

# Advance Reservation and Co-Scheduling Report

## Scheduling Working Group

7/14/2008

### 1. Introduction

For the past several months, the Scheduling Working Group in the TeraGrid has been examining tools that support advance reservations and co-scheduling for deployment on TeraGrid. This effort grew out of the results of two TeraGrid Requirements Analysis Teams (RATs): The Scheduling RAT and the Metascheduling RAT. One of the results of these RATs was that there are TeraGrid users that want to use advance reservations and co-scheduling in their work.

To be precise, we define these capabilities as:

- **Advance reservations.** An *advance reservation* dedicates a set of resources (compute, visualization, etc.) to a user or group of users at a specific time for a specific duration. For instance, a user may need resources for a demonstration, to meet a deadline, or for simultaneous access to TeraGrid and non-TeraGrid resources.
- **Co-scheduling.** There are some simulations or analyses that require simultaneous use of multiple resources. For example, a user needs to run a simulation with extremely large memory requirements that cannot be met by using a single resource. By using multiple resources at the same time, *co-scheduling* enables this kind of capability.

The Scheduling Working Group examined three tools that support these capabilities: the Globus Advance Reservation Service (GARS), the Grid Universal Remote (GUR), and the Highly-Available Resource Coallocator (HARC). The original intent of the working group was to select a subset of these tools for deployment. However, during the course of our evaluation process, it became clear that both GUR and HARC have users that would prefer not to switch to a different tool and that there might be users that would prefer GARS. Furthermore, we realized that the developers of these tools would continue developing them whether or not we recommended that they be deployed.

With these observations, the working group decided to change its process from determining which advance reservation and co-scheduling tools should be deployed, to supporting the deployment of any of these tools that are production ready on TeraGrid. This includes an evaluation of the tool to determine whether it is ready to be deployed and assisting with deployment and support after deployment.

There are advantages and disadvantages of this new approach, but we believe the advantages outweigh the disadvantages. One disadvantage is that there will be multiple tools to choose from that provide the same functionality that can be deployed. For the RP, it may cause confusion on

which tool(s) to install. Installing multiple tools instead of just one would result in extra work. We plan to mitigate the amount of extra work by ensuring that the install processes for these tools are straightforward and can be accomplished in a minimal amount of time. Another disadvantage is that users may not know which tool is the best one for them. When multiple reservation and co-scheduling tools are available in production on TeraGrid, we will mitigate this problem by providing documentation comparing the tools so that users can more easily decide which one(s) to use.

There are also several advantages to this approach. There are already users of two of the tools, so we will not eliminate either from deployment on TeraGrid and perhaps lose users. Furthermore, we did not foresee a clear choice for which tool(s) to select. Our current approach will avoid a contentious selection process that most likely would not have a clear result and would harm future collaboration.

In the rest of this document, we summarize the tools we examined, describe the current deployment status of these tools on TeraGrid, provide more detail about our future plans, and summarize our results. An appendix provides more details about the tools we examined.

## **2. Summary of Tools**

Previous TeraGrid RATs identified the GUR and HARC tools as candidates for supporting advance reservation and co-scheduling on TeraGrid. In addition, the Globus project began work on GARS, a tool for advance reservation only. We therefore examined all three of these tools.

The GARS software is developed by the University of Chicago at Argonne National Laboratory by the Globus group. It is a service implemented in the Web Service Resource Framework, runs in a Globus web service container, and provides command line, Java API, and web services interfaces. GARS supports advance reservations of a system that the GARS service is installed on. It does not provide co-scheduling, but multiple requests to different GARS services can be used to co-schedule resources from multiple systems.

The GUR software is developed by the San Diego Supercomputing Center. It is implemented on top of SSH (including the SSH version that supports the Grid Security Infrastructure), does not have any of its own services or daemons, and provides a command line interface. A GUR client provides reservations of single systems or co-scheduling of multiple systems for any systems that it is configured to know about.

The HARC software is developed by the Center for Computation and Technology at Louisiana State University. It is implemented using a service on each TeraGrid system that accepts reservations, a set of services that manage reservations, and client software that the user interacts with. The client software provides a command line interface to users. HARC provides reservations of single systems and co-scheduling of multiple systems for any systems that the HARC reservation management services know about.

