RUDA, A Distributed Grid Accounting System*

M.L. Chen, A. Geist, K. Chanchio, D.L. Million
CSM Division, *Oak Ridge National Laboratory*

Abstract

This paper presents a distributed resource usage data management, accounting, and allocation System, RUDA[1]. It is designed and developed for Science Grid of the Department of Energy. To meet the challenges of the Grid environment, RUDA performs resource usage data management and accounting without centralized servers or databases, and supports heterogeneous resources with no significant impact on their local systems. It can easily be integrated into any Grid infrastructures, and maintains the integrity of the Grid security features.

1. Introduction

Modern science and technology, such as High Energy and Nuclear Physics, Astronomy, Climate and Materials science, are increasingly collaborative and span wide disciplinary and geographical areas. They often demand huge resources of computing, storage, and instruments, which individual research institutes could not possess. The widely deployed Internet links geographically distributed resources and makes resource and data sharing possible. However, it is still awkward for users to overcome the barriers formed by specific software of different vendors and sites, and to integrate a diverse set of resources and tools into a problem-solving environment. A distributed infrastructure - Grids - was created in early 2000 [1]. Grid technology, such as the protocols and services developed by Globus [2], reduces the barriers and enables flexible, controlled resource sharing on a large scale. Driven by the demand and attracted by the promising future of Grids, Grid applications multiply swiftly and in turn drive the rapid development of Grid technology.

While many Grid services are maturing, Grid accounting still remains an issue of research. It is well understood that the ability to dynamically monitor, manage, and account for usage data of Grid resources, such as computational, network, and storage resources, is one of the key issues for putting Grids into production. However, the Grid environment, which contains a large and growing number of widely distributed sites with heterogeneous resources, poses great challenges to Grid accounting. Rapid changing of Grid software infrastructures and the increasing security concerns of Wide Area Network (WAN) add additional complications. Therefore, for an accounting system functioning in a Grid environment, except managing and accounting for the resource usage data dynamically, it should be flexible, decentralized, and scalable [3]. Though many methods and tools exist for resource usage management and accounting, they tend to be local and vary from site to site, even from machine to machine. They are centralized systems, of which the scalability is limited by the load capability and data size of the centralized server and database. In addition, there is lack of convenient means for remote users to monitor their resource usage and accounting data. The existing accounting methods and tools could not satisfy the requirements of Grid accounting.

RUDA has been designed and developed at Oak Ridge National Laboratory (ORNL) for Science Grid (SG) [4] of the Department of Energy (DOE). To meet the challenges of the Grid environment, RUDA is designed as a distributed accounting system. It collects resource usage data from local systems, such as job schedulers, local accounting systems, and so on, of individual resource entities and stores the data locally. For users, projects, collaborations, sites, and other organizations to perform accounting tasks, RUDA can build hierarchical structures while still storing data in a distributed manner. RUDA software uses a client/server structure. The RUDA server is constructed of core software, which performs major services, plus interfaces which accommodate customized routines for communications with local systems and

[1] This work is sponsored by the U.S. DOE Office of Science under the auspices of the Scientific Discovery through Advanced Computing program (SciDAC).

outside world. This structure provides flexibility to support widely diversified resources and to minimize the impact on local systems. RUDA is an essentially self-contained model and can be easily integrated into any Grid software infrastructures. Security features are among the most important considerations of RUDA design, as well as convenient remote data monitoring and system administration.

A RUDA prototype has been developed to show the feasibility of the RUDA design. It has been deployed onto a number of computers of the Science Grid test bed [4] and Earth System Grid II (ESG) [5], and used daily within the ESG project to monitor resource loads generated by large-scale file movement operations and provide accounting information.

The following sections present a more detailed overview of the RUDA system. Section 2 outlines the architecture of the system. Section 3 describes the major features of RUDA and the mechanisms behind them. The software structure of the prototype is described in section 4, and section 5 discusses its performance. Section 6 is a summary.

