Chapter 25

Using Secure Auctions to Build a Distributed Meta-scheduler for the Grid

Kyle Chard and Kris Bubendorfer

25.1 Introduction

The previous chapter introduced a number of different techniques for establishing trustworthy Grid resource or service auctions. These protocols permit new architectures that are community or peer oriented to be developed without compromising the integrity of the resource or service allocations. In this chapter we will look at an architecture that is designed to take advantage of trustworthy protocols in a Grid context. DRIVE (Distributed Resource Infrastructure for a Virtual Economy)[1] is a community meta-scheduler implemented within a Virtual Organization (VO)[2], using resources contributed by the members of the VO to conduct the auction. This distribution of effort decentralizes and shares the computational burden of allocation over all participating resource providers.

DRIVE uses auction protocols to quickly and efficiently establish the market price for a resource or service. DRIVE is designed to allocate jobs to Local Resource Managers

(LRMs) in both small and large scale Grids. This flexibility is achieved in part through a modular auction protocol plug-in architecture, allowing users to define and select protocols based on context specific requirements.

The allocation mechanism is dynamically composed from a set of *obligation* Grid services individually contributed by resource providers. As these services are hosted by potentially untrusted members of dynamic VOs there are obvious risks in terms of trust and security. These risks can be mitigated by choosing an appropriate protocol while also considering the tradeoff between security and cost. For example, within a single organization there is an inherent level of trust in the members of the VO, thus a simple open auction protocol could be used to quickly allocate resources. Whereas, in a global Grid this trust does not exist and users may wish to use secure and trustworthy auction protocols, such as those presented in Chapter 24, to ensure the allocation is done fairly and sensitive information is not disclosed.

DRIVE is built using a collection of Globus Toolkit 4 (GT4)[3] compliant WSRF web services with a java-based client side broker. Importantly, the broker has no requirement for GT4 or any form of web service container.

25.2 Background

Economic allocation mechanisms commonly use some form of market abstraction to efficiently determine the price for a good (or service). There are various types of useful market abstractions, including commodity markets, posted price and auctions. Single bid

auctions are most often preferred in distributed system resource allocation auctions as they are an efficient means of producing optimal allocations with minimal communication and usually minimal computation. 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 some Quality of Service (QoS) advantages over other methods[4].

A VO[2] 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 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)[5] 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.

Globus Toolkit[3], the predominant Grid toolkit, 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. Globus 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.

25.3 Related Work

Nimrod/G [6] is a Grid resource broker that uses market based mechanisms for resource allocation. Distribution is achieved through interaction between hierarchical schedulers. The GridBus Broker[7] 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 allocation based on data transfer. Nimrod/G and the GridBus Broker are discussed in more detail in earlier chapters.

EMPEROR[8] provides a framework for implementing scheduling algorithms focusing 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 does not use distributed allocation nor does it support economic allocation mechanisms.

GridWay [9] is a lightweight client side meta-scheduler which facilitates large scale resource sharing across administrative domains. GridWay provides support for simple scheduling mechanisms with the ability for user defined extensions through its modular architecture. GridWay utilizes common Globus components for data transfer, resource discovery and job execution. Unlike many meta-schedulers, GridWay has a single site level installation which users can connect to without requiring local GridWay or Globus installations. However it does not support market based allocation and uses an organization level server for each domain.

The Community Scheduler Framework (CSF)[10] 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. Advanced reservation is supported through a reservation service which interacts directly with reservation compliant LRMs.

DRIVE is radically different from current meta-scheduling technology. Existing meta-schedulers are typically deployed per client or using an organizational level server implementation. 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 also provides a rich economic model for allocations, including support for secure combinatorial allocations, privacy and verifiability.

25.4 Use Cases

In the process of designing DRIVE a number of use cases have been considered that typify common usage of Grids. A range of situations, infrastructures and application complexities have been investigated to ensure flexibility in the architecture. A major goal of the DRIVE architecture is to facilitate use in a wide variety of Grid scenarios ranging from single user single organization Grids to large scale collaborations using resources provisioned from a global Grid. The following section outlines three use cases: a small scale single institution computation of the Tutte polynomial[11]; CyberShake[12], a large scale multi-provider geophysics computation; and a potential Utility Computing scenario.

Tutte Polynomial

As the first use case we consider the needs of a single scientist (or small group of scientists) operating within their organization. Imagine that our scientist wants to compute Tutte polynomials [11] which are used in a number of different disciplines. For a single researcher it would take weeks to setup and run each of the algorithms against a test suite of graphs. When these experiments are broken into Grid jobs they can be run on a small single organizational Grid over a weekend.

In graph theory Tutte polynomials can be used to find the number of spanning trees, forests, or connected spanning subgraphs. Tutte polynomials can also be used in areas

such as micro-biology to classify knots, where a knot is a form somewhat like a tangled cord with its ends joined. An interesting application of knots is in the study of DNA where the double helix can be viewed as two tangled strands forming a knot. The computation of Tutte polynomials is computationally intensive and is well suited to running on a Grid.