There are some similarities of interface and function in these tools, but different design philosophies. All three tools provide command line interfaces while GARS also provides a Java

API and web service interface. All three tools support advance reservation while only GUR and HARC provide co-scheduling. All of the tools support similar types of reservation/co-scheduling requests (e.g. exact time or range of times, exact number of nodes or range of nodes). One important difference is that GARS and GUR do not currently allow you to query the status of a previously made reservation, while HARC does. It is useful to be able to do this to ensure that a reservation is still in place and wasn't canceled for some reason.

As far as design, GARS is designed to fit within the Globus WSRF framework that is already deployed on TeraGrid. GUR is designed to be simple and have as few dependencies on other software as possible (it only needs a version of SSH and a local resource manager that supports user-settable reservations). HARC is designed for reliability and fault tolerance, resulting in a more complex implementation.

All three tools provide a command line interface, and the number of users that desire other interfaces remains to be seen. At least one science gateway would like a web service interface, and other science gateways may prefer APIs or web service interfaces. The "best" design for TeraGrid is also unknown – opinions vary even within the Scheduling Working Group. The design choices made by tool developers will be tested as they move toward production deployment on TeraGrid.

See Section **Error! Reference source not found.** for more detailed information about GARS, GUR, and HARC.

### 3. Current TeraGrid Deployments

GARS, GUR, and HARC are deployed on TeraGrid systems both as part of our exploration of these tools and independently by the tool developers. GARS is deployed in testbed form on the NCSA, SDSC, and UC/ANL Itanium clusters. GUR is deployed in production on the NCSA and SDSC Itanium clusters as well as SDSC Datastar. GUR is also deployed in testbed form on the UC/ANL Itanium cluster. HARC is deployed in production at LONI/LSU and in testbed form on the NCSA Itanium cluster. Furthermore, the clients for these tools are available on the server [tg-t004.uc.teragrid.org](http://tg-t004.uc.teragrid.org) and `softenv` has been configured on that system to include keys to configure the user environment for these clients.

In addition to these deployments, the Scheduling Working Group has provided documentation about the installations on TeraGrid systems and user guides that describe how to use these software packages on the TeraGrid. The user guides are on the TeraGrid wiki and can be found from the GARS, GUR, and HARC testbeds pages that are linked from the Scheduling Working Group home page at [http://www.teragridforum.org/mediawiki/index.php?title=Scheduling\\_WG](http://www.teragridforum.org/mediawiki/index.php?title=Scheduling_WG).

An important point to note is that having GARS, GUR, or HARC working properly is not the only requirement for useful advance reservations or co-scheduling on TeraGrid: The local resource managers must also be configured to allow reservations. This configuration includes setting policies such as when reservations can be made, which users can make reservations, how users are charged for reservations, and when/if reservations are canceled if a user doesn't submit

jobs to it. It can be difficult to perform these configurations correctly so that specific users can successfully make reservations.

## 4. Future Plans

In the area of advance reservation and co-scheduling, the working group will support the deployment of any of these three tools on TeraGrid. However, the developers of these tools are primarily responsible for preparing them for deployment. The working group will:

- Help developers prepare their tool for deployment. The developers of these three tools are welcome to prepare their tools for TeraGrid deployment as Common TeraGrid Software Stack (CTSS) components. This preparation consists of tasks such as packaging the tool as a pacman package, preparing documentation for installation and usage, and so on. While these tasks are primarily the responsibility of the tool developers, the working group will provide guidance in these tasks.
- Determine whether a tool is ready for inclusion in CTSS and inform the Software Working Group when it is. The reason the Scheduling Working Group will determine when a tool is ready is that we know the most about these tools and can reduce the workload on the Software Working Group by providing an initial gate for reservation and co-scheduling tools to pass. To determine whether an advance reservation or co-scheduling tool is ready for inclusion in CTSS, we will evaluate the installation process on a variety of TeraGrid systems, the documentation, and the support process.
- Provide documentation on these tools to TeraGrid users.
- Help tool developers support TeraGrid Resource Providers (RPs) as they install any of these tools.
- Help tool developers support TeraGrid users as they use these tools.

These are the working group plans only in the area of advance reservation and co-scheduling. We may or may not decide to use a similar process for other types of scheduling tools (e.g. automatic resource selection).