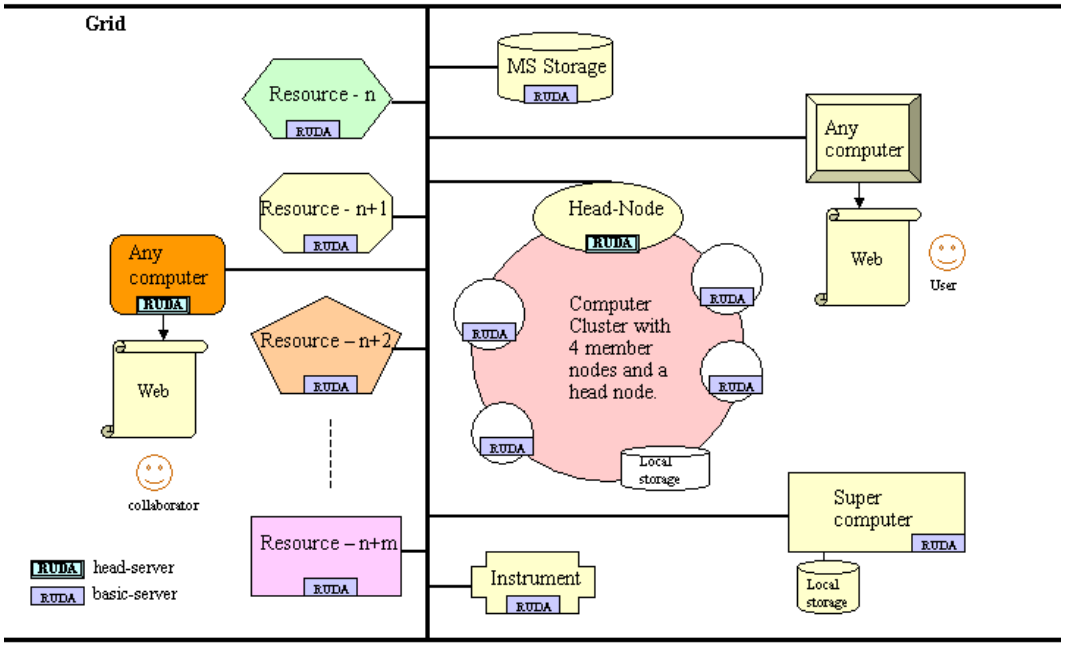

Figure 1. System architecture – 1: Grid environment

2. System Architecture

In a Grid environment as shown in Fig. 1, resources such as computers, storage systems, and instruments owned by unconnected sites are distributed in a large geographical area which may cross states, countries, or even continents. There are also computers, typically laptops or PCs, which may not provide resources to the Grid but be connected or temporarily plugged in by users to perform resource usage-related tasks, such as data monitoring and accounting. The basic assumption is that all these entities are connected through high-speed networks and that a Grid infrastructure has been established in this environment. Substantial numbers of Grid resource users are organized into research project groups or collaborations led by Principle Investigators (PIs).

RUDA software is constructed on a client/server model. The client processes are for users and administrators to interface with the server, and to perform data monitoring and server reconfiguration. The multithreaded server provides services such as dynamic configuration, data collection and management, accounting, and web based data monitoring. The server can be configured in two running modes and called basic-server and head-server respectively.

In a RUDA system, each resource entity has a RUDA basic-server running as a daemon. The basic-server collects resource usage information periodically from the local system, according to a configurable time schedule. In the case of a computer cluster resource, a basic-server runs on each of its member nodes, and a head-server runs on its head node. The head-server summarizes the resource usage data collected from basic-servers of the

member nodes, and manages them collectively as a single data set. The cluster appears as a single resource entity to the Grid [Fig. 1].

For data management and accounting purposes, one can run a head-server on any computer on the Grid and configure its member resource entities. The server then collects selected data segments from the basic-servers running on specified members, and performs management and accounting for these data collectively. For instance, a project PI can run a head-server which, according to the configuration, collects relevant resource usage data from all resources the project uses, and perform accounting collectively for his group members and the project. The RUDA system also provides the capability to build hierarchical structures for large organizations to perform management and accounting.