The algorithms presented by Haggard et.al.[11] provide a way to compute the Tutte polynomial of large graphs. In order to develop the proposed algorithms a variety of graph sizes and complexities are considered. The Grid used to test the three Tutte algorithms contained approximately 150 desktop machines running Sun Grid Engine[13]. Each experiment tested a single algorithm and was made up of a few hundred jobs (207, 144 and 428 respectively), with each job performing analysis on 100 different sized graphs. Each graph took between a few seconds and a few hours to compute due to the wide range of size and complexity. The total runtime of the three experiments was less than 60 hours, with less than 1 MB of data moved between nodes for each job and a resulting data set of approximately 50 MB for all three experiments.

In this use case the requirements for security, trust and privacy are minimal due to the fact the jobs are running within the user's organization. There are no hard deadlines or QoS constraints and the jobs have a relatively short running time with no dependencies between them. For this scenario we can use a simple 1st price auction protocol that does not need to support advanced reservation, deadlines, QoS constraints, trust or privacy.

CyberShake

The second use case considered is the Southern California Earthquake Center (SCEC), a large scale VO encompassing 50 earth science research institutions across the world. SCEC has a production Grid infrastructure used to share participating institution resources amongst collaborating parties. This infrastructure provides the ability for researchers to utilize a large pool of resources, however for many projects the SCEC capabilities are not sufficient. An example is the CyberShake project[12] which involves calculation of Probabilistic Seismic Hazard Analysis (PSHA) maps which can be used when designing buildings. Recent advancements in geophysics models allow these PSHA maps to be calculated based on earthquake wave propagation simulations, however these calculations require a large amount of computation and storage. In order to facilitate such large scale calculations large portions of the computation must be outsourced to high end facilities such as TeraGrid.

PSHA calculations are conducted for each site of interest in stages, with some stages containing thousands of individual jobs. To produce a hazard map of a region at least 1,000 sites need to be calculated. The first stage of Strain Green Tensor (SGT) generation runs over a period of days using 144 or 288 processors and producing up to 10 TB of data. The second stage runs several thousand (average 5000) individual rupture forecasts each on a single processor for between a few minutes and 3 days. The third stage calculates the spectral acceleration and involves the same number of jobs each running for a matter of minutes. Finally the spectral acceleration values are combined and hazard maps are constructed.

The experiments shown in [12] are based on calculation of PSHA maps for two sites of interest using resources provisioned from SCEC and TeraGrid. The resulting experiments involve over 250,000 individual jobs with a total running time equivalent to 1.8 CPU years. The reservation of nodes at two major computing centers for 9.5 days and 7 days respectively are sufficient to perform the calculations.

Scenarios like CyberShake show common infrastructural requirements for large scale science applications. Providing QoS guarantees and enforcing deadlines is much more important than in small scale experiments due to the need for coordinated computation and storage over a large pool of resources. In order to support scenarios like CyberShake DRIVE must be able to provision resources from dynamic VOs spanning multiple organizations. As there is no expectation of inter organizational trust we need to adopt an auction protocol that acts in place of a trusted third party to establish trustworthy resource allocation. In this case we would use a verifiable, privacy preserving auction protocol such as those presented in the previous chapter. Additionally advanced reservations and contracts are vital in assuring deadlines are met and levels of QoS can be specified and provided.

Utility Computing Wholesale Markets

The third use case is a utility computing scenario. In a utility computing model computation and storage is viewed as a commodity in which users pay for the resources they use much the same as we currently pay for electricity. Although a true utility computing infrastructure remains unrealized, we might imagine that such a scheme could

be implemented on a global scale using a tiered approach. In this scheme there would be three tiers: a wholesale computation market, a retail computation market and an optional organizational market. Wholesalers buy large chunks of computation from resource providers and on sell it to retailers who in turn sell to organizations or individuals. An organization may also choose to use auctions to divide their allocation amongst their users. The requirements at each of these tiers are different and can be separately met by different scheduling and trust arrangements:

- **Wholesale to retail:** the requirements at the wholesale to retail tier are secure, verifiable, privacy preserving allocations. This market represents a very competitive environment in which allocations are for huge quantities of resources worth large amounts of money, thus ensuring the correct allocations are made without any sensitive data being released is imperative. Due to the large quantities of resources being reserved and exposure to risk, it is essential to use a more complex yet safer allocation process. The cost of which may be more than a simple allocation mechanism but in a large scale case this represents a small fraction of the total resources provisioned. The allocation process should be secure and verifiable whilst also keeping commercially sensitive information private.

- **Retail to organization:** the requirements at the retail to organization tier are more relaxed, allocations will be of a fine to medium grained nature with less risk than the wholesale market. Resources may be allocated using different mechanisms to those used at the wholesale level, for example perhaps a provider will use a fixed price

model. The allocation mechanism in the most part should be faster with less emphasis placed on trust, security and privacy.

