SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2009; **39**:1419–1438
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.942

Ontology-based Grid resource management

Balachandar R. Amarnath[1,*,†], Thamarai Selvi Somasundaram[1], Mahendran Ellappan[1] and Rajkumar Buyya[2]

[1]*CARE, Department of Information Technology, Madras Institute of Technology, Anna University Chennai, Chromepet, Chennai 600 044, India*
[2]*Cloud Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, VIC 3010, Australia*

SUMMARY

Grid resources are typically diverse in nature with respect to their software and hardware configurations, resource usage policies and the kind of application they support. Aggregating and monitoring these resources, and discovering suitable resources for the applications become a challenging issue. This is partially due to the representation of Grid metadata supported by the existing Grid middleware which offers limited scope for matching the job requirements that directly affect scheduling decisions. This paper proposes a semantic component in conventional Grid architecture to support ontology-based representation of Grid metadata and facilitate context-based information retrieval that complements Grid schedulers for effective resource management. Web Ontology language is used for creating Grid resource ontology and Algernon inference engine has been used for resource discovery. This semantic component has been integrated with conventional Grid schedulers. Several experiments have also been carried out to investigate the performance overhead that arises while integrating this component with Grid schedulers.
Copyright © 2009 John Wiley & Sons, Ltd.

Received 9 June 2009; Revised 28 August 2009; Accepted 2 September 2009

KEY WORDS: Grid computing; semantic Grid; ontology; Grid resource; management

1. INTRODUCTION

The emergence of scientific and industrial applications requiring large amount of computational power needs seamless access to geographically distributed computational resources. This forces the respective institutions and organizations to harness idle computers on an international scale.

*Correspondence to: Balachandar R. Amarnath, CARE, Department of Information Technology, Madras Institute of Technology, Anna University Chennai, Chromepet, Chennai 600 044, India.
†E-mail: balachandar.ra@gmail.com

Contract/grant sponsor: Ministry of Communication and Information Technology, Government of India
Contract/grant sponsor: Australian Department of Innovation, Industry, Science and Research

Copyright © 2009 John Wiley & Sons, Ltd.

However, there was no technology to support flexible and controlled sharing of various types of resources that are needed to solve computationally intensive applications. To address this issue, an extended distributed computing technology, termed as 'Grid', was proposed which facilitates the aggregation of distributed computational resources that spans beyond organizational boundaries, and their coordinated utilization to meet the requirements of advanced science and engineering applications [1–3]. Consequently, Grid middleware has been proposed, that performs basic authentication and authorization of the participants of the Grid and governs execution of job and monitors the dynamic state of the participating resources. However, due to the diverse nature of participating Grid resources, Grid middlewares are limited in functionality with respect to coordinating and managing the resources for application execution. Further, Grid applications often require computational resources with different software and hardware configurations. Hence, discovering suitable resources that match the application requirements are really a difficult task. In such an environment, the role of Grid schedulers is very important as they often need to manage various resources information and their status during application scheduling. For instance, as soon as a job arrives in the job queue, the Grid scheduler aggregates Grid resource information and performs matchmaking to find out suitable Grid resource information. A sophisticated Grid scheduling mechanism also considers resource usage policies before scheduling application to the resources, and establishes service level agreement to ensure the proper delivery of the requested Quality of Service (QoS) parameters. However, the representation of these metadata supports the keyword-based discovery of resources that often miss relatively capable resources. For instance, an application requests Linux Operating System for execution. If this resource is not available, conventional Grid schedulers put the application in pending state due to non-availability of suitable resource. However, in most cases, this application can also be executed in resource that has Fedora or other Unix-based Operating Systems. Unfortunately, currently existing Grid schedulers do not possess the capability to infer the relationship between the two operating systems. This is due to the fact that the underlying Grid middleware such as Globus [4], gLite [5] and Unicore [6] defines and implements mechanisms for resource discovery and monitoring, which supports traditional service matching based on symmetric, attribute-based matching and does not support semantic descriptions of Grid resources.

