On the Use of Services to Support Numerical Weather Prediction

Jay Alameda, Albert L. Rossi, Shawn Hampton
University of Illinois at Urbana-Champaign, National Center for Supercomputing Applications, 1205 W. Clark St., Urbana, IL, 61801,
jalameda@ncsa.uiuc.edu

The challenges of building an effective grid-based problem solving environment that truly extends and embraces a computational scientist's traditional tools are multifold. It is far too easy to build simple stovepipes that allow fixed use patterns, that don't extend a scientist's desktop, and fail to encompass the full range of patterns that a scientist needs to find such a problem-solving environment as a liberating and enabling tool. In the LEAD project, we have focused on the most challenging users of numerical weather prediction, namely, the atmospheric science researchers, who are prone to use their own tools, their own modified versions of community codes such as the Weather Research and Forecasting (WRF) model, and are typically comfortable with elaborate shell scripts to perform the work they find to be necessary to succeed, to drive our development efforts. Our response to these challenges includes a multi-level workflow engine, to handle both the challenges of ensemble description and execution, as well as the detailed patterns of workflow on each computational resource; services to support the peculiarities of each platform being used to do the modeling (such as on TeraGrid), and the use of an RDF triple store and message bus together as the backbone of our notification, logging, and metadata infrastructure. The design of our problem-solving environment elements attempts to come to grips with lack of control of elements surrounding and supporting the environment; we achieve this through multiple mechanisms including using the OSGI plug-in architecture, as well as the use of RDF triples as our finest-grain descriptive element. This combination, we believe, is an important stepping stone to building a cyber environment, which aims to provide flexibility and ease of use far beyond the current range of typical problem solving environments.

1 Numerical Weather Prediction as a science driver (Linked Environments for Atmospheric Discovery, LEAD)

Numerical Weather Prediction, in its current form, revolves around pre-scheduled computations, running models such as the Weather Research and Forecasting (WRF) [WRF] code on fixed grids, using observations statically obtained on a fixed schedule, to generate predictions of future weather, as depicted in Figure 1.

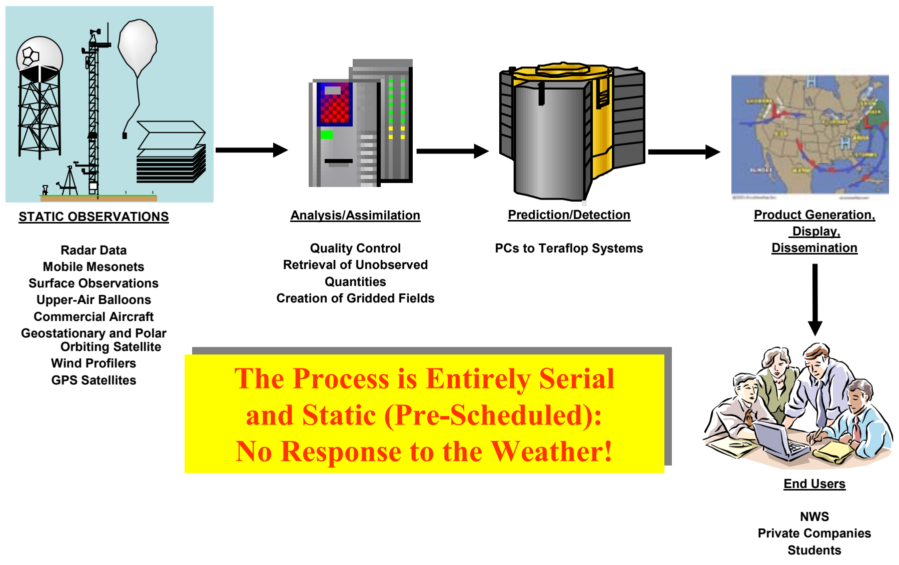


Figure 1. Traditional Numerical Weather Prediction Methodology (credit: K. Droegemeier, U Oklahoma)