Figure 2 shows a RUDA system example. Sites A and B, which may be geographically far away from each other, provide 4 and 10 resource computers respectively to Grid computing. Each individual resource computer runs a basic-server to collect resource usage data from its local system and store them locally. The server also monitors and accounts for resource usage for each individual computer at the levels of jobs, users, and projects or collaborations. C and D are two collaborations who use some of the resources. To manage their resource usage data, each of them runs a head-server on a machine on the Grid. The server is configured to collect the resource usage data of their collaborators from all relevant resource computers and to manage the data as a single data set.

Figure 2 also shows a simple example of RUDA hierarchical structure. The resource computers of site B belong to divisions 1 and 2. Each division sets up a RUDA head-server to manage resources usage data and perform accounting for their Grid users, projects, and collaborations. The division head-servers need not to collect detailed data at job level unless desired. An administrator wants to monitor the current resource usage and accounting status of all Grid projects and collaborations at site B. Instead of collecting data from 10 resource computers, he/she runs a head-server and collects the data of project/collaboration level from two divisions' head-servers. Large organizations can build multi-layer hierarchical structures to facilitate management at different levels and to minimize the traffic on the wide area network.

The web based monitoring functions of the RUDA server provides services for authorized users accessing their data through the web. The head-server provides the collective data set information at project/collaboration, user, and job levels. It also provides the links to its member server web sites for collaborators to navigate to web pages with the detailed data on each computer. Grid users can monitor their current resource usage data from any computer on the Grid, which needs not to have RUDA software installed. The RUDA web interface also allows authorized users to request resource allocations and PIs to redistribute the allocation among their group members. Command Line Interfaces (CLIs) are provided for users to query resource usage data and accounting information for specified resources, and for administrators to do management tasks.

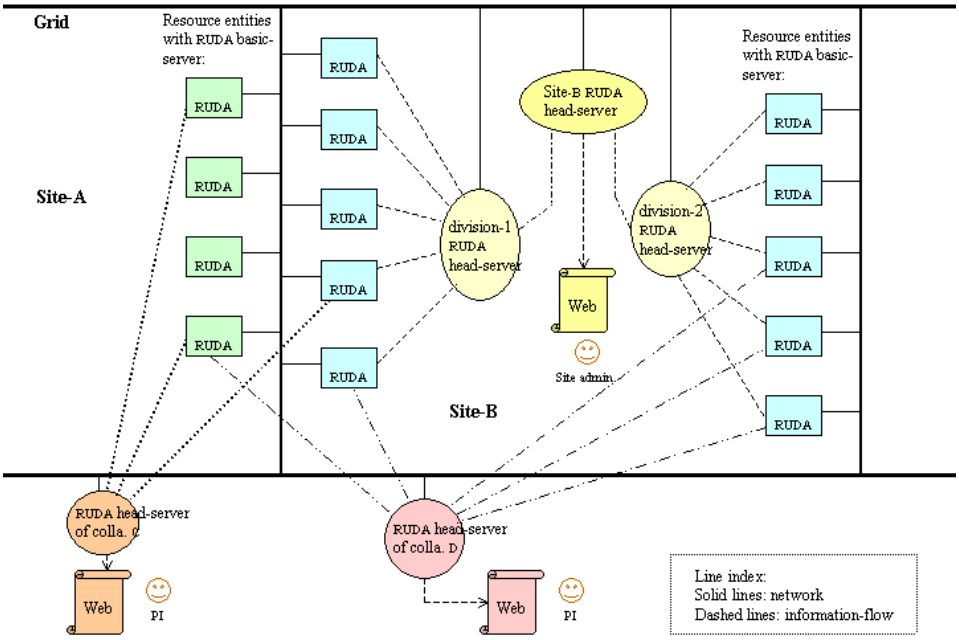

Figure 2. System Architecture –2 : A segment of RUDA system

3. Design Features and Approaches

RUDA is designed to meet the criteria that, in addition to dynamically monitoring, managing, and accounting for the resource usage data, a Grid accounting system should be decentralized, scalable, flexible, and having minimum impact on local systems.

3.1. Dynamic data management and accounting

RUDA monitors, manages, and accounts for Grid resource usage data dynamically. It is able to monitor and account for resource usage of a job at job run time, if the relevant information can be extracted from the local system, and at any time after the job is completed.