## 5. Summary

Previous work on the TeraGrid found that there are users that would like to be able to reserve resources in advance on either single systems or multiple systems. For the past several months, the Scheduling Working Group has been examining tools that support these capabilities. The tools we examined were the Globus Advance Reservation Service, the Grid Universal Remote, and the Highly-Available Resource Coallocator.

The original intent of the working group was to select a subset of these tools for deployment, but existing usage of the tools and other factors made us realize that this was not the best approach to take. The working group decided to change its goals to supporting the deployment of any of these tools that the developers wish to move into production on TeraGrid. This includes helping developers prepare their tool for deployment, determining whether a tool is ready to be deployed, and assisting with deployment and support. These are the next tasks of the Scheduling Working Group in the area of advance reservation and co-scheduling.

## 6. Evaluations

This section provides the detailed evaluation information about the three tools for advance reservation and co-scheduling that were examined.

### 6.1. Globus Advance Reservation Service

1. General information
  - a. What is the name of the tool? **Globus Advance Reservation Service (GARS)**
  - b. Where is the web site for the tool?  
**<http://confluence.globus.org/display/gars/Home>**  
**<http://confluence.globus.org/display/gars/Alpha+Version>**
  - c. Cost/Licensing
    - i. Is the tool free to use? **Yes** Is support free? **Yes**
    - ii. How is the tool licensed? (GPL, Commercial, etc.) **Globus license (Apache v2)**
  - d. Code Availability
    - i. Is the code open-source? **Yes**
    - ii. Is there a mechanism for the developer to accept code changes from TeraGrid? **Yes**
  - e. Support/Documentation
    - i. What support does the scheduler developer provide? (24/7, user forums, faqs) **There are an open gars-user and gars-dev email lists for questions. There is an open jira gars project for bug submission/tracking.**
    - ii. What is the quality of the documentation? **Basic admin and user documentation at**  
**<http://confluence.globus.org/display/gars/Alpha+Version>**
    - iii. Is the web site for the tool helpful and informative? **Yes**
  - f. Product Maturity
    - i. How long has the product been available? **8 months**
    - ii. What is the production status of the code (prototype/alpha/beta/production)? **Alpha**
    - iii. How many other production grids use this software today? **None**
    - iv. Approximately how many users of this software are there? **One**
    - v. Approximately how many developers support this product? **Two**
2. Functionality. Does the tool support the following functionality at this time? (also indicate if the functionality is planned in the future and a timeline, if known)
  - a. Advanced reservations of individual systems? **Yes**
  - b. Co-scheduling/co-allocation across more than one system? **No**
  - c. Request a reservation/co-allocation starting at a specific time? **Yes**
  - d. Request a reservation/co-allocation starting within a window of time? **Yes**
  - e. Request a reservation/co-allocation with an exact number of nodes/resources? **Yes**

- f. Request a reservation/co-allocation with a range of number of nodes/resources? **No**
  - g. Request a reservation/co-allocation on an exact set of systems? **Yes**
  - h. Request a reservation/co-allocation on a subset of a candidate set of systems? **No**
  - i. Query the status of a previously made reservation/co-allocation? **No**
  - j. Query for when a specific reservation/co-allocation can be made? **No**
  - k. Provide useful information about why a reservation/co-allocation cannot be made so that a user can adjust their request? What are examples of reasons the tool provides? **No**
  - l. Submission of reservation requests directly to the local scheduler (i.e. without going through any global services)? **Yes**
  - m. Please list other relevant functionality not described above. **The GARS service is not fault tolerant. In the event of a container or host restart, knowledge about any reservations is lost.**
3. Installation
- a. Were the installation instructions clear? **Yes, except for host cert generation and installation.**
  - b. What Resource Managers does the tool interface with? (e.g. LSF, PBS, LoadLeveler, SGE) **Moab, Catalina**
  - c. Does the scheduler require any modifications to local resource manager? Are these modifications straightforward?  
**The local schedulers must be configured to allow users to set reservations. This mod is straightforward on Catalina. Similar to other reservation and co-scheduling grid tools, the local resource manager must be configured to allow users to set reservations and a policy must be defined that specifies under what conditions users can make reservations. When reservations are made available via any mechanism, consideration should be given to their impact on batch jobs.**
  - d. How long did installation take in hours of work?  
**~2 hours of typing/sitting. Longer in wallclock. This included installing a separate container for the GARS service.**
  - e. What additional software is required in order to support the tool and where must it be installed? For each software dependency, is that software already in CTSS?
    - i. Each Teragrid resource (for example, GridFTP)  
**Globus Java WS Core (on TG already)**  
**Java 1.4.2 or 1.5+ (on TG already)**  
**Ant 1.6+ (on TG already)**  
**sudo (on TG platforms already)**
    - ii. Somewhere on the TeraGrid (for example, MDS, Myproxy)  
**Certificate Authority for host certs (already on TG)**
    - iii. On the same machine as the metascheduler (for example, OS, MySQL)  
**None**
  - f. Did you ask any questions of the developers? **Yes.** If so, were the developers responsive? **They were responsive.**
  - g. What customization was necessary to get the software to work? Was this customization easy or difficult?

