Chapter 23
Architectural Elements of Resource Sharing Networks

Marcos Dias de Assunção
The University of Melbourne, Australia

Rajkumar Buyya
The University of Melbourne, Australia

ABSTRACT

This chapter first presents taxonomies on approaches for resource allocation across resource sharing networks such as Grids. It then examines existing systems and classifies them under their architectures, operational models, support for the life-cycle of virtual organisations, and resource control techniques. Resource sharing networks have been established and used for various scientific applications over the last decade. The early ideas of Grid computing have foreseen a global and scalable network that would provide users with resources on demand. In spite of the extensive literature on resource allocation and scheduling across organisational boundaries, these resource sharing networks mostly work in isolation, thus contrasting with the original idea of Grid computing. Several efforts have been made towards providing architectures, mechanisms, policies and standards that may enable resource allocation across Grids. A survey and classification of these systems are relevant for the understanding of different approaches utilised for connecting resources across organisations and virtualisation techniques. In addition, a classification also sets the ground for future work on inter-operation of Grids.

INTRODUCTION

Since the formulation of the early ideas on meta-computing (Smarr & Catlett, 1992), several research activities have focused on mechanisms to connect worldwide distributed resources. Advances in distributed computing have enabled the creation of Grid-based resource sharing networks such as TeraGrid (Catlett, Beckman, Skow, & Foster, 2006) and Open Science Grid (2005). These networks, composed of multiple resource providers, enable collaborative work and sharing of resources such as computers,

DOI: 10.4018/978-1-60566-661-7.ch023

storage devices and network links among groups of individuals and organisations. These collaborations, widely known as Virtual Organisations (VOs) (Foster, Kesselman, & Tuecke, 2001), require resources from multiple computing sites. In this chapter we focus on networks established by organisations to share computing resources.

Despite the extensive literature on resource allocation and scheduling across organisational boundaries (Butt, Zhang, & Hu, 2003: Grimme, Lepping, & Papaspyrou, 2008; Iosup, Epema, Tannenbaum, Farrellee, & Livny, 2007; Ranjan, Rahman, & Buyya, 2008; Fu, Chase, Chun, Schwab, & Vahdat, 2003; Irwin et al., 2006; Peterson, Muir, Roscoe, & Klingaman, 2006; Ramakrishnan et al., 2006; Huang, Casanova, & Chien, 2006), existing resource sharing networks mostly work in isolation and with different utilisation levels (Assunção, Buyya, & Venugopal, 2008; Iosup et al., 2007), thus contrasting with the original idea of Grid computing (Foster et al., 2001). The early ideas of Grid computing have foreseen a global and scalable network that would provide users with resources on demand.

We have previously demonstrated that there can exist benefits for Grids to share resources with one another such as reducing the costs incurred by over-provisioning (Assunção & Buyya, in press). Hence, it is relevant to survey and classify existing work on mechanisms that can be used to interconnect resources from multiple Grids. A survey and classification of these systems are important in order to understand the different approaches utilised for connecting resources across organisations and to set the ground for future work on inter-operation of resource sharing networks, such as Grids. Taxonomies on resource management systems for resource sharing networks have been proposed (Iosup et al., 2007; Grit, 2005). Buyya et al. (2000) and Iosup et al. (2007) have described the architectures used by meta-scheduler systems and how jobs are directed to the resources where they execute. Grit (2005) has classified the roles of intermediate parties, such as brokers, in resource allocation for virtual computing environments.

This chapter extends existing taxonomies, thus making the following contributions:

- It examines additional systems and classifies them under a larger property spectrum namely resource control techniques, scheduling considering virtual organisations and arrangements for resource sharing.
- It provides classifications and a survey of work on resource allocation and scheduling across organisations, such as centralised scheduling, meta-scheduling and resource brokering in Grid computing. This survey aims to show different approaches to federate organisations in a resource sharing network and to allocate resources to its users. We also present a mapping of the surveyed systems against the proposed classifications.

BACKGROUND

Several of the organisational models followed by existing Grids are based on the idea of VOs. The VO scenario is characterised by resource providers offering different shares of resources to different VOs via some kind of agreement or contract; these shares are further aggregated and allocated to users and groups within each VO. The life-cycle of a VO can be divided into four distinct phases namely creation, operation, maintenance, and dissolution. During the creation phase, an organisation looks for collaborators and then selects a list of potential partners to start the VO. The operation phase is concerned with resource management, task distribution, and usage policy enforcement (Wasson & Humphrey, 2003; Dumitrescu & Foster, 2004). The maintenance phase deals with the adaptation of the VO, such as al-

location of additional resources according to its users' demands. The VO dissolution involves legal and economic issues such as determining the success or failure of the VO, intellectual property and revocation of access and usage privileges.

The problem of managing resources within VOs in Grid computing is further complicated by the fact that resource control is generally performed at the job level. Grid-based resource sharing networks have users with units of work to execute, also called jobs; some entities decide when and where these jobs will execute. The task of deciding where and when to run the users' work units is termed as scheduling. The resources contributed by providers are generally clusters of computers and the scheduling in these resources is commonly performed by Local Resource Management Systems (LRMSs) such as PBS (2005) and SGE (Bulhões, Byun, Castrapel, & Hassaine, 2004). Scheduling of Grid users' applications and allocation of resources contributed by providers is carried out by Grid Resource Management Systems (GRMSs). A GRMS may comprise components such as:

- Meta-schedulers, which communicate with LRMSs to place jobs at the provider sites;
- Schedulers that allocate resources considering how providers and users are organised in virtual organisations (Dumitrescu & Foster, 2005); and
- Resource brokers, which represent users or organisations by scheduling and managing job execution on their behalf.

These components interact with providers' LRMSs either directly or via interfaces provided by the Grid middleware. The Grid schedulers can communicate with one another in various ways, which include via sharing agreements, hierarchical scheduling, Peer-to-Peer (P2P) networks, among others.

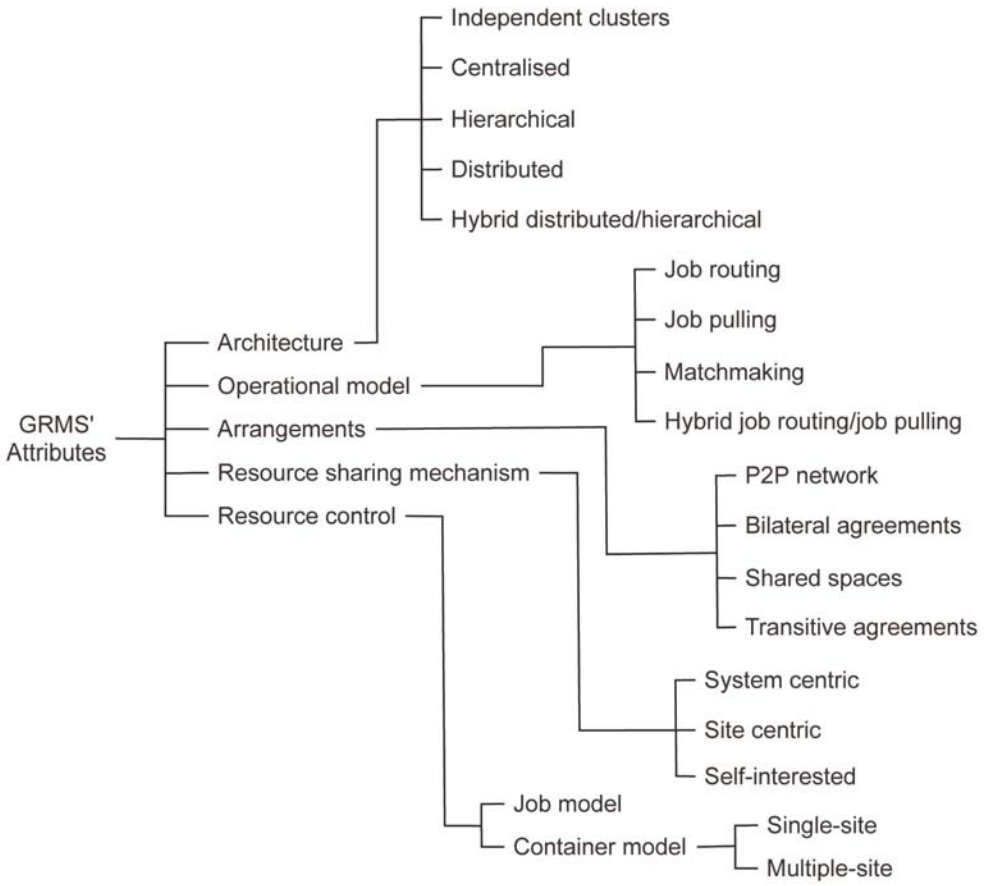
Recently, utility data centres have deployed resource managers that allow the partitioning of physical resources and the allocation of raw resources that can be customised with the operating system and software of the user's preference. This partitioning is made possible by virtualisation technologies such as Xen (Barham et al., 2003; Padala et al., 2007) and VMWare[1]. The use of virtualisation technologies for resource allocation enables the creation of customised virtual clusters (Foster et al., 2006; Chase, Irwin, Grit, Moore, & Sprenkle, 2003; Keahey, Foster, Freeman, & Zhang, 2006). The use of virtualisation technology allows for another form of resource control termed containment (Ramakrishnan et al., 2006), in which remote resources are bound to the users' local computing site on demand. The resource shares can be exchanged across sites by intermediate parties. Thereby, a VO can allocate resources on demand from multiple resource providers and bind them to a customised environment, while maintaining it isolated from other VOs (Ramakrishnan et al., 2006).

In the following sections, we classify existing systems according to their support to the life-cycle of VOs, their resource control techniques and the mechanisms for inter-operation with other systems. We also survey representative work and map them according to the proposed taxonomies.

CLASSIFICATIONS FOR GRID RESOURCE MANAGEMENT SYSTEMS

Buyya et al. (2000) and Iosup et al. (2007) have classified systems according to their architectures and operational models. We present their taxonomy in this section because it classifies the way that schedulers can be organised in a resource sharing network. We have included a new operational model to the taxonomy (i.e. hybrid of job routing and job pulling). Moreover, systems with similar architecture may

Figure 1. Taxonomy on Grid resource management systems



still differ in terms of the mechanisms employed for resource sharing, the self-interest of the system's participants, and the communication model. A Grid system can use decentralised scheduling wherein schedulers communicate their decisions with one another in a co-operative manner, thus guaranteeing the maximisation of the global utility of the system. On the other hand, a broker may represent a particular user community within the Grid, can have contracts with other brokers in order to use the resources they control and allocate resources that maximise its own utility (generally given by the achieved profit). We classify the arrangements between brokers in this section. Furthermore, systems can also differ according to their resource control techniques and support to different stages of the VO life-cycle. This section classifies resource control techniques and the systems' support for virtual organisations. The attributes of GRMSs and the taxonomy are summarised in Figure 1.

Architecture and Operational Models of GRMSs

This section describes several manners in which schedulers and brokers can be organised in Grid systems. Iosup et al. (2007) considered a multiple cluster scenario and classified the architectures possibly used

as Grid resource management systems. They classified the architectures in the following categories:

- **Independent clusters** - each cluster has its LRMS and there is no meta-scheduler component. Users submit their jobs to the clusters of the organisations to which they belong or on which they have accounts. We extend this category by including single-user Grid resource brokers. In this case, the user sends her jobs to a broker, which on behalf of the user submits jobs to clusters the user can access.
- **Centralised meta-scheduler** - there is a centralised entity to which jobs are forwarded. Jobs are then sent by the centralised entity to the clusters where they are executed. The centralised component is responsible for determining which resources are allocated to the job and, in some cases, for migrating jobs if the load conditions change.
- **Hierarchical meta-scheduler** - schedulers are organised in a hierarchy. Jobs arrive either at the root of the hierarchy or at the LRMSs.
- **Distributed meta-scheduler** - cluster schedulers can share jobs that arrive at their LRMSs with one another. Links can be defined either in a static manner (i.e. by the system administrator at the system's startup phase) or in a dynamic fashion (i.e. peers are selected dynamically at runtime). Grit (2007) discusses the types of contracts that schedulers (or brokers) can establish with one another.
- **Hybrid distributed/hierarchical meta-scheduler** - each Grid site is managed by a hierarchical meta-scheduler. Additionally, the root meta-schedulers can share the load with one another.