The National Science Foundation funded information technology research project, Linked Environments for Atmospheric Discovery (LEAD), was funded to address the shortcomings in the traditional forecasting methodology, and, for the first time, provide a means for people and technologies to interact with the weather [LEAD]. This project, a partnership of researchers from the University of Oklahoma, Indiana University, University of North Carolina at Chapel Hill, University of Alabama in Huntsville, the University of Illinois at Urbana-Champaign, Millersville University, Howard University, Colorado State University and University Corporation for Atmospheric Research (UCAR), is doing basic information technology research into the issues to enable models to respond to observations, as well as have models and algorithms drive sensors. This research is resulting in an integrated, scalable framework that allows analysis tools, forecast models and data repositories to be used as dynamically adaptive, on-demand systems that adapt in response to the weather, respond to users, initiate processes automatically, steer remote observatories, and operate independent of data format and location, as well as location of compute resources. The group at Illinois has been focusing on the use cases provided by our atmospheric science researchers. These researchers can be characterized by:

- having their own research computing allocations
- need to modify community codes or write their own codes
- work both remotely and locally on their own workstation
- always looking to innovate by asking tough questions: i.e., what will the weather look like with today's moderate risk of severe weather?
- Need to work with tens to hundreds of simulations

In this paper, we will describe the particular infrastructure we developed to meet these requirements, starting with our context within the LEAD Architecture.

2 LEAD Architecture

The LEAD project has broadly defined a service architecture, depicted abstractly in Figure 2. In this figure, user interfaces are clients of a set of crosscutting and configuration and execution services, which in turn access distributed resources through resource access services, such as those provided by the Globus Project [Globus]

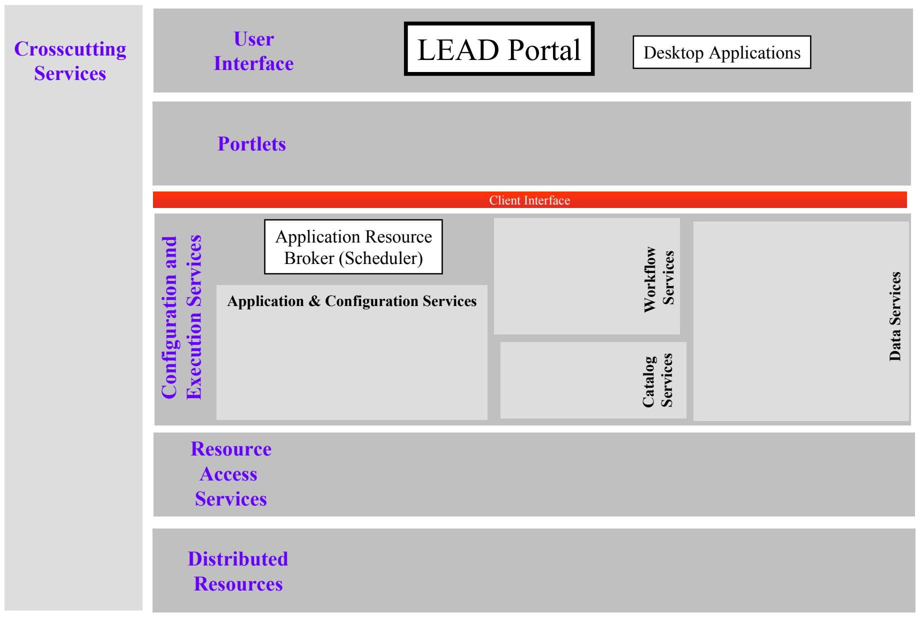


Figure 2. The LEAD service architecture. The majority of work within LEAD is in the area of crosscutting services, user interfaces, and configuration/execution services.

For our work with researchers, our focus has been on services and interface elements as depicted in Figure 3.

