GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration

Alexander Barmouta
Dept. of Computer Science and Software Engineering
The University of Western Australia
Nedlands, Western Australia, 6009
barmouta@csse.uwa.edu.au

Rajkumar Buyya
Grid Computing and Distributed Systems (GRIDS) Lab
Dept. of Computer Science and Software Engineering
The University of Melbourne, Australia
raj@cs.mu.oz.au

Abstract

Computational Grids are emerging as a new infrastructure for Internet-based parallel and distributed computing. They enable the sharing, exchange, discovery, and aggregation of resources distributed across multiple administrative domains, organizations and enterprises. To accomplish this, Grids need infrastructure that supports various services: security, uniform access, resource management, scheduling, application composition, computational economy, and accounting. Although several initiatives are engaged in the development of Grid technologies, Grid accounting issues are yet to be addressed. To overcome this limitation, we propose a new infrastructure called Grid Bank that provides services for accounting. The support of computational economy and accounting services can lead to a self-regulated accountability in grid computing. This paper presents requirements of Grid accounting and different economic models within which it can operate and proposes a Grid Accounting Services Architecture to meet them. The paper highlights implementation issues with a detailed discussion on the format for various records/databases that the GridBank needs to maintain. It also presents protocols for interaction between the GridBank and various components within Grid computing environments.

Keywords: Computational economy, Grid accounting, GridBank, Payment schemes, and Grid Scheduling.

1 Introduction

As the trend towards Internet-based computing continues, applications are able to harness distributed resources for solving large-scale computationally intensive problems. Resources interconnected via the Internet with middleware supporting remote execution of applications constitute what is called the computational Grid [2,11,12,14]. The Grid couples a variety of heterogeneous computational resources, storage systems, databases and other special-purpose computing devices and presents them as a unified integrated resource. In the global Grid environment, users submit their applications to a Grid Resource Broker, which then discovers resources, negotiates for service costs, performs resource selection, schedules tasks to resources and monitors their execution. Resource providers advertise their services with the discovery service and run the Grid Trade Service (GTS) used by the Grid Resource Broker (GRB) to negotiate service cost. Such a setup allows open market trade of computational services to take place on the Grid. The Grid resource services are offered at different prices, and those prices are negotiated using one of several economic models from the real world [2,4].

It was observed that the utility delivered by resources is enhanced when resource allocation is performed based on user's quality-of-service (QoS) requirements/constraints (e.g. deadline and budget constraints to minimise time/cost) [2]. In a global computing environment all users would prefer to use powerful resources, which would cause some resources to be oversubscribed and others undersubscribed. This is where computational economy and suitable service pricing strategies come into play. Resource owners are permitted to solicit an open market price in a way that achieves maximum profit and resource consumers are allowed to choose resources that meet their QoS requirements. That is, when there is less demand for resources, the price is lowered; when there is high demand, the price is raised. This helps in regulating the supply-and-demand for access to Grid resources and services.

The Gridbus project [3] is developing technologies that enable service-oriented cluster and grid computing. It is driven by a distributed computational economy approach to the sharing, exchange, discovery, and aggregation of resources. It builds on the GRid Architecture for Computational Economy (GRACE) framework for managing the supply-and-demand for resources based on users' quality-of-service requirements [2,4]. The GRACE framework has been implemented in the following components for resource selection and scheduling:

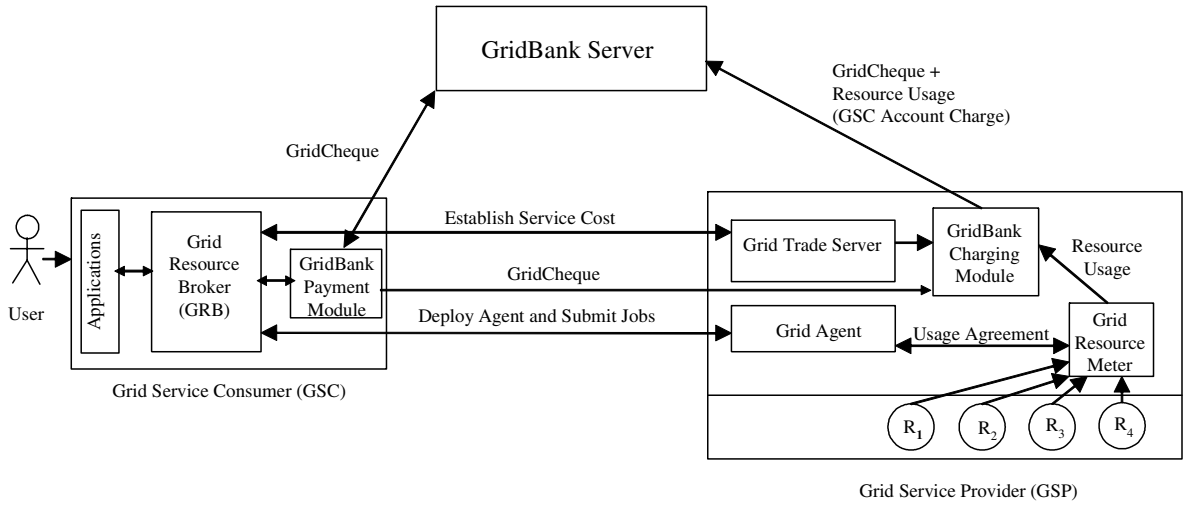
- Nimrod-G (Grid Resource Broker designed for parameterized applications)
- Grid Resource and Market Information Server
- Grid Open Trading protocols and APIs
- Grid Trade Manager (part of broker involved in establishing service price)
- Grid Trade Server (for resource owners)