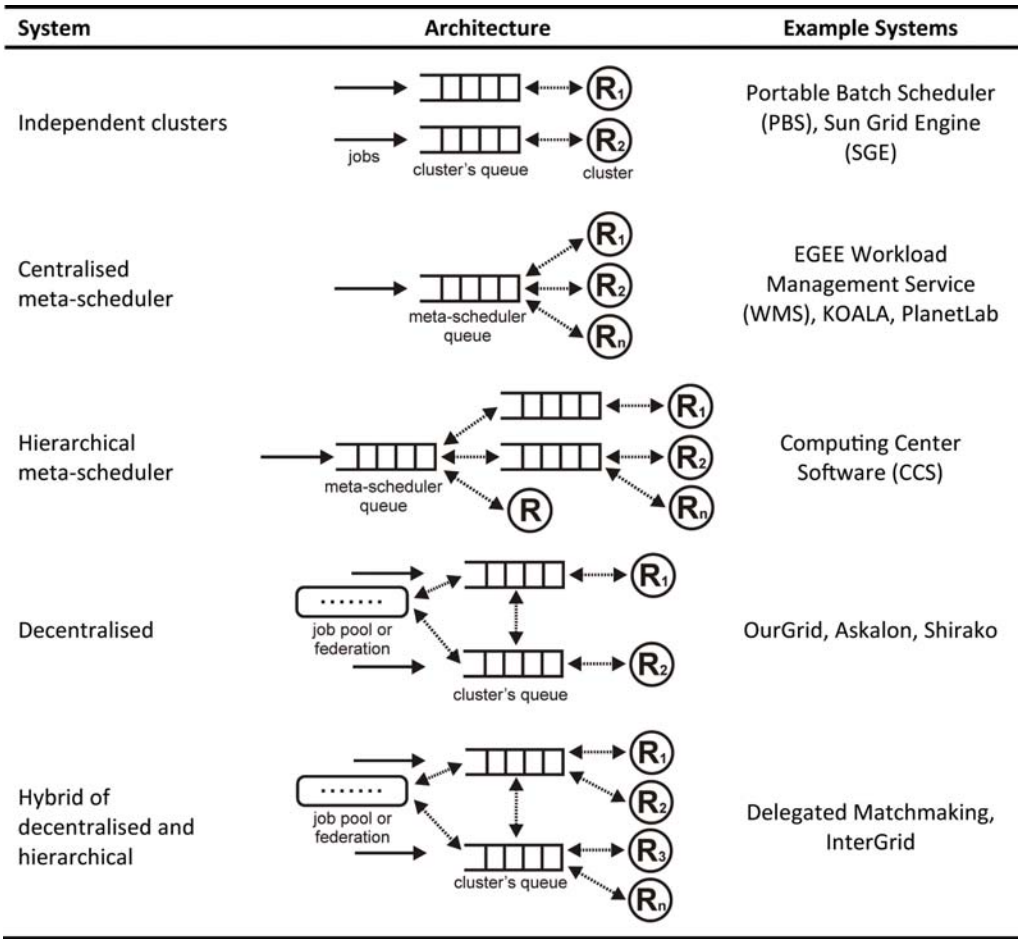
This classification is comprehensive since it captures the main forms through which schedulers and brokers can be organised in resource sharing networks. However, some categories can be further extended. For example, the site schedulers can be organised in several decentralised ways and use varying mechanisms for resource sharing, such as a mesh network in which contracts are established between brokers (Irwin et al., 2006; Fu et al., 2003) or via a P2P network with a bartering-inspired economic mechanism for resource sharing (Andrade, Brasileiro, Cirne, & Mowbray, 2007).

Iosup et al. also classified a group of systems according to their operational model; the operational model corresponds to the mechanism that ensures jobs entering the system arrive at the resource in which they run. They have identified three operational models:

- **Job routing**, whereby jobs are routed by the schedulers from the arrival point to the resources where they run through a push operation (scheduler-initiated routing);
- **Job pulling**, through which jobs are pulled from a higher-level scheduler by resources (resource-initiated routing); and
- **Matchmaking**, wherein jobs and resources are connected to one another by the resource manager, which acts as a broker matching requests from both sides.

We add a fourth category to the classification above in which the operational model can be a **hybrid of job routing and job pulling**. Examples of such cases include those that use a job pool to (from) which jobs are pushed (pulled) by busy (unoccupied) site schedulers (Grimme et al., 2008). (See Figure 2).

Figure 2. Architecture models of GRMSs



Arrangements Between Brokers in Resource Sharing Networks

This section describes the types of arrangements that can be established between clusters in resource sharing networks when decentralised or semi-decentralised architectures are in place. It is important to distinguish between the way links between sites are established and their communication pattern; from the mechanism used for negotiating the resource shares. We classify the work according to the communication model in the following categories:

- **P2P network** - the sites of the resource sharing network are peers in a P2P network. They use the network to locate sites where the jobs can run (Butt et al., 2003; Andrade, Cirne, Brasileiro, & Roisenberg, 2003).
- **Bilateral sharing agreements** - sites establish bilateral agreements through which a site can locate another suitable site to run a given job. The redirection or acceptance of jobs occurs only

between sites that have a sharing agreement (Epema, Livny, Dantzig, Evers, & Pruyne, 1996).
- **Shared spaces** - sites co-ordinate resource sharing via shared spaces such as federation directories and tuple spaces (Grimme et al., 2008; Ranjan et al., 2008).
- **Transitive agreements** - this is similar to bilateral agreements. However, a site can utilise resources from another site with which it has no direct agreement (Fu et al., 2003; Irwin et al., 2006).

Although existing work can present similar communication models or similar organisational forms for brokers or schedulers, the resource sharing mechanisms can differ. The schedulers or brokers can use mechanisms for resource sharing from the following categories:

- **System centric** - the mechanism is designed with the goal of maximising the overall utility of the participants. Such mechanisms aim to, for example, balance the load between sites (Iosup et al., 2007) and prevent free-riding (Andrade et al., 2007).
- **Site centric** - brokers and schedulers are driven by the interest of maximising the utility of the participants within the site they represent without the explicit goal of maximising the overall utility across the system (Butt et al., 2003; Ranjan, Harwood, & Buyya, 2006).
- **Self-interested** - brokers act with the goal of maximising their own utility, generally given by profit, yet satisfying the requirements of their users. They do not take into account the utility of the whole system (Irwin et al., 2006).

Resource Control Techniques

The emergence of virtualisation technologies has resulted in the creation of testbeds wherein multiple-site slices (i.e. multiple-site containers) are allocated to different communities (Peterson et al., 2006). In this way, slices run concurrently and are isolated from each other. This approach, wherein resources are bound to a virtual execution environment or workspace where a service or application can run, is termed here as a container model. Most of the existing Grid middleware employ a job model in which jobs are routed until they reach the sites' local batch schedulers for execution. It is clear that both models can co-exist, thus an existing Grid technology can be deployed in a workspace enabled by container-based resource management (Ramakrishnan et al., 2006; Montero, Huedo, & Llorente, 2008). We classify systems in the following categories:

- **Job model** - this is the model currently utilised by most of the Grid systems. The jobs are directed or pulled across the network until they arrive at the nodes where they are finally executed.
- **Container-based** - resource managers in this category can manage a cluster of computers within a site by means of virtualisation technologies (Keahey et al., 2006; Chase et al., 2003). They bind resources to virtual clusters or workspaces according to a customer's demand. They commonly provide an interface through which one can allocate a set of nodes (generally virtual machines) and configure them with the operating system and software of choice.
    - **Single-site** - these container-based resource managers allow the user to create a customised virtual cluster using shares of the physical machines available at the site. These resource managers are termed here as single-site because they usually manage the resources of one administrative site (Fontán, Vázquez, Gonzalez, Montero, & Llorente, 2008; Chase et al.,

2003), although they can be extended to enable container-based resource control at multiple sites (Montero et al., 2008).

- **Multiple-site** - existing systems utilise the features of single-site container-based resource managers to create networks of virtual machines on which an application or existing Grid middleware can be deployed (Ramakrishnan et al., 2006). These networks of virtual machines are termed here as multiple-site containers because they can comprise resources bound to workspaces at multiple administrative sites. These systems allow a user to allocate resources from multiple computing sites thus forming a network of virtual machines or a multiple-site container (Irwin et al., 2006; Shoykhet, Lange, & Dinda, 2004; Ruth, Jiang, Xu, & Goasguen, 2005; Ramakrishnan et al., 2006). This network of virtual machines is also referred to as virtual Grid (Huang et al., 2006) or slice (Peterson et al., 2006).

Some systems such as Shirako (Irwin et al., 2006) and VioCluster (Ruth, McGachey, & Xu, 2005) provide container-based resource control. Shirako also offers resource control at the job level (Ramakrishnan et al., 2006) by providing a component that is aware of the resources leased. This component gives recommendations on which site can execute a given job.

Taxonomy on Virtual Organisations

The idea of user communities or virtual organisations underlies several of the organisational models adopted by Grid systems and guides many of the efforts on providing fair resource allocation for Grids. Consequently, the systems can be classified according to the VO awareness of their scheduling and resource allocation mechanisms. One may easily advocate that several systems, that were not explicitly designed to support VOs, can be used for resource management within a VO. We restrict ourselves to provide a taxonomy that classifies systems according to (i) the VO awareness of their resource allocation and scheduling mechanisms; and (ii) the provision of tools for handling different issues related to the VO life-cycle. For the VO awareness of scheduling mechanisms we can classify the systems in:

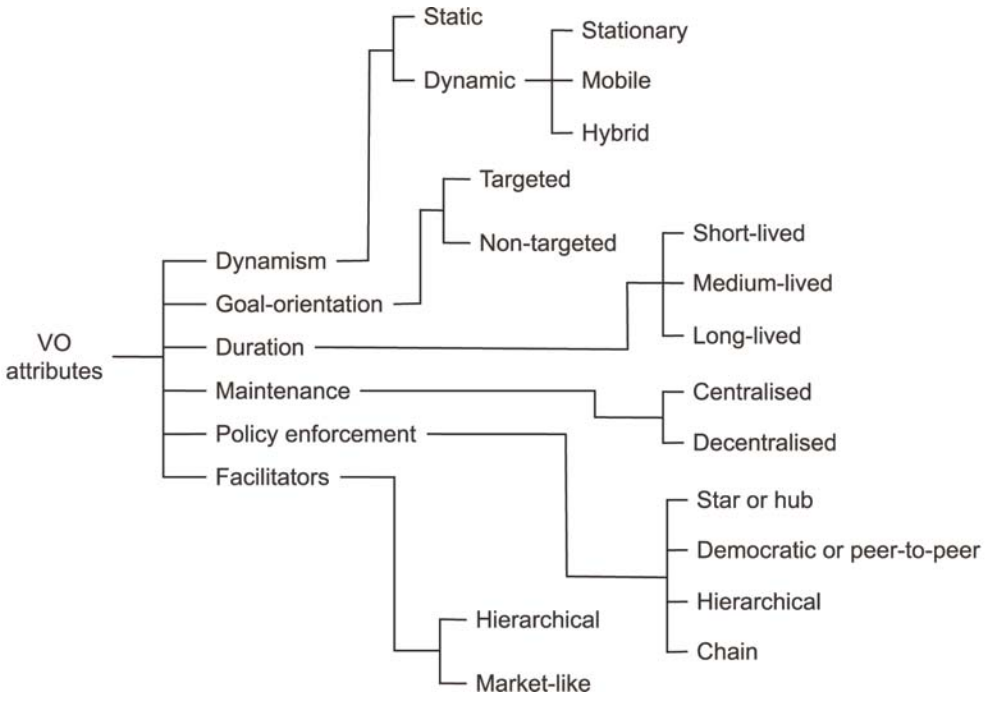
- **Multiple VOs** - those scheduling mechanisms that perform scheduling and allocation taking into consideration the various VOs existing within a Grid; and
- **Single VO** - those mechanisms that can be used for scheduling within a VO.