Meanwhile, semantic web technology offers greater support in describing a particular domain that makes it possible to infer semantic relationship between two entities. Exploiting such technologies in Grid resource management provides greater flexibility in making scheduling decisions such as resource discovery for application execution, policy negotiation and establishing Service Level Agreements (SLA) between the users and Grid resource providers. This initiative has led to the emergence of Semantic Grid technology. The Semantic Grid is an extension of the current Grid in which information and services are given well-defined meaning through machine understandable descriptions which helps in making intelligent scheduling decisions with a high degree of automation in computational Grid infrastructure [7]. Metadata in Grid includes available Grid resource information, job execution status and information related to resource usage policies. Representing such information with the help of semantic web technologies would support context-based information retrieval that can complement Grid schedulers while making scheduling decisions such as resource allocation and policy management. The objective of semantic Grid's approach is to make information 'understandable' by computers. Information must therefore be described in such a way that computers can interpret it and derive its meaning. The concept of ontology helps to describe the information by expressing the relationship between them. The semantic representation of information is exploited while retrieving information using an ontology reasoning language. Hence,

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

these technologies enable computers to work more intelligently with the information in assisting humans in classifying, filtering and discovering information. Integrating such semantic component in Grid architecture requires clear understanding of the Grid middleware and the mechanism it uses for resource aggregation across the Grid infrastructure, and the type of information that the Grid scheduler requires during resource management.

In this paper, a four-layered conceptual Grid architecture is proposed with knowledge layer incorporated in the high-level middleware layer. The semantic component implemented in the knowledge layer supports ontological representation of Grid metadata. It also implements information retrieval component that matches user's application requirements against the available Grid resources represented in ontological form and takes appropriate scheduling decisions. The requirements are matched against the capability of existing Grid resources based on the semantic relationship between them. It uses web ontology language (OWL) [8] for representing ontology of Grid resources through protégé ontology editor. A rule-based inference engine called Algernon [9] is used for information retrieval from the ontology knowledge base. This component has been tested with Grid scheduler such as Gridbus Broker and Moab [10]. Several experiments have been carried out to analyze the performance of the semantic component with respect to information retrieval, performance of Algernon inference engine against its counterparts, and overhead arises in ontology-based matchmaking mechanisms against database and the one implemented in the popular 'Gridway' metascheduler [11]. The contributions of this paper are summarized below:

- Ontology description module that dynamically aggregates Grid resource information from the underlying Grid resources describes them onto the ontology template and thus creating Grid resources knowledge base.
- Ontology-based resource discovery module that matches the requirements of applications against the available Grid resources based on the 'context' of the request instead of keyword-based discovery mechanism. Further, an approach is proposed that stores new keywords present in the request in the database that allows the discovery module to learn them.
- Integration of the semantic component with Grid schedulers such as Gridbus broker, and Moab metascheduler used in Garuda, a national Grid computing initiative of India, and
- Comparative study of effective information retrieval using several inference engines.
- The overhead involved in implementing semantic component in Grid infrastructure is compared against other standard approaches.

The remainder of the paper is organized as follows: Section 2 highlights few important research works related to our proposal. Section 3 describes background concepts such as ontology, semantic web and other related technologies exploited in our work. Section 4 presents the proposed architecture and their high-level description. The creation of ontology and knowledge base has been explained in Section 5. The integration of the proposed semantic component with Gridbus broker and with Garuda infrastructure is explained in Section 6. Section 7 discusses the experiments carried out to analyze the overhead and performance issues arising due to this semantic component. Section 8 concludes the paper by highlighting the features and further scope of the proposed work.

2. RELATED WORK

Several researches have been carried out in this field and here we discuss in brief some of the works closely related to our work.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

2.1. Matchmaking system in grid

Gridway [11] is a Grid metascheduler built over Globus middleware. It supports job scheduling across Globus-based Grid resources, aggregates resource information and performs keyword-based matchmaking to discover suitable resources that match application requirements.

The Monitoring and Discovery System (MDS) implemented in Globus middleware [12] is designed to provide a standard interface for publishing and discovering Grid resources status and configuration information in GLUE schema as well as its own schema called MDS schema. The representation of the information aggregated by MDS is used by Grid schedulers and it supports keyword-based matchmaking of resource requirements against the available resources.

Condor [13] is a cluster manager that supports job submission across several clusters. The Workload Management System (WMS) implemented in gLite middleware uses condor for matchmaking user's application requirements against the available resources in the Grid. Condor specifies a standard for describing jobs, workstations and other resources. However, with this representation, it is not possible to understand the semantic relationship between the available resource information and the requested ones.

