A Distributed Economic Meta-scheduler for the Grid

Kyle Chard and Kris Bubendorfer
School of Mathematics, Statistics and Computer Science
Victoria University of Wellington
P.O.Box 600, Wellington
New Zealand
kyle@mcs.vuw.ac.nz, kris@mcs.vuw.ac.nz

***Abstract*—In this paper we present *DRIVE*, a novel architecture for a Virtual Organization (VO) based distributed economic meta-scheduler in which members of the VO collaboratively allocate Grid resources. Resource Providers joining the VO contribute *Obligation* services to the VO. These contributed services are in effect membership 'dues' and are used in the running of the VO's operations - allocation, advertising, management, etc. We use an auction plug-in mechanism to support arbitrary auction protocols which allows users to choose a protocol based on specific requirements and infrastructural availability. For instance, within a single organization, where internal trust exists, users can achieve maximum allocation performance by choosing a conventional sealed bid auction plug-in. In a global utility Grid no such trust exists. The same meta-scheduler architecture can be used with a more expensive secure auction protocol plug-in which ensures the allocation is carried out fairly in the absence of trust. The DRIVE prototype has been implemented as a collection of standard WSRF Web services and includes a distributed implementation of a secure combinatorial Garbled Circuit protocol.**

I. Introduction

The task of allocating jobs to resources in a large scale Grid is a complex problem. Local resource managers (LRMs) have only a local or organizational view of the Grid and are unable to collaborate when allocating jobs. Grid meta-schedulers focus on global allocation by acting as a higher level entity able to submit workload to numerous LRMs. However, most meta-schedulers are implemented on a per-client basis and are therefore only concerned with their own allocations rather than achieving globally optimal allocation across LRMs. Essentially client oriented meta-schedulers compete by submitting jobs to resources with no knowledge of other meta-schedulers actions therefore reducing efficiency.

This paper introduces DRIVE (Distributed Resource Infrastructure for a Virtual Economy), a distributed economic meta-scheduler that is able to efficiently allocate resources over large scale Grids. Economic markets have been used as a means of resource allocation in a number of different human and computational domains. The use of markets is particularly suited to scalable resource economies as they are naturally decentralized, efficient and produce optimal resource allocations [1], [2], [3]. The basic premise of the DRIVE architecture is built on the concept of a Virtual Organization (VO) [4]. Resource providers join a VO in order to offer services in exchange for some form of compensation. The meta-scheduler is distributed to share the computational burden of allocation over all participating resource providers. To this effect, the auction mechanism is comprised of a set of Grid services contributed by resource providers that are used to collaboratively conduct allocation. An economic approach is used to efficiently allocate resources between competing users belonging to different administrative domains. Specifically DRIVE uses auction protocols to quickly and efficiently establish the market price for a good (or service). The components of DRIVE are designed with standardized interfaces to aid interoperability and are built as Globus Toolkit 4 (GT4) compliant WSRF web services.

The author is in the second year of his candidature as a PhD student with an expected completion date of July 2009

Our main contributions: (1) We have developed a distributed economic VO meta-scheduler that uses services contributed by resource providers to conduct resource allocations. (2) We have developed an auction plug-in mechanism to support arbitrary auction protocols, making the meta-scheduler more suitable for a wide range of Grid types and sizes. (3) The meta-scheduler prototype has been implemented as a collection of standard WSRF Web services. We have also implemented an auction plug-in for the prototype that is secure, privacy preserving, and verifiable. The auction plug-in is a novel web services based implementation of the combinatorial Garbled Circuit protocol [5], [6], and can be used safely for resource allocation in an open Grid environment where trust has not been previously established.

The rest of the paper is organized as follows: in section II we provide an overview of background information, in section III we investigate related work, in section IV we describe the architecture of DRIVE, section V outlines our initial prototype and presents our experimental results, section VI explores future work, and finally in section VII we conclude the paper.