The RUDA basic-server runs on each resource entity and, by way of a customized interface routine, collects resource usage data from the local system. The data collection process runs according to a configurable time schedule, and it can also be initiated by a user request.

The RUDA server owns a fully configurable dynamic data structure. The structure is built of data blocks which are organized in layers and chains. The data blocks contain categorized resource usage data of collaborations/projects, users and their current running jobs if any. On the top layer of the structure, a data block records the current resource status of this machine. Upon receiving newly collected data, such as, wall time, CPU time, storage space used by each running jobs, the server updates the data recorded in each job data block and performs accounting for each job according to a customized formula. The formula used in the prototype is a simple summary over weighted resource usages in standard charge unit:

$$total\_change = \sum_{i=1}^{n} (Usage\_amount(i)) \times (weight(i))$$

Here, n is the total number of resource categories in the server data structure, i spanning from 1 to n represents each specific resource category. Weight(i) presents the weight factor of resource category-i. The weight factors are determined, through configuration parameters, by the local resource owner according to their administrative policy. The formula and the weight factor information are available for users upon request through a client CLI. The current accounting result of a running job is recorded in its data block.

The server summarizes the newly updated usage data and accounting results of relevant current jobs for each user, project/collaboration respectively. It calculates the charges of each resource category and the totals that are resulted from the current jobs, for each user and project/collaboration and stores the results in corresponding data blocks. All resource usage data and charges of completed jobs are accumulated and recorded properly in the data structure before moved into a local database. Based on the data of both current and completed jobs, the server calculates, for each user and project, the individual categories and total charges accumulated since the beginning of current accounting period. By means of a head-server, any authorized user, project/collaboration, and site can monitor and account for their Grid usage data dynamically.

3.2. Scalability

The large and growing scale of the Grid environment poses great challenges to the accounting system design. To ease the concern of server load and database size of a centralized system, RUDA is designed to be a distributed system. The data collection and storage are local on each resource entity and therefore decoupled from the scale of the Grid. The head-server of RUDA, although it can be used for computer clusters and virtual machines, is specially designed for sites, collaborations and other organizations to perform usage data monitoring and accounting. A PI can run a RUDA head-server and configure a number of basic-servers as its members. The head-server, by means of Grid software infrastructure, initiates a client process on its remote member machine with data selection criteria as input. The criteria specify the required project name, and user names if not all users are selected. They can also define the lowest data block level, and this function can be used, for example, to cut off the detailed data of individual jobs. This process selects concerned data segments according to the criteria and transfers them back to the head-server. The server organizes the data from all its members together and manages them as a single collective data set. The data size of the head-server managed depends on the size of the collaboration or project, not on the scale of the Grid. Furthermore, since the resource usage history is accumulated and recorded locally by the RUDA basic-server in individual resource entities, the PI can run the head-server only when he/she wants to access the data.

RUDA also provides the capability to organize a hierarchy structure to perform accounting tasks when needed. A PI can run a head-server and configure a group of head-servers and/or basic-servers as its members. The high level head-server then collects selected data segments from its lower level members and manages them collectively. Since the data each server manages is irrelevant to the Grid scale, this design again does not raise the concern about server load and data size when the Grid scales up. Besides, constructing a properly organized hierarchy structure to perform management and accounting may significantly reduce the wide area network load and contribute to the scalability of the

system. For example, an international collaboration uses resources located in many areas of a few countries. A head-server hierarchy can be built with lowest level servers for each area, middle level servers for each country, and a high level server for the collaboration. In comparison with one head-server structure, this scheme keeps most network traffic localized and minimizes wide-area and especially international traffic.

3.3. Flexibility

RUDA software is designed to be applicable to heterogeneous resources with various local systems. By means of an isolation layer, major RUDA server functions are separated from the environment. Interfaces of the RUDA basic-server accommodate customized RUDA data collection methods to communicate with resource local systems. For example, to enable communications between the RUDA basic-server and a new type of local accounting system, a developer creates a customized data collection routine, which extracts the relevant data from the local accounting system and loads them into a RUDA standard data cargo. This routine can then be plugged into the RUDA basic-server to perform the data collection task without modifying the resource local accounting system. This structure allows RUDA to support widely diversified resources and minimizes impact on local systems.