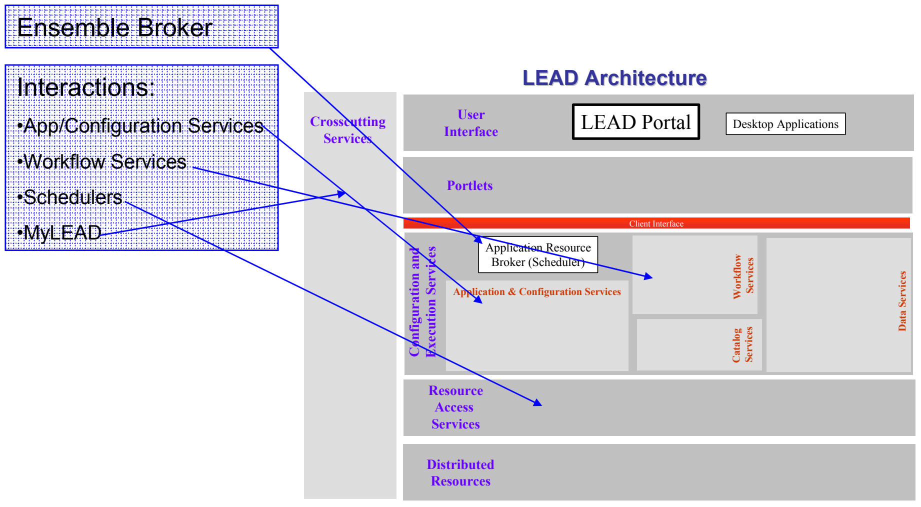


Figure 3. Context of UIUC contributions to LEAD within the LEAD service architecture. We are building a facility for managing atmospheric science ensembles, the Ensemble Broker, and a group of supporting services for this facility.

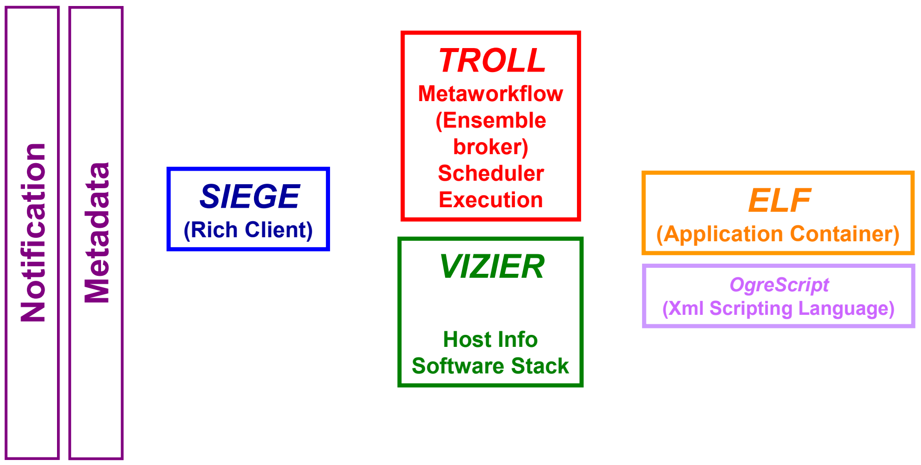


Figure 4. Siege and Ensemble Broker for managing large numbers of runs. Troll manages the coordination and execution of runs, vizier supplies information necessary for correct execution of workflows on remote hosts, and ELF+ogrescript manages all aspects of cluster-local execution. Tying this all together is notification and metadata (in progress).

Our application partners have the following requirements that have shaped our work. First, they need a facility to manage a large number of runs. Secondly, they need efficient interactions with their desktop platform. Thirdly, they need to be able to make modifications in a simple and powerful way – for instance, to add their own application, or to change their pattern of work. As a result, we developed a simple workflow description (ensemble builder to describe the overall interactions between jobs, and ogrescript for the compute resource-local orchestration), and a desktop tool (Siege) to manage their work. These pieces, which fit in the context of Figure 3, can be depicted schematically in Figure 4, which shows the desktop client, Siege [Siege], which interacts with the Troll ensemble broker stack and Vizier information services, to deploy applications controlled by the remote application container, ELF, which implements our own scripting language, OgreScript. Tying the system together are the notification systems, currently using the Java Messaging Service (JMS) [JMS] channel ActiveMQ [ActiveMQ], and a metadata system (we are planning to integrate myLEAD into the JMS channel, to be able to create LEAD metadata objects [LMS] from metadata events published to the channel).

The Siege desktop client is built using the Eclipse Rich Client Platform (RCP) [RCP], which has many advantages including ease of interface mockup, pluggable modules, and well-defined extension points. We are also using RCP to build our services, as it provides a nice modularity to better manage dependencies, especially on third-party libraries such as the jglobus [jglobus] client libraries that we use to access grid capabilities such as job submission and file management.

With Siege, the user authenticates to a myproxy server backed with a Kerberos realm [myproxy] to allow delegation of short term grid credentials to the Siege client by mere use of a user's Kerberos login at NCSA or on TeraGrid. This login, which is granted as normal part of a user's allocation at NCSA, allows users seamless access to computational resources through Siege, by use of their familiar Kerberos login (as depicted in Figure 5).