Gridbus broker [14] is a resource broker designed to support scheduling of both computational and data Grid applications. However, the resource discovery module implemented in the Gridbus broker does not support semantic description and discovery of Grid resources, and it uses the Globus Grid Index Information Services (GIIS) [15] or Grid Market Directory (GMD) [16] to gather Grid resource information.

In all the above middlewares and metaschedulers, the matchmaking is carried out based on the keywords present in the request. The work presented in this paper can bypass the conventional matchmaking system and performs semantic-based resource discovery. It can also be easily integrated with Grid metaschedulers and we describe two such integrations in the following sections.

2.2. Semantic grid projects

A reference architecture that extends OGSA to support the explicit handling of semantics is proposed in [17]. It defines the associated knowledge services to support a spectrum of service capabilities, and a model, the capabilities and the mechanisms for the semantic Grid. It extends the capabilities of Grid middleware to include semantic provisioning services and semantically aware Grid services. An extensive survey on semantic web and semantic Grid technologies is presented in [18]. It explores the potential and predicted impact of emerging technologies on collaborative industrial design, and outlines a reference architecture that can be implemented to meet the specific requirements of industrial business.

The U.K. myGrid [19] project uses ontologies to describe and select web-based services used in the Life Sciences; the U.K. Geodise project uses ontologies to guide aeronautical engineers to select and conFigure Matlab scripts [20]. Mirza Pahlevi Said and Isao Kojima in [21] proposed Semantic Monitoring and Discovery System (S-MDS) built on the top of WS-Resource Framework. Here, they extended MDS4 of Globus middleware to support RDF/OWL descriptions and accept query from the SPARQL inference engine. The authors provided an extension to S-MDS in [22] by incorporating the rule-based approach in conjunction with inference capability. However, the S-MDS approach is tightly coupled with the underlying Globus middleware, and the authors did not provide information about support for other middlewares.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

An ontology-based Matchmaker Service proposed in [23] supports dynamic resource discovery and resource descriptions. However, the request is expressed using request ontology and hence there is a need to compile the user request as ontology descriptions. Pernas *et al.* in [24] use ontology for describing Grid resources thereby enabling semantic discovery. They define the concept-classification tree to create resource ontology which uses protégé axioms to discover resources semantically. Similarly, the literature [25] exploits semantic web technology for resource matching in the Grid. This literature also utilizes background ontology and matchmaking rules to perform matchmaking of the resources. Zhang *et al.* in [26] proposed a semantic Grid infrastructure for e-government applications. They argue the necessity of such infrastructure for the management of e-government resources in the form of services across virtual government agencies. We differ from this work with respect to the domain considered. Yin *et al.* in [27] presented an agent-based semantic Grid for collaboration across virtual enterprises by forming a super peer network over the semantic Grid. Michael Hartung *et al.* in [28] provided a platform and a metamodel that allows the user to create and edit Grid related metadata present across the Grid infrastructure. They have also implemented this platform in German D-Grid. However, our focus is to devise a semantic system to complement Grid scheduling decisions and hence must be adaptable to integrate with Grid schedulers. Further, it requires a different approach to represent and manage distributed computational Grid resources.

Although in these works, several approaches have been proposed for semantic-based Grid resource management, an extensive analysis in devising a mechanism for resource representation, selection of reasoning engine and integration of such component with Grid scheduler is still missing.

The semantic component proposed in our earlier work [29] supports context-based Grid resource discovery and is incorporated with Gridbus Broker. In this paper, we propose a generic architecture that supports ontology-based representation and discovery of computational Grid resources. The semantic component has been integrated with the existing Grid schedulers. Further, the performance of the component has been improved by coupling the technique of updating the new *keywords* used for request in the database. The selection of Algernon inference engine has been justified by comparing the performances with other popular inference engines. In addition, the performance of this semantic component against other conventional approaches has also been discussed.