**The sudoers file needs to be modified. This was technically easy, but approval from our security group took time.**

**The local scheduler must be configured to allow reservations by users.**

**Technically easy for Catalina.**

- h. Are there installation problems that you expect would occur on many installations? **No. Although getting host certificates from an acceptable CA could be an issue.**
  - i. For the software components that would be installed by RPs, are there any barriers to installing these components automatically as part of a CTSS kit? **No.**
4. Operation
- a. How reliable is the software (failures/week)? **Unknown – not enough experience.**
  - b. What failures were encountered? **Reservation failure to ANL GARS service. This was due to my user not being allowed to make user reservations in the Moab reservation config.**
  - c. Does the software provide logging? Can the amount of logging be adjusted? **Yes. GARS prints log messages to stdout which goes to the Globus container.log file.**
  - d. What amount of resources are typically used by the software? On what systems? (e.g. central server, login node) (e.g. disk space, physical/virtual memory, CPU time)  
**Login node: 21MB disk, 100MB/100MB physical/virtual memory, negligible CPU time**
5. User Experience
- a. What is the quality of the user documentation? **Sufficient for install and basic usage.**
  - b. What client interfaces are provided (GUI, command line, web interface, etc.)  
**There is a command line interface, a Java API, and a WSRF interface.**
  - c. For each user interface evaluated (e.g. GUI, API, command line):
    - i. Provide the interface name: **Command line**
    - ii. Is it well documented? **Yes**
    - iii. Is it easy to understand and use? **Yes**
    - iv. Are there any changes to the interface that would improve it? **Provide node ranges (0-N nodes requested)**
  - d. Where any problems encountered? (e.g. documentation not matching interface, unimplemented features) **Installation of host certs was not well documented.**
  - e. Are the error messages clear and helpful for debugging problems? **No. It was not clear that my user was not configured to make Moab reservations at ANL.**
  - f. What is the average response time of the software? **A few seconds of overhead on top of the time it takes the LRMS to make the reservation. This could be ten seconds for Moab or 2 minutes for Catalina.**
  - g. How does the software perform under load? At what amount of load does the software begin to respond slowly? (e.g. twice as slow as unloaded response time)  
**Not done**
  - h. For each TeraGrid user helping evaluate:

- i. Does this software meet your needs? **Yes, this could be used by GUR to set reservations.**
  - ii. Is this your preferred software for performing advance reservation and/or co-scheduling? **No.**
- 6. Any other evaluator comments?  
**I did this install as a complete standalone service, so I needed to install Java WS Core and host certs. If this is part of the Globus package already installed on TG systems, it would be faster.**  
**In the evaluator's opinion, instead of requiring a modification to the sudoers file, it would be much better for gars-client to invoke the remote reservation command as the requesting user.**