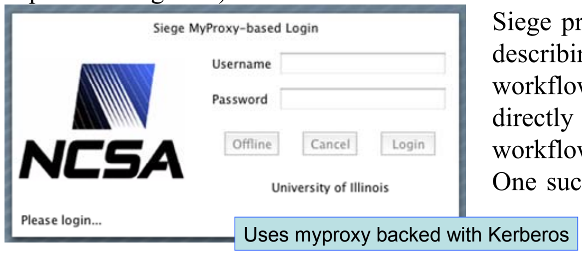
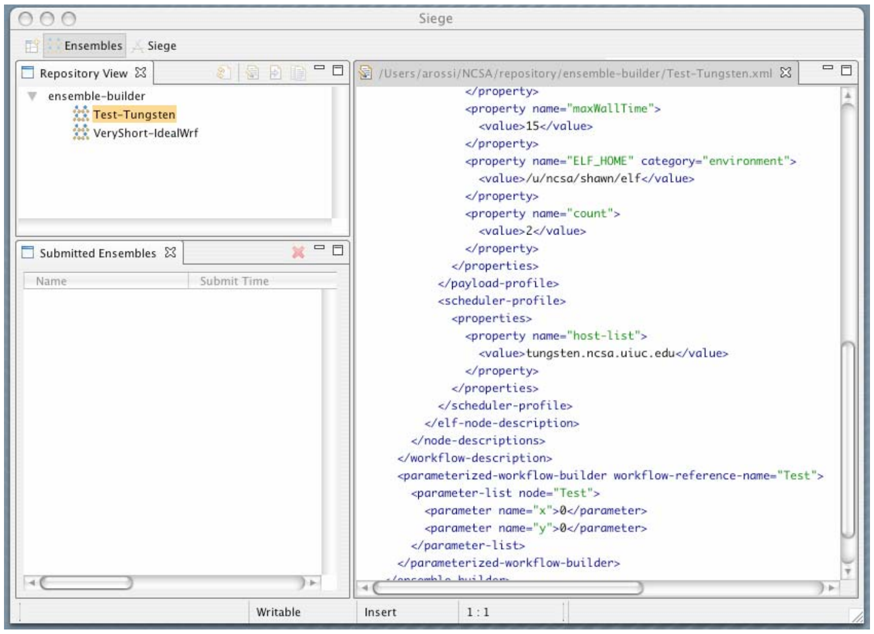


Figure 5 Siege MyProxy login, which uses Kerberos authentication for NCSA and TeraGrid.

Siege provides simple mechanisms for describing, executing and monitoring workflows. For instance, a user can directly edit the xml description of the workflow, as depicted in Figure 6. One such user prefers this, as the user can directly manipulate ranges of parameters to be explored in a parameter space study. The XML description of the ensemble is expanded into the full graph of execution at job submission time.

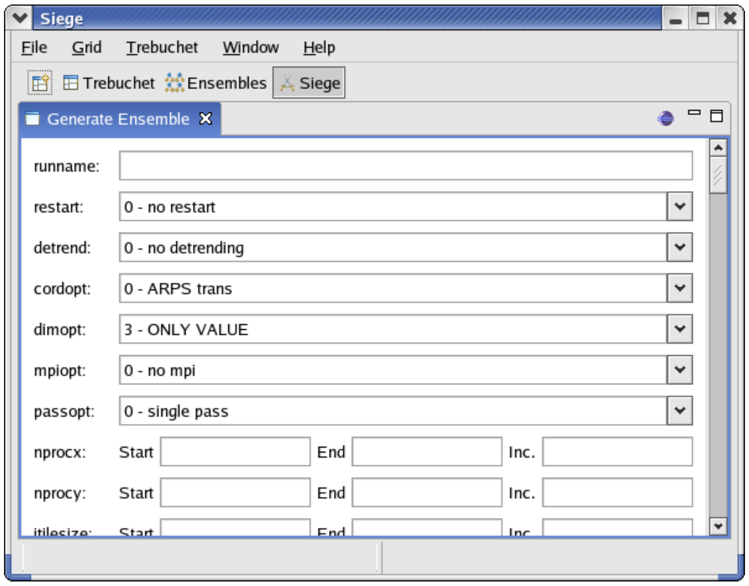
We are also prototyping user interfaces which guide the user to describe ranges and intervals of variables to be studied, in this case for a research weather code. The prototype, shown in Figure 7, depicts indication of default ranges and variable value possibilities as well.