3. FUNDAMENTALS OF SEMANTIC GRID

The emergence of Semantic Web concepts offers sophisticated tools and mechanism to describe a particular domain of interest, and querying and rule-based languages for information retrieval. They provide a new dimension for describing web content for efficient information retrieval and solve interoperability issues in collaborative computing. Semantic web, proposed by Berners Lee *et al.* in [30], is a collection of information described in a hierarchical manner so that they are easily understandable by computers thereby bringing human and computer, a step closer. Such explicit specification of conceptualization is called Ontology. Ontologies facilitate the interchange and integration of heterogeneous information. They are used to capture knowledge about some domain of interest. Ontology describes the concepts in the domain and also the relationships that hold between those concepts a.k.a. classes. It also allows to model the real-time entities of a domain as *instances* of classes. The ontology together with instances forms *knowledge base* of that domain. Semantic web uses Resource Description Framework (RDF) [31] over XML-based representation

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

of data as a standard data interchange format to create ontologies. Unlike traditional XML, RDF schemas allows to provide a semantic relationship between the two concepts such as 'Cluster is a *type of* computingResource' where 'Cluster' and 'computingResource' are referred to as 'classes' and '*a type of*' is referred to as '*property*'. With the wide adoption of RDF by the semantic web communities, RDF continued to grow. The most recent development in standard ontology language is Web Ontology Language (OWL) from the World Wide Web Consortium. OWL is developed as a vocabulary extension to RDF and is capable of representing several relationships such as disjoint and cardinality. *Classes* and *Properties* described using ontology languages are identified as URIs, and can be easily accessible through internet which enables flexible sharing and reusing them.

To facilitate the creation of ontologies, several editors were developed. Protégé [32] is one such editor that supports the creation of ontology using OWL language and is widely used for developing ontology-based applications. An ontology description of a particular domain supports *semantic*- or *context*-based information retrieval especially in Artificial Intelligence (AI) applications where semantic relationships between concepts are established using description logic. There are inference engines such as Algernon that provides versatile DL-based querying mechanism for knowledge retrieval from ontology knowledge base. Further, using rule languages like SWRL [33], allows even more sophisticated information retrieval mechanisms. Such developments in semantic web technologies tempted Grid researchers to integrate them with Grid environment to address, especially, Grid resource and data management. This initiative has led to the notion of 'Semantic Grid' that suggests *machine processable descriptions of particular domain to maximize the potential for sharing and reuse, and thus better enabling computers and people to work in cooperation* [34]. In computational Grid environment, users and software agents should be able to discover, invoke, compose and monitor Grid nodes offering the required services and possessing particular properties. Describing these resources and services using the concepts of semantic web technology supports understanding the meaning of the request. This would help the Grid scheduler to identify computational resources that are not exactly matching with the application requirements but can still execute the application. For instance, an application has been compiled for Linux operating system and submits a request to the Grid scheduler for such resources in the Grid. If such resources are not available, the Grid scheduler can suggest resources with Fedora operating system for executing this application. This is for the simplest case of scheduling decision. In real environments, there would be a large amount of metadata that need to be processed for making versatile scheduling decisions. In such situation, the conventional description of metadata supported by current middleware components offers limited flexibility with regard to knowledge retrieval. For instance, a Globus 2.4 middleware-based Grid aggregates Grid resource information and stores in volatile *LDAP* server. The description of resource is found to be an 'object' and corresponding attribute. In the advanced versions of Globus such as Globus 4.x, information is stored as XML documents. High-level services such as Resource management services and Resource brokering services use such conventional description of resources and make appropriate scheduling decisions such as resource discovery, verifying resource usage policies and application scheduling to selected resources. They compare the characteristics requested by the applications with the available resource configuration based on the keyword present in the request. Hence, such discovery offers limited flexibility and restricted amount of information retrieval. However, if the Grid metadata are described using the concepts of ontology, the Grid scheduler can employ semantic-based information retrieval mechanism using the appropriate inference engine. In this paper, the scope has been limited to semantic-based resource discovery which allows us to provide more details on the implementation and experimental results.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

4. ARCHITECTURE

The semantic Grid architecture interacts with the underlying Grid resources and creates its ontology representation to support semantic-based information retrieval. Further, it should not impose any special requirements to be fulfilled by the participating Grid resources. In addition, semantic component must be able to incorporate with different Grid middlewares. To meet the above requirements, the implementation of semantic component is not tightly coupled with the underlying Grid middleware. It uses the standard way of interacting with the Grid middleware and automatically creates ontology description of Grid resources.