## 6.2. Grid Universal Remote

- 1. General information
  - a. What is the name of the tool? **Grid Universal Remote (GUR)**
  - b. Where is the web site for the tool?  
**<http://www.sdsc.edu/scheduler/gur.html>**  
**SDSC user documentation: <http://www.sdsc.edu/us/tools/coschedule.html>**
  - c. Cost/Licensing
    - i. Is the tool free to use? Is support free? **Free to use and support is free**
    - ii. How is the tool licensed? (GPL, Commercial,etc.)  
**No fee for education, research, non-profit use. Contact UCSD for use in commercial systems.**
  - d. Code Availability
    - i. Is the code open-source? **Yes, with commercial use restriction.**
    - ii. Is there a mechanism for the developer to accept code changes from TeraGrid? **Yes, the developer of GUR is Kenneth Yoshimoto of SDSC. He participates in TeraGrid**
  - e. Support/Documentation
    - i. What support does the scheduler developer provide? (24/7, user forums, faqs) **Email to the developer. Tickets to SDSC, NCSA, or TeraGrid.**
    - ii. What is the quality of the documentation?  
**The user documentation is sufficient.**  
**The administrator documentation is sufficient.**  
**Developer documentation is not available.**
    - iii. Is the web site for the tool helpful and informative?  
**The web site is sparse. It could use improvement if the goal is to promote the use of GUR.**
  - f. Product Maturity
    - i. How long has the product been available?  
**3 years, including prototypes**

- ii. What is the production status of the code (prototype/alpha/beta/production)? **Production.**
  - iii. How many other production grids use this software today? **TeraGrid only.**
  - iv. Approximately how many users of this software are there? **Approximately 10, based on GUR usage logs. 5 of these are TeraGrid end users.**
  - v. Approximately how many developers support this product? **One.**
2. Functionality. Does the tool support the following functionality at this time? (also indicate if the functionality is planned in the future and a timeline, if known)
- a. Advanced reservations of individual systems? **Yes**
  - b. Co-scheduling/co-allocation across more than one system? **Yes**
  - c. Request a reservation/co-allocation starting at a specific time? **Yes**
  - d. Request a reservation/co-allocation starting within a window of time? **Yes**
  - e. Request a reservation/co-allocation with an exact number of nodes/resources? **Yes**
  - f. Request a reservation/co-allocation with a range of number of nodes/resources? **Yes**
  - g. Request a reservation/co-allocation on an exact set of systems? **Yes**
  - h. Request a reservation/co-allocation on a subset of a candidate set of systems? **Yes**
  - i. Query the status of a previously made reservation/co-allocation? **No**
  - j. Query for when a specific reservation/co-allocation can be made? **No**
  - k. Provide useful information about why a reservation/co-allocation cannot be made so that a user can adjust their request? What are examples of reasons the tool provides? **Yes. GUR appears to pass back the errors generated by the LRMS reservation tool that it uses. I doubt there is any more information that could be provided to GUR users. Here's an example:**

**Reservation creation failed (ReserveFailure).  
 Failure message: (SERVERMODE (NORMAL)  
 forcing TZ to (PST8PDT)  
 TZ (PST8PDT)  
 Bad account format**

- l. Submission of reservation requests directly to the local scheduler (i.e. without going through any global services)? **Yes**
- m. Please list other relevant functionality not described above.  
**The gur.py program has a mode that interactively prompts the user for information and generates a co-scheduling description file.**

**GUR allows the user to specify a total number of nodes they want and the minimum and maximum number of nodes they want to use on each machine. GUR will allocate the total number of nodes within the min and max constraints.**

- 3. Installation
  - a. Were the installation instructions clear? **Yes. The only thing that came up is when editing the config file, it wasn't clear what the full set of available**

variables is. I had to look at the python code to figure some things out. This editing probably isn't necessary with the user-prompt based install script.

- b. What Resource Managers does the tool interface with? (e.g. LSF, PBS, LoadLeveler, SGE) **Moab, Catalina. A prototype LSF interface has been created. GUR can also use GARS, and therefore whatever resource managers that GARS interfaces to**
  - c. Does the scheduler require any modifications to local resource manager? Are these modifications straightforward? **The only modifications are enabling user-settable reservations and then specifying the policies for those reservations.**
  - d. How long did installation take in hours of work? **It took about 4 hours to install GUR, create a prototype LSF interface, and performance tests. This is using the old install method of editing the GUR config file. I estimate 60 minutes to understand GUR and do an install without implementing a LRMS interface.**
  - e. What additional software is required in order to support the tool and where must it be installed? For each software dependency, is that software already in CTSS?
    - i. Each TeraGrid resource (for example, GridFTP) **Python, a LRMS that supports user-settable reservations.**
    - ii. Somewhere on the TeraGrid (for example, MDS, Myproxy) **None**
    - iii. On the same machine as the metascheduler (for example, OS, MySQL) **Python**
  - f. Did you ask any questions of the developers? If so, were the developers responsive? **Yes and Yes.**
  - g. What customization was necessary to get the software to work? **Configuration by either answering a set of questions from a config script or editing a configuration file. I also had to write the interface to LSF since that wasn't available. I wouldn't expect this to be the case for the majority of system administrators.**