To avoid the complications caused by changes in Grid software infrastructure, RUDA software is built to be essentially self-contained. RUDA uses Grid software infrastructure for WAN communications and data transfers, but do not depend on details of the infrastructure. The utilization of Grid infrastructure is limited to its high level application programming interfaces, for which changes are less frequent. Infrastructure modifications beneath the high level API commands have no effect on the operation of RUDA. Furthermore, the API calls are assembled in a couple of customization routines. By modifying a few lines of customization routines, RUDA can be easily integrated into Grid software with different infrastructures.

3.4. Security

Security is certainly one of the most crucial issues for Grid applications. Grid software development has made huge efforts to gain good security in network communications. For example, Grid Security Infrastructure of Globus extends existing standard protocols and APIs, such as SSL/TLS, X.509 & CA, GSS-API to single sign-on and delegation, and has gradually formed a consummate security system [6-8].

RUDA uses Grid infrastructure to ferry information/data through the WAN and runs both server and client locally. Grid users can use the RUDA CLIs to access a remote computer and transfer selected resource data. These RUDA CLIs are built based on the Grid infrastructure APIs, in which the Grid security features are built. For example, the prototype of RUDA uses Globus infrastructure to run the remote RUDA client procedure to obtain required data sets and to transfer the data back to the user. This design simplifies the security issues of WAN communication for the RUDA accounting system, and maintains the integrity of Grid infrastructure security features.

3.5. Manageability

To provide maximum control to the site administrator, the RUDA server is fully configurable. According to local resources and policies, administrators can modify control parameters, such as project and user maps, data update time interval, data backup method, and so on, which consequently change the amount of memory, disk space, and CPU-time used by the RUDA server. RUDA provides CLIs for easy configuration operation. Authorized administrators can use these CLIs to control the RUDA server remotely from any computers on the Grid. In addition, the RUDA server supports dynamic reconfiguration. The site administrator can reconfigure the server whenever needed at run time. After an incident, such as power failure, the server can automatically recover current RUDA data structures without losing data.

4. Prototype Software

A prototype of RUDA has been developed on a DOE Science Grid test-bed at ORNL. The prototype contains most of the major components of RUDA to test the feasibility of RUDA design. A client/server structure is established to provide users and administrators convenient data access and system configuration.

The RUDA server [Fig. 3] is a multi-threaded process which performs tasks such as dynamic configuration, resource usage data collection, data management, accounting, and Web based data monitoring. The server is fully configurable. It updates its configuration automatically at a frequency specified by a configuration parameter to allow the site administrator to configure the server dynamically. Three data collection interfaces are provided which accept input data files and data structures in RUDA standard forms. Customized data collection routines are called by the interfaces to input data from various local systems. The customization routines are built in a way which provides skeletons to minimize the effort of adapting analogous local systems. According to the configuration, the data loading process inputs data through a proper data collection interface. Upon receiving new data, the data management process

first checks the validity of the data owner according to the project and user maps provided by the system administrator. The data which belong to valid projects and users will be accepted and processed. The current version of the prototype only processes data of central processing unit time, wall time, and memory high water mark. However, the data structure shown in Fig. 4 has been set up for a full data set defined according to Global Grid Forum suggestions, a DOE lab survey, and an investigation into ORNL local accounting systems [9-11]. The user/project resource usage data are calculated from historical records and current jobs, and reflect the resource usage dynamic status. The data are then recorded in the data structure. The structure management process checks through the data structure, records completed jobs into history data sets and remove them from the data structure, remove obsolete/invalid user/project, and records the updated data set in a local database. The status of the computer resources is also recorded. Lower limits of available resources can be specified by configuration parameters. In case an available resource is lower than its threshold, emails are sent to warn the users/admin on a mailing list.