- **Organization to user:** the requirements at the organization to user level are very fine grained with little to no risk. The organization level tier would be commonly seen within organizations but is not an essential part of the overall utility computing model. The main concern of the allocation mechanism within the organization is efficient allocation with minimal overhead. As the allocation takes place within a trusted domain a simple 1st price auction protocol could be used with no regard for trust or privacy. Specialized sub schedulers such as Falkon[14] may also be used to allocate resources within the organization.

Summary

These use cases show a range of requirements placed on Grid systems. The allocation infrastructure needs to scale to support dynamic multi-organization collaborations whilst also providing efficiency in small scale Grid scenarios. In order to do this mechanisms are required to deal with trust, security, and privacy whilst also providing the ability to reserve resources and enforce contracts between participants. The range of use cases shows that no one mechanism will satisfy every scenario, in order to provide the required flexibility DRIVE must have a modular architecture that is capable of supporting different allocation plug-ins. Generally when provisioning large quantities of resources the overhead associated with secure allocation protocols is tolerable considering the benefits gained. However for smaller allocations this overhead could be a significant percentage of the total job and therefore a simpler protocol may be more suitable.

25.5 DRIVE Architecture

Ideally a meta-scheduler should be flexible in terms of suitability for use in different Grid scenarios, ranging from small scale local Grids within single organizations to large scale global Grids that span administrative boundaries. DRIVE achieves this flexibility through the use of a VO model to represent Grid systems and an auction plug-in architecture for allocating resources. The VO model is used to group resource providers and users to specific Grid systems and a VO membership service controls VO access to services and resources by authenticating participants.

The plug-in auction protocol is vital in tailoring allocation mechanisms to specific scenarios. For instance, within an organization internal trust may exist – 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 [15] 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 [14] to provide rapid execution of many smaller tasks and amortize overhead. To permit the deployment of different auction mechanisms (both secure and insecure), we make use of the General Auction Framework (GAF)[16] to provide a protocol independent shell for a wide range of auction types. This is important as it allows protocols to be seamlessly selected for

different tasks due to the wide ranging complexity of privacy preserving, trustworthy and verifiable auction protocols.

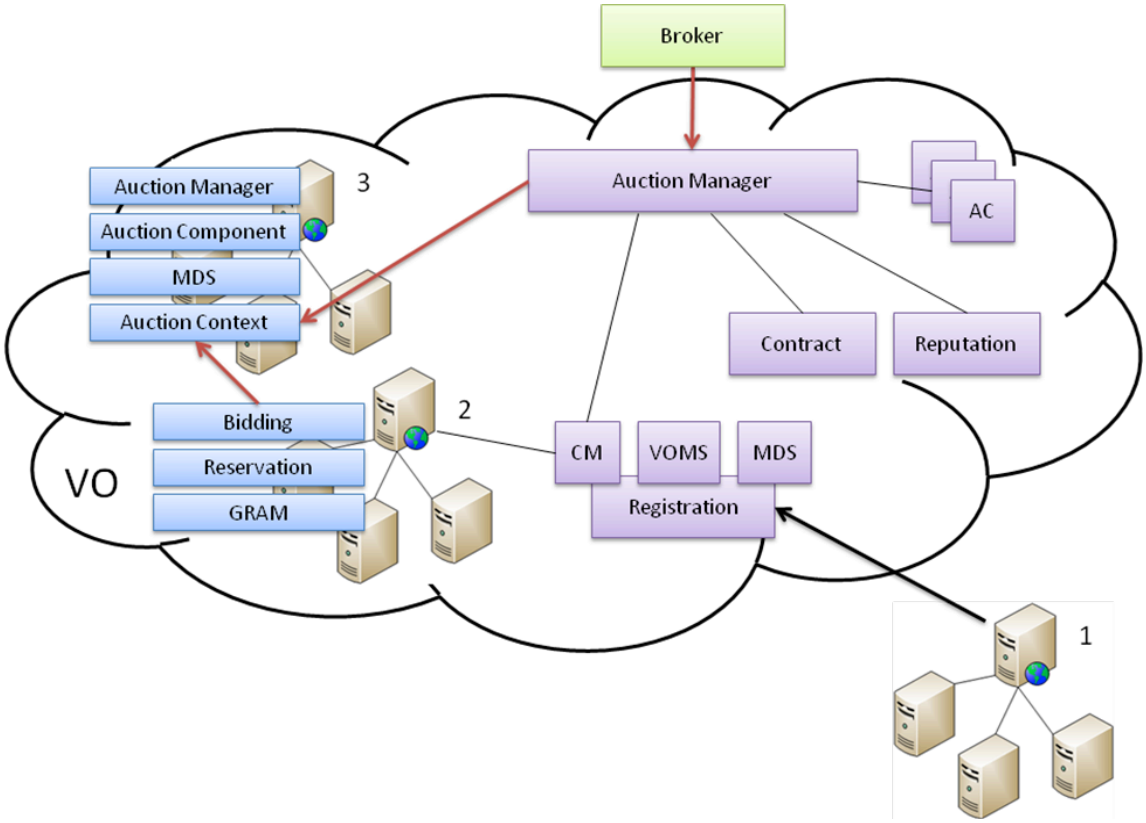

Figure 25.1 Overview of the DRIVE Architectural Components

25.5.1 High Level Description

Figure 25.1 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 the VO, 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 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), include the Bidding Service, Reservation Service 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 redesign 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 where internal trust exists, 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 or web service containers 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 MDS and is free to choose services based on reputation[17]. 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 (containing SLAs) describing the agreed upon commitments and the job is dispatched to the GRAM service on the allocated hosts.