Furthermore, the idea of VO has been used in slightly different ways in the Grid computing context. For example, in the Open Science Grid (OSG), VOs are recursive and may overlap. We use several criteria to classify VOs as presented in Figure 3.

With regard to dynamism, we classify VOs as static and dynamic (Figure 3). Although Grid computing is mentioned as the enabler for dynamic VOs, it has been used to create more static and long-term collaborations such as APAC (2005), EGEE (2005), the UK National e-Science Centre (2005), and TeraGrid (Catlett et al., 2006). A static VO has a pre-defined number of participants and its structure does not change over time. A dynamic VO presents a number of participants that changes constantly as the VO evolves (Wesner, Dimitrakos, & Jeffrey, 2004). New participants can join, whereas existing participants may leave.

A dynamic VO can be stationary or mobile. A stationary VO is generally composed of highly specialised resources including supercomputers, clusters of computers, personal computers and data resources. The

Figure 3. Taxonomy on Grid facilitated VOs



components of the VO are not mobile. In contrast, a mobile VO is composed of mobile resources such as Personal Digital Assistants (PDAs), mobile phones. The VO is highly responsive and adapts to different contexts (Wesner et al., 2004). Mobile VOs can be found in disaster handling and crisis management situations. Moreover, a VO can be hybrid, having both stationary and mobile components.

Considering goal-orientation, we divide VOs into two categories: targeted and non-targeted (Figure 3). A targeted VO can be an alliance or collaboration created to explore a market opportunity or achieve a common research goal. A VO for e-Science collaboration is an example of a targeted VO as the participants have a common goal (Hey & Trefethen, 2002). A non-targeted VO is characterised by the absence of a common goal; it generally comprises participants who pursue different goals, yet benefit from the VO by pooling resources. This VO is highly dynamic because participants can leave when they achieve their goals.

VOs can be short-, medium- or long-lived (Figure 3). A short-lived VO lasts for minutes or hours. A medium-lived VO lasts for weeks and is formed, for example, when a scientist needs to carry out experiments that take several days to finish. Data may be required to carry out such experiments. This scenario may be simplified if the VO model is used; the VO may not be needed as soon as the experiments have been carried out. A long-lived VO is formed to explore a market opportunity (goal-oriented) or to pool resources to achieve disparate objectives (non-targeted). Such endeavours normally last from months to years; hence, we consider a long-lived VO to last for several months or years.

As discussed in the previous section, the formation and maintenance of a VO present several challenges. These challenges have been tackled in different ways, which in turn have created different formation and

maintenance approaches. We thus classify the formation and membership, or maintenance, as centralised and decentralised (Figure 3). The formation and membership of a centralised VO is controlled by a trusted third party, such as Open Science Grid (2005) or the Enabling Grids for E-SciencE (2005). OSG provides an open market where providers and users can advertise their needs and intentions; a provider or user may form a VO for a given purpose. EGEE provides a hierarchical infrastructure to enable the formation of VOs. On the other hand, in a decentralised controlled VO, no third party is responsible for enabling or controlling the formation and maintenance. This kind of VO can be complex as it can require multiple Service Level Agreements (SLAs) to be negotiated among multiple participants. In addition, the monitoring of SLAs and commitment of the members are difficult to control. The VO also needs to self-adapt when participants leave or new participants join.

Regarding the enforcement of policies, VOs can follow different approaches, such as hub or democratic. This is also referred to as topology. Katzy et al. (2005) classify VOs in terms of topology, identifying the following types: chain, star or hub, and peer-to-peer. Sairamesh et al. (2005) identify business models for VOs; the business models are analogous to topologies. However, by discussing the business models for VOs, the authors are concerned with a larger set of problems, including enforcement of policies, management, trust and security, and financial aspects. In our taxonomy, we classify the enforcement and monitoring of policies as star or hub, democratic or peer-to-peer, hierarchical, and chain (Figure 3).

Some projects such as Open Science Grid (2005) and EGEE (2005) aim to establish consortiums or clusters of organisations, which in turn allow the creation of dynamic VOs. Although not very related to the core issues of VOs, they aim to address an important problem: the establishment of trust between organisations and the means for them to look for and find potential partners. These consortiums can be classified as hierarchical and market-like (Figure 3). A market-like structure is any infrastructure that offers a market place, which organisations can join and present interests in starting a new collaboration or accepting to participate in an ongoing collaboration. These infrastructures may make use of economic models such as auctions, bartering, and bilateral negotiation.

A SURVEY OF EXISTING WORK

This section describes relevant work of the proposed taxonomy in more detail. First, it describes work on a range of systems that have a decentralised architecture. Some systems present a hierarchy of scheduling whereby jobs are submitted to the root of the hierarchy or to their leaves; in either case, the jobs execute at the leaves of the hierarchical structure. Second, this section presents systems of hierarchical structure, resource brokers and meta-scheduling frameworks. During the past few years, several Grid-based resource sharing networks and other testbeds have been created. Third, we discuss the work on inter-operation between resource sharing networks. Finally, this section discusses relevant work focusing on VO issues.

Distributed Architecture Based Systems

Condor Flocking: The flocking mechanism used by Condor (Epema et al., 1996) provides a software approach to interconnect pools of Condor resources. The mechanism requires manual configuration of sharing agreements between Condor pools. Each pool owner and each workstation owner maintains full control of when their resources can be used by external jobs.

The developers of Condor flocking opted for a layered design for the flocking mechanism, which enables the Condor's Central Manager (CM) (Litzkow, Livny, & Mutka, 1988) and other Condor machines to remain unmodified and operate transparently from the flock.

The basis of the flocking mechanism is formed by Gateway Machines (GW). There is at least one GW in each Condor pool. GWs act as resource brokers between pools. Each GW has a configuration file describing the subset of connections it maintains with other GWs. Periodically, a GW queries the status of its pool from the CM. From the list of resources obtained, the GW makes a list of those resources that are idle. The GW then sends this list to the other GWs to which it is connected. Periodically, the GW that received this list chooses a machine from the list, and advertises itself to the CM with the characteristics of this machine. The flocking protocol (which is a modified version of the normal Condor protocol) allows the GWs to create shadow processes that so that a submission machine is under the impression of contacting the execution machine directly.

Self-Organizing Flock of Condors: The original flocking scheme of Condor has the drawback that knowledge about all pools with which resources can be shared need to be known a priori before starting Condor (Epema et al., 1996). This static information poses limitations regarding the number of resources available and resource discovery. Butt et al. (2003) introduced a self-organising resource discovery mechanism for Condor, which allows pools to discover one another and resources available dynamically. The P2P network used by the flocking mechanism is based on Pastry and takes into account the network proximity. This may result in saved bandwidth in data transfer and faster communications. Experiments with this implementation considering four pools with four machines each were provided. Additionally, simulation results demonstrated the performance of the flocking mechanism when interconnecting 1,000 pools.

Shirako: Shirako (Irwin et al., 2006) is a system for on-demand leasing of shared networked resources across clusters. Shirako's design goals include: autonomous providers, who may offer resources to the system on a temporary basis and retain the ultimate control over them; adaptive guest applications that lease resources from the providers according to changing demand; pluggable resource types, allowing participants to include various types of resources, such as network links, storage and computing; brokers that provide guest applications with an interface to acquire resources from resource providers; and allocation policies at guest applications, brokers and providers, which define the manner resources are allocated in the system.

Shirako utilises a leasing abstraction in which authorities representing provider sites offer their resources to be provisioned by brokers to guest applications. Shirako brokers are responsible for coordinating resource allocation across provider sites. The provisioning of resources determines how much of each resource each guest application receives, when and where. The site authorities define how much resource is given to which brokers. The authorities also define which resources are assigned to serve requests approved by a broker. When a broker approves a request, it issues a ticket that can be redeemed for a lease at a site authority. The ticket specifies the type of resource, the number of resource units granted and the interval over which the ticket is valid. Sites issue tickets for their resources to brokers; the brokers' polices may decide to subdivide or aggregate tickets.

A service manager is a component that represents the guest application and uses the lease API provided by Shirako to request resources from the broker. The service manager determines when and how to redeem existing tickets, extend existing leases, or acquire new leases to meet changing demand. The system allows guest applications to renew or extend their leases. The broker and site authorities match accumulated pending requests with resources under the authorities' control. The broker prioritises requests

and selects resource types and quantities to serve them. The site authority assigns specific resource units from its inventory to fulfill lease requests that are backed by a valid ticket. Site authorities use Cluster on Demand (Chase et al., 2003) to configure the resources allocated at the remote sites.

The leasing abstraction provided by Shirako is a useful basis to co-ordinate resource sharing for systems that create distributed virtual execution environments of networked virtual machines (Keahey et al., 2006; Ruth, Rhee, Xu, Kennell, & Goasguen, 2006; Adabala et al., 2005; Shoykhet et al., 2004).

Ramakrishnan et al. (2006) used Shirako to provide a hosting model wherein Grid deployments run in multiple-site containers isolated from one another. An Application Manager (AM), which is the entry point of jobs from a VO or Grid, interacts with a Grid Resource Oversight Coordinator (GROC) to obtain a recommendation of a site to which jobs can be submitted. The hosting model uses Shirako's leasing core. A GROC performs the functions of leasing resources from computing sites and recommending sites for task submission. At the computing site, Cluster on Demand is utilised to provide a virtual cluster used to run Globus 4 along with Torque/MAUI.

VioCluster: VioCluster is a system that enables dynamic machine trading across clusters of computers (Ruth, McGachey, & Xu, 2005). VioCluster introduces the idea of virtual domain. A virtual domain, originally comprising its physical domain of origin (i.e. a cluster of computers), can grow in the number of computing resources, thus dynamically allocating resources from other physical domains according to the demands of its user applications.

VioCluster presents two important system components: the creation of dynamic virtual domains and the mechanism through which resource sharing is negotiated. VioCluster uses machine and network virtualisation technology to move machines between domains. Each virtual domain has a broker that interacts with other domains. A broker has a borrowing policy and a lending policy. The borrowing policy determines under which circumstances the broker will attempt to obtain more machines. The lending policy governs when it is willing to let another virtual domain make use of machines within its physical domain.

The broker represents a virtual domain when negotiating trade agreements with other virtual domains. It is the broker's responsibility to determine whether trades should occur. The policies for negotiating the resources specify: the reclamation, that is, when the resources will be returned to their home domain; machine properties, which represent the machines to be borrowed; and the machines' location as some applications require communication. The borrowing policy must be aware of the communication requirements of user applications.

Machine virtualisation simplifies the transfer of machines between domains. When a machine belonging to a physical domain B is borrowed by a virtual domain A, it is utilised to run a virtual machine. This virtual machine matches the configuration of the machines in physical domain A. Network virtualisation enables the establishment of virtual network links connecting the new virtual machine to the nodes of domain A. For the presented prototype, PBS is used to manage the nodes of the virtual domain. PBS is aware of the computers' heterogeneity and never schedules jobs on a mixture of virtual and physical machines. The size of the work queue in PBS was used as a measure of the demand within a domain.

OurGrid: OurGrid (Andrade et al., 2003) is a resource sharing system organised as a P2P network of sites that share resources equitably in order to form a Grid to which they all have access. OurGrid was designed with the goal of easing the assembly of Grids, thus it provides connected sites with access to the Grid resources with a minimum of guarantees needed. OurGrid is used to execute Bag-of-Tasks (BoT) applications. BoT are parallel applications composed of a set of independent tasks that do not communicate with one another during their execution. In contrast to other Grid infrastructures, the system

does not require offline negotiations if a resource owner wants to offer her resources to the Grid.

OurGrid uses a resource exchange mechanism termed network of favours. A participant A is doing a favour to participant B when A allows B to use her resources. According to the network of favours, every participant does favours to other participants expecting the favours to be reciprocated. In conflicting situations, a participant prioritises those who have done favours to it in the past. The more favours a participant does, the more it expects to be rewarded. The participants locally account their favours and cannot profit from them in another way than expecting other participants to do them some favours. Detailed experiments have demonstrated the scalability of the network of favours (Andrade et al., 2007), showing that the larger the network becomes, the more fair the mechanism performs.