The Gridbus project's service-oriented computing technologies support the end-to-end quality-of-services driven by computational economy at various levels—cluster, peer-to-peer (P2P) networks, and the Grid—for the management of computational, data, and application services. At the cluster level, the Libra scheduler has been developed to support economy-driven cluster resource management [3]. It is used within a single administrative domain for distributing computational tasks among resources that belong to a cluster. At the P2P network level, the CPM (compute-power-market) infrastructure is being developed through the Jxta community [7]. At the Grid level, various tools are being developed to support a QoS based management of resources and scheduling of applications. To enable performance evaluation, a Grid simulation toolkit called GridSim has been developed. It supports the modelling and simulation of application scheduling on simulated Grid resources. To support the accounting of resource or service usage and enable sustainable resource sharing across virtual organizations, we are developing Grid Accounting Services infrastructure.

The early efforts in Grid computing and usage scenarios were mostly academic or exploratory in nature and did not enforce the grid economy mechanisms. The recent move towards a multi-institutional production scale Grid infrastructures such as the TeraGrid facility, the need for Grid economy and accounting is being increasingly felt [6]. In order to enable the sharing of resources across multiple administrative domains, the accounting infrastructure needs to support unambiguous recording of user identities against resource usage. In the context of the Gridbus project, an infrastructure providing such a service is called the GridBank.

GridBank (GB) is a secure Grid-wide accounting and (micro) payment handling system. It maintains the user's (consumers and providers) accounts and resource usage records in the database. It supports protocols that enable its interaction with the resource brokers of Grid Service Consumers (GSCs) and the resource traders of Grid Service Providers (GSPs). It has been primarily envisioned to provide services for enabling Grid computing economy; however, we envision its usage in E-commerce applications. The GridBank services can be used in both co-operative and competitive distributed computing environments.

GridBank can be thought of as a web service for Grid accounting and payment. GridBank uses SOAP over Globus toolkit's sockets, which are optimised for security. Clients use the same user proxy/component to access GridBank as they use to access other resources on the Grid. A user proxy is a certificate signed by the user, which is later used to repeatedly authenticate the user to resources [9,11,17]. This preserves the Grid's single sign-in policy and avoids repeatedly entering the user password. Using existing payment systems for the Grid would not satisfy this policy.



1) GRB negotiates service cost per time unit (e.g. $ per hour)
2) GridBank Payment Module requests GridCheque for the GSP whose service GSC wants to use. GridBank issues GridCheque provided GSC has sufficient funds.
3) GridBank Payment Module forwards GridCheque to GridBank Charging Module.
4) GRB deploys Grid Agent and submits jobs for execution on the resource.
5) Grid Resource Meter gathers Resource Usage Records from all resources used to provide the service, optionally aggregates individual records into one Resource Usage Record and forwards it to the GridBank Charging Module. Grid Resource Meter optionally performs usage check with Grid Agent.
6) GridBank Charging Module contacts GridBank and redeems all outstanding payments. It can do so in batches rather than after each transaction.

Figure 1: Interaction of GridBank with other Grid components.