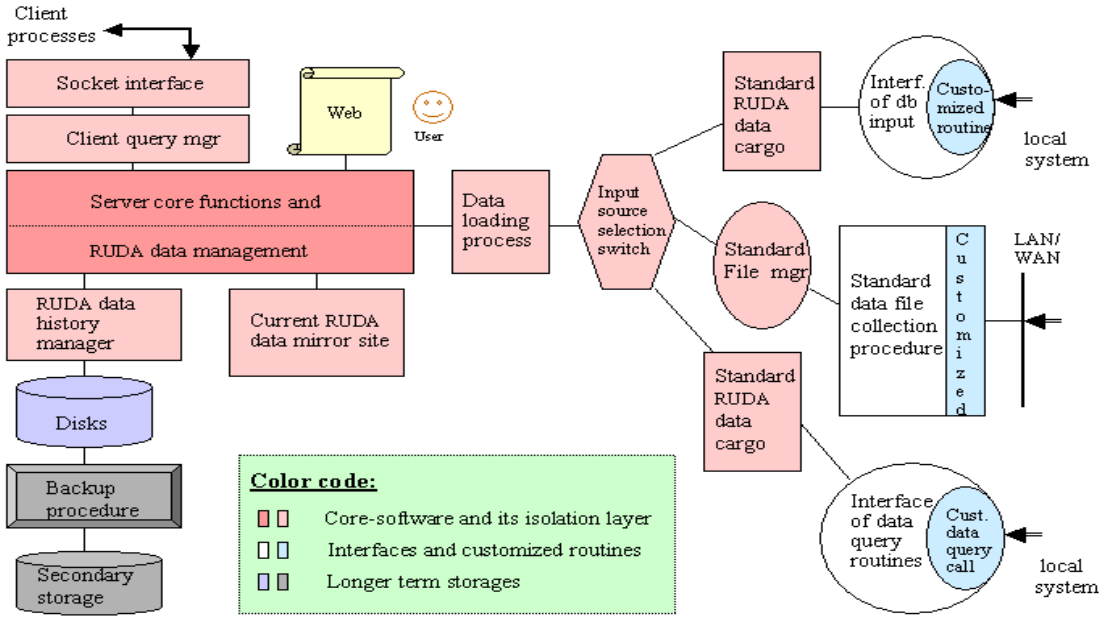

Figure 3. RUDA server software structure

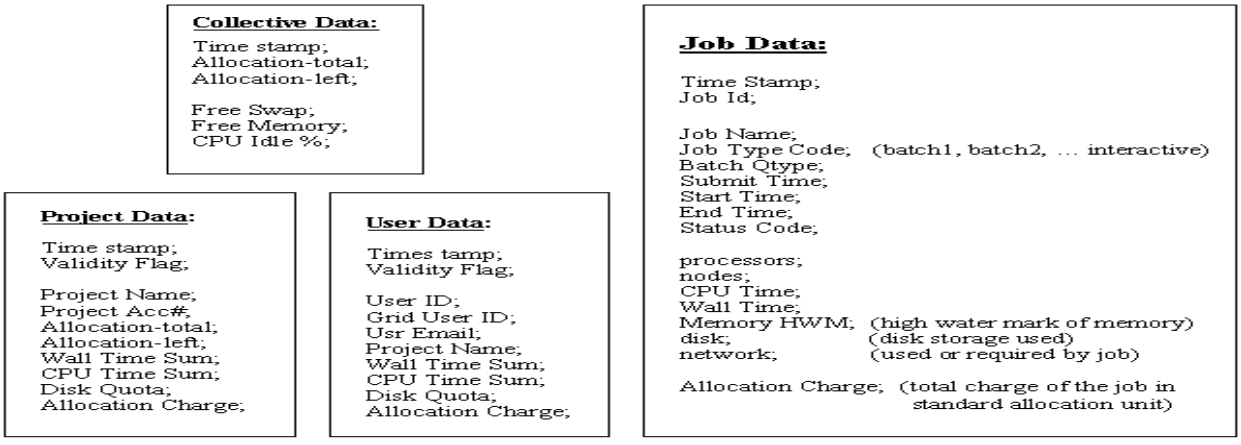

Figure 4. RUDA Data Sets

The server provides web interface services including resource usage data monitoring and an allocation application interface. A head-server web page displays the Grid resource usage data collectively for each relevant project and user. It also provides the links to its member server web sites. The basic-server web facility displays data summarized at project and user level, and navigates to detailed current resource usage and accounting information of individual users. The current resource status of the concerned resource entity is also displayed on the web. To attract better attention, the background colors of the status display change from green to yellow or red while the resource status changes from sufficient to low or too low respectively. Allocation on major resources of Science Grid requires pre-approval, and automatic allocation is not allowed in current practice. RUDA prototype provides a web interface for allocation application. By means of this web interface, a PI can check the available resources and apply for allocation. The requests are sent to the resource administrator through email for approval. The status of the allocation application can be monitored through the web interface. And an email will be sent to the applicant upon approval. PIs can also monitor and redistribute the allocation among their project members by using this interface.

The client processes [Fig. 5] are provided for users and administrators to interface with the server locally or remotely. The client processes send user queries to a server and return the required data or processing acknowledgment back. CLIs are provided for users to get online help and to query a specific part or whole set of resource usage data. Administrators can use CLIs to configure and control the server, or dump the whole data set for diagnostic purposes. By means of Globus APIs, the infrastructure used in the prototype, users and administrator can access the server from any computers on the Grid remotely.

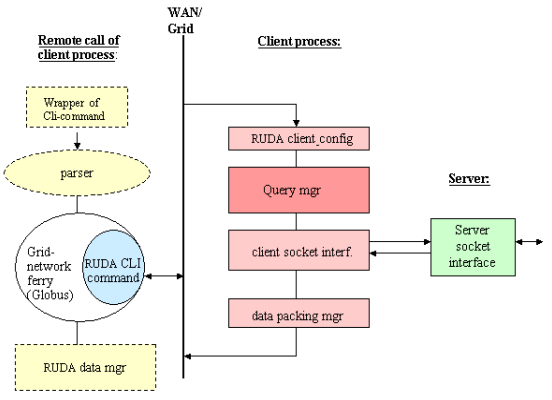

Figure 5. RUDA client software structure

