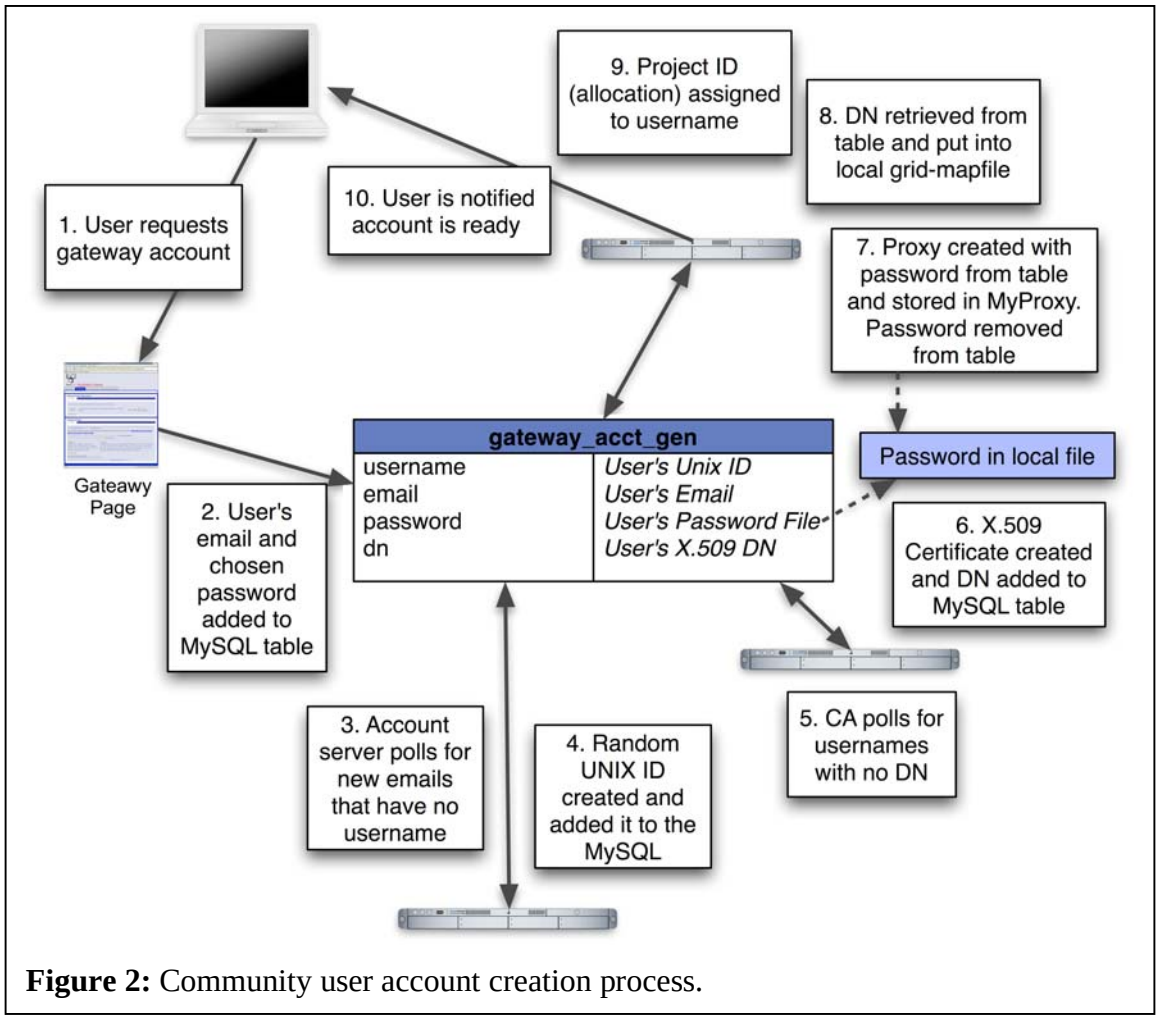

Figure 2: Community user account creation process.

discovered, a series of actions is initiated. First, in step 4, a randomly named Unix account is dynamically created on the UC TeraGrid cluster, and the name is entered into the account generation table in the database. The user will never have direct access to this account, which will have restricted capabilities, including no access to a login shell. It will be used only to enable the portal to access resources on the user's behalf. This local Unix account is necessary in order to submit jobs to the local resource scheduler. A single local account could have been used for all community users of the gateway, but having separate accounts simplifies usage accounting, and gives better control over user data access.