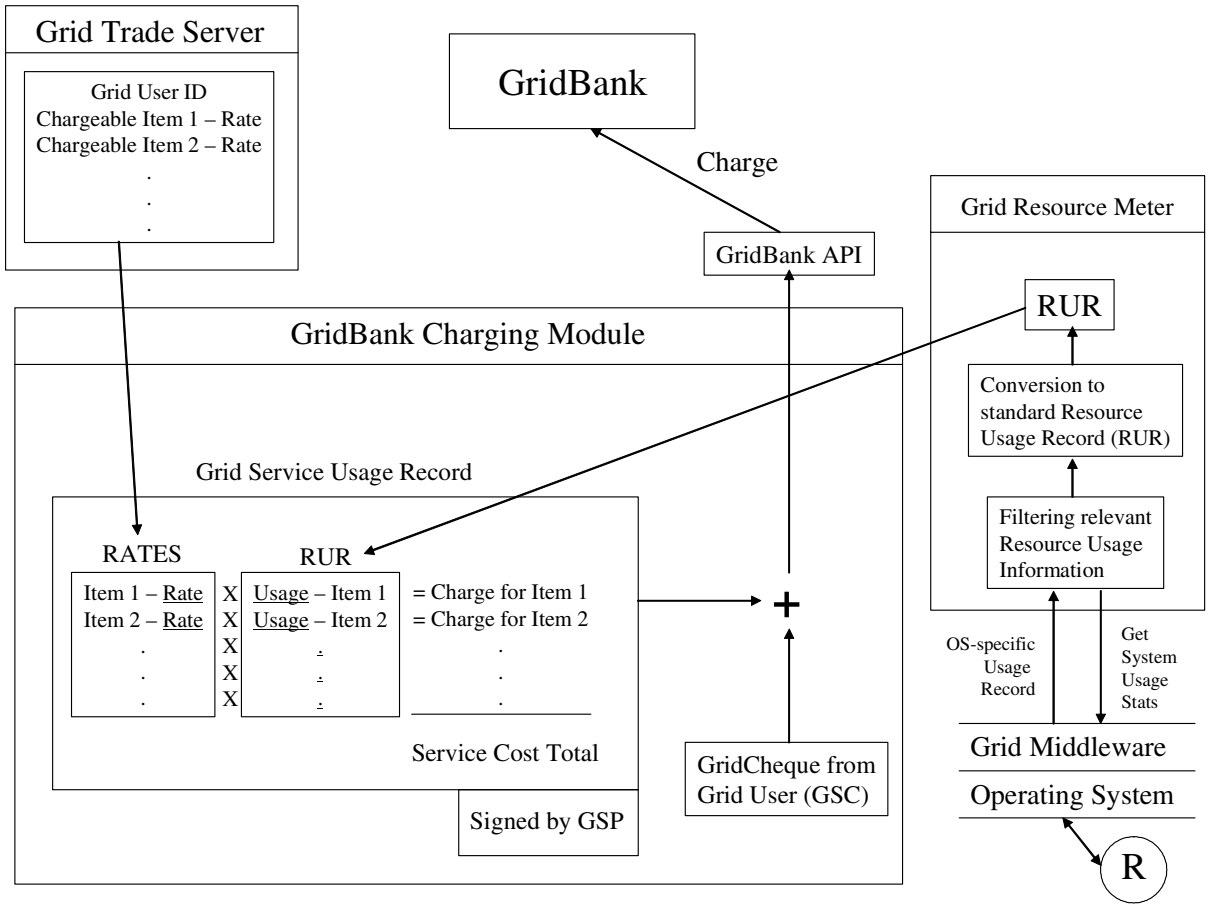


Figure 2: Interaction between the GTS, GBCM, GRM and GridBank.

2 Components Interaction

Use Case: The interaction between GridBank Server and various components of Grid is shown in Figure 1. GSPs and GSCs open an account with GridBank. Then, the user submits the application processing requirements along with the QoS requirements (e.g., deadline and budget) to the Grid Resource Broker (GRB). The GRB interacts with GSP's Grid Trading Service (GTS) or Grid Market Directory (GMD) to establish the cost of services and then selects a suitable GSP. It then submits user jobs to the GSP for processing along with details of its chargeable account ID in the GridBank or GridCheque purchased from the GridBank. The GSP provides the service by executing the user job and the GSP's Grid Resource Meter measures the resources consumed while processing the user job. The GSP's charging module contacts the GridBank with a request to charge the user account. It also passes information related to the reason for charging (resource usage record).

2.1 GSP Components Interaction

The Grid Resource Meter (GRM) module will interface with the local resource allocation system (e.g. cluster scheduler) or user-level Grid agents (such as Nimrod-G agent) to extract resource usage information. The interfacing can be mediated through Grid middleware tools such as Globus [10], by extending it such that a native operating system usage function is called after a user application finished execution, and another function that is called by GRM to collect the data. Once GRM obtains the raw usage statistics (see Figure 2), it filters relevant fields in the record and passes them to the conversion unit, which generates a standard OS-independent Resource Usage Record (RUR). The RUR is a combined effort of the Grid community at the Global Grid Forum [15,23]. GridBank stores this record in its database, which provides evidence that a transaction took place. RUR is then forwarded to GridBank Charging Module (GBCM) for calculation of the total service cost. GBCM obtains service rates for the user from the Grid Trade Server (GTS). The user ID (Certificate Name) passed to GBCM is contained in the GridCheque (or any other payment instrument). The service rates record generated by the GTS and the Resource Usage Record must conform to each other - for every chargeable item in the rates record there must be a corresponding item in the RUR. Chargeable items to be considered are [2,4] :

- Processors: User CPU time
- Main Memory
- Secondary Storage
- I/O channels (such as networking)
- Software Libraries: System CPU time

The total charge is calculated by multiplying rate by usage for each item and then adding up individual charges. The rate for CPU time is G$ (Grid currency) per CPU hour and the usage is time. The rate for memory and storage is G$ per MB*hour and usage is MB*hour. I/O service rate is G$ per MB and usage is MB of total "traffic". These calculations along with the rates and RUR records are signed by GSP to provide non-repudiation of the transaction, and are submitted together with GridCheque (or other payment instrument) to GridBank Server for processing.