5. Deployment of Prototype

RUDA has been running smoothly on a number of Solaris, Linux, and AIX machines of the Science Grid test bed at ORNL and the Earth System Grid II at the National Center for Atmospheric Research. RUDA servers collect and monitor the resource usage data, and perform the accounting of users/projects according to a customized formula successfully. Both modes of RUDA servers function as expected. RUDA now is being used daily within the Earth System Grid II project to monitor loads generated by large-scale, long-haul data movement operations and provide resource usage accounting information. The success of the RUDA prototype operation has proved the feasibility of the RUDA design.

6. Summary

RUDA, a Grid Resource Usage Data Management, Accounting, and Allocation system is designed for the DOE Science Grid. RUDA is a distributed accounting system with features specially designed to meet the challenges of widely diversified resources and the growing scale of the Grid environment. A prototype of RUDA has been developed and tested on a Science Grid test bed, and is currently used daily within the Earth System Grid II project.

To benefit the Grid communities, the RUDA software packages have been published on the SG web site. One can check http://www.sciencegrid.org to find more detailed documentation and RUDA software packages.

References

[1] Foster, I. & Kesselman, C. (Eds), "The Grid: Blueprint for a New Computing Infrastructure," Morgan-Kaufmann, 1999.
[2] Foster, I. & Kesselman, C. Globus, "A Toolkit-Based Grid Architecture," ibid, p. 278, Morgan-Kaufmann, 1999.
[3]Bill Thigpen & Tom Hacker, "Distributed Accounting on the GRID," Global Grid Forum (GGF), March 4-7, 2001.
[4] http://www.doesciencegrid.org.
[5] http://www.earthsystemgrid.org.
[6] http://www.globus.org/security.
[7] Eric Rescorla, "SSL and TLS: Designing and Building Secure Systems," Addison-Wesley, 2000.
[8] Richard E. Smith, "Authentication: From Passwords to Public Keys," Addison-Wesley, 2001.
[9] http://www.gridforum.org.
[10] Mi young Koo, "Usage Record Fields – Survey Results and Proposed Minimum Set," GGF, October 2002.
[11] Laura F. McGinnis, "Resource Accounting – Current Practices," GGF, February 2001.