Was this customization easy or difficult? **Relatively easy. It took a while to get through all of the configuration file editing. It should be significantly quicker if the administrator uses the configuration script that prompts the installer for input.**
  - h. Are there installation problems that you expect would occur on many installations? **Perhaps mistakes when editing the config file.**
  - i. For the software components that would be installed by RPs, are there any barriers to installing these components automatically as part of a CTSS kit? **Not with a little work. Pacman supports asking questions of the installer, so I believe the questions asked by the configuration script could be incorporated into the pacman process.**
4. Operation
- a. How reliable is the software (failures/week)? **Once the software is installed, I don't believe failures of the software will occur often. It is relatively simple and would only break if the LRMS reservation interfaces change.**
  - b. What failures were encountered? **None with GUR. MPIg testing encountered errors using the reservations after GUR made them via the LRMS**
  - c. Does the software provide logging? **Yes.** Can the amount of logging be adjusted? **Unknown.**

- d. What amount of resources are typically used by the software? On what systems? (e.g. central server, login node) (e.g. disk space, physical/virtual memory, CPU time) **I didn't do an exact measurement, but the amount of resources used is very minimal. It is simply a client-side python script that interacts with the user and uses ssh to invoke LRMS commands on the server side.**
5. User Experience
- a. What is the quality of the user documentation? **It is a bit brief in general. The quickstart guide could use more work, in particular. It was easy to determine how to create reservation scripts using the co-scheduling documents at NCSA and SDSC. The documentation included example scripts that were helpful. The learning curve was short and one user group was excited that they could reserve resources as late as 1 day before the reservation start time. There is a useful GUR document on the TeraGrid wiki that the user group didn't know about until recently. Links to documentation from teragrid.org are important.**
- b. What client interfaces are provided (GUI, command line, web interface, etc.) **Command line.**
- c. For each user interface evaluated (e.g. GUI, API, command line):
- i. Provide the interface name: **Command line.**
  - ii. Is it well documented? **Good enough. "gur.py -help" is useful.**
  - iii. Is it easy to understand and use? **Yes.**
  - iv. Are there any changes to the interface that would improve it?  
**The flags -jobfile and -metajobfile could perhaps be renamed to be more clear. -jobfile is the description of the job before it is submitted and -metajobfile is a description of a job after it is submitted.**
- Instead of just having gur.py, commands like gur\_reserve, gur\_cancel, etc. could be written.**
- The GUR reservation script format has redundant information in it, for example total\_nodes and (min\_int, max\_int) in each resource section.**
- The method to use to select the number of nodes and time frame for GUR to return a success is tricky. Is there a way for GUR to help more with this?**
- A web interface/service for GUR would be helpful.**
- d. Where any problems encountered? (e.g. documentation not matching interface, unimplemented features)
- e. Are the error messages clear and helpful for debugging problems? **Mostly. When errors occur, the output with -debug is very lengthy. Error messages can sometimes be a bit cryptic.**
- f. What is the average response time of the software? **It very much depends on the response time of the LRMS being contacted. It seems to take an additional**

**second or two in addition to the LRMS response time. For example, 5-10 seconds to NCSA, 40-100 seconds to SDSC.**

- g. How does the software perform under load? At what amount of load does the software begin to respond slowly? (e.g. twice as slow as unloaded response time) **Untested.**
  - h. For each TeraGrid user helping evaluate:
    - i. Does this software meet your needs? **For MPIg: . For GiSolve: Yes.**
    - ii. Is this your preferred software for performing advance reservation and/or co-scheduling?  
**For MPIg:**  
**For GiSolve: Yes**
6. Any other evaluator comments?  
**Another good addition would be a way to list reservations. Even better, to do it without having to list metajob file(s). Perhaps the whole metajob thing can be done away with and a list of reservations could be kept in \$HOME/.gur or something? The current interface supports printing information about a job by specifying the metajob file (using the --dump option).**