In addition to the Unix account, an X.509 certificate will also be required to access the TGviz resources. For obvious security reasons, it is unlikely that any of the existing certificate authorities accepted by the TeraGrid would issue a certificate to this virtually anonymous user. Therefore, the UC TeraGrid resource will run its own SimpleCA [12] server to issue credentials for this dynamically created user. Likewise, it is unlikely that the other TeraGrid resources would accept certificates generated by this SimpleCA, so these credentials will be valid only on the UC TeraGrid cluster. This is why it is necessary for the GridFTP portlet to support the use of multiple credentials: the gateway

user will be known by the UC TeraGrid cluster only as this dynamic user account and its credentials, while said account and credentials will not be recognized on any other resource. Another process on the management node, step 5 in Figure 2, polls the database checking for usernames that have no DN. When one is found, step 6, it creates and signs an X.509 certificate/key pair with the password the user provided. This certificate will be valid only for the length of time the user's account is valid, as regulated by our account policy. The certificate DN is then stored in the MySQL database for later use.

A long-term proxy for this newly created certificate is then generated in step 7. It is stored in a MyProxy server running locally on the UC TeraGrid cluster using the password from the MySQL database, which the user provided. At this point the temporary file containing the user's password can be deleted. Although other MyProxy servers are available on the TeraGrid, they are likely configured to only host proxies for certificates issued by CAs that they trust. Since it is expected that the UC SimpleCA will only be trusted by the UC TeraGrid resource, we must also run our own MyProxy server. This proxy will be valid only as long as the user's account; the user won't have direct access to his certificate to renew the proxy when it expires.

The subject (i.e., the DN) for this certificate is retrieved from the MySQL database, step 8, and put into the grid-mapfile on the UC TeraGrid resource and associated with the dynamic Unix account created for this user. If the user later wishes to change his gateway password his existing certificate and proxy will be destroyed and removed from the system. New ones will be created, with a lifetime equal to that which was remaining from the original certificate. This new proxy is then stored in the MyProxy server using the new password. Finally, in step 9, the dynamic user account is assigned a project ID, which corresponds to a community user allocation. Email is then sent to the user, notifying him that an account has been created. When the user returns to the gateway and logs in using the password chosen at the time of the account request, the portal uses it to automatically retrieve the proxy from the MyProxy server on the user's behalf. The functionality of the GridFTP and ParaView portlets are also enabled, and the user is ready to start using the resource.

Another backend process, which enforces the established policies, is periodically run. This process checks to see whether any of the gateway accounts has expired or exceeded established quotas. For security purposes, community user credentials will only be valid for a relatively short period of time, on the order of thirty days, depending on policy. If the user's account has expired appropriate action is taken to clean up the user's account. This includes such tasks as disabling the gateway login, cleaning up certificates and proxies, and removing the user's DN from the local grid-mapfile. Any data that the user has stored could remain intact for some predetermined amount of time, during which the user could renew their account, up to a maximum number of times, dictated by the established account policy. The process for renewing the user's account would consist of allowing the user to set a new gateway password, and creating new credentials, as was done in the initial account creation process. Once the maximum number of renewals is exceeded, all information, including any stored data would be removed, and the user would need to reapply for a new account.

Accounting for Usage. Because the TeraGrid is a nationally funded project, it must account for all of its usage. All resources used by community users of the gateway are charged against a community user allocation. Local accounting information can be used to track the usage of individual users, based on their dynamically created Unix account, and mapped back to a real user derived from the information in the MySQL database, if desired. Local accounting information and the database could also be used to determine usage based on scientific community, using the information gathered at the time of gateway account creation.

Additional Future Work

Many of the components for the community access TGviz gateway have been implemented and are in place, while others are still under development. As we continue to work toward finishing and deploying the completed gateway, we look ahead for ways of improving it and offering new capabilities. One way is to provide additional services by developing portal interfaces to more visualization applications, as was done for ParaView. While other applications have not yet been integrated, now that we have practical experience we expect that it should be straightforward to apply this approach to other client/server type applications. Other more batch-oriented applications could potentially be enabled in a similar fashion. Customizing portlet applications to provide services for enabling the following specific communities is also being targeted: atmospheric sciences, astrophysics, fluid dynamics, life sciences, nanotechnology, materials science, high-energy physics, and astronomy. The modular design of ParaView allows for the creation of custom clients that interface with this same ParaView server. Different customized ParaView clients could be tailored to these specific scientific domains, simplifying the visualization and data analysis process for the scientist. The mechanism for launching the ParaView server and client could be reused, virtually unchanged.