Besides the basic functionality, the GRM provides different levels of accounting information depending on the kind of payment protocol GridBank Charging Module is using. Different protocols might require different resource usage statistics. For example, in Figure 1, each individual resource (R_1 – R_4) used to provide computational service presents its usage record to Grid Resource Meter (GRM). GRM might choose to aggregate individual records into the standard RUR to reflect the charge for the combined GSP's service.

2.2 GSC Components Interaction

Grid Resource Broker (in our implementation Nimrod-G resource broker [2,5]) performs service cost (rates) negotiations with GSP's Grid Trade Server and deploys the Grid Agent responsible for setting up the execution environment on GSP's machine and downloading the application and data from remote locations if they are not already on the machine [2]. However, before the broker can submit a job, a local account on the remote host must exist [17]. Section 2.3 addresses this issue.

The GRB interacts with the GridBank Payment Module to manage funds on user's behalf. The user can then set the budget to prevent overspending. GRB submits a job for execution on the resource in similar fashion as normally (using Globus's submit-job command [17]), but the call is made using GBPM's API such that it initially forwards payment details to GridBank Charging Module in order to authorize access to the service.

2.3 Access Scalability

Grid middleware technologies such as Globus assume that local access to resources is managed by the resource owner who has to create and manage local accounts for each user. Thousands (or even millions) of GSCs can be clients of GridBank and the requirement to have a local account at each resource is simply not realistic. GridBank Charging Module alleviates the problem. This can be achieved by enhancing GridBank to provide "arbitration" and "resource access authorization" services like a market-maker that brings a buyer and seller together. It also requires enhancement to Grid job submission systems.

In order to access the service, the GSC has to present credentials to the GSP. In the context of GridBank we consider such credentials to be a payment instrument that GSC obtains from the GridBank. The payment instrument contains GSP's identification (the Certificate Name) and GridBank account details.

Grid Service Consumer (GSC) initiates the process by negotiating service rates with GSP's Grid Trade Service (GTS). Upon mutual agreement on service rates, Grid Resource Broker (GRB) contacts GridBank Payment Module (GBPM) with request for access to the GSP.

GBPM obtains a payment instruction, for example, a GridCheque, from the GridBank Server. It then forwards the payment and agreed service rates to the GridBank Charging Module (GBCM). GBCM confirms that the payment and service rates offer are valid. If so, the module decides to grant access to the GSC. The access to the GSP is accomplished in the usual fashion by using Globus Toolkit. This implies that the GSP needs a local account to be associated with the GSC [17]. In order to achieve scalability, a small number of (template) accounts are maintained by the system and are dynamically allocated to GSCs as they request for service.

GSP maintains a pool of template accounts [18]. These accounts are local system accounts that are not associated with any particular user. When a GSC contacts GSP to execute some application, provided GSC presents a well-formed payment instrument, GSP dynamically assigns one of the template accounts from the pool of free accounts. GSC's Certificate Name is temporarily mapped to the local account (in grid-mapfile [17]) to indicate the dynamic relationship between the account and current user. GSP retains the fine-grained access control to its resources by specifying permissions on the template accounts.

When application has finished execution, Grid Resource Meter collects information about resource usage against the local account and forwards it to the GBCM. The GBCM calculates total cost based on the resource usage and the mutually agreed service rates. These charges are recorded against the GSC associated with the local account. The GBCM then removes the association by deleting the entry corresponding to the GSC in the grid-mapfile [17] and returning the local account to the pool of free accounts. The GBCM then forwards the payment instrument (e.g. GridCheque) along with the Resource Usage Record to the GridBank for processing.

3 Grid Bank System Architecture

3.1 Payment Strategies

We employed a layered and modular architecture for GridBank to leverage existing technologies and manage them as separate components (Figure 3). This approach allows different payment schemes such as digital cheques, coins, hash chains and other [1,20,21,22,24,25] to be easily integrated with other GridBank modules. Depending on charging strategy, the GSC and GSP can select appropriate protocol and exchange service for currency. The charging policies include [2,4] :

1. Pay before use
2. Pay as you go
3. Pay after use

The first policy is appropriate for services that have a fixed cost, for example, to access a directory service. A simple funds transfer protocol is designed to enable GSC to request funds transfer with the confirmation send to the GSP. The GSC establishes a secure connection with GridBank to provide account details of GSC and GSP as well as amount and URL of GSP. GridBank performs the funds transfer and sends the confirmation to the specified URL of the GSP via another secure channel.

The second policy might be used to eliminate unnecessary trust relationships between GSCs and GSPs. A hash chain scheme based on PayWord [24,25] would allow service consumers to dynamically pay service providers for CPU time or per each computation result delivered.