25.5.2 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 resource is interested in hosting. When an auction is conducted the Category Manager is consulted and appropriate resource providers are selected based on their resource profiles. This has the effect of reducing auction size and therefore increasing the efficiency of the allocation process.
- 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. It is also used to monitor SLAs contained in the contracts.

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 which then creates the appropriate auction resources and selects the services required for the particular auction protocol.
- Auction Context: Stores state for individual auctions, including status and protocol specific state. 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. The Auction Component takes different forms depending on the protocol, for example in a verifiable Secure Generalized Vickery Auction (vSGVA) the Auction Component is an Auction Evaluator used to compute the winner of the auction. In the Garbled Circuit protocol the Auction Component is an Auction Issuer used to garble the circuit and is required for the VPOT protocol[18].

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 consideration of 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 its Reservation Service prior to bidding on advanced reservation auctions.

25.5.3 Contract Structure

There is potential for considerable latency in the allocation process between the bidding phase and auction completion. This latency greatly impacts utilization when resources are reserved while waiting for the result of the auction, this problem is exaggerated by the fact that all but one of the bidders will not win the auction. Overbooking can be used to

minimize this problem by allowing resources to bid on auctions that may exceed their capacity in the knowledge it is unlikely that they will win all auctions. To permit overbooking DRIVE makes use of a progressive contract structure[19] to ensure that resources are available at the time of the bid and when the contract is finalized. Backup resources are used to reduce the impact of overbooking in the case of unexpected changes to resource availability.

In DRIVE an agreement is returned by the Auction Manager to the client upon the result of the auction, this agreement 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 contract containing Service Level Agreements (SLAs). A group of back-up resource providers is computed by the Auction Manager to allow for the possibility that an agreement may not be honored. Before a client can use the resources the agreement must be converted into a contract, to do this the resource providers are contacted to check the current resource availability.

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 their resource state changes. Penalties may apply for breaking an agreement, for example the party that breaks the agreement may have to pay the difference of price between its bid and the backup options computed by the Auction Manager.

SLAs are specified in the hardened contract to ensure the agreed upon QoS is delivered by the resource providers. The details of the SLA tie specific user requirements with resource provider capabilities as detailed in the contract. The SLA is made up of a number of measurable properties that are monitored for the duration of usage. These properties map to specific QoS parameters. The SLA may also define a set of penalties in the event a certain capability cannot be met by the resource provider. SLAs are discussed in more detail earlier in this book.

25.5.5 Co-allocation

Co-allocation is the process of simultaneously allocating resources in predetermined capacities over a group of resource providers. Co-allocation is often required in Grid computing due to requirements for QoS, replication, and parallelism. Take for example, a large scale science application that has data sites worldwide, overall efficiency can be greatly improved by running an application in parallel using many 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 (and backup resources) are accepted for the auction. The construction of the subsequent co-allocative contracts is done via CORA (Coallocative Oversubscribing Resource Allocation) [19] through the Contract Service.

25.5.6 Advanced Reservation

Advanced reservation is a key component in Grid resource provisioning, often jobs require particular resources to be available at certain times in order to run efficiently. An example would be a requirement for a large amount of data storage as an intermediary stage in a workflow in order to temporarily store results of a job. This storage must be available when the job completes and for the duration of the subsequent processing stage. To support advanced reservation DRIVE makes use of a Reservation Service per resource provider, this service stores commitments made by providers allowing future commitments to be considered prior to bidding. We expect to extend the Reservation Service to leverage existing functionality in advanced reservation compliant LRMs and GRAM.

25.5.7 Job Description

There are a number of distributed resource management systems designed to deploy jobs to remote Grid resources, unfortunately most use proprietary languages and interfaces to describe and deploy jobs. Grid environments commonly require communication between different types of job management systems running on heterogeneous hosts. For this reason there have been efforts to define a standardized job description language in order to provide transparency over distributed resource managers.

DRIVE makes use of the Job Submission Description Language (JSDL) to describe jobs submitted to the VO. JSDL is a language designed to describe requirements of jobs for submission to Grids. JSDL is an XML based language with an XML schema that allows

the expression of requirements as a set of XML elements. The JDSL Working Group is in the process of extending the language to encompass a wider range of jobs and is producing translation tables to and from scheduling languages (job requirements and resource descriptions) used by the major distributed resource managers and Grid projects.