A four-layered semantic Grid architecture is modeled with the knowledge layer sitting at the top of the Grid scheduler in the high-level Grid middleware layer as shown in Figure 1.

Fabric layer: The Grid Fabric layer provides the resources to which shared access is mediated by Grid protocols. The resources may be computational resources, storage systems, catalogues, network resources and sensors or may be a logical entity, such as a distributed file system, computer cluster or distributed computer pool.

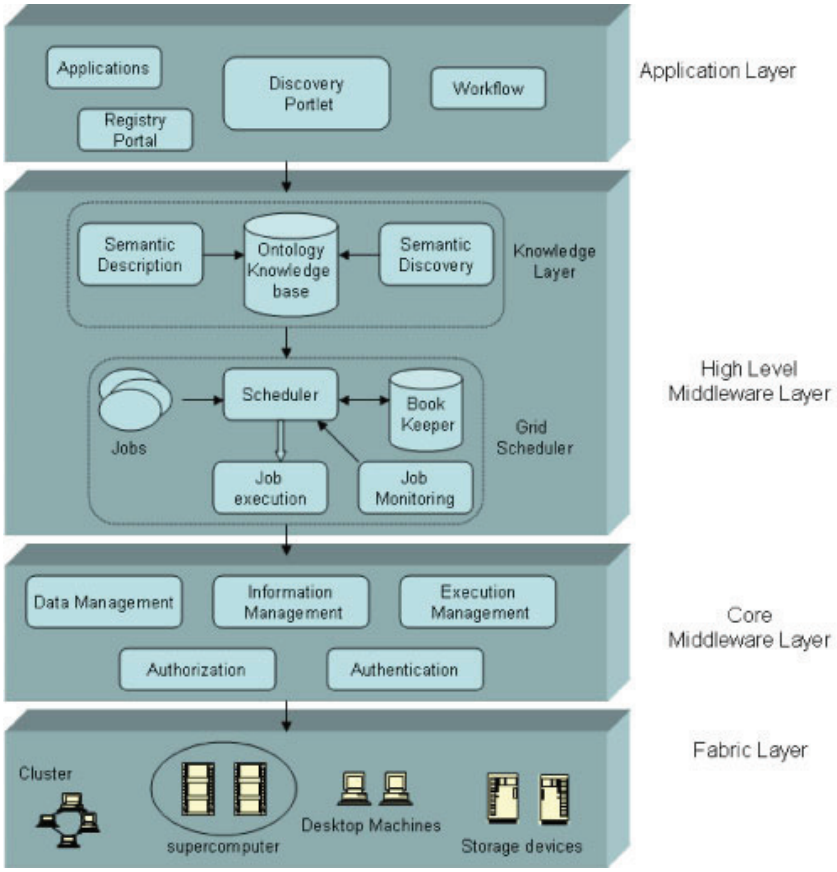

Figure 1. Semantic grid architecture.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Core middleware layer: This layer consists of low-level middleware that provides secure and unified access to remote resources. Depending on the type of resources, we can choose different middlewares such as Globus, Unicore, Alchemi and SRB. Using the services of such low-level middleware layer, one can create high-level middleware services that support rapid creation and deployment of applications on global Grids.

High-level middleware layer: In our architecture, this layer is proposed to be implemented using a Grid scheduler. It discovers suitable resources that match the user requirements and schedules job to that selected resource. Further, the scheduler monitors the execution of the job in the resource and aggregates the results of the execution. It obtains the services provided by the knowledge layer for discovery of suitable resource. The knowledge layer provides knowledge discovery from the data aggregated from the information services of the underlying middleware layer. Moreover, this layer is domain oriented and usually uses domain knowledge built with domain ontology. In the proposed architecture, the knowledge layer contains necessary functions and services for creating ontology description of the available Grid resource information. It also provides programming and user interfaces that enable semantic-based discovery of suitable Grid resources using the ontology description maintained in this layer. The knowledge layer is modeled as a separate layer and is not tightly coupled with the underlying middleware layer. In this paper, it has been implemented in such a way that, it creates ontology description of Grid resources present in a text file or from a database. In addition, it can directly query the Globus-based Grid resources to aggregate the resource information and creates ontology description.