The third payment strategy emulates credit card payment model. When the service charge is unknown beforehand, the GSC forwards a payment order in the form of a digital cheque to GSP. The cheque is made out to GSP so no one else can redeem it. After computation has finished, GSP calculates total cost and forwards the cheque along with resource usage record to GridBank for processing. This can be done in batches. Such scheme is based on NetCheque protocol [21,25] and relies on public key cryptography.

Secure communication between parties involved in a transaction is provided by the Security Layer (see Figure 3). In our implementation we chose Public Key Infrastructure (PKI) since it is already provided by Globus Toolkit's GSI (Globus Security Infrastructure) [13,17]. Client authentication and authorization are part of GSI. Secure communication between all participants of any GridBank transaction use Globus I/O API, which implements GSS (Generic Security Service) API. GSS API also provides symmetric data encryption based on SSL technologies to securely exchange sensitive financial information.

3.2 Server Architecture

GridBank's server architecture (see Figure 3) consists of several modules that can be enhanced or replaced without affecting other modules. These modules are organized into three layers. Accounts Layer deals with database and account operations. Payment Protocol Layer defines payment schemes, message formats and communication protocols. Security Layer ensures that any messages passed to Payment Protocol and consequently invoking operations in Accounts Layers are authentic.

GB database module is a relational database that stores account and transaction information.

GB Accounts is the core module interacting with the GB database. It provides functions for basic account operations such as creation of accounts, requesting and updating account details, transfer of funds from one account to another, locking funds and transfer from locked funds. This module is independent of payment scheme, protocols used and underlying security model. Its purpose is to perform database operations that deal with manipulating and managing GridBank's database.

GB Admin module provides account management such as deposit, withdrawal, change credit limit, cancel transfers and close account functions. These functions are performed by GridBank's administrators who are responsible for transferring real money to and from clients. In the future, this process can be automated by using other payment systems such as PayPal or credit card transactions by importing data from those systems. Administrators have a privileged access and the credentials for such access are checked by GB Administration module.

The payment protocol layer represents payment schemes and associated protocols that interact with GB Accounts. The protocols are described in section 3.1. *GridCheque Protocol* module implements the pay-after-use scheme. *GridHash Protocol* module implements pay-as-you-go payment strategy. Pay-before-use protocol does not involve generation of any payment instruments. Transaction is performed on-line; secure connections guarantee authenticity of participants. Any other payment scheme that defines its own data structures and communication protocol can be added without need to modify GB Accounts or GB Security modules.

GB Security Protocol module performs authentication and authorization of GridBank's clients. Authentication process uses Generic Security Services (GSS) API, which is implemented by Globus Toolkit I/O module [16]. It is based on Public Key Infrastructure (PKI) using X509v3 certificates [8,13]. Certificates can be issued by the GridBank's own certificate authority (CA) or the third party CAs [17]. Once clients are authenticated, the certificate subject name is retrieved using Globus I/O API and is checked against the database (Figure 3). If the subject name appears either in the accounts or in administrator tables, then the client is authorized to establish a connection. Otherwise connection is refused, and this provides a mechanism to limit denial-of-service attacks. Clients simply cannot send any requests before a connection is established. Only clients with existing account or administrator privilege are authorized and connected.

The GSP entities, such Grid Trade Server working with Grid Resource Meter, are responsible for enforcing accounting by billing the GSC's account. For example, Nimrod-G broker has an entity called Grid-Agent responsible for execution of the user's job on a Grid node, and also keeping track of resource consumption, which can be used by the Trade Server to enforce accounting.

3.3 Client Architecture

The Security Layer is identical to the server. The Protocol Layer has same protocol modules as the server with corresponding client functionality. GridBank API provides an interface to the Protocol layer, which is responsible for obtaining payment instruments or performing direct transfers. GridBank Payment Module and GridBank Charging Module interface to GridBank API module to invoke GridBank operations. The GBPM requests payments whereas GBCM redeems payments.