25.5.8 VO Security

Security is a major issue to be considered in DRIVE. Globus security mechanisms rely on certificates to authenticate access to resources and use policy decision points to decide permissible actions. DRIVE relies heavily on VOs to group users and provide the infrastructure required for general VO functionality such as allocation and management. In order for users to join VOs and prove they are a member of the VO we use the Virtual Organization Membership Service (VOMS)[5]. VOMS is a database-based mechanism used to manage authorization information within multi-institutional collaborations. User roles and capabilities are stored in a database and a set of tools are provided for accessing and manipulating data. Users join a VO by contacting the VOMS service to gain credentials and resources check local access control lists to determine if these credentials are appropriate.

Credentials for users and providers are generated when required based on authorization information stored in the database. These credentials extend authorization information stored in standard X.509 Grid proxies by including role and capability information as well as VOMS server credentials. The resource provider authenticating the certificate keeps an access control list to make authorization decisions based on groups, roles, or capabilities. VOMS has the advantage that resource sites do not need to retrieve all VO

lists multiple times a day, rather VO information is pushed through the certificate. We use VOMS in DRIVE to take advantage of single sign on, backward compatibility with standard certificates, and the ability for users to join multiple VOs. Another important aspect of VOs is the ability to define policy. Although VOMS does not provide a standard way to define fine grained policy, it can be achieved by adding policy tables to the database or through a separate policy management service.

25.5.8 Interaction Phases

There are three major phases in the lifecycle of a member of a VO. First resource providers register with a VO making obligation and participation services available to other entities of the VO. Having joined the VO users can request resource allocations using the contributed services to determine suitable resource providers. Finally jobs can be submitted and executed on members of the VO.

Registration

Before a resource provider can offer its services it must first be a member of a VO. The process of registration is as shown in Figure 25.2:

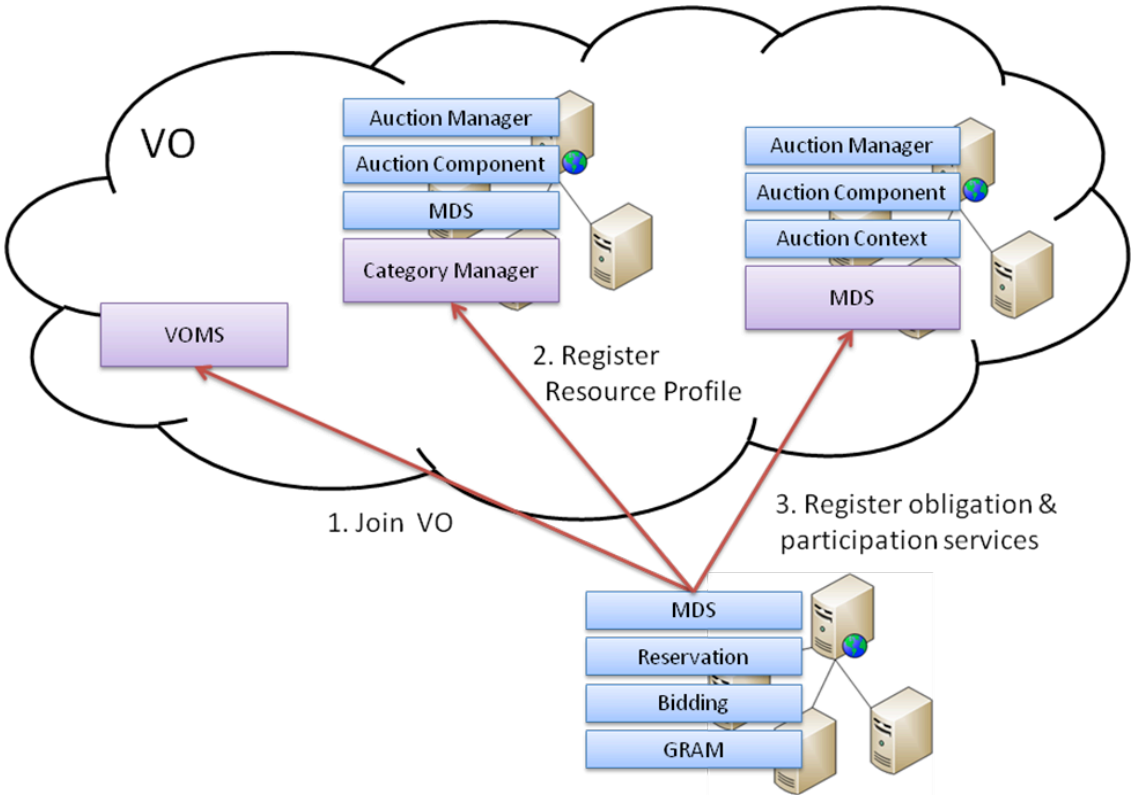

Figure 25.2 DRIVE Resource Provider Registration

1. The first step of registration is joining the VO, to do this each resource provider registers with VOMS. VOMS records user information and issues certificates to be used when authenticating access. The resource provider is then able to pull authentication information from VOMS and use it to maintain an access control list containing VO/user rights.

2. Participation service details and high level resource profiles are then submitted to a Category Manager in order to categorize providers according to the type of auctions they are interested in bidding on, for example CPU intensive jobs.

3. Finally the address of any contributed services and the address of the participation services are registered in the MDS index service for discovery purposes by clients, providers or other services in the VO.