Application layer: The application layer enables the use of resources in a Grid environment through various collaboration and resource access protocols. The discovery portal present in this layer allows the resource requester to submit resource requirements to find out suitable resources for application execution. It also includes software and tools to support application workflow and composition.

5. KNOWLEDGE LAYER

The components proposed in the knowledge layer can be divided into two types based on their functionalities as resource description and discovery component as shown in Figure 2. The resource description module supports representation of Grid metadata whereas discovery module discovers suitable resources for application execution. Both the modules implemented in the knowledge layer use semantic web's approach of making information understandable by computers.

Further, based on Karlshruhe ontology model [35], we define the following relations that can exist between the resources viz.,

- Subsumption relation, $\leq_c : (C \times C) U (R \times R) \rightarrow \{\text{true}, \text{false}\}$ for example '$\leq c(g,h) = \text{true}$' means: '$g$ is a subtype of h'.
- Conformity relation, which relates each individual instance (I) in I to one, and one only concept type in C.
- conf: $I \rightarrow C$ where C is concept instantiation and R is relation instantiation.

The semantic component exploits the abovementioned properties in creating ontology of Grid resource and suitable resource discovery.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

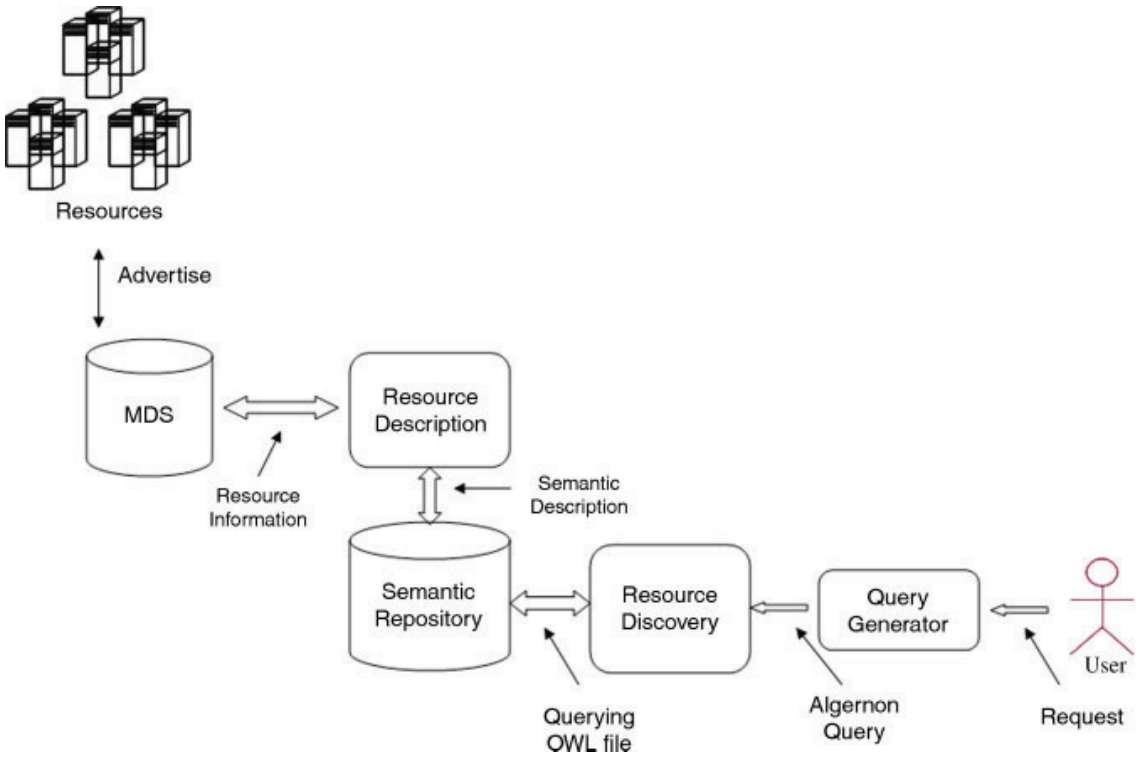

Figure 2. Semantic component.