II. Background

A VO [4] is a dynamic collaboration of administratively and geographically disjoint individuals or institutions who share computing resources to meet a common goal. A Grid can be organized into a number of VOs, each with different policy requirements. As such, the VO provides a way to

978-0-7695-3156-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CCGRID.2008.48

manage complex Grid systems by establishing and enforcing agreements between resource providers and the VO, and the VO and its members. The Virtual Organization Membership Service (VOMS) [7] is used to manage authorization information within multi-organization collaborations (VOs). VOMS defines user groups, roles and capabilities, and includes tools to provide much of the VO management functionality DRIVE requires. Users may be members of any number of VOs, with any number of roles assigned in each VO. The structure of a VO is somewhat representative of the Grid, it can support complex hierarchical structures containing groups and subgroups under separate administration.

Economic resource allocation often uses some form of market to efficiently determine the price for a good (or service). It is for this reason they have been widely adopted as a means of resource allocation in distributed systems [1], [2], [8], [3]. There are various types of useful market abstractions, including commodity markets, posted price and auctions. Single bid auctions are most often preferred in distributed systems auctions as they are an efficient means of producing optimal allocations with minimal communication. To meet quality-of-service (QoS) criteria, an auction for a single *representative* resource (for example, CPU time is only one of the many resources required for execution) is not normally sufficient. This problem is solved by using combinatorial auctions where bidders bid on collections of goods. Auctions in general have been shown to provide Quality of Service (QoS) advantages over other methods [9].

Globus Toolkit [10] uses a decentralized scheduling model in which scheduling decisions are made by application level schedulers and brokers. Resource brokers translate application or user requirements into appropriate Grid specific resource requirements. Brokers often include schedulers and perform many aspects of meta-scheduling. In the Globus model brokers are responsible for discovering available resources using the Monitoring and Discovery System (MDS). Jobs are passed to the appropriate Grid Resource Allocation and Management (GRAM) service which in turn interacts with the LRM to execute the job.

III. Related Work

Nimrod/G [11] is a Grid resource broker that uses market based mechanisms for resource allocation. Distribution is achieved through interaction between hierarchical schedulers. The GridBus Broker [12] is a meta-scheduler that extends Nimrod/G to take a data centric approach to scheduling, in particular it is able to access remote data repositories and optimize based on data transfer.

EMPEROR [13] provides a framework for implementing scheduling algorithms based on performance criteria. The implementation is based on the Open Grid Services Architecture (OGSA) and makes use of common Globus services for tasks such as monitoring, discovery and job execution. EMPEROR is focused on resource performance prediction and is not distributed nor does it support economic allocation mechanisms.

GridWay [14] is a client side meta-scheduler which facilitates large scale resource sharing across administrative domains. Unlike many meta-schedulers, GridWay has a single site level installation which users can connect to without requiring local GridWay or Globus installations. GridWay does not support market based allocation and requires a single point of contact for each domain.

The Community Scheduler Framework (CSF) [15] is an open source framework for implementing meta-schedulers (Community schedulers) using a collection of Grid services. CSF provides a queuing service where job submissions are assigned to resources by applying policies, the scheduling mechanisms are user defined and implemented using a plug-in architecture. This method can support limited economic scheduling via the queuing mechanism.

Our meta-scheduler (DRIVE) is radically different from current meta-scheduling technology. Existing meta-schedulers are typically deployed on a per client or per organization basis. DRIVE is not centered about a single client or client domain, but rather abstracts over collections of resource providers that are members of a VO. DRIVE uses a collaborative architecture in which the workload when making allocations is distributed amongst resource providers belonging to a VO. DRIVE also provides a richer economic model for allocations - including support for combinatorial allocations, privacy and verifiability.

IV. Architecture

The general philosophy behind DRIVE is the concept of distributing the cost of resource allocation across the members of the Grid. A VO model is used to provide the components of DRIVE, this VO model represents the dynamic nature of Grid systems. Ideally a meta-scheduler should be suitable for use in small scale local Grids within single organizations or large scale global Grids that span organizational boundaries. A VO model provides a way to do this by grouping resource providers and users to the specific Grid system. VOMS controls VO access by authenticating participants.