Resource Allocation

Resource allocation is performed in the VO using auction based mechanisms and is conducted using a subset of the contributed obligation services. From a users perspective jobs are described using Job Submission Definition Language (JSDL) documents and are 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 allocation process is shown in Figure 25.3:

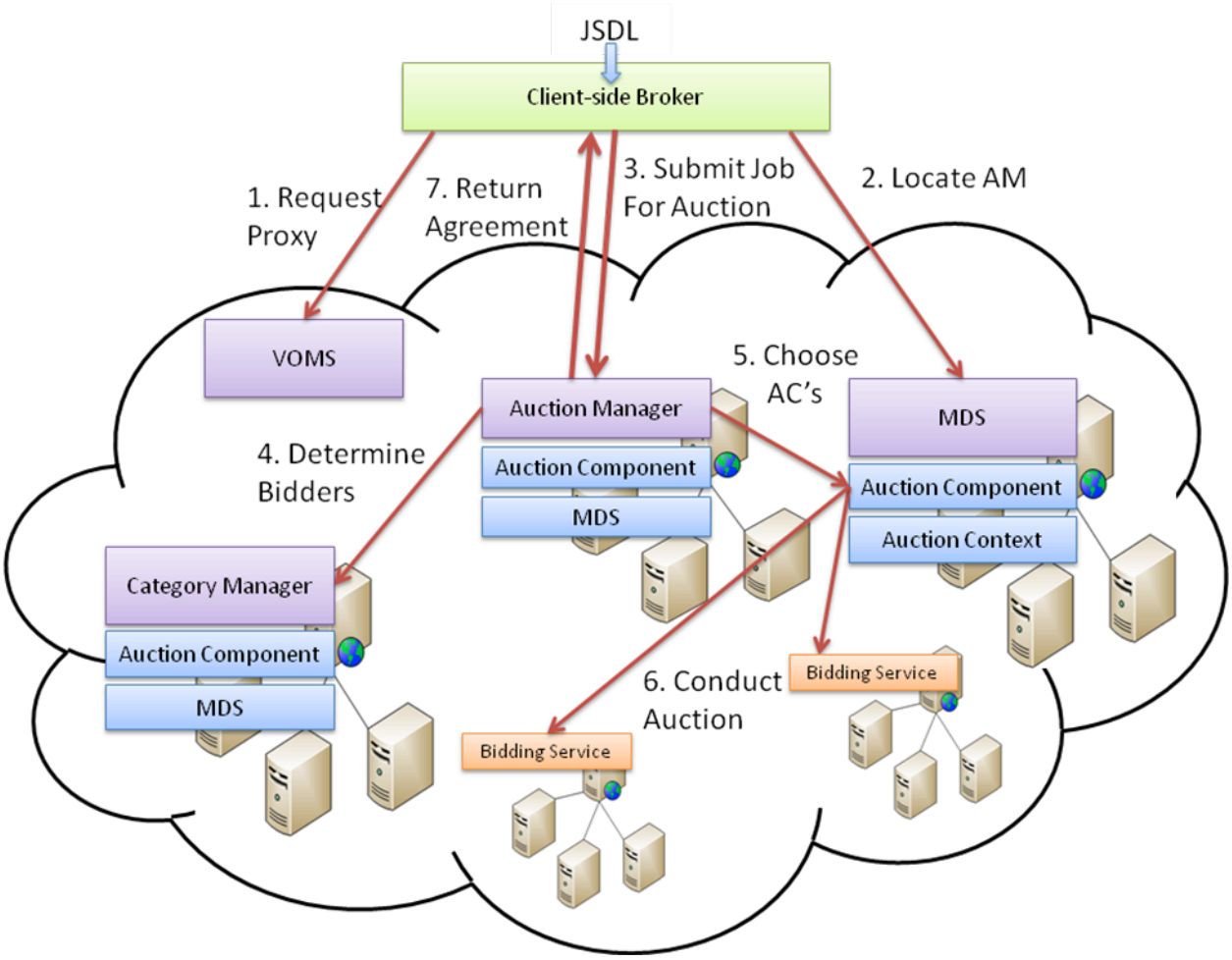

Figure 25.3 DRIVE Resource Allocation

1. First the broker must authenticate itself on behalf of the user with VOMS, a working proxy is created and subsequent access to VO services is provided via this proxy.

2. The broker locates a viable Auction Manager using the MDS index service. The choice of Auction Manager can be based on protocol support, reputation, location or any other user requirements. Having discovered a suitable Auction Manager all allocation is now performed though this service.

3. The job is submitted to the chosen Auction Manager to coordinate the auction process.

4. Suitable resource providers are selected by consulting Category Manager resource profiles. These 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 the auction publishing services.

5. Depending on the auction protocol selected different Auction Components will be instantiated and used, for example for the Garbled Circuit protocol an Auction Issuer is constructed, for the homomorphic and polynomial auction protocols a group of Auction Evaluators will be created.

6. Resource Providers each host a Bidding Service which is responsible for implementing bidding policies, pricing goods in the auction according to a user defined pricing function and following the auction protocol to submit bids. The Bidding Service is invoked by the Auction Manager or Auction Component depending on the protocol.