5.1. Resource description using ontology template

Creating ontology is inevitably a very laborious process, and there is a need to at least partially automate the process of ontology creation and knowledge extraction. Hence, we can imagine a predefined ontology of concepts and relationships, plus a knowledge base of instances [36]. To realize this, the resource description module defines *resource ontology template* created using Protégé editor and it provides necessary concepts and properties with which a resource can be described. Various possible computing resources are considered for creating the ontology template.

The resource description module defines resource ontology template created using Protégé editor and provides necessary concepts and properties with which a resource can be described. Different possible computing resources are considered for creating ontology template. Our structuring of the ontology of resources is motivated by the need to provide semantic information about a resource and also to provide transparent access to Grid resources. We propose the following precise definitions to explain the motivation behind the creation of ontology template and how it can be used for semantic description.

Definition 1. An ontology template is a domain specific ontology that provides hierarchy of concepts along with properties to define their characteristics.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Definition 2. Any resource can be modeled as an instance of a specific concept provided that the resource can be described using the properties defined in that concept.

Once the ontology template is created, knowledge base can be built with the instances and the specific property instantiations. Together the ontology and the knowledge base make up a semantic repository. When a resource is registered in the Grid, its information can be obtained using a Grid resource monitoring tool such as MDS of Globus Middleware. With this information, an instance of the appropriate resource concept in the ontology template is created for every computing resource in the Grid. The characteristics of the resource for ex, freeRAMSpace, are also defined in the respective property of the appropriate resource concept in the ontology template. For example, an existence of a computer with Linux OS can be represented in the ontology template by creating an instance for the concept 'computer' and the 'hasOS' property of the concept 'computer' will be assigned the value 'Linux'.

Protégé-OWL APIs are used to dynamically create instances of a particular concept and also to assign values to appropriate properties in the resource ontology template. With these features, the resource information of the entire Grid environment can be described semantically which in turn enables semantic discovery of Grid resources.

5.2. Semantics-based resource discovery

Grid schedulers obtain resource requirement from the users for application execution, perform matchmaking of available resources against the requested ones, and discover suitable resources that match the requirements. In Grid, every participating resource advertises its capabilities in terms of *properties* such as Number of CPUs, Memory, storage capacity, CPU Load, etc., through a Grid scheduler. Matchmaking refers to capability matching of the requested Grid resources with the advertised ones. Grid scheduler implements a mechanism to support expression of application requirements that include nature of task, type of computing resources it requires, their configuration in terms of *properties* mentioned above in addition to other parameters such as budget, deadline and at what time the resource is actually needed. Some schedulers such as *condor* also allow to specify the resource requirements through mathematical expression (such as $<$, $>$, $<=$, $>=$, and $==$). For instance, a sample job submission request supported by condor is given below:

```
 Executable       = /bin/ls
Requirements      = Memory > = 128 &&
                    OpSys == 'LINUX_2.4' &&
                    Arch == 'i686'
Rank                = Memory >= 256
Error              = ls.err.$(Process)
Output              = ls.out.$(Process)
```

The matchmaking algorithm evaluates the requirements present in this request and finds out the suitable resources available in the Grid by matching the *keywords* present in the request. Such keyword-based matchmaking mechanism retrieves the resource that exactly matches the requirements. However, there are situations when potential resources are available, which can possibly execute the application but still *miss* from the search. Keyword-based matchmaking mechanism

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

is not able to find such resources and hence no suggestion can be made to the user about these resources. This is due to the fact that the algorithm does not understand the semantic relationship between the keywords. However, a semantic relationship can be established by the ontology representation of Grid resources and their characteristics and modifying matchmaking algorithm to exploit the same during resource discovery phase.

The resource discovery module proposed in this paper implements an ontology-based matchmaking mechanism that determines the semantic relationship between the request and the advertised resource information and hence determines closely related Grid resources when exact match fails. The algorithm classifies resources into three broad categories, viz *exact*, resources that exactly match with the requested resource requirements, *subsume*, if the advertised resources have more capabilities than that requested, *plugin*, an exact contrary to the previous case, that is, the application expects more capabilities than that is advertised, and *disjoint*, which actually is not a match but infers that both the request and the available resources are completely different [37].