The three participants in the OurGrid's resource sharing protocol are clients, consumers, and providers. A client requires access to the Grid resources to run her applications. The consumer receives requests from the client to find resources. When the client sends a request to the consumer, the consumer first finds the resources able to serve the request and then executes the tasks on the resources. The provider manages the resources shared in the community and provides them to consumers.

Delegated Matchmaking: Iosup et al. (2007) introduced a matchmaking protocol in which a computing site binds resources from remote sites to its local environment. A network of sites, created on top of the local cluster schedulers, manages the resources of the interconnected Grids. Sites are organised according to administrative and political agreements so that parent-child links can be established. Then, a hierarchy of sites is formed with the Grid clusters at the leaves of the hierarchy. After that, supplementary to the hierarchical links, sibling links are established between sites that are at the same hierarchical level and operate under the same parent site. The proposed delegated matchmaking mechanism enables requests for resources to be delegated up and down the hierarchy thus achieving a decentralised network.

The architecture is different from work wherein a scheduler forwards jobs to be executed on a remote site. The main idea of the matchmaking mechanism is to delegate ownership of resources to the user who requested them through this network of sites, and add the resources transparently to the user's local site. When a request cannot be satisfied locally, the matchmaking mechanism adds remote resources to the user's site. This simplifies security issues since the mechanism adds the resources to the trusted local resource pool. Simulation results show that the mechanism leads to an increase in the number of requests served by the interconnected sites.

Grid Federation: Ranjan et al. (2005) proposed a system that federates clusters of computers via a shared directory. Grid Federation Agents (GFAs), representing the federated clusters, post quotes about idle resources (i.e. a claim stating that a given resource is available) and, upon the arrival of a job, query the directory to find a resource suitable to execute the job. The directory is a shared-space implemented as a Distributed Hash Table (DHT) P2P network that can match quotes and user requests (Ranjan et al., 2008).

An SLA driven co-ordination mechanism for Grid superscheduling has also been proposed (Ranjan et al., 2006). GFAs negotiate SLAs and redirect requests through a Contract-Net protocol. GFAs use a greedy policy to evaluate resource requests. A GFA is a cluster resource manager and has control over the cluster's resources. GFAs engage into bilateral negotiations for each request they receive, without considering network locality.

Askalon: Siddiqui et al. (2006) introduced a capacity planning architecture with a three-layer negotiation protocol for advance reservation on Grid resources. The architecture is composed of allocators that make reservations of individual nodes and co-allocators that reserve multiple nodes for a single Grid application. A co-allocator receives requests from users and generates alternative offers that the user

can utilise to run her application. A co-allocation request can comprise a set of allocation requests, each allocation request corresponding to an activity of the Grid application. A workflow with a list of activities is an example of Grid application requiring co-allocation of resources. Co-allocators aim to agree on Grid resource sharing. The proposed co-ordination mechanism produces contention-free schedules either by eliminating conflicting offers or by lowering the objective level of some of the allocators.

GRUBER/DI-GRUBER: Dumitrescu et al. (2005) highlighted that challenging usage policies can arise in VOs that comprise participants and resources from different physical organisations. Participants want to delegate access to their resources to a VO, while maintaining such resources under the control of local usage policies. They seek to address the following issues:

- How usage policies are enforced at the resource and VO levels.
- What mechanisms are used by a VO to ensure policy enforcement.
- How the distribution of policies to the enforcement points is carried out.
- How policies are made available to VO job and data planners.

They have proposed a policy management model in which participants can specify the maximum percentage of resources delegated to a VO. A VO in turn can specify the maximum percentage of resource usage it wishes to delegate to a given VO's group. Based on this model above, they have proposed a Grid resource broker termed GRUBER (Dumitrescu & Foster, 2005). GRUBER architecture is composed of four components, namely:

- Engine: which implements several algorithms to detect available resources.
- Site monitoring: is one of the data providers for the GRUBER engine. It is responsible for collecting data on the status of Grid elements.
- Site selectors: consist of tools that communicate with the engine and provide information about which sites can execute the jobs.
- Queue manager: resides on the submitting host and decides how many jobs should be executed and when.

Users who want to execute jobs, do so by sending them to submitting hosts. The integration of existing external schedulers with GRUBER is made in the submitting hosts. The external scheduler utilises GRUBER either as the queue manager that controls the start time of jobs and enforces VO policies, or as a site recommender. The second case is applicable if the queue manager is not available.

DI-GRUBER, a distributed version of GRUBER, has also been presented (Dumitrescu, Raicu, & Foster, 2005). DI-GRUBER works with multiple decision points, which gather information to steer resource allocations defined by Usage Service Level Agreements (USLAs). These points make decisions on a per-job basis to comply with resource allocations to VO groups. Authors advocated that 4 to 5 decision points are enough to handle the job scheduling of a Grid 10 times larger than Grid3 at the time the work was carried out (Dumitrescu, Raicu, & Foster, 2005).

Other important work: Balazinska et al. (2004) have proposed a load balancing mechanism for Medusa. Medusa is a stream processing system that allows the migration of stream processing operators from overloaded to under-utilised resources. The request offloading is performed based on the marginal cost of the request. The marginal cost for one participant is given by the increase (decrease) in the cost curve given by the acceptance (removal) of the request from the requests served by the participant.

NWIRE (Schwiegelshohn & Yahyapour, 1999) links various resources to a metacomputing system, also termed meta-system. It also enables the scheduling in these environments. A meta-system comprises interconnected MetaDomains. Each MetaDomain is managed by a MetaManager that manages a set of ResourceManagers. A ResourceManager interfaces the scheduler at the cluster level. The MetaManager permanently collects information about all of its resources. It handles all requests inside its MetaDomain and works as a resource broker to other MetaDomains. In this way, requests received by a MetaManager can be submitted either by users within its MetaDomain or by other MetaManagers. Each MetaManager contains a scheduler that maps requests for resources to a specific resource in its MetaDomain.

Grimme et al. (2008) have presented a mechanism for collaboration between resource providers by means of job interchange though a central job pool. According to this mechanism, a cluster scheduler adds to the central pool jobs that cannot be started immediately. After scheduling local jobs, a local scheduler can schedule jobs from the central pool if resources are available.

Dixon et al. (2006) have provided a tit-for-tat or bartering mechanism based on local, non-transferable currency for resource allocation in large-scale distributed infrastructures such as PlanetLab. The currency is maintained locally within each domain in the form of credit given to other domains for providing resources in the past. This creates pair-wise relationships between administrative domains. The mechanism resembles OurGrid's network of favours (Andrade et al., 2003). The information about exchanged resources decays with time, so that recent behaviour is more important. Simulation results showed that, for an infrastructure like PlanetLab, the proposed mechanism is more fair than the free-for-all approach currently adopted by PlanetLab.

Graupner et al. (2002) have introduced a resource control architecture for federated utility data centres. In this architecture, physical resources are grouped in virtual servers and services are mapped to virtual servers. The meta-system is the upper layer implemented as an overlay network whose nodes contain descriptive data about the two layers below. Allocations change according to service demand, which requires to the control algorithms to be reactive and deliver quality solutions. The control layer performs allocation of services to virtual server environments and its use has been demonstrated by a capacity control example for a homogeneous Grid cluster.

Hierarchical Systems, Brokers and Meta-Scheduling

This section describes some systems that are organised in a hierarchical manner. We also describe work on Grid resource brokering and frameworks that can be used to build meta-schedulers.

Computing Center Software (CCS): CCS (Brune, Gehring, Keller, & Reinefeld, 1999) is a system for managing geographically distributed high-performance computers. It consists of three components, namely: the CCS, which is a vendor-independent resource management software for local HPC systems; the Resource and Service Description (RSD), used by the CCS to specify and map hardware and software components of computing environments; and the Service Coordination Layer (SCL), which co-ordinates the use of resources across computing sites.

The CCS controls the mapping and scheduling of interactive and parallel jobs on massively parallel systems. It uses the concept of island, wherein each island has components for user interface, authorisation and accounting, scheduling of user requests, access to the physical parallel system, system control, and management of the island. At the meta-computing level, the Center Resource Manager (CRM) exposes scheduling and brokering features of the islands. The CRM is a management tool atop the CCS islands. When a user submits an application, the CRM maps the user request to the static and dynamic informa-

tion on resources available. Once the resources are found, CRM requests the allocation of all required resources at all the islands involved. If not all resources are available, the CRM either re-schedules the request or rejects it. Center Information Server (CIS) is a passive component that contains information about resources and their statuses, and is analogous to Globus Metacomputing Directory Service (MDS) (Foster & Kesselman, 1997). It is used by the CRM to obtain information about resources available.

The Service Co-ordination Layer (SCL) is located one level above the local resource management systems. The SCL co-ordinates the use of resources across the network of islands. It is organised as a network of co-operating servers, wherein each server represents one computing centre. The centres determine which resources are made available to others and retain full autonomy over them.

EGEE Workload Management System (WMS): EGEE WMS (Vázquez-Poletti, Huedo, Montero, & Llorente, 2007) has a semi-centralised architecture. One or more schedulerscan be installed in the Grid infrastructure, each providing scheduling functionality for a group of VOs. The EGEE WMS components are: The User Interface (UI) from where the user dispatches the jobs; the Resource Broker (RB), which uses Condor-G (Frey, Tannenbaum, Livny, Foster, & Tuecke, 2001); the Computing Element (CE), which is the cluster front-end; the Worker Nodes (WNs), which are the cluster nodes; the Storage Element (SE), used for job files storage; and the Logging and Bookkeeping service (LB) that registers job events.

Condor-G: Condor-G (Frey et al., 2001) leverages software from Globus and Condor (Frey et al., 2001) and allows users to utilise resources spanning multiple domains as if they all belong to one personal domain. Although Condor-G can be viewed as a resource broker itself (Venugopal, Nadiminti, Gibbins, & Buyya, 2008), it can also provide a framework to build meta-schedulers.

The GlideIn mechanism of Condor-G is used to start a daemon process on a remote resource. The process uses standard Condor mechanisms to advertise the resource availability to a Condor collector process, which is then queried by the Scheduler to learn about available resources. Condor-G uses Condor mechanisms to match locally queued jobs to the resources advertised by these daemons and to execute them on those resources. Condor-G submits an initial GlideIn executable (a portable shell script), which in turn uses GSI-authenticated GridFTP to retrieve the Condor executables from a central repository. By submitting GlideIns to all remote resources capable of serving a job, Condor-G can guarantee optimal queuing times to user applications.

Gridbus Broker: Gridbus Grid resource broker (Venugopal et al., 2008) is user-centric broker that provides scheduling algorithms for both computing- and data-intensive applications. In Gridbus, each user has her own broker, which represents the user by (i) selecting resources that minimise the user's quality of service constraints such as execution deadline and budget spent; (ii) submitting jobs to remote resources; and (iii) copying input and output files. Gridbus interacts with various Grid middleware's (Venugopal et al., 2008).

Gridway: GridWay (Huedo, Montero, & Llorente, 2004) is a Globus based resource broker that provides a framework for execution of jobs in a 'submit and forget' fashion. The framework performs job submission and execution monitoring. Job execution adapts itself to dynamic resource conditions and application demands in order to improve performance. The adaptation is performed through application migration following performance degradation, sophisticated resource discovery, requirements change, or remote resource failure.

The framework is modular wherein the following modules can be set on a per-job basis: resource selector, performance degradation evaluator, prolog, wrapper and epilog. The name of the first two modules or steps are intuitive, so we describe here only the last three. During prolog, the component

responsible for job submission (i.e. submission manager) submits the prolog executable, which configures the remote system and transfers executable and input files. In the case of restart of an execution, the prolog also transfers restart files. The wrapper executable is submitted after prolog and wraps the actual job in order to obtain its exit code. The epilog is a script that transfers the output files and cleans the remote resource.

GridWay also enables the deployment of virtual machines in a Globus Grid (Rubio-Montero, Huedo, Montero, & Llorente, 2007). The scheduling and selection of suitable resources is performed by GridWay whereas a virtual workspace is provided for each Grid job. A pre-wrapper phase is responsible for performing advanced job configuration routines, whereas the wrapper script starts a virtual machine and triggers the application job on it.