7. When the auction is complete a winner is determined and an agreement is created by the Contract Service and returned to the client by the Auction Manager, a list of alternative resource providers is maintained in case the agreement cannot be honored. The resource provider records the successful allocation and subsequent agreement through its Reservation Service, this information can then be considered prior to bidding on future auctions.

Job Submission

When an auction is completed the winner and losers are notified of the results, an agreement will be created between the client and the winning bidder(s). At some point after the conclusion of the auction this agreement will need to be confirmed and converted into a hardened contract between the participating parties. Having been allocated resources and presented with the final contract the job is ready to be executed. The hardening and job submission process is shown in Figure 25.4:

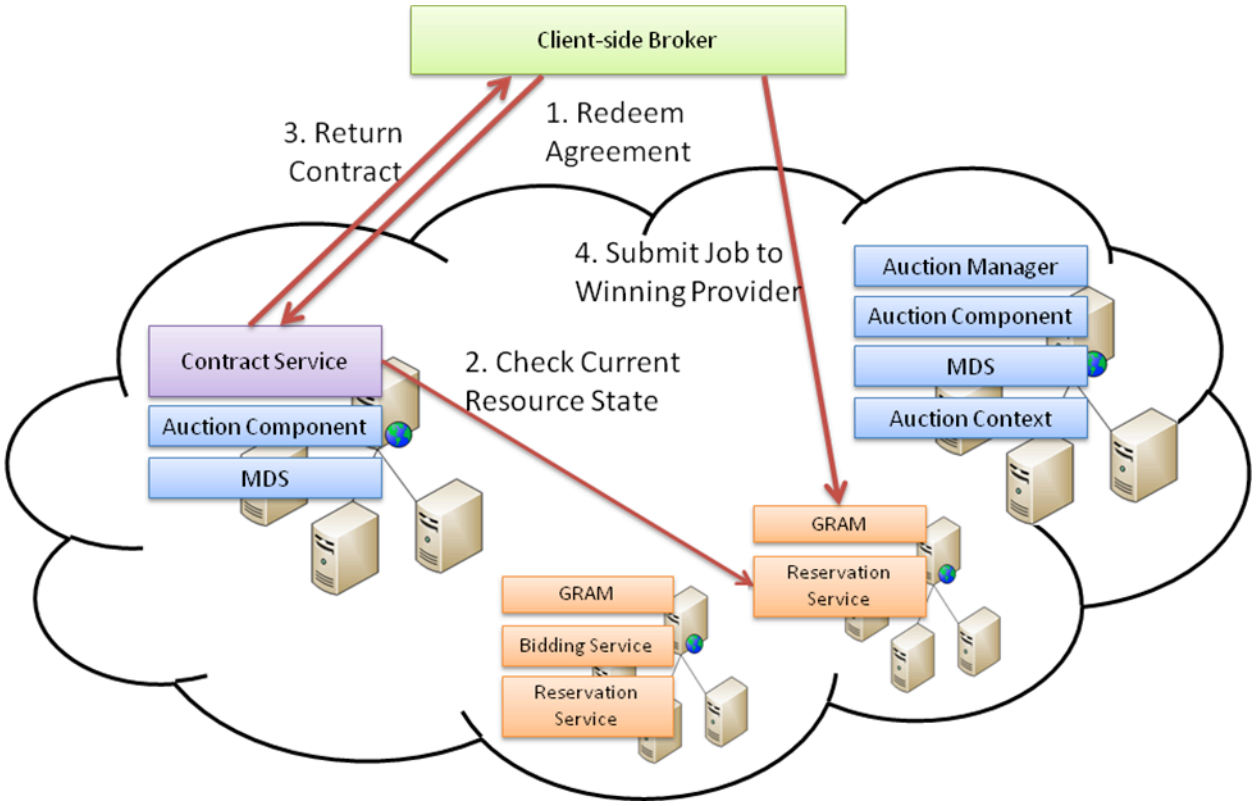

Figure 25.4 DRIVE Job Submission and Execution

1. The agreement created in the allocation process must be converted to a contract before execution. To do this the agreement is passed to the Contract Service for redemption.

2. The Contract Service checks the resource provider Reservation Service for availability which in turn checks load and future commitments. If the resource is still available the local Reservation Service is updated to reflect the hardened contract

3. If the resource is determined to be available the Contract Service returns a contract to the client containing the agreed upon service guarantees in the form of SLAs. If there is no availability the backup list is checked for availability in an attempt to satisfy the agreement elsewhere, if the contract cannot be honored some form of penalty may be imposed (reputation or financial).