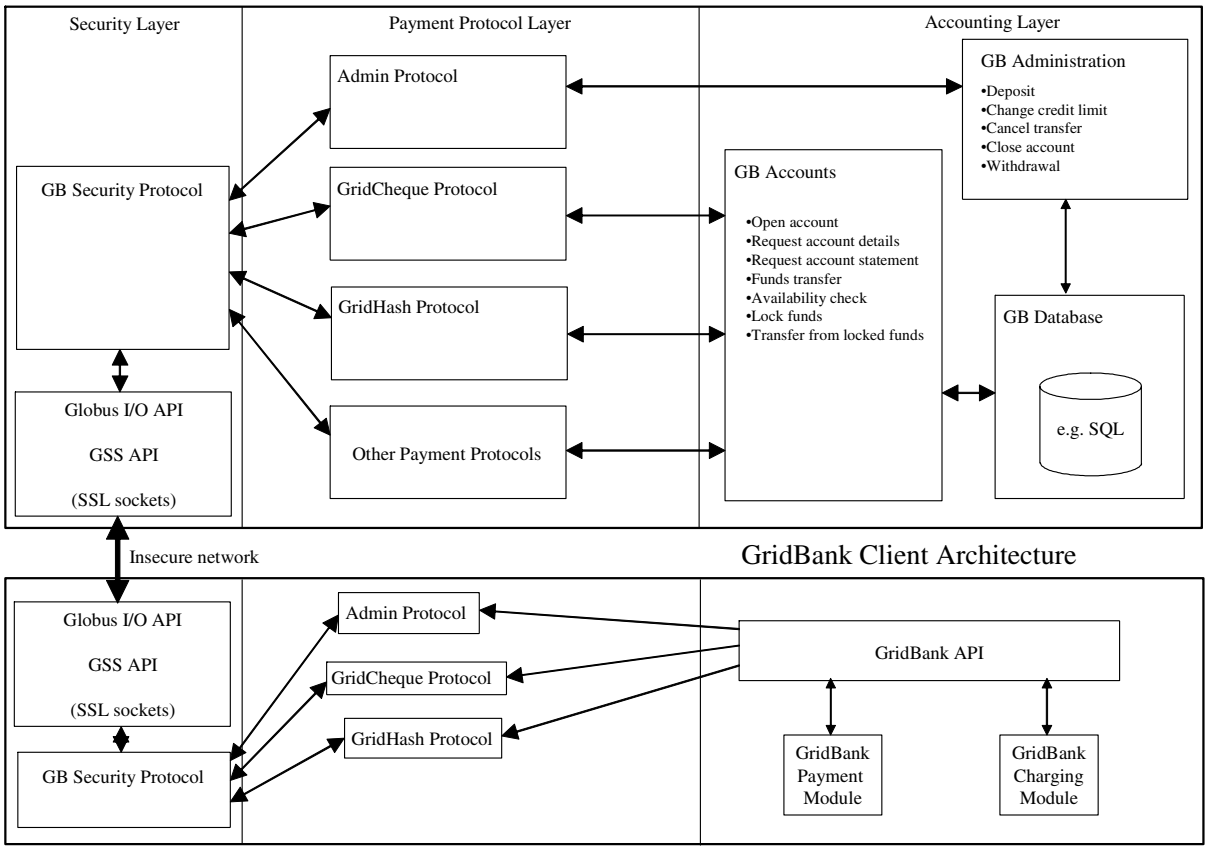


Figure 3: GridBank Server Architecture.

3.4 Payment Guarantee

When a chargeable service has a fixed price as in Pay Before Use strategy or the client obtains hash chains, there are no issues regarding availability of funds; a client could never overspend since the account is checked and decremented beforehand. On the other hand, when a credit card approach is taken as in Pay After Use strategy when the total cost is not known beforehand, clients can easily spend more than they have in the account (even if considering credit limit, this is also an issue if cost exceeds available balance together with credit limit). To guarantee payment when issuing GridCheques, GridBank will have to lock a certain amount of funds for the cheque to be valid. The exact amount will depend on the budget constraint set with the GRB. Each GSP will receive a cheque with a reserved amount, which is transferred to the "locked" balance of the GSC's account.

4 Grid Bank Usage/Operating Models

4.1 Co-operative Model

In co-operative computing environments, all participants both consume and provide services; when participants provide services, they earn credits. They can use/spend those credits to get access to other resources within the community when needed [2,4]. GridBank supports this form of resource bartering. When new participants join the community, a certain number of credits are allocated to them. The allocation could depend on the value of the resource the participant is contributing to the community. A decision on the value assigned to participant's resources need to be determined by the community and is outside the scope of GridBank.

To achieve price equilibrium, supply and demand need to be carefully regulated in such a way so that GSPs are paid approximately as much currency as they will use to access other Grid services. Otherwise the whole environment will end up in a state where some participants, who do not require any services, have all the money while others who want to access GSPs have none. A community based resource valuation and pricing authority is needed to control prices of Grid resources.

Co-operative Resource Sharing Use Case: Four Grid Services Providers, who are consumers as well, use GridBank to record how much of other resources they have consumed. To maintain price equilibrium the participants should ensure that they provide approximately the same value of resources. In Figure 4, the GridBank accounts show how much of Grid currency each client have consumed and provided. Although computations on some resources are faster because of better hardware, the slower resources have to compensate by running longer.