KOALA: Mohamed and Epema (in press) have presented the design and implementation of KOALA, a Grid scheduler that supports resource co-allocation. KOALA Grid scheduler interacts with cluster batch schedulers for the execution of jobs. The work proposes an alternative to advance reservation at local resource managers, when reservation features are not available. This alternative allows processors to be allocated from multiple sites at the same time.

SNAP-Based Community Resource Broker: The Service Negotiation and Acquisition Protocol (SNAP)-based community resource broker uses an interesting three-phase commit protocol. SNAP is proposed because traditional advance reservation facilities cannot cope with the fact that information availability may change between the moment at which resource availability is queried and the time when the reservation of resources is actually performed (Haji, Gourlay, Djemame, & Dew, 2005). The three phases of SNAP protocol consist of (i) a step in which resource availability is queried and probers are deployed, which inform the broker in case the resource status changes; (ii) then, the resources are selected and reserved; and (iii) after that, the job is deployed on the reserved resources.

Platform Community Scheduler Framework (CSF): CSF (2003) provides a set of tools that can be utilised to create a Grid meta-scheduler or a community scheduler. The meta-scheduler enables users to define the protocols to interact with resource managers in a system independent manner. The interface with a resource manager is performed via a component termed Resource Manager (RM) Adapter. A RM Adapter interfaces a cluster resource manager. CSF supports the GRAM protocol to access the services of the resource managers that do not support the RM Adapter interface.

Platform's LSF and MultiCluster products leverage the CSF to provide a framework for implementing meta-scheduling. Grid Gateway is an interface that integrates Platform LSF and CSF. A scheduling plug-in for Platform LSF scheduler decides which LSF jobs are forwarded to the meta-scheduler. This decision is based on information obtained from an information service provided by the Grid Gateway. When a job is forwarded to the meta-scheduler, the job submission and monitoring tools dispatch the job and query its status information through the Grid Gateway. The Grid Gateway uses the job submission, monitoring and reservation services from the CSF. Platform MultiCluster also allows multiple clusters using LSF to forward jobs to one another transparently to the end-user.

Other important work: Kertész et al. (2008) introduced a meta-brokering system in which the meta-broker, invoked through a Web portal, submits jobs, monitors job status and copies output files using brokers from different Grid middleware, such as NorduGrid Broker and EGEE WMS.

Kim and Buyya (2007) tackle the problem of fair-share resource allocation in hierarchical VOs. They provide a model for hierarchical VO environments based on a resource sharing policy; and provide a heuristic solution for fair-share resource allocation in hierarchical VOs.

Inter-Operation of Resource Sharing Networks

Relevant work on the attempts to enable inter-operation between resource sharing networks is discussed in this section.

PlanetLab: PlanetLab (Peterson et al., 2006) is a large-scale testbed that enables the creation of slices, that is, distributed environments based on virtualisation technology. A slice is a set of virtual machines, each running on a unique node. The individual virtual machines that make up a slice contain no information about the other virtual machines in the set and are managed by the service running in the slice. Each service deployed on PlanetLab runs on a slice of PlanetLab's global pool of resources. Multiple slices can run concurrently and each slice is like a network container that isolates services from other containers.

The principals in PlanetLab are:

- **Owner:** organisation that hosts (owns) one or more PlanetLab nodes.
- **User:** researcher who deploys a service on a set of PlanetLab nodes.
- **PlanetLab Consortium (PLC):** centralised trusted intermediary that manages nodes on behalf of a group of owners and creates slices on those nodes on behalf of a group of users.

When PLC acts as a Slice Authority (SA), it maintains the state of the set of system-wide slices for which the PLC is responsible. The SA provides an interface through which users register themselves, create slices, bind users to slices, and request the slice to be instantiated on a set of nodes. PLC, acting as a Management Authority (MA), maintains a server that installs and updates the software running on the nodes it manages and monitors these nodes for correct behavior, taking appropriate action when anomalies and failures are detected. The MA maintains a database of registered nodes. Each node is affiliated with an organization (owner) and is located at a site belonging to the organization. MA provides an interface used by node owners to register their nodes with the PLC and allows users and slices authorities to obtain information about the set of nodes managed by the MA.

PlanetLab's architecture has evolved to enable decentralised control or federations of PlanetLabs (Peterson et al., 2006). The PLC has been split into two components namely the MA and SA, which allow PLC-like entities to evolve these two components independently. Therefore, autonomous organisations can federate and define peering relationships with each other. For example, peering relationships with other infrastructure is one of the goals of PlanetLab Europe (2008). A resource owner may choose a MA to which it wants to provide resources. MAs, in turn, may blacklist particular SAs. A SA may trust only certain MAs to provide it with the virtual machines it needs for its users. This enables various types of agreements between SAs and MAs.

It is also important to mention that Ricci et al. (2006) have discussed issues related to the design of a general resource allocation interface that is sufficiently wide for allocators in a large variety of current and future testbeds. An allocator is a component that receives as input the users' abstract description for the required resources and the resource status from a resource discoverer and produces allocations performed by a deployment service. The goal of an allocator is to allow users to specify characteristics of their slice in high-level terms and find resources to match these requirements. Authors have described their experience in designing PlanetLab and Emulab and among several important issues, they have advocated that:

- In future infrastructures, several allocators may co-exist and it might be difficult for them to co-exist without interfering into one another;
- With the current proportional-share philosophy of PlanetLab, where multiple management services can co-exist, allocators do not have guarantees over any resources;
- Thus, co-ordination between the allocators may be required.

Grid Interoperability Now - Community Group (GIN-CG): GIN-CG (2006) has been working on providing interoperability between Grids by developing components and adapters that enable secure and standard job submissions, data transfers, and information queries. These efforts provide the basis for load management across Grids by facilitating standard job submission and request redirection. They also enable secure access to resources and data across Grids. Although GIN-CG's efforts are relevant, its members also highlight the need for common allocation and brokering of resources across Grids.[2]

InterGrid: Assunção et al. (2008) have proposed an architecture and policies to enable the interoperation of Grids. This set of architecture and policies is termed as the InterGrid. InterGrid is inspired by the peering agreements between Internet Service Providers (ISPs). The Internet is composed of competing ISPs that agree to allow traffic into one another's networks. These agreements between ISPs are commonly termed as peering and transit arrangements (Metz, 2001).

In the InterGrid, a Resource Provider (RP) contributes a share of computational resources, storage resources, networks, application services or other type of resource to a Grid in return for regular payments. An RP has local users whose resource demands need to be satisfied, yet it delegates provisioning rights over spare resources to an InterGrid Gateway (IGG) by providing information about the resources available in the form of free time slots (Assunção & Buyya, 2008). A free time slot includes information about the number of resources available, their configuration and time frame over which they will be available. The control over resource shares offered by providers is performed via a container model, in which the resources are used to run virtual machines. Internally, each Grid may have a resource management system organised in a hierarchical manner. However, for the sake of simplicity, experimental results consider that RPs delegate provisioning rights directly to an IGG (Assunção & Buyya, in press).

A Grid has pre-defined peering arrangements with other Grids, managed by IGGs and, through which they co-ordinate the use of resources of the InterGrid. An IGG is aware of the terms of the peering with other Grids; provides Grid selection capabilities by selecting a suitable Grid able to provide the required resources; and replies to requests from other IGGs. The peering arrangement between two Grids is represented as a contract. Request redirection policies determine which peering Grid is selected to process a request and at what price the processing is performed (Assunção & Buyya, in press).

Other important work: Boghosian et al. (2006) have performed experiments using resources from more than one Grid for three projects, namely Nektar, SPICE and Vortonics. The applications in these three projects require massive numbers of computing resources only achievable through Grids of Grids. Although resources from multiple Grids were used during the experiments, they emphasised that several human interactions and negotiations are required in order to use federated resources. The authors highlighted that even if interoperability at the middleware level existed, it would not guarantee that the federated Grids can be utilised for large-scale distributed applications because there are important additional requirements such as compatible and consistent usage policies, automated advanced reservations and co-scheduling.

Caromel et al. (2007) have proposed the use of a P2P network to acquire resources dynamically from a Grid infrastructure (i.e. Grid'5000) and desktop machines in order to run compute intensive applica-

tions. The communication between the P2P network and Grid'5000 is performed through SSH tunnels. Moreover, the allocation of nodes for the P2P network uses the deployment framework of ProActive by deploying Java Virtual Machines on the allocated nodes.

In addition to GIN-CG's efforts, other Grid middleware interoperability approaches have been presented. Wang et al. (2007) have described a gateway approach to achieve interoperability between gLite (2005) (the middleware used in EGEE) and CNGrid GOS (2007) (the middleware of the Chinese National Grid (2007)). The work focuses on job management interoperability, but also describes interoperability between the different protocols used for data management as well as resource information. In the proposed interoperability approach, gLite is viewed as a type of site job manager by GOS, whereas the submission to GOS resources by gLite is implemented in a different manner; an extended job manager is instantiated for each job submitted to a GOS resource. The extended job manager sends the whole batch job to be executed in the CNGrid.

Virtual Organisations

We have also carried out a survey on how projects address different challenges in the VO life-cycle. Two main categories of projects have been identified: the facilitators for VOs, which provide means for building clusters of organisations hence enabling collaboration and formation of VOs; and enablers for VOs, which provide middleware and tools to help in the formation, management, maintenance and dissolution of VOs. The classification is not strict because a project can fall into two categories, providing software for enabling VOs and working as a consortium, which organisations can join and start collaborations that are more dynamic. We divide our survey into three parts: middleware and software infrastructure for enabling VOs; consortiums and charters that facilitate the formation of VOs; and other relevant work that addresses issues related to a VO's life-cycle.

Enabling Technology

Enabling a VO means to provide the required software tools to help in the different phases of the life-cycle of a VO. As we present in this section, due to the complex challenges in the life-cycle, many projects do not address all the phases. We discuss relevant work in this section.

The CONOISE Project: CONOISE (Patel et al., 2005) uses a marketplace (auctions) for the formation of VOs (Norman et al., 2004). The auctions are combinatorial; combinatorial auctions allow a good degree of flexibility so that VO initiators can specify a broad range of requirements. A combinatorial auction allows multiple units of a single item or multiple items to be sold simultaneously. However, combinatorial auctions lack on means for bid representation and efficient clearing algorithms to determine prices, quantities and winners. As demonstrated by Dang (2004), clearing combinatorial auctions is an NP-Complete problem. Thus, polynomial and sub-optimal auction clearing algorithms for combinatorial auctions have been proposed.

Stakeholders in VOs enabled by CONOISE are called agents. As example of VO formation, a user may request a service to an agent, who in turn verifies if it is able to provide the service requested at the time specified. If the agent cannot provide the service, it looks for the Service Providers (SPs) offering the service required. The Requesting Agent (RA) then starts a combinatorial auction and sends call for bids to SPs. Once RA receives the bids, it determines the best set of partners and then, starts the formation of the VO. Once the VO is formed, RA becomes the VO manager.

An agent that receives a call for bids has the following options: (a) she can decide not to bid for the auction; (b) she can bid considering its resources; (c) she may bid using resources from an existing collaboration; (d) she may identify the need to start a new VO to provide the extra resources required. Note that call for bids are recursive. CONOISE uses a cumulative scheduling based on a Constraint Satisfaction Program (CSP) to model the decision process of an agent.

CONOISE also focuses on the operation and maintenance phases of VOs. Once a VO is formed, it uses principles of coalition formation for distributing tasks amongst the member agents (Patel et al., 2005). An algorithm for coalition structure generation, which is bound from the optimal, is presented and evaluated (Dang, 2004). Although not very focused on authorisation issues, the CONOISE project also deals with issues regarding trust and reputation in VOs by providing reputation and policing mechanisms to ensure minimum quality of service.