Because DRIVE aims to support both local and global Grids the ability to select appropriate protocols is vital. For instance, within an organization internal trust exists – in this case DRIVE users can achieve maximum allocation performance by choosing a conventional sealed bid auction plug-in which does not utilize expensive trust or privacy preserving mechanisms. In a global utility Grid no such trust exists. The same DRIVE architecture can be used with a secure auction protocol [16] plug-in ensuring the allocation is carried out fairly and bid details are kept private. Due to the high overhead of secure protocols we suggest the use of sub-scheduling tools such as Falkon [17] to provide rapid execution of many smaller tasks and amortize the overhead. To permit the deployment of different auction mechanisms (both secure and insecure), we make use of the General Auction Framework (GAF) [18] to provide a communication independent shell for a wide range of auction types. It is important to be able to seamlessly select different protocols for different tasks due to the wide ranging complexity of privacy preserving, trustworthy and verifiable auction protocols.

A. High Level Description

Figure 1 shows the high level DRIVE architecture. The cloud outlines the scope of the VO. Resource providers (2) and (3) are members of the VO, while resource provider (1) is attempting to join the VO. On joining, a resource provider exposes a set of *obligation* services which it contributes to the VO for the running of the meta-scheduler. Obligation services, shown by resource provider (3), can include the Auction Manager, Auction Context, Auction Component, Reputation Service and MDS. Once a member of the VO, the resource provider also exposes *participation* services which are needed to bid for, accept and schedule jobs on the resource provider. Participation services, shown by resource provider (2), are Bidding, Reservation and GRAM. The remaining *trusted core* services: VOMS, Category Manager and Contract Service, are in the current architecture hosted by a set of trusted hosts, although it is our aim to modify these services to also run on hosts without needing to establish trust. The actual obligation services are defined by the use-case. For instance, within an organization internal trust exists – and therefore all trusted core services could be deployed as obligation services to achieve a wide distribution of load within the VO.

The client side broker provides a transparent interface to DRIVE, supporting job monitoring and submission to allocated resource providers (via GRAM). The broker requires no specific Grid components and can be set up as a client side application or deployed as a domain level server. All correspondence between the broker and a DRIVE VO is through web service interfaces. The broker is able to discover meta-scheduler services via DRIVE using the standard Globus monitoring and Discovery Service (MDS) and is free to choose services based on reputation. The broker submits job requests to an Auction Manager which instantiates an Auction Context for the duration of the auction. At the end of the auction, the winner and broker receive contracts (SLAs) describing the agreed upon commitments and the job is dispatched to the GRAM service on the allocated hosts.

B. Architectural Components

The DRIVE architecture is built from WSRF web services. Each service has a well defined task and is accessible through a standard interface. These services collectively perform resource allocation through plug-in auction protocols. The services below are categorized into trusted core, obligation, and participation services. For brevity, the list excludes standard Globus services such MDS, GRAM, etc.

Trusted Core Services

- Category Manager: A registration service for resource providers. Resource providers register their bidding profile when joining the VO. This profile is used to categorize the type of job a resources is interested in hosting. This has the effect of reducing auction size and therefore increasing the efficiency of the allocation process.

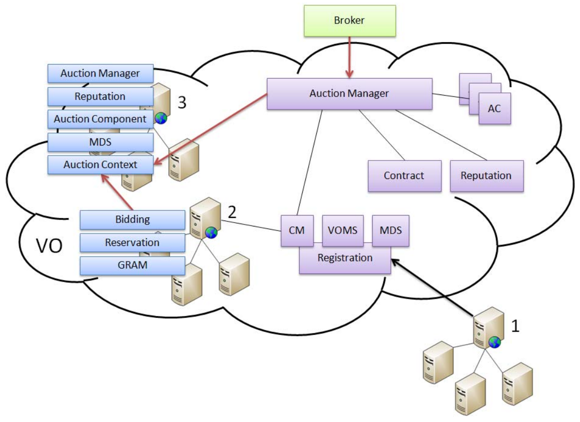

Fig. 1. Overview of the DRIVE Architecture

- Contract Service: Stores contracts that have been issued within the VO. It is used in the contract hardening process before redemption, ensuring that the contract can be honored.