Finally, we have prototyped a specific interface which is designed to allow the user to use Unidata's Integrated Data Viewer (IDV) [IDV] to select the center of a



```
                </property>
                <property name="maxWallTime">
                  <value>15</value>
                </property>
                <property name="ELF_HOME" category="environment">
                  <value>/u/ncsa/shawn/elf</value>
                </property>
                <property name="count">
                  <value>2</value>
                </property>
              </properties>
            </payload-profile>
            <scheduler-profile>
              <properties>
                <property name="host-list">
                  <value>tungsten.ncsa.uiuc.edu</value>
                </property>
              </properties>
            </scheduler-profile>
          </elf-node-description>
        </node-descriptions>
      </workflow-description>
      <parameterized-workflow-builder workflow-reference-name="Test">
        <parameter-list node="Test">
          <parameter name="x">0</parameter>
          <parameter name="y">0</parameter>
        </parameter-list>
      </parameterized-workflow-builder>
```

Figure 6. Direct editing of ensemble-builder workflow description is provided as one possibility within Siege. The XML description of the ensemble is expanded into the full graph of execution at job submission time



runname:					
restart:	0 - no restart				
detrend:	0 - no detrending				
cordopt:	0 - ARPS trans				
dimopt:	3 - ONLY VALUE				
mpiopt:	0 - no mpi				
passopt:	0 - single pass				
nprocx:	Start	End		Inc.	
nprocy:	Start	End		Inc.	
itilesize:	Start	End		Inc.	

Figure 7. Prototype parameter study interface for weather research code

.

domain to model the atmosphere using WRF, and then simply launch and monitor the resulting model on TeraGrid resources. This interface was integrated to allow straightforward visualization of the model results in IDV as well, and is depicted in Figure 8.

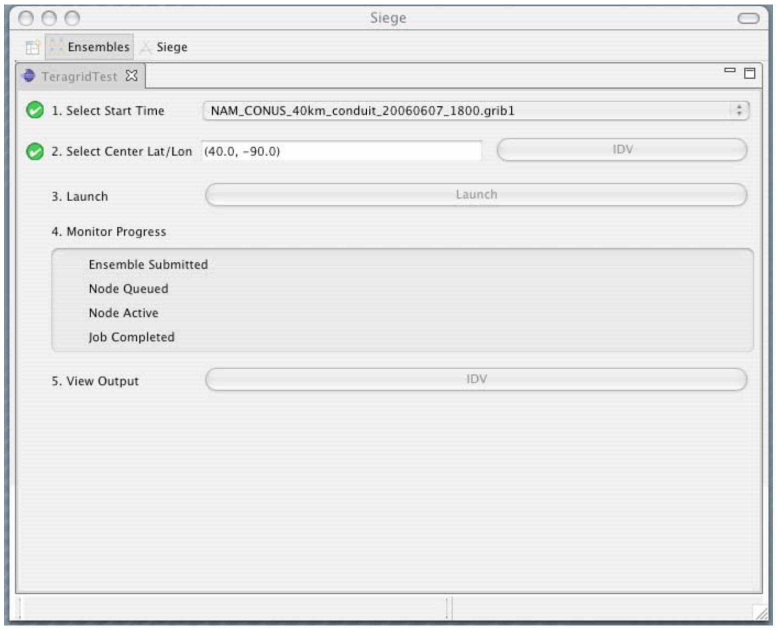


Figure 8 Simplified Unidata Workshop facility, which is integrated with the Unidata IDV, and allows simple selection of domain center latitude and longitude, as well as the NAM dataset for model initialization.

3 Real Grid Resources

The cyber infrastructure we have described so far, Siege plus its supporting services, are designed to work with grid resources [grid] as their target for supporting computations. In the context of NSF-funded computational resources, all computational resources are under the TeraGrid [TeraGrid] umbrella. The implications of this are multifold – first, one desirable result is that all TeraGrid platforms support common grid services. Secondly, there is supposed to be a common environment on all platforms (a goal which has proven harder to achieve than previously suspected), and lastly, there are no other grid resources available for a user to check against teragrid – unless you had your own test bed. One additional complexity: in TeraGrid in 2006, over 20 completely different computational architectures are represented. This means that the pain of porting and validation are very much there – not to mention performance optimization for each platform.