The TrustCoM Project: TrustCoM (2005) addresses issues related to the establishment of trust throughout the life-cycle of VOs. Its members envision that the establishment of Service Oriented Architectures (SOAs) and the dynamic open electronic marketplaces will allow dynamic alliances and VOs among enterprises to respond quickly to market opportunities. The establishment of trust, not only at a resource level but also at a business process level, is hence of importance. In this light, TrustCoM aims to provide a framework for trust, security and contract management to enable on-demand and self-managed dynamic VOs (Dimitrakos, Golby, & Kearley, 2004; Svirskas, Arevas, Wilson, & Matthews, 2005).

The framework extends current VO membership services (Svirskas et al., 2005) by providing means to: (i) identify potential VO partners through reputation management; (ii) manage users according to the roles defined in the business process models that VO partners perform; (iii) define and manage the SLA obligations on security and privacy; (iv) enable the enforcement of policies based on the SLAs and contracts. From a corporate perspective, Sairamesh et al (2005) provide examples of business models on the enforcement of security policies and the VO management.

While the goal is to enable dynamic VOs, TrustCoM focuses on the security requirements for the establishment of VOs composed of enterprises. Studies and market analysis to identify the main issues and requirements to build a secure environment in which VOs form and operate have been performed.

Facilitators or Breeding Environments

In order to address the problem of trust between organisations, projects have created federations and consortiums which physical organisations or Grids can join to start VOs based on common interests. We describe the main projects in this field and explain some of the technologies they use.

Open Science Grid (OSG): OSG (2005) can be considered as a facilitator for VOs. The reason is that the project aims at forming a cluster or consortium of organisations and suggests them to follow a policy that states how collaboration takes place and how a VO is formed. To join the consortium and consequently form a VO, it is necessary to have a minimum infrastructure and preferably use the middleware suggested by OSG. In addition, OSG provides tools to check the status and monitor existing VOs. OSG facilitates the formation of VOs by providing an open-market-like infrastructure that allows the consortium members to advertise their resources and goals and establish VOs to explore their objectives. The VO concept is used in a recursive manner; VOs may be composed of sub-VOs. For more information we refer to the Blueprint for the OSG (2004).

A basic infrastructure must be provided to form a VO, including a VO Membership Service (VOMS) and operation support. The operation support's main goal is to provide technical support services at

the request of a member site. As OSG intends to federate across heterogeneous Grid environments, the resources of the member sites and users are organised in VOs under the contracts that result from negotiations among the sites, which in turn have to follow the consortium's policies. Such contracts are defined at the middleware layer and can be negotiated in an automated fashion; however, thus far there is no easily responsive means to form a VO and the formation requires complex multilateral agreements among the involved sites.

OSG middleware uses VOMS to support authorisation services for VO members hence helping in the maintenance and operation phases. Additionally, for the sake of scalability and easiness of administration, Grid User Management System (GUMS) facilitates the mapping of Grid credentials to site-specific credentials. GUMS and VOMS provide means to facilitate the authorisation in the operation and maintenance phases. GridCat provides maps and statistics on jobs running and storage capacity of the member sites. This information can guide schedulers and brokers on job submission and in turn facilitate the operation phase. Additionally, MonALISA (MONitoring Agents using a Large Integrated Services Architecture) (Legrand et al., 2004) has been utilised to monitor computational nodes, applications and network performance of the VOs within the consortium.

Enabling Grids for E-sciencE (EGEE): Similarly to OSG, EGEE (2005) federates resource centres to enable a global infrastructure for researchers. EGEE's resource centres are hierarchically organised: an Operations Manager Centre (OMC) located at CERN, Regional Operations Centres (ROC) located in different countries, Core Infrastructures Centres (CIC) and Resource Centres (RC) responsible for providing resources to the Grid. A ROC carries out activities as supporting deployment and operations; negotiating SLAs within its region and organising certification authorities. CICs are in charge of providing VO-services, such as maintaining VO-Servers and registration; VO-specific services such as databases, resource brokers and user interfaces; and other activities such as accounting and resource usage. The OMC interfaces with international Grid efforts. It is also responsible for activities such as approving connection with new RCs, promoting cross-trust among CAs, and enabling cooperation and agreements with user communities, VOs and existing national and regional infrastructures.

To join EGEE, in addition to the installation of the Grid middleware, there is a need for a formal request and further assessment from special committees. Once the application is considered suitable to EGEE, a VO will be formed. Accounting is based on the use of resources by members of the VO. EGEE currently utilises LCG-2/gLite (2005).

Other Important Work

Resource allocation in a VO depends on, and is driven by, many conditions and rules: the VO can be formed by physical organisations under different, sometimes conflicting, resource usage policies. Participating organisations provide their resources to the VO, which can be defined in terms of SLAs, and agree to enforce VO level policies defining who has access to the resources in the VO. Different models can be adopted for negotiation and enforcement of SLAs. One model is by relying on a trusted VO manager. Resource providers supply resources to the VO according to SLAs established with the VO manager. The VO manager in turn assigns resource quotas to VO groups and users based on a commonly agreed VO-level policy. In contrast, a VO can follow a democratic or P2P sharing approach, in which "you give what you can and get what others can offer" or "you get what give" (Wasson & Humphrey, 2003).

Elmroth and Gardfj¨all (2005) presented an approach for enabling Grid-wide fair-share scheduling. The work introduces a scheduling framework that enforces fair-share policies in a Grid-wide scale. The

policies are hierarchical in the sense that they can be subdivided recursively to form a tree of shares. Although the policies are hierarchical, they are enforced in a flat and decentralised manner. In the proposed framework, resources have local policies and split the available resources to given VOs. These local policies have references to the VO-level policies. Although the proposed framework and algorithm do not require a centralised scheduler, it may impose certain overhead in locally caching global usage information.

MAPPING OF SURVEYED WORK AGAINST THE TAXONOMIES

This section presents the mapping of the surveyed projects against the proposed taxonomies. For simplicity, the mapping only considers selected work from those surveyed to be included in the tables presented in this section.

Table 1 classifies existing work according to their architectures and operational models. Gridbus Broker, GridWay, and SNAP-based community resource broker are resource brokers that act on behalf of users to submit jobs to Grid resources to which they have access. They follow the operational model based on job routing. Although GridWay provides means for the deployment of virtual machines, this deployment takes place on a job basis (Rubio-Montero et al., 2007). DI-GRUBER, VioCluster, Condor flocking and CSF have a distributed-scheduler architecture in which brokers or meta-schedulers have bilateral sharing agreements between them (Table 2). OurGrid and Self-organising flock of Condors utilise P2P networks of brokers or schedulers, whereas Grid federation uses a P2P network to build a shared space utilised by providers and users to post resource claims and requests respectively (Table 2). VioCluster and Shirako enable the creation of virtualised environments in which job routing or job pulling based systems can be deployed. However, in these last two systems, resources are controlled at the level of containment or virtual machines.

Table 2 summarises the communication models and sharing mechanisms utilised by distributed-scheduler based systems. Shirako uses transitive agreements in which brokers can exchange claims of resources issued by site authorities who represent the resource providers. It allows brokers to delegate access to resources multiple times.

The resource control techniques employed by the surveyed systems are summarised in Table 3. As described beforehand, VioCluster and Shirako use containment based resource control, whereas the remaining systems utilise the job model. EGEE WMS and DI-GRUBER take into account the scheduling of jobs according to the VOs to which users belong and the shares contributed by resource providers. The other systems can be utilised to form a single VO wherein jobs can be controlled on a user basis.

The support of various works to the VO life-cycle phases is depicted in Table 4. We select a subset of the surveyed work, particularly the work that focuses on VO related issues such as their formation and operation. DI-GRUBER and gLite schedule jobs by considering the resource shares of multiple VOs. EGEE and OSG also work as facilitators of VOs by providing consortiums to which organisations can join and start VOs (Table 5). However, the process is not automated and requires the establishment of contracts between the consortium and the physical resource providers. Shirako enables the creation of virtualised environments spanning multiple providers, which can be used for hosting multiple VOs (Ramakrishnan et al., 2006).

The systems characteristics and the VOs they enable are summarised in Table 5. Conoise and Akogrimo allow the formation of dynamic VOs in which the VO can be started by a user utilising a mobile

Table 1. GRMSs according to architectures and operational models

System	Architecture	Operational Model
SGE and PBS	Independent clusters	Job routing
Condor-G	Independent clusters*	Job routing
Gridbus Broker	Resource Broker	Job routing
GridWay	Resource Broker	Job routing**
SNAP-Based Community Resource Broker	Resource Broker	Job routing
EGEE WMS	Centralised	Job routing
KOALA	Centralised	Job routing
PlanetLab	Centralised	N/A***
Computing Center Software (CCS)	Hierarchical	Job routing
GRUBER/DI-GRUBER	Distributed/static	Job routing
VioCluster	Distributed/static	N/A***
Condor flocking	Distributed/static	Matchmaking
Community Scheduler Framework	Distributed/static	Job routing
OurGrid	Distributed/dynamic	Job routing
Self-organising flock of Condors	Distributed/dynamic	Matchmaking
Grid federation	Distributed/dynamic	Job routing
Askalon	Distributed/dynamic	Job routing
SHARP/Shirako	Distributed/dynamic	N/A***
Delegated Matchmaking	Hybrid	Matchmaking

*	Condor-G provides software that can be used to build meta-schedulers.
**	GridWay also manages the deployment of virtual machines.
***	PlanetLab, VioCluster and Shirako use resource control at the containment level, even though they enable the creation of virtual execution environments on which systems based on job routing can be deployed.

Table 2. Classification of GRMSs according to their sharing arrangements

System	Communication Pattern	Sharing Mechanism
GRUBER/DI-GRUBER	Bilateral agreements	System centric
VioCluster	Bilateral agreements	Site centric
Condor flocking	Bilateral agreements	Site centric
OurGrid	P2P network	System centric
Self-organising flock of Condors	P2P network	Site centric
Grid federation	Shared space	Site centric
Askalon	Bilateral agreements	Site centric
SHARP/Shirako	Transitive agreements	Self-interest
Delegated MatchMaking	Bilateral agreements	Site centric

Table 3. Classification of GRMSs according to their support for VOs and resource control

System	Support for VOs	Resource Control
EGEE WMS	Multiple VO	Job model
KOALA	Single VO	Job model
GRUBER/DI-GRUBER	Multiple VO	Job model
VioCluster	Single VO	Container model/multiple site*
Condor flocking	Single VO	Job model
OurGrid	Single VO	Job model
Self-organising flock of Condors	Single VO	Job model
Grid federation	Single VO	Job model
Askalon	Single VO	Job model
SHARP/Shirako	Multiple VO**	Container model/multiple site***
Delegated MatchMaking	Single VO	Job model

*	VioCluster supports containment at both single site and multiple site levels.
**	Shirako enables the creation of multiple containers that can in turn be used by multiple VOs, even though it does not handle issues on job scheduling amongst multiple VOs.
***	Shirako supports containment at both (i) single site level through Cluster on Demand and (ii) multiple-site level. Shirako also explores resource control at job level by providing recommendations on the site in which jobs should be executed.

Table 4. Support to the phases of the VO's lifecycle by the projects analysed

Project Name	Support to the phases of the VO life-cycle				Support for short term collaborations
	Creation	Operation	Maintenance	Dissolution	
OSG*	Partial	Partial	Not available	Not available	Not available
EGEE/gLite*	Partial	Available	Not available	Not available	Not available
CONOISE	Available	Available	Available	Not available	Available
TrustCoM	Mainly related to security issues	Mainly related to security issues	Not available	Not available	Not available
DI-GRUBER	Not available	Available	Partial**	Not available	Not available
Akogrimo***	Partial	Partial	Partial	Partial	Partial
Shirako	Not available	Available	Available	Not available	Not available

*	OSG and EGEE work as consortiums enabling trust among organisations and facilitating the formation of VOs. They also provide tools for monitoring status of resources and job submissions. EGEE's WMS performs the scheduling taking into account multiple VOs.
**	DI-GRUPER's policy decision points allow for the re-adjustment of the VOs according to the current resource shares offered by providers and the status of the Grid.
***	Akogrimo aims at enabling collaboration between doctors upon the patient's request or in case of a health emergency.