Obligation Services

- Auction Manager: The central auction service, it manages the auction process and advertises the auction to suitable bidders. The broker starts an auction by passing the job description (JSDL) to the auction manager who then creates the resources required for the particular protocol.
- Auction Context: Stores state for individual auctions, clients can register for notifications or poll the service for status, results and contracts.
- Auction Component: A universal component which can be used for different auction protocols, it is a shell that wraps protocol specific services. It takes different forms depending on the protocol, for example in a SGVA auction it is an Auction Evaluator used to compute the winner of the auction, in the Garbled Circuit protocol it is an Auction Issuer used to garble the circuit.
- Reputation Service: Stores reputation information for all entities in the VO using distributed reputation algorithms.

Participation Services

- Bidding Service: Responds to auction events from the Auction Manager and computes bids for jobs it wishes to host using pricing algorithms, local policy, and future commitments.
- Reservation Service: Stores any commitments made by the provider, normally in the form of contracts. This service is the key component for supporting advanced reservation in DRIVE, the Bidding Service is able to check resource commitments using the Reservation Service before bidding on advanced reservation auctions.

C. Contract Structure

DRIVE uses the progressive contract structure from CORA (Coallocative, Oversubscribing Resource Allocation) [8]. A soft contract is returned by the Auction Manager to the client upon the result of the auction, this soft contract does not guarantee resource availability rather it is a strong indication that the resources will be available at the specified time, it can be thought of as a place holder for the eventual hard contract containing SLAs. A group of back-up resource providers is computed by the Auction Manager to allow for the possibility a soft contract may not be honored. Before a client can use the resources the soft contract must be hardened into a hard contract, to do this the resource providers are contacted to check the current resource situation. Progressive contracts are used due to the latency in the auction process, a soft contract allows negotiation to be cheaply aborted at an earlier stage if resource availability changes within the system. This approach is also applicable to the problem of advanced reservation, it makes it more likely resources will bid on advanced reservation auctions in the knowledge that they will not be penalized as harshly if resource state changes. It is generally advantageous for an application to have a cheaper initial auction process with the chance that the contract may be broken and a new auction conducted. Penalties for breaking a soft contract need to be considered, for example the party that breaks the contract may have to pay the difference of price between its bid and the backup options computed by the Auction Manager.

D. *Co-allocation*

Co-allocation is the process of simultaneously allocating resources in predetermined capacities over an ad-hoc group of resource providers. Co-allocation is often required in Grid computing due to QoS, replication, and parallelism requirements, take for example a scientific application that has data sites worldwide, the efficiency can be greatly improved by running an application in parallel using processing resources close to the individual data sites. Co-allocation auctions are performed in the same way as conventional single auctions, but rather than a single contract, a number of contracts are created for each co-allocated task. Co-allocated tasks are listed in the auction description which is taken into account during the bidding phase and multiple winners are accepted for the auction. The construction of the subsequent co-allocative contracts is done via CORA.

E. *Interaction Phases*

There are three important use-cases when interacting with DRIVE. First resource providers must register with the VO to make their resources available to users. Users can then submit job requests to the allocation mechanism resulting in resources being allocated and a contract returned. Finally the user can redeem the contract and submit the job to the allocated resource providers for execution.

1) Registration: For resource providers to offer services they must first be a member of a VO, only then are their resources considered in the allocation process and are therefore able to host user jobs. To join a VO each resource provider registers with VOMS, VOMS records user information and issues certificates to be used when authenticating access. A key component of the DRIVE architecture is the obligation services collaboratively used to perform allocation, these services are contributed by resource providers in exchange for compensation. The address of any contributed services is registered in the MDS index service for discovery purposes by clients, providers or other services in the VO. Participation service addresses and high level resource profiles are submitted to a Category Manager (CM) in order to categorize providers according to the type of jobs they are willing to host. CMs are consulted for every auction to determine suitable resource providers from which to solicit bids.

2) Resource Allocation: Resource allocation is conducted using auction based mechanisms hosted on obligation VO services. From a users perspective jobs are described using Job Submission Definition Language (JSDL) documents and submitted to a client broker. The process of economic allocation (auctioning) is transparent to the user. The DRIVE architecture does not specify the type of job that can be submitted only that it is defined using JSDL. Workflows can be decomposed into a set of jobs each described with a JSDL document, the set of JSDL documents is passed to the broker to be submitted to the Grid.