Because the dynamic user accounts and the credentials associated with them are recognized only by the UC TeraGrid resource, currently only the UC resources can be accessed. We are investigating ways in which these user accounts could be made more secure, such as through the use of the Kerberos CA mentioned earlier. Because a Kerberos CA does not require the user to have long lived certificates, the system could be made more secure by not having a valid proxy left in MyProxy. Proxies are generated and issued when the user authenticates and are valid for very short periods, typically 8-10 hours. Also, since a Kerberos CA does not require a UNIX account for authentication, it will further secure the RP site by not having UNIX accounts created for each of the gateway users. The goal is to satisfy the security requirements of additional TeraGrid sites and provide users with access to the diverse resources available at those sites.

This initial design to utilize local UC accounts was to allow for the quick development of a gateway that gives users access to UC TG resources, while in parallel investigating the best approach for expanding access to include resources at other TG sites. The TeraGrid has since started supporting community accounts, which allow for multiple users to

access resources through a single user account. These community accounts simplify the scalability issue, as they will be enabled TeraGrid-wide. Hence, using a community account with the gateway will provide access to resources across the TG. While simplifying some aspects of the gateway using the community account raises other issues, many which are as yet unresolved. For instance, securing the data of the individual gateway users who are utilizing the community account will need to be addressed. But the TG community as a whole will be investigating these issues, because most, if not all, Science Gateways on the TG will be adopting the use of community accounts. Because we will be able to then leverage from, and contribute to, the work of the rest of the community, it will be advantageous to switch to this model.

Also, there are likely to be cases where users have data that they would like to move onto the TeraGrid but that doesn't reside on a GridFTP server. Enabling transfers from additional types of storage facilities that recognize different types of credentials, such as ssh keys and Kerberos tickets, could be achieved through additional augmentation of the ProxyManager and GridFTP portlets. Offering access to a service such as the Reliable File Transfer Service (RFT) [13] would further enable users to transfer data reliably and asynchronously. This will be an important capability as the size of data sets increases, since it will become impractical for the user to stay connected to the gateway waiting for a transfer to complete.

Additional work could also be done in the area of accounting. In the implementation currently under development, there is no way to differentiate cycles that an existing TeraGrid user consumes via access through the gateway and those she utilizes through other means. Being able to distinguish between these two types of usage would be valuable while evaluating the effectiveness of the gateway to provide access to advanced resources. One possible approach to this would be to implement usage accounting within the gateway. The gateway already has the job ID of any jobs submitted through the gateway. These could be used to query the resource's local database for usage statistics associated with these jobs, which could then be stored in the gateway's database and later compared to the user's total usage on the resource.

Conclusion

We have discussed the lack of sufficient access to advanced visualization resources by scientific researchers, in particular those with large datasets. To address this problem a Visualization Gateway was proposed, which would make such resources available to a wide range of users. The current implementation of the gateway, which provides access to existing TeraGrid users, was described, including the components of the ParaView portlet. A design for a gateway that would also enable community users to access these resources was also described, along with technical and political issues that were raised. Finally, future directions, including a proposed switch to an approach that uses community accounts, were discussed.

Acknowledgements

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. ANI-01-22296.

References

[1] TeraGrid, www.teragrid.org
[2] OGCE2, www.collab-ogce.org
[3] uPortal, www.uportal.org
[4] MyProxy, grid.ncsa.uiuc.edu/myproxy
[5] W. Allcock, A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets**.** *Journal of Network and Computer Applications*, 23:187 - 200, 2001
[6] G. von Laszewski, I. Foster, J. Gawor, and P. Lane. A Java Commodity Grid Toolkit. *Concurrency: Practice and Experience*, 13, nos. 8 – 9, 2001, 643 - 662.
[7] I. Foster, C. Kesselman, Globus: A Toolkit-Based Grid Architecture. In Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 259 - 278; and http://www.globus.org.
[8] ParaView, www.paraview.org
[9] J. Novotny, M. Russell, and O. Wehrens, GridSphere: A Portal Framework For Building Collaborations, *1st International Workshop on Middleware for Grid Computing*, Rio de Janeiro, Brazil, June 2003.
[10] MySQL, www.mysql.com
[11] J. T. Kohl, B. C. Neuman, and T. Y. T'so, The Evolution of the Kerberos Authentication System. In *Distributed Open Systems*, pages 78-94. IEEE Computer Society Press, 1994.
[12] SimpleCA, www.globus.org/toolkit/docs/4.0/security/simpleca/index.html
[13] W.E. Allcock, I. Foster, R. Madduri. Reliable Data Transport: A Critical Service for the Grid. *Building Service Based Grids Workshop, Global Grid Forum 11*, June 2004.