**It would be nice if the install script detected paths in the user's search path and used them as defaults when asking the user for paths.**

**Having a file on each client system that describes all of the servers isn't a very good approach for maintainability. I know the developers are already thinking about how to change this.**

**A user group (gateway) suggested that a portal interface would be useful for their users.**

**This isn't an issue with just GUR, but some reservations were not respected. One situation is that when a reservation was made for a class demonstration, the instructor didn't get to job submissions until a little while into the class. The reservation was already canceled by the system because it hadn't been used. Other times, for unknown reasons. An explanation of why this happens would be good.**

### **6.3. Highly Available Resource Coallocator**

- 1. General information
  - a. What is the name of the tool? **HARC**
  - b. Where is the web site for the tool? **<https://wiki.cct.lsu.edu/harc/HARC>**
  - c. Cost/Licensing
    - i. Is the tool free to use? **Yes** Is support free? **Yes**
    - ii. How is the tool licensed? (GPL, Commercial,etc.) **BSD**
  - d. Code Availability
    - i. Is the code open-source? **Yes**

- ii. Is there a mechanism for the developer to accept code changes from TeraGrid? **Yes**
  - e. Support/Documentation
    - i. What support does the scheduler developer provide? (24/7, user forums, faqs) **On line documentation, email list, and wiki are available. See <https://wiki.cct.lsu.edu/harc/HARC>**
    - ii. What is the quality of the documentation?  
**Very good**
    - iii. Is the web site for the tool helpful and informative?  
**Yes, very helpful**
  - f. Product Maturity
    - i. How long has the product been available? **Unknown**
    - ii. What is the production status of the code (prototype/alpha/beta/production)? **The code is in production on many resources in the UK and at LSU/LONI. NCSA is also supporting HARC in production to a limited number of users (GENIUS).**
    - iii. How many other production grids use this software today?  
**LONI Infrastructure  
EnLIGHTened Project  
TeraGrid  
See  
[http://wiki.realitygrid.org/wiki/GENIUS\\_HARC#Deployment\\_STAT](http://wiki.realitygrid.org/wiki/GENIUS_HARC#Deployment_STAT)  
**US for the deployment status of the GENIUS project****
    - iv. Approximately how many users of this software are there? **Unknown, but would guess a small number of projects with on the order of tens of users**
    - v. Approximately how many developers support this product? **One part time developer currently with a full time developer being hired.**
- 2. Functionality. Does the tool support the following functionality at this time? (also indicate if the functionality is planned in the future and a timeline, if known)
  - a. Advanced reservations of individual systems? **Yes**
  - b. Co-scheduling/co-allocation across more than one system? **Yes**
  - c. Request a reservation/co-allocation starting at a specific time? **Yes**
  - d. Request a reservation/co-allocation starting within a window of time? **Yes**
  - e. Request a reservation/co-allocation with an exact number of nodes/resources? **Yes**
  - f. Request a reservation/co-allocation with a range of number of nodes/resources?  
**Yes**
  - g. Request a reservation/co-allocation on an exact set of systems? **Yes**
  - h. Request a reservation/co-allocation on a subset of a candidate set of systems? **Yes**
  - i. Query the status of a previously made reservation/co-allocation? **Yes**
  - j. Query for when a specific reservation/co-allocation can be made? **Yes**
  - k. Provide useful information about why a reservation/co-allocation cannot be made so that a user can adjust their request? **Yes** What are examples of reasons the tool provides?  
**Debugging output is available from the client that indicates problems with communication between client and servers, problems contacting resource**