The broker authenticates on behalf of the user with VOMS, a working proxy is created and subsequent access is provided via this proxy. The broker locates a viable Auction Manager (AM) using the index service. The choice of AM can be based on protocol support, reputation, location or user requirements. Having discovered a suitable AM all allocation is now performed though this service. Depending on the chosen auction protocol the Auction Components will be instantiated and used, for example for the Garbled Circuit protocol an Auction Issuer is constructed. The auction is then conducted, first suitable resource providers are selected by consulting CM resource profiles, selected resource providers are then notified of the auction. Resource providers that are not explicitly chosen are still able to bid, but must discover the auction through auction publishing services. Resource Providers each host a Bidding Service which is responsible for implementing bidding policies, pricing goods in the auction according to some pricing function and submitting bids to the AM. When the auction is complete a winner is determined and a soft contract is created by the Contract Service and returned to the client, a list of alternative resource providers is maintained in case the soft contract cannot be honored. The resource provider records the successful allocation and subsequent contract through its Reservation Service.

3) Job Submission: Having been allocated resources a job is ready to be executed, the soft contract is passed to the Contract Service (CS) for redemption. The CS checks the resource provider Reservation Service for availability which in turn checks load and other commitments, if the resource is determined to be available the CS returns a hardened contract to the client. If there is no availability the CS checks the backup list for availability in an attempt to satisfy the soft contract elsewhere, if the contract cannot be honored some form of penalty will be imposed (reputation or financial).

The hardened contract is then redeemed by the client at the appropriate resource provider and the job is submitted to the LRM through GRAM. If there is no suitable backup a new auction will be conducted.

V. Prototype and Evaluation

The DRIVE prototype is built as a collection of WSRF Grid services deployed to GT4 containers with a simple Java client to request allocations and submit jobs to GRAM. The focus of this implementation is to experiment with auction protocols in a distributed Grid service environment and to serve as a proof of concept for the auction components of the DRIVE architecture. The prototype supports two distributed auction protocols, a standard insecure sealed bid second-price auction and a secure, verifiable Garbled Circuit[5], [6]. The following experiments look at the effect the number of bidders and goods have on auction time and resource creation overhead. All experiments are run on 5 Pentium 4 3.2 MHz machines running Fedora Core 5 and GT4, jobs are submitted using the Java client from a machine outside the testbed.

The auction times are recorded from when the auction manager receives a job request until the winner is determined. Auctions are concluded when all bids are received rather than using explicit end times. Overhead is the time taken to setup auction components, for example WSRF resources storing auction state and any protocol specific computation that can be performed before the auction commences such as the circuit creation and garbling in the Garbled Circuit protocol. Communication between the broker and auction services is not included in the auction or overhead time. The Garbled Circuit protocol is using 10 bit price representation.

A. Number of Bidders

Scalability of auction protocols is determined by the number of bidders they can effectively support. The time taken to solicit bids and determine a winner in both the sealed bid (Figure 2) and Garbled Circuit (Figure 3) protocols is shown to be linear. The Garbled Circuit protocol has substantial overhead which increases linearly with the number of bidders, this is due to the extra cost of encryption and circuit creation used to provide trust and privacy. The overhead of a sealed bid auction is almost constant as there is only a single WSRF resource created and no preliminary computation. Comparison between Figure 2 and Figure 3 shows the substantial computational cost in terms of auction time and overhead of using complex trustworthy protocols, the sealed bid auction takes 5 seconds to determine a winner with 50 bidders whereas the Garbled Circuit protocol takes almost 30 seconds with only 10 bidders. As mentioned previously we envision the use of more complex trustworthy protocols in larger value auctions where you would expect fewer bidders and would be willing to sacrifice auction time for secure results.

B. Number of Goods

Combinatorial auctions can be used to describe dependent resource requirements often needed in Grid systems, for example when expressing QoS constraints. Figure 4 shows the time taken to compute the winner using the Garbled Circuit protocol and the associated overhead on a logarithmic scale. The time taken and overhead is shown to be exponential with the increase in the number of goods, this is due to the exponential increase in circuit size and the increased computation required for each resource combination.