4. The final 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.

25.6 Future Trends

As Grid computing evolves towards the vision of a global utility Grid infrastructure a wider range of applications will be seen. We will see an increase in the number of large dynamic collaborations of users all over the world, placing new requirements on Grid infrastructure. This Grid infrastructure will need to dynamically adapt to the rapidly

changing VO structure and scenarios posed by jobs. Trust within the components of the VO will need to be established using techniques such as distributed, secure and verifiable allocation protocols. Additionally trust and security will be incorporated from a lower level than is currently the case, leveraging these capabilities in the Grid infrastructure will facilitate the use of more secure mechanisms. Global reputation services will be available to provide feedback on users and providers in the Grid. One might imagine reputation services going far beyond that of the Grid world, establishing global reputation for users over a variety of domains. A global VO management system is also likely as VOs become larger and more dynamic. It may be created as a hierarchical DNS type system for managing dynamic VOs in much the same way as IP information is dynamically updated today.

25.7 Summary

Grid meta-schedulers turn the focus of resource allocation from the resource to the client, the focus of global allocation is achieved by submitting 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, or coordination with, other meta-schedulers actions therefore reducing system efficiency. Current meta-scheduling technology requires dedicated resources set aside for allocation rather than exploiting the large pool of available VO resources.

DRIVE is a distributed economic meta-scheduler based on a dynamic VO model, in which VO members collaboratively conduct resource allocation using services in the VO. Resource providers contribute a set of obligation Grid services to the VO as a condition of joining and these services are used to host allocation and management services This collaborative resource allocation has the effect of spreading the burden of allocation across participating entities whilst also providing the ability to define distributed allocation protocols. 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. All services are designed using GT4 compliant WSRF web services. The client side broker is light weight with no dependencies on web service containers or Globus.

DRIVE makes use of a plug-in auction mechanism that facilitates the use of arbitrary auction protocols making it a suitable meta-scheduler for various Grid topologies. Users are able to select protocols based on specific requirements, infrastructure availability and their level of trust relationships with the entities participating in 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.

References

1. K. Chard and K. Bubendorfer, *A Distributed Economic Meta-scheduler for the Grid*, in *The 8th IEEE Internation Symposium on Cluster Computing and the Grid (CCGrid '08)*. 2008: Lyon France.

2. I. Foster, C. Kesselman and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, in *Lecture Notes in Computer Science*. 2001.
3. I. Foster and C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit*, in *The International Journal of Supercomputer Applications and High Performance Computing*. 1997. p. 115--128.
4. S. Krawczyk and K. Bubendorfer, *Grid Resource Allocation: Utilisation Patterns*, in *Proceedings of the 6th Australasian Symposium on Grid Computing and e-Research (AusGrid)*. 2008: Wollongong, Australia.
5. R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Frohner, K. LHrentey and F.Spataro, *From gridmap-file to VOMS: managing authorization in a Grid environment.* Future Generation Computer Systems, 2005. **21**(4): p. 549--558.
6. 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)*. 2000. p. 283--289.
7. S. Venugopal, R. Buyya and L. Winton, *A grid service broker for scheduling distributed data-oriented applications on global grids*, in *Proceedings of the 2nd Workshop on Middleware in Grid Computing (MGC '04)*. 2004. p. 75--80.
8. L. Adzigogov, J. Soldatos and L. Polymenakos, *EMPEROR: An OGSA Grid Meta-Scheduler Based on Dynamic Resource Predictions*, in *Journal of Grid Computing*. 2005. p. 19--37.
9. E. Huedo, R.S. Montero and I.M. Llorente, *A Framework for Adaptive Execution on Grids*, in *Software - Practice and Experience*. 2004. p. 631-651.
10. *Technical WhitePaper: Open source metascheduling for Virtual Organizations with the Community Scheduler Framework*. 2003, Platform Computing.
11. G. Haggard, D. Pearce and G. Royle, *Computing Tutte Polynomials*, in*????* 2008.
12. D. Ewa, C. Scott, F. Edward, F. Hunter, G. Robert, G. Nitin, G. Vipin, H.J. Thomas, K. Carl, M. Philip, M. John, M. Gaurang, O. David, V. Karan and Z. Li, *Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance Tracking: The CyberShake Example*, in *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*. 2006, IEEE Computer Society.
13. W. Gentzsch, *Sun Grid Engine: Towards Creating a Compute Power Grid*, in *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*. 2001, IEEE Computer Society.
14. I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster and M. Wilde, *Falkon: a Fast and Light-weight tasK executiON framework*, in *SuperComputing*. 2007.
15. K. Bubendorfer, B. Palmer and I. Welch, eds. *Encyclopedia of Grid Computing Technologies and Applications, ch Trust and Privacy in Grid Resource Auctions*. Information Science Reference, ed. E. Udoh and F. Wang. 2008.
16. W. Thomson, *A Framework for Secure Auctions*, in *Department of Mathematics, Statistics and Computer Science*. 2008, Victoria University of Wellington.
17. J.D. Sonnek and J.B. Weissman, *A Quantitative Comparison of Reputation Systems in the Grid*, in *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*. 2005, IEEE Computer Society.

18. A. Juels and M. Szydlo, *A Two-Server, Sealed-Bid Auction Protocol*, in *Proceedings of the 6th Financial Cryptography Conference (FC 2002)*. 2003. p. 72--86.
19. 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*. 2006: Amsterdam, Holland.