In the Grid context, it is possible to consider the resources that fall into first two categories for application execution while the *plugin* resources cannot execute the application. This can be very well explained with the following scenario. Let us consider that an application requires Linux 2.4 version of Operating system and five numbers of CPUs for execution. An instance of Linux 2.4 or Linux 2.6 operating system with more than five CPUs falls into subsume categories whereas an instance of Linux 2.2 operating system with less than five numbers of CPUs would fall into *plugin* category. In this case, the *plugin* resource cannot execute the application as it expects more capabilities than that are available.

To determine the semantic relationship, the matchmaking system has to interact with the ontology knowledge base and retrieve information from it. Inference engines are widely used for this purpose and in this paper, the Algernon inference engine, developed by Stanford University is used for information retrieval from the knowledge base. It offers versatile queries that can be executed onto the knowledge base. Algernon is a rule-based inference engine that allows specifying rules and executing query according to the rules. The discovery module based on the user's request generates appropriate Algernon-based query and executes it into knowledge base. If the exactly matching resources are not available, it then determines the resources that exhibit subsumption relation with the request. This kind of inference is possible if the requested concept is modeled as a subconcept of the advertised concept in the ontology template, then it is possible to conclude that there exists a subsume match between the advertised and requested capabilities. For instance, while constructing computational resource ontology, a resource with Linux operating system would be modeled as a subconcept of the Unix-based resource. In such cases, it can be concluded that a request of resource with Unix operating system subsumes an advertisement of a resource with Linux operating system.

The resources identified from this module can be suggested to the user or Grid scheduler for making scheduling decisions and to execute application in it.

6. DESIGN AND IMPLEMENTATION

The proposed semantic component is implemented and tested in the Grid Computing Laboratory of Anna University. The necessary software including Globus Toolkit 4.0 and protégé was successfully installed in all local machines. Various components of Globus were successfully configured

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

and tested for proper operation. In addition, the MDS component has been tested properly for aggregating resource information about the local machines. Further, one of the local machines are designated as 'submission node' and configured so as to aggregate resource information about all the local machines. The semantic component is installed in this machine which includes the semantic description and discovery module. Ontology template is also maintained in this machine. The process of creating ontology template and the implementation aspect of discovery mechanism is explained in detail in the following subsections.

6.1. Creation of knowledge base

Ontology template has been created by considering different computing resources in the Grid. The *concept* of these resources has been defined properly using relations and *properties* so that the characteristics of any resource can be defined by its properties. The resource description module accesses the Grid nodes and retrieves resource information by executing *ldap* queries on those nodes and updates into the ontology template. For every type of information retrieved from the Grid node, the module creates an instance of the appropriate concept in the ontology template forming *conformity* relation between instances and their respective concept types. Further, an instance will be exactly related to only one type of concept. In addition, the values of various properties retrieved from MDS are assigned to the respective *properties* of the appropriate concepts in the ontology template.

At this point, the ontology template with concepts and properties and the corresponding instances and property values together constitute the knowledge base of the Grid resources. To create such knowledge base, protégé-OWL libraries are used. The libraries contain necessary APIs to create instances for a concept, assign values to properties and other ontology operations. This semantic description of resources facilitates the use of inference engines to interact with the knowledge base and retrieves information semantically. Moreover, the description module is made to execute periodically so that addition and removal of resources is updated in the knowledge base dynamically. A portion of the ontology template is shown in Figure 3.

6.2. Resource discovery mechanism

The discovery module relies on the power of the Algernon inference engine. A query tags with the format *label:label_value* is considered in which the properties of the resource are denoted as *label* and the requested value as *label_value*. It may also include operators in query label for flexible querying. For instance, if the user wants to search for machines with free RAM greater than 200 MB, the query should be RAM: >200. Currently, the system supports >, <, = and also NOT operators. In addition, the query mechanism allows querying a resource with multiple constraints. For example, if the user wants to query a machine with free RAM of 200 MB and free Hard disk space of 10 000 MB, then the query *'freeRAM:200 freeHDD:10 000'* will retrieve all resources with 200 MB and hard disk space with 10 000 MB. The Query generator module parses the user query using regular expression, stores left tag and right tag in a vector and converts it into suitable Algernon query. For example, the query 'freeRAM:200' will be converted into the following machine understandable query.

((instance RAM ?inst)(hasFreeMB ?inst ?val)(:TEST(:LISP(=?val"+rightTag+"))) (presentIn-Computer?inst ?instanceComputer)).

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