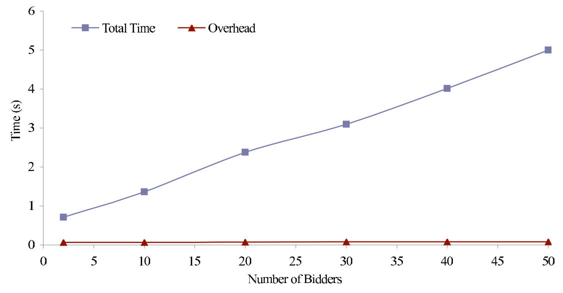

Fig. 2. Number of Bidders vs Auction Time and Overhead for a Sealed Bid Second Price Auction with 1 Good

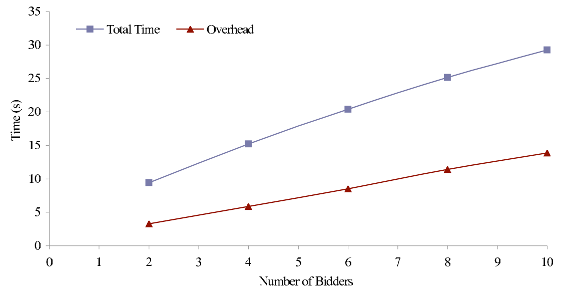

Fig. 3. Number of Bidders vs Auction Time and Overhead for Garbled Circuits with 1 Good

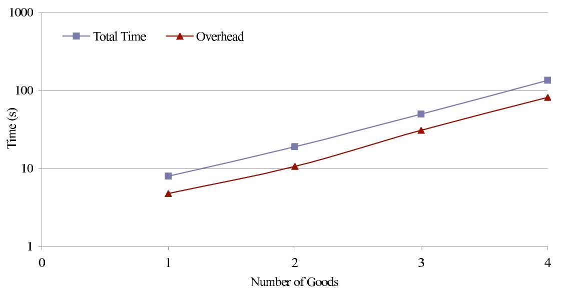

Fig. 4. Number of Goods vs Auction Time and Overhead for Garbled Circuits

VI. Future Work

A major focus of our future work is reducing the trusted core services so that the infrastructural service requirements are a minimal subset of the overall services. To do this we plan to redesign the Category Manager and Contract Service to use distributed encryption techniques permitting them to be executed on untrusted VO members. It might also be possible to similarly distribute an encrypted VOMS like service - however this is not an immediate goal.

The prototype serves as a proof of concept for the auction component of the DRIVE architecture, we aim to extend this to facilitate the use of arbitrary auction protocols through the GAF framework, in particular homomorphic and polynomial auction protocols have properties that would make them suitable for large scale Grids. We are yet to develop the Reputation Service and the associated feedback mechanisms for VO participants, this will be implemented when we add verification to auction protocols. We will also be adding support for policy specification and management within the VO. Presently bidders in our system use naïve pricing schemes, these will be extended to incorporate budget considerations and advanced reservation.

Table I presents an outline of our staged development plan, we have completed the initial stage of creating a prototype used to evaluate protocol performance and are currently including the GAF framework in our prototype.

TABLE I
TIME LINE OF FUTURE RESEARCH

Timeline	Description
Jan 2008 – Mar 2008	Fully incorporate GAF framework, implement polynomial and homomorphic auction schemes
Mar 2008 – Aug 2008	Implement and evaluate advanced reservation and co-allocation mechanisms
Aug 2008 – Oct 2008	Add auction verification and reputation algorithms
Oct 2008 – Dec 2008	Include policy specification and management
Dec 2008 – Feb 2009	Evaluate prototype in large scale experiments
Aug 2009	Expected completion of PhD

VII. CONCLUSIONS

We have presented DRIVE, a novel meta-scheduler architecture for efficient resource allocation over large scale Grids. DRIVE is a distributed economic meta-scheduler based on a VO model, in which VO members collaboratively conduct resource allocation auctions thus spreading the burden of allocation across participating entities. Resource providers contribute a set of obligation Grid services to the VO as a condition of joining and these services are used to host auctions in exchange for compensation. A group of trusted services is hosted in a secure section of the VO providing a trust anchor for security throughout the VO. Resource providers also have a set of participation services used to bid on auctions and manage local allocation. DRIVE services are designed using GT4 compliant WSRF web services. DRIVE uses a light weight client architecture that has no dependencies on web service containers or Globus.