**managers on target resources, timeout problems, authentication problems, etc.**

- l. Submission of reservation requests directly to the local scheduler (i.e. without going through any global services)? **No, the client contacts a set of centralized services which in turn communicates with a resource manager running on a target system.**
  - m. Please list other relevant functionality not described above. **Can query for the status of a reservation and can cancel a reservation.**
3. Installation
- a. Were the installation instructions clear? **Yes, but rather complex with many prerequisites such as additional Perl modules and specific versions of Java.**
  - b. What Resource Managers does the tool interface with? (e.g. LSF, PBS, LoadLeveler, SGE) **PBS, Catalina, LoadLeveler**
  - c. Does the scheduler require any modifications to local resource manager? **No, not to the resource manager, but the underlying scheduler needs to support user settable advance reservations and it must be configured to allow these type of reservations to be made.** Are these modifications straightforward? **Yes**
  - d. How long did installation take in hours of work? **Installation consists of 3 parts; the client, the resource manager, and one or more acceptors. The client software install is relatively easy and should be able to be completed in a matter of a few hours. The resource manager installation is more complex and may require a couple of days. The acceptor is significantly more complex with many components like tomcat and apache and may require several days depending on the familiarity of the installer with these other technologies.**
  - e. What additional software is required in order to support the tool and where must it be installed? For each software dependency, is that software already in CTSS?
    - i. Each Teragrid resource (for example, GridFTP) **For the client, additional perl modules on the system**
    - ii. Somewhere on the TeraGrid (for example, MDS, Myproxy) **One or more acceptors could be deployed on independent system(s). The acceptors require**
    - iii. On the same machine as the metascheduler (for example, OS, MySQL) **The HARC resource manager needs to reside on a system with access to the local resource manager and local scheduler. The resource manager requires**
  - f. Did you ask any questions of the developers? **Yes** If so, were the developers responsive? **Very much, but the main developer is no longer with the project**
  - g. What customization was necessary to get the software to work? **None** Was this customization easy or difficult? **N/A**
  - h. Are there installation problems that you expect would occur on many installations? **No**
  - i. For the software components that would be installed by RPs, are there any barriers to installing these components automatically as part of a CTSS kit? **Yes, due to licensing issues or other software?**
4. Operation
- a. How reliable is the software (failures/week)? **Seems quite reliable**

- b. What failures were encountered? **Some failures were experienced with some of the systems at LSU that were running the acceptors**
  - c. Does the software provide logging? **Yes** Can the amount of logging be adjusted? **No**
  - d. What amount of resources are typically used by the software? On what systems? (e.g. central server, login node) (e.g. disk space, physical/virtual memory, CPU time) **Very little**
5. User Experience
- a. What is the quality of the user documentation? **Client documentation is sufficient. Brief command syntax is available for each command and text files resident in the client distribution provide additional information and examples.**
  - b. What client interfaces are provided (GUI, command line, web interface, etc.) **Command line**
    - i. For each user interface evaluated (e.g. GUI, API, command line):
    - ii. Provide the interface name: **command line**
    - iii. Is it well documented? **Yes, there are clear examples on the wiki and via text pages within the client distribution**
    - iv. Is it easy to understand and use? **Yes**
    - v. Are there any changes to the interface that would improve it? **Node ranges in reservation requests would be good. Documentation available in man page format would be convenient**
  - c. Where any problems encountered? (e.g. documentation not matching interface, unimplemented features) **No**
  - d. Are the error messages clear and helpful for debugging problems? **Not always. An example is "RM did not respond". This could be due to a large number of issues. It is not easy to identify the cause of this.**
  - e. What is the average response time of the software? **Depends on the underlying local scheduler. Pretty fast for Moab, slow for Catalina**
  - f. How does the software perform under load? At what amount of load does the software begin to respond slowly? (e.g. twice as slow as unloaded response time) **Untested**
  - g. For each TeraGrid user helping evaluate:
    - i. Does this software meet your needs? **Yes, the GENIUS project uses this co scheduling implementation exclusively**
    - ii. Is this your preferred software for performing advance reservation and/or co-scheduling? **Yes, for the GENIUS project**
6. Any other evaluator comments?
- The major uncertainty associated with this implementation is the deployment and management of the Acceptor systems. While the implementation provides fault tolerance it is at the cost of complexity and cost. The complexity is that additional Acceptor software needs to be deployed and supported with an inherent support cost as well as hardware systems needing to run the acceptor software. The testing has been done using existing acceptor systems that are managed by LSU. While the acceptor software was deployed on a test system its installation, configuration, and testing required significant time and effort. Therefore, the deployment and support**

**of this software is non trivial and should be considered when making a recommendation for its wider deployment on TeraGrid.**