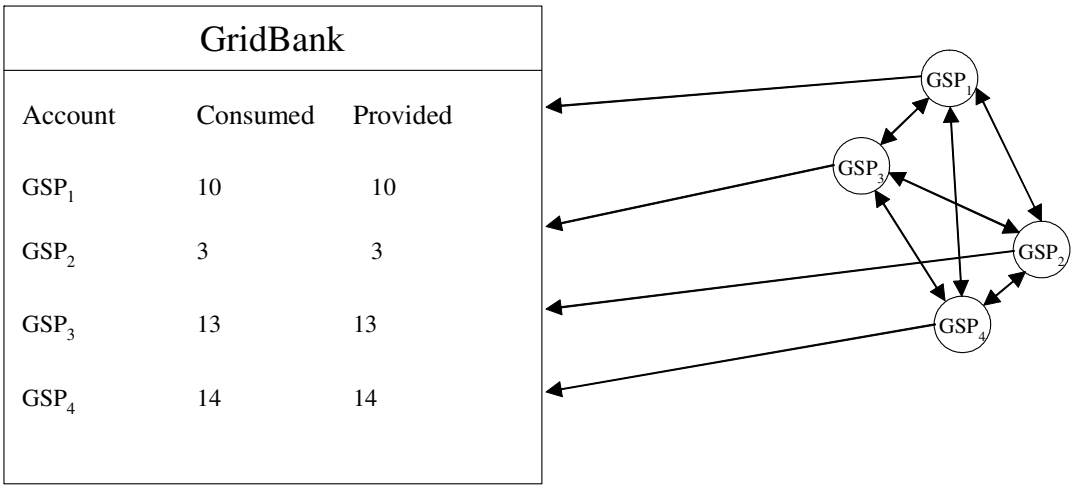


Figure 4: Co-operative Resource Sharing.

4.2 Competitive Model

In competitive computing environments GSPs are allowed to solicit any price [2,4]. However, to attract customers they need estimates of market value of their resources. GridBank's transaction history can assist in deciding how much a computational service is worth. Such transaction history is confidential and cannot be disclosed as is. Therefore GridBank would receive a description of the resource, process the information in its database regarding prices paid for resources of similar type, and then produce an estimate. The simplest approach to compare resources is to consider hardware parameters such as processor speed, number of processors, amount of main memory and secondary storage, network bandwidth, etc. More sophisticated methods are to be considered as GridBank evolves.

5 Resource Usage Record (RUR)

The Resource Usage Record passed between GridBank Payment, GridBank Charging Modules and GridBank Server is an XML document, allowing RUR to be extended by each site to include site-specific information. This approach will achieve greater flexibility as the RUR can be independently defined by the Grid sites. GridBank Server database can then be modified to retain extra information. However, a set of essential resource usage measurements has to be agreed upon by all sites.

A list of essential elements is being defined by the Grid community effort [15,23]. Currently, the following items are being associated with RUR:

- User details
  - Host name / IP address
  - Certificate Name (Grid-wide unique ID of GSC)
- Job details
  - Job ID (it can be a combination of local process ID and the application job ID as assigned by the systems such as Nimrod-G)
  - Application name
  - Job start date (includes time)
  - Job end date
- Resource details
  - Host name / IP address
  - Certificate Name (Grid-wide unique ID of GSP)
  - Host type (e.g. Cray; optional)
  - Local job id (local OS process id to settle disputes about resource consumption)
  - Wall clock time + price per time unit (e.g. Per second)
  - CPU time + price per time unit
  - Main memory + price per time unit
  - Secondary storage + price per time unit
  - Network activity + price per time unit
  - Software service + price per time unit
- Total price

6 Conclusion and Future Work

We presented new Grid components in the context of the Grid-wide banking and accounting service that will enable participants to engage in global computational economy. Grid Resource Meter extracts resource usage information from the operating system and converts it into a Grid-wide standard form. GridBank Charging Module is responsible for determining legitimacy of payment instruments passed to it by the GridBank Payment Module, setting up and removing (after execution of user application) temporary local accounts, calculating total charge using the Resource Usage Record and the service rates passed by the Grid Trade Service, and redeeming the payment with the GridBank server. GridBank Payment Module receives requests for job execution from the Grid Resource Broker, obtains a payment instrument from the GridBank, forwards the payment to GBCM and submits the job when GBCM notifies GBPM that a local account has been set up. Grid Trade Server negotiates service cost/rates with GRB and provides interface for GBCM to obtain the information. Negotiation protocols are already defined in [2,4].

In the future, GridBank system will be expanded to provide multiple servers/branches across the Grid to achieve scalability in similar manner as the currency

servers in NetCash [20] and NetCheque [21] systems. It is precisely for this purpose that GridBank accounts have branch numbers. Each Virtual Organization (VO) [12,14], which is a collaboration/site of resources, associates a GridBank server that all participants of the organization use. If a GSC is from one VO and GSP is from another, then their respective servers will need to define protocols for settling accounts between the branches. Moreover, if another payment system is introduced to the Grid, then that system can use different bank number and additional protocols can be defined to settle accounts between multiple banks.