Table 5. Mapping of the systems against the propsed VO taxonomies

System	Dynamism	Goal Orientation	Duration	Control	Policy Enforcement	Facilitators
Conoise*	Dynamic/Hybrid	Targeted	Medium-lived	Decentralised	Democratic	N/A
TrustCoM**	Static	Targeted	Long-lived	N/A	N/A	N/A
GRUBER/DI-GRUBER	Static	Targeted	Long-lived	Decentralised	Decentralised***	N/A
gLite/EGEE	Static	Targeted	Long-lived	Centralised	Centralised	Centralised+
Open Science Grid	Static	Targeted	Long-lived	Hierarchical	Centralised	Market-like
Akogrimo	Dynamic/Hybrid	Targeted	Short or Medium-lived	Decentralised	Democratic	N/A
Shirako	Dynamic	Non-targeted	Medium-lived	Decentralised	Democratic	N/A

*	Conoise and Akogrimo allow a client using a mobile device to start a VO, thus the VO can comprise fixed and mobile resources.
**	TrustCoM deals with security issues and does not provide tools for the management and policy enforcement in VOs.
***	DI-GRUBER uses a network of decision points to guide submitting hosts and schedulers about which resources can execute the jobs.
+	EGEE Workload Management System is aware of the VOs and schedules jobs accordingly to the VOs in the system.

device. The virtual environments enabled by Shirako can be adapted by leasing additional resources or terminating leases according to the demands of the virtual organisation it is hosting (Ramakrishnan et al., 2006). Resource providers in Shirako may offer their resources in return for economic compensation meaning that the resource providers may not have a common target in solving a particular resource challenge. This makes the VOs non-targeted.

FUTURE TRENDS

Over the last decade, the distributed computing realm has been characterised by the deployment of large-scale Grids such as EGEE and TeraGrid. Such Grids have provided the research community with an unprecedented number of resources, which have been used for various scientific research. However, the hardware and software heterogeneity of the resources provided by the organisations within a Grid have increased the complexity of deploying applications in these environments. Recently, application deployment has been facilitated by the intensifying use of virtualisation technologies.

The increasing ubiquity of virtual machine technologies has enabled the creation of customised environments atop a physical infrastructure and the emergence of new business models such as virtualised data centres and cloud computing. The use of virtual machines brings several benefits such as: server consolidation, the ability to create VMs to run legacy code without interfering in other applications' APIs, improved security through the creation of application sandboxes, dynamic provisioning of virtual machines to services, and performance isolation.

Existing virtual-machine based resource management systems can manage a cluster of computers within a site allowing the creation of virtual workspaces (Keahey et al., 2006) or virtual clusters (Foster et al., 2006; Montero et al., 2008; Chase et al., 2003). They can bind resources to virtual clusters or workspaces according to a customer's demand. These resource managers allow the user to create customised virtual clusters using shares of the physical machines available at the site. In addition, current data centres are using virtualisation technology to provide users with the look and feel of taping into a dedicated computing and storage infrastructure for which they are charged a fee based on usage (e.g. Amazon Elastic Computing Cloud[3] and 3Tera[4]).

These factors are resulting in the creation of virtual execution environments or slices that span both commercial and academic computing sites. Virtualisation technologies minimise many of the concerns that previously prevented the peering of resource sharing networks, such as the execution of unknown applications and the lack of guarantees over resource control. For the resource provider, substantial work is being carried out on the provisioning of resources to services and user applications. Techniques such as workload forecasts along with resource overbooking can reduce the need for over-provisioning a computing infrastructure. Users can benefit from the improved reliability, the performance isolation, and the environment isolation offered by virtualisation technologies.

We are likely to see an increase in the number of virtual organisations enabled by virtual machines, thus allocating resources from both commercial data centres and research testbeds. We suggest that emerging applications will require the prompt formation of VOs, which are also quickly responsive and automated. VOs can have dynamic resource demands, which are quickly responded by data centres relying on virtualisation technologies. There can also be an increase in business workflows relying on globally available messaging based systems for process synchronisation[5]. Our current research focuses on connecting computing sites managed by virtualisation technologies for creating distributed virtual environments which are used by the user applications.

CONCLUSION

This chapter presents classifications and a survey of systems that can provide means for inter-operating resource sharing networks. It also provides taxonomies on Virtual Organisations (VOs) with a focus on Grid computing practices. Hence, we initially discussed the challenges in VOs and presented a background on the life-cycle of VOs and on resource sharing networks. This chapter suggests that future applications will require the prompt formation of VOs, which are also quickly responsive and automated. This may be enabled by virtualisation technology and corroborates the current trends on multiple site containers or virtual workspaces. Relevant work and technology in the area were presented and discussed.

ACKNOWLEDGMENT

We thank Marco Netto, Alexandre di Costanzo and Chee Shin Yeo for sharing their thoughts on the topic and helping in improving the structure of this chapter. We are grateful to Mukaddim Pathan for proof reading a preliminary version of this chapter. This work is supported by research grants from the Australian Research Council (ARC) and Australian Department of Innovation, Industry, Science and Research (DIISR). Marcos' PhD research is partially supported by NICTA.

REFERENCES

A Blueprint for the Open Science Grids. (2004, December). Snapshot v0.9.

Adabala, S., Chadha, V., Chawla, P., Figueiredo, R., Fortes, J., & Krsul, I. (2005, June). From virtualized resources to virtual computing Grids: the In-VIGO system. *Future Generation Computer Systems*, *21*(6), 896–909. doi:10.1016/j.future.2003.12.021

Andrade, N., Brasileiro, F., Cirne, W., & Mowbray, M. (2007). Automatic Grid assembly by promoting collaboration in peer-to-peer Grids. *Journal of Parallel and Distributed Computing*, *67*(8), 957–966. doi:10.1016/j.jpdc.2007.04.011

Andrade, N., Cirne, W., Brasileiro, F., & Roisenberg, P. (2003). OurGrid: An approach to easily assemble Grids with equitable resource sharing. In *9th Workshop on Job Scheduling Strategies for Parallel Processing* (Vol. 2862, pp. 61–86). Berlin/Heidelberg: Springer.

Australian Partnership for Advanced Computing (APAC) Grid. (2005). Retrieved from http://www.apac.edu.au/programs/GRID/index.html.

Balazinska, M., Balakrishnan, H., & Stonebraker, M. (2004, March). Contract-based load management in federated distributed systems. In *1st Symposium on Networked Systems Design and Implementation (NSDI)* (pp. 197-210). San Francisco: USENIX Association.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., et al. (2003). Xen and the art of virtualization. In *19th ACM Symposium on Operating Systems Principles (SOSP '03)* (pp. 164–177). New York: ACM Press.

Boghosian, B., Coveney, P., Dong, S., Finn, L., Jha, S., Karniadakis, G. E., et al. (2006, June). Nektar, SPICE and vortonics: Using federated Grids for large scale scientific applications. In *IEEE Workshop on Challenges of Large Applications in Distributed Environments (CLADE)*. Paris: IEEE Computing Society.

Brune, M., Gehring, J., Keller, A., & Reinefeld, A. (1999). Managing clusters of geographically distributed high-performance computers. *Concurrency (Chichester, England)*, *11*(15), 887–911. doi:10.1002/(SICI)1096-9128(19991225)11:15<887::AID-CPE459>3.0.CO;2-J

Bulhões, P. T., Byun, C., Castrapel, R., & Hassaine, O. (2004, May). *N1 Grid Engine 6 Features and Capabilities* [White Paper]. Phoenix, AZ: Sun Microsystems.

Butt, A. R., Zhang, R., & Hu, Y. C. (2003). A self-organizing flock of condors. In *2003 ACM/IEEE Conference on Supercomputing (SC 2003)* (p. 42). Washington, DC: IEEE Computer Society.

Buyya, R., Abramson, D., & Giddy, J. (2000, June). An economy driven resource management architecture for global computational power grids. In *7th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000)*. Las Vegas, AZ: CSREA Press.

Caromel, D., di Costanzo, A., & Mathieu, C. (2007). Peer-to-peer for computational Grids: Mixing clusters and desktop machines. *Parallel Computing*, *33*(4–5), 275–288. doi:10.1016/j.parco.2007.02.011

Catlett, C., Beckman, P., Skow, D., & Foster, I. (2006, May). Creating and operating national-scale cyberinfrastructure services. *Cyberinfrastructure Technology Watch Quarterly*, *2*(2), 2–10.

Chase, J. S., Irwin, D. E., Grit, L. E., Moore, J. D., & Sprenkle, S. E. (2003). Dynamic virtual clusters in a Grid site manager. In *12th IEEE International Symposium on High Performance Distributed Computing (HPDC 2003)* (p. 90). Washington, DC: IEEE Computer Society.

Chinese National Grid (CNGrid) Project Web Site. (2007). Retrieved from http://www.cngrid.org/

CNGrid GOS Project Web site. *(2007).* Retrieved from http://vega.ict.ac.cn

Dang, V. D. (2004). *Coalition Formation and Operation in Virtual Organisations.* PhD thesis, Faculty of Engineering, Science and Mathematics, School of Electronics and Computer Science, University of Southampton, Southampton, UK.

de Assunção, M. D., & Buyya, R. (2008, December). Performance analysis of multiple site resource provisioning: Effects of the precision of availability information [Technical Report]. In *International Conference on High Performance Computing (HiPC 2008)* (Vol. 5374, pp. 157–168). Berlin/Heidelberg: Springer.

de Assunção, M. D., & Buyya, R. (in press). Performance analysis of allocation policies for interGrid resource provisioning. *Information and Software Technology*.

de Assunção, M. D., Buyya, R., & Venugopal, S. (2008, June). InterGrid: A case for internetworking islands of Grids. [CCPE]. *Concurrency and Computation*, *20*(8), 997–1024. doi:10.1002/cpe.1249

Dimitrakos, T., Golby, D., & Kearley, P. (2004, October). Towards a trust and contract management framework for dynamic virtual organisations. In *eChallenges*. Vienna, Austria.

Dixon, C., Bragin, T., Krishnamurthy, A., & Anderson, T. (2006, September). Tit-for-Tat Distributed Resource Allocation [Poster]. *The ACM SIGCOMM 2006 Conference*.

Dumitrescu, C., & Foster, I. (2004). Usage policy-based CPU sharing in virtual organizations. In *5th IEEE/ACM International Workshop on Grid Computing (Grid 2004)* (pp. 53–60). Washington, DC: IEEE Computer Society.

Dumitrescu, C., & Foster, I. (2005, August). GRUBER: A Grid resource usage SLA broker. In J. C. Cunha & P. D. Medeiros (Eds.), *Euro-Par 2005* (Vol. 3648, pp. 465–474). Berlin/Heidelberg: Springer.

Dumitrescu, C., Raicu, I., & Foster, I. (2005). DI-GRUBER: A distributed approach to Grid resource brokering. In *2005 ACM/IEEE Conference on Supercomputing (SC 2005)* (p. 38). Washington, DC: IEEE Computer Society.

Dumitrescu, C., Wilde, M., & Foster, I. (2005, June). A model for usage policy-based resource allocation in Grids. In *6th IEEE International Workshop on Policies for Distributed Systems and Networks* (pp. 191–200). Washington, DC: IEEE Computer Society.

Elmroth, E., & Gardfjäll, P. (2005, December). Design and evaluation of a decentralized system for Grid-wide fairshare scheduling. In *1st IEEE International Conference on e-Science and Grid Computing* (pp. 221–229). Melbourne, Australia: IEEE Computer Society Press.

Enabling Grids for E-sciencE (EGEE) project. (2005). Retrieved from http://public.eu-egee.org.

Epema, D. H. J., Livny, M., van Dantzig, R., Evers, X., & Pruyne, J. (1996). A worldwide flock of condors: Load sharing among workstation clusters. *Future Generation Computer Systems*, *12*(1), 53–65. doi:10.1016/0167-739X(95)00035-Q

Fontán, J., Vázquez, T., Gonzalez, L., Montero, R. S., & Llorente, I. M. (2008, May). OpenNEbula: The open source virtual machine manager for cluster computing. In *Open Source Grid and Cluster Software Conference – Book of Abstracts*. San Francisco.

Foster, I., Freeman, T., Keahey, K., Scheftner, D., Sotomayor, B., & Zhang, X. (2006, May). Virtual clusters for Grid communities. In *6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006)* (pp. 513–520). Washington, DC: IEEE Computer Society.