DRIVE has a plug-in auction mechanism that facilitates the use of arbitrary combinatorial auction protocols making it a suitable meta-scheduler for all Grid types and sizes. Users are able to select protocols based on specific allocation requirements, available infrastructure and their level of trust in the entities conducting the auction. This flexibility allows the same DRIVE infrastructure to be used within a trusted organization or on a global Grid where no such trust exists. Thus avoiding the use of complex privacy preserving or trustworthy protocols within a trusted environment whilst still providing bid privacy and confidence in auction results in an untrusted environment. The distributed nature of the auction framework in DRIVE allows implementation of complex auction protocols that have not yet been explored in Grid systems.

REFERENCES

[1] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta, "Spawn: A distributed computational economy," *Software Engineering*, vol. 18, no. 2, pp. 103–117, 1992.

[2] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *Concurrency and Computation: Practice and Experience (CCPE)*, vol. 14, no. 13-15, pp. 1507–1542, May 2002.

[3] C.-H. Chien, P. H.-M. Chang, and V.-W. Soo, "Market-oriented multiple resource scheduling in grid computing environments," in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA '05)*, vol. 1, no. 28-30, March 2005, pp. 867–872.

[4] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Lecture Notes in Computer Science*, vol. 2150, 2001.

[5] A. Juels and M. Szydlo, "A two-server, sealed-bid auction protocol," in *Proceedings of the 6th Financial Cryptography Conference (FC 2002)*. Springer-Verlag, 2003, pp. 72–86.

[6] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proceedings of the 1st ACM Conference on Electronic Commerce (EC '99)*. ACM, 1999, pp. 129–139.

[7] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, K. Lőrentey, and F. Spataro, "From gridmap-file to VOMS: managing authorization in a grid environment," *Future Generation Computer Systems*, vol. 21, no. 4, pp. 549–558, 2005.

[8] K. Bubendorfer, "Fine grained resource reservation in open grid economies," in *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing (e-Science '06)*, Amsterdam, Holland, December 2006.

[9] S. Krawczyk and K. Bubendorfer, "Grid resource allocation: Utilisation patterns," in *Proceedings of the 6th Australasian Symposium on Grid Computing and e-Research (AusGrid)*, ser. CRPIT, Wollongong, Australia, January 2008.

[10] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, 1997.

[11] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/g: an architecture for a resource management and scheduling system in a global computational grid," in *Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000)*, vol. 1, 2000, pp. 283–289.

[12] S. Venugopal, R. Buyya, and L. Winton, "A grid service broker for scheduling e-science applications on global data grids," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 6, pp. 685–699, May 2006.

[13] L. Adzigogov, J. Soldatos, and L. Polymenakos, "EMPEROR: An OGSA grid meta-scheduler based on dynamic resource predictions," *Journal of Grid Computing*, vol. 3, no. 1–2, pp. 19–37, June 2005.

[14] E. Huedo, R. Montero, and I. Llorente, "A framework for adaptive execution on grids," *Software - Practice and Experience*, vol. 34, no. 7, pp. 631–651, June 2004.

[15] Platform Computing, "Technical WhitePaper: Open source metascheduling for Virtual Organizations with the Community Scheduler Framework (CSF)," August 2003.

[16] K. Bubendorfer, B. Palmer, and I. Welch, *Encyclopedia of Grid Computing Technologies and Applications*. Information Science Reference, 2008, ch. Trust and Privacy in Grid Resource Auctions.

[17] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, "Falkon: a Fast and Light-weight tasK executiON framework," in *SuperComputing*, 2007.

[18] W. Thompson, "A framework for secure auctions," Master's thesis, Victoria University of Wellington, 2008.