As mentioned in section 4.2 and 5, GridBank Server at each site can perform data mining of resource usage records and can advise on charging strategies. Although a lot of research has been done on strategic allocation of performance critical resources [19] for businesses, none that we know of has been done in computational resource planning. Competitive environment is one of the primary driving forces behind pricing strategies. Grid resource advance reservation module can keep track of dynamics of rates offered on the 'computational resource' marketplace and automate market driven brokerage process. This component is currently under investigation.

7. Acknowledgements

We would like thank Chris McDonald (University of Western Australia), Andrew Twigg (Cambridge University), Scott Jackson (Pacific Northwest National Laboratory, USA), and Srikumar Venugopal for their constructive comments.

8. References

[1] Bellare, M., ET. AL., Design, Implementation and Deployment of a Secure Account-Based Electronic Payment System. *Online at http://citeseer.nj.nec.com/99445.html.*

[2] Buyya, R., Economic-based Distributed Resource Management and Scheduling for Grid Computing, PhD Thesis, Monash University, Melbourne, Australia, April 12, 2002. Online at http://www.buyya.com/thesis/

[3] Buyya, R., *Grid Economy Comes of Age: Gridbus Technologies for Service-Oriented Cluster and Grid Computing*, Proceedings of the 2nd IEEE International Conference on Peer-to-Peer Computing (P2P 2002), Linkoping, Sweden, Sept. 5-7, 2002. Project website: http://www.gridbus.org

[4] Buyya, R., Abramson, D., Giddy, J., *A Case for Economy Grid Architecture for Service Oriented Grid Computing*, Proceedings of International Parallel and Distributed Processing Symposium: Heterogeneous Computing Workshop (HCW 2001), San Francisco, USA.

[5] Buyya, R., Abramson, D., Giddy, J., *Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid*, Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region, 2002, Beijing, China.

[6] Catlett, C., *TeraGrid Primer*, http://www.teragrid.org/about.html

[7] Buyya, R., Haron, C., and Yong, C., Compute Power Market (CPM) Project, http://compute-power-market.jxta.org/

[8] Feghhi, J., Feghhi, J., Williams, P., *Digital Certificates: Applied Internet Security*. Addison Wesley Longman Inc., 1998. ISBN 0-201-30980-7, 1998.

[9] Foster, I., Kesselman, C., Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications, 11(2): 115-128, 1997. Online at http://www.globus.org/research/papers.html*

[10] Foster, I., Kesselman, C., *The Globus Project: A Status Report*, *Proceedings of the IPPS/SPDP '98. Heterogeneous Computing Workshop (HCW'98),* March 1998, Florida, USA.

[11] Foster, I., Kesselman, C. (editors), *The Grid: Blueprint for a New Computing Infrastructure*, ISBN 1-55860-475-8, Morgan Kaufmann Publishers, USA, 1999.

[12] Foster, I., Kesselman, C., Nick, J., Tuecke, S., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.* http://www.globus.org/research/papers.html (23/08/2002)

[13] Foster, I., Kesselman, C., Tsudik, G., Tuecke, S., *A Security Architecture for Computational Grids*, Proceedings of the 5th ACM Conference on Computer and Communication Security, 1998.

[14] Foster, I., Kesselman, C., Tuecke, S., The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *International Journal of Supercomputer Applications, 15(3), Sage Publications. 2001.*

[15] GGF (Global Grid Forum) – http://www.gridforum.org

[16] GLOBUS I/O API - http://www-unix.globus.org/api/c-globus-2.0-beta1/globus_io/html/index.html

[17] Globus Project – http://www.globus.org

[18] Hacker, T., Athey, B., *Account Allocations on the Grid,* http://www.nas.nasa.gov/~thigpen/accounts-wg/Documents/accounttemplates.pdf (23/08/2002)

[19] Hitt, M., Clifford, P., Nixon, R., Coyne, K, *Dynamic Strategic Resources: Development, Diffusion and Integration.* The strategic management series, ISBN 0-471-62533-7, Wiley Publishers, 1999.

[20] Medvinsky, G., Neuman, B., *NetCash: A design for practical electronic currency on the Internet,* Proceedings of the 1st ACM Conference on Computer and Communication Security, November 1993.

[21] Medvinsky, G., Neuman, B.,, *Requirements for Network Payment: The NetCheque Perspective*, *Proceedings of IEEE COMPCON'95. March 1995*

[22] Pierce, M., *Multi-Party Electronic Payments for Mobile Communications*, PhD Thesis, University of Dublin, Trinity College, Dublin, Ireland, 2000.

[23] Global Grid Forum, *RUR - Resource Usage Record Working Group*, http://www.gridforum.org/3_SRM/ur.htm

[24] Rivest, R., Shamir, A., PayWord and MicroMint: Two simple micro-payment schemes, *CryptoBytes, volume 2, number 1, RSA Laboratories, Spring 1996.*

[25] Wayner, P., *Digital Cash*, 2nd Edition. ISBN 0-12-788772-5, Academic Press Limited, 1997, US