Foster, I., & Kesselman, C. (1997, Summer). Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications*, *11*(2), 115–128.

Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *The International Journal of Supercomputer Applications*, *15*(3), 200–222.

Frey, J., Tannenbaum, T., Livny, M., Foster, I. T., & Tuecke, S. (2001, August). Condor-G: A computation management agent for multi-institutional Grids. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC 2001)* (pp. 55–63). San Francisco: IEEE Computer Society.

Fu, Y., Chase, J., Chun, B., Schwab, S., & Vahdat, A. (2003). SHARP: An architecture for secure resource peering. In *19th ACM Symposium on Operating Systems Principles (SOSP 2003)* (pp. 133–148). New York: ACM Press.

gLite - Lightweight Middleware for Grid Computing. (2005). Retrieved from http://glite.web.cern.ch/glite.

Graupner, S., Kotov, V., Andrzejak, A., & Trinks, H. (2002, August). *Control Architecture for Service Grids in a Federation of Utility Data Centers* (Technical Report No. HPL-2002-235). Palo Alto, CA: HP Laboratories Palo Alto.

Grid Interoperability Now Community Group (GIN-CG). (2006). Retrieved from http://forge.ogf.org/sf/projects/gin.

Grimme, C., Lepping, J., & Papaspyrou, A. (2008, April). Prospects of collaboration between compute providers by means of job interchange. In *Job Scheduling Strategies for Parallel Processing* (Vol. 4942, p. 132-151). Berlin / Heidelberg: Springer.

Grit, L. E. (2005, October). *Broker Architectures for Service-Oriented Systems* [Technical Report]. Durham, NC: Department of Computer Science, Duke University.

Grit, L. E. (2007). *Extensible Resource Management for Networked Virtual Computing*. PhD thesis, Department of Computer Science, Duke University, Durham, NC. (Adviser: Jeffrey S. Chase)

Haji, M. H., Gourlay, I., Djemame, K., & Dew, P. M. (2005). A SNAP-based community resource broker using a three-phase commit protocol: A performance study. *The Computer Journal*, *48*(3), 333–346. doi:10.1093/comjnl/bxh088

Hey, T., & Trefethen, A. E. (2002). The UK e-science core programme and the Grid. *Future Generation Computer Systems*, *18*(8), 1017–1031. doi:10.1016/S0167-739X(02)00082-1

Huang, R., Casanova, H., & Chien, A. A. (2006, April). Using virtual Grids to simplify application scheduling. In *20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*. Rhodes Island, Greece: IEEE.

Huedo, E., Montero, R. S., & Llorente, I. M. (2004). A framework for adaptive execution in Grids. *Software, Practice & Experience*, *34*(7), 631–651. doi:10.1002/spe.584

Iosup, A., Epema, D. H. J., Tannenbaum, T., Farrellee, M., & Livny, M. (2007, November). Inter-operating Grids through delegated matchmaking. In *2007 ACM/IEEE Conference on Supercomputing (SC 2007)* (pp. 1–12). New York: ACM Press.

Irwin, D., Chase, J., Grit, L., Yumerefendi, A., Becker, D., & Yocum, K. G. (2006, June). Sharing networked resources with brokered leases. In *USENIX Annual Technical Conference* (pp. 199–212). Berkeley, CA: USENIX Association.

Katzy, B., Zhang, C., & Löh, H. (2005). Virtual organizations: Systems and practices. In L. M. Camarinha-Matos, H. Afsarmanesh, & M. Ollus (Eds.), (p. 45-58). New York: Springer Science+Business Media, Inc.

Keahey, K., Foster, I., Freeman, T., & Zhang, X. (2006). Virtual workspaces: Achieving quality of service and quality of life in the Grids. *Science Progress*, *13*(4), 265–275.

Kertész, A., Farkas, Z., Kacsuk, P., & Kiss, T. (2008, April). Grid enabled remote instrumentation. In F. Davoli, N. Meyer, R. Pugliese, & S. Zappatore (Eds.), *2nd International Workshop on Distributed Cooperative Laboratories: Instrumenting the Grid (INGRID 2007)* (pp. 303–312). New York: Springer US.

Kim, K. H., & Buyya, R. (2007, September). Fair resource sharing in hierarchical virtual organizations for global Grids. In *8th IEEE/ACM International Conference on Grid Computing (Grid 2007)* (pp. 50–57). Austin, TX: IEEE.

Legrand, I., Newman, H., Voicu, R., Cirstoiu, C., Grigoras, C., Toarta, M., et al. (2004, September-October). Monalisa: An agent based, dynamic service system to monitor, control and optimize Grid based applications. In *Computing in High Energy and Nuclear Physics (CHEP)*, Interlaken, Switzerland.

Litzkow, M. J., Livny, M., & Mutka, M. W. (1988, June). Condor – a hunter of idle workstations. In *8th International Conference of Distributed Computing Systems* (pp. 104–111). San Jose, CA: Computer Society.

Metz, C. (2001). Interconnecting ISP networks. *IEEE Internet Computing*, *5*(2), 74–80. doi:10.1109/4236.914650

Mohamed, H., & Epema, D. (in press). KOALA: A co-allocating Grid scheduler. *Concurrency and Computation*.

Montero, R. S., Huedo, E., & Llorente, I. M. (2008, September/October). Dynamic deployment of custom execution environments in Grids. In *2nd International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP '08)* (pp. 33–38). Valencia, Spain: IEEE Computer Society.

National e-Science Centre. (2005). Retrieved from http://www.nesc.ac.uk.

Norman, T. J., Preece, A., Chalmers, S., Jennings, N. R., Luck, M., & Dang, V. D. (2004). Agent-based formation of virtual organisations. *Knowledge-Based Systems*, *17*, 103–111. doi:10.1016/j.knosys.2004.03.005

Open Science Grid. (2005). Retrieved from http://www.opensciencegrid.org

Open Source Metascheduling for Virtual Organizations with the Community Scheduler Framework (CSF) (Tech. Rep.) (2003, August). Ontario, Canada: Platform Computing.

OpenPBS. *The portable batch system software.* (2005). Veridian Systems, Inc., Mountain View, CA. Retrieved from http://www.openpbs.org/scheduler.html

Padala, P., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., et al. (2007, March). Adaptive control of virtualized resources in utility computing environments. In *2007 Conference on EuroSys (EuroSys 2007)* (pp. 289-302). Lisbon, Portugal: ACM Press.

Patel, J., Teacy, L. W. T., Jennings, N. R., Luck, M., Chalmers, S., & Oren, N. (2005). Agent-based virtual organisations for the Grids. *International Journal of Multi-Agent and Grid Systems*, *1*(4), 237–249.

Peterson, L., Muir, S., Roscoe, T., & Klingaman, A. (2006, May). *PlanetLab Architecture: An Overview* (Tech. Rep. No. PDN-06-031). Princeton, NJ: PlanetLab Consortium.

PlanetLab Europe. (2008). Retrieved from http://www.planet-lab.eu/.

Ramakrishnan, L., Irwin, D., Grit, L., Yumerefendi, A., Iamnitchi, A., & Chase, J. (2006). Toward a doctrine of containment: Grid hosting with adaptive resource control. In *2006 ACM/IEEE Conference on Supercomputing (SC 2006)* (p. 101). New York: ACM Press.

Ranjan, R., Buyya, R., & Harwood, A. (2005, September). A case for cooperative and incentive-based coupling of distributed clusters. In *7th IEEE International Conference on Cluster Computing*. Boston, MA: IEEE CS Press.

Ranjan, R., Harwood, A., & Buyya, R. (2006, September). SLA-based coordinated superscheduling scheme for computational Grids. In *IEEE International Conference on Cluster Computing (Cluster 2006)* (pp. 1–8). Barcelona, Spain: IEEE.

Ranjan, R., Rahman, M., & Buyya, R. (2008, May). A decentralized and cooperative workflow scheduling algorithm. In *8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2008)*. Lyon, France: IEEE Computer Society.

Ricci, R., Oppenheimer, D., Lepreau, J., & Vahdat, A. (2006, January). Lessons from resource allocators for large-scale multiuser testbeds. *SIGOPS Operating Systems Review*, *40*(1), 25–32. doi:10.1145/1113361.1113369

Rubio-Montero, A., Huedo, E., Montero, R., & Llorente, I. (2007, March). Management of virtual machines on globus Grids using GridWay. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007)* (pp. 1–7). Long Beach, USA: IEEE Computer Society.

Ruth, P., Jiang, X., Xu, D., & Goasguen, S. (2005, May). Virtual distributed environments in a shared infrastructure. *IEEE Computer*, *38*(5), 63–69.

Ruth, P., McGachey, P., & Xu, D. (2005, September). VioCluster: Virtualization for dynamic computational domain. In *IEEE International on Cluster Computing (Cluster 2005)* (pp. 1–10). Burlington, MA: IEEE.

Ruth, P., Rhee, J., Xu, D., Kennell, R., & Goasguen, S. (2006, June). Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. In *3rd IEEE International Conference on Autonomic Computing (ICAC 2006)* (pp. 5-14). Dublin, Ireland: IEEE.

Sairamesh, J., Stanbridge, P., Ausio, J., Keser, C., & Karabulut, Y. (2005, March). *Business Models for Virtual Organization Management and Interoperability* (Deliverable A - WP8&15 WP - Business & Economic Models No. V.1.5). Deliverable document 01945 prepared for TrustCom and the European Commission.

Schwiegelshohn, U., & Yahyapour, R. (1999). Resource allocation and scheduling in metasystems. In *7th International Conference on High-Performance Computing and Networking (HPCN Europe '99)* (pp. 851–860). London, UK: Springer-Verlag.

Shoykhet, A., Lange, J., & Dinda, P. (2004, July). *Virtuoso: A System For Virtual Machine Marketplaces* [Technical Report No. NWU-CS-04-39]. Evanston/Chicago: Electrical Engineering and Computer Science Department, Northwestern University.

Siddiqui, M., Villazón, A., & Fahringer, T. (2006). Grid capacity planning with negotiation-based advance reservation for optimized QoS. In *2006 ACM/IEEE Conference on Supercomputing (SC 2006)* (pp. 21–21). New York: ACM.

Smarr, L., & Catlett, C. E. (1992, June). Metacomputing. *Communications of the ACM*, *35*(6), 44–52. doi:10.1145/129888.129890

Svirskas, A., Arevas, A., Wilson, M., & Matthews, B. (2005, October). Secure and trusted virtual organization management. *ERCIM News* (63).

The TrustCoM Project. (2005). Retrieved from http://www.eu-trustcom.com.

Vázquez-Poletti, J. L., Huedo, E., Montero, R. S., & Llorente, I. M. (2007). A comparison between two grid scheduling philosophies: EGEE WMS and Grid Way. *Multiagent and Grid Systems*, *3*(4), 429–439.

Venugopal, S., Nadiminti, K., Gibbins, H., & Buyya, R. (2008). Designing a resource broker for heterogeneous Grids. *Software, Practice & Experience*, *38*(8), 793–825. doi:10.1002/spe.849

Wang, Y., Scardaci, D., Yan, B., & Huang, Y. (2007). Interconnect EGEE and CNGRID e-infrastructures through interoperability between gLite and GOS middlewares. In *International Grid Interoperability and Interoperation Workshop (IGIIW 2007) with e-Science 2007* (pp. 553–560). Bangalore, India: IEEE Computer Society.

Wasson, G., & Humphrey, M. (2003). Policy and enforcement in virtual organizations. In *4th International Workshop on Grid Computing* (pp. 125–132). Washington, DC: IEEE Computer Society.

Wesner, S., Dimitrakos, T., & Jeffrey, K. (2004, October). Akogrimo - the Grid goes mobile. *ERCIM News,* (59), 32-33.

ENDNOTES

1 http://www.vmware.com/
2 The personal communication amongst GIN-CG members is online at: http://www.ogf.org/pipermail/gin-ops/2007-July/000142.html
3 http://aws.amazon.com/ec2/
4 http://www.3tera.com/
5 http://aws.amazon.com/sqs/