The mechanism used in TeraGrid to invoke the correct environment is softenv. Softenv essentially sources a script, establishing all relevant paths. The configuration of the environment is controlled through a user-configurable ".soft" file, which in the end establishes the source file. This is fine for users that are using the resources in a traditional fashion, i.e., doing local batch submissions of codes and problems that were built on the resource in question, but, it makes little sense for users of Siege who, in the end, we hope don't need to log onto a resource to take advantage of its capabilities. So, as a rule, we don't want to rely on users modifying their ".soft" files, with potential adverse effects to their environment – and, to be sure, we even want to use accounts where users may have inadvertently broken their environment, by establishing a correct environment for the scientific application in question. We would also like to invoke the correct environment at job-submission time, through the Globus GRAM Resource Specification Language (RSL) [RSL].

4 Results

We have been able to use the Siege user interface and service architecture to manage runs made in the course of the Unidata workshop in July 2006 [UnidataWorkshop], as well as to support one of the coauthors (Jewett) work in understanding parametric sensitivities for interacting thunderstorms. Through this, we have observed that we have a number of issues to address with respect to scalability and robustness – in no particular order:

- in-memory service state is hard to manage, we need to persist all such state to a relational database so that services can be brought back up seamlessly without loss of information
- gridFTP servers appear to have an issue handling many clients, with difficult-to-diagnose failures as a result
- the web services in the stack need to be few in number, as well as close to transactional as possible, with small items being used per interaction rather than complex objects

But, even with the issues identified for resolution, we were able to successfully support the modeling efforts of the workshop attendees as well as our coauthor, and have charted a path for improved performance and scalability of the underlying service stack. We also have shown that the Eclipse Rich Client Platform provides a powerful, flexible alternative user interface which integrates well with a user's desktop platform. The use of JMS for messaging has also proven to be a good choice, for its inherent flexibility and malleability to a variety of other messaging systems, as well as its performance, and finally, the ability to perform cluster-local orchestration readily facilitates local monitoring of batch processes

5 Acknowledgements

LEAD is a Large Information Technology Research (ITR) Grant funded by the National Science Foundation under the following Cooperative Agreements: ATM-0331594 (University of Oklahoma), ATM-0331591 (Colorado State University), ATM-0331574 (Millersville University), ATM-0331480 (Indiana University), ATM-

0331579 (University of Alabama in Huntsville), ATM03-31586 (Howard University), ATM-0331587 (University Corporation for Atmospheric Research), and ATM-0331578 (University of Illinois at Urbana-Champaign, with a sub-contract to the University of North Carolina). This work has also been supported by SCI03-30554, SCI04-38712, and SCI96-19019. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

6 References

[WRF] The Weather Research & Forecasting Website, http://www.wrf-model.org/index.php.

[LEAD] Droegemeier, K. K. and Co-Authors, 2004: Linked environments for atmospheric discovery (LEAD): A cyberinfrastructure for mesoscale meteorology research and education. Preprints, 20th Conf. on Interactive Info. Processing Systems for Meteorology, Oceanography, and Hydrology, Seattle , WA , Amer. Meteor. Soc

[Globus], The Globus Toolkit, http://www.globus.org/toolkit/.

[grid], Grid Computing, Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Grid_computing.
[TeraGrid] TeraGrid, http://www.teragrid.org/.

[Siege] Siege – MRD-Public – Confluence, http://torcida.ncsa.uiuc.edu:8080/confluence/display/MRDPUB/Siege

[JMS] Java Message Service, http://java.sun.com/products/jms/

[ActiveMQ] ActiveMQ Home, http://www.activemq.org/site/home.html

[LMS] LEAD Metadata Schema Repository, http://www.extreme.indiana.edu/rescat/metadata/.

[RCP], Rich Client Platform, http://wiki.eclipse.org/index.php/Rich_Client_Platform.

[jglobus] CoG jglobus, http://dev.globus.org/wiki/CoG_jglobus.

[myproxy] Myproxy Credential Management Service, http://grid.ncsa.uiuc.edu/myproxy/

[TeraGrid] TeraGrid, http://www.teragrid.org/.

[IDV] Unidata Integrated Data Viewer (IDV),
http://www.unidata.ucar.edu/software/idv/.

[UnidataWorkshop] 2006 Unidata Users Workshop: Expanding the Use of Models as Educational Tools in the Atmospheric & Related Sciences,
http://www.unidata.ucar.edu/community/2006workshop/

[RSL] GT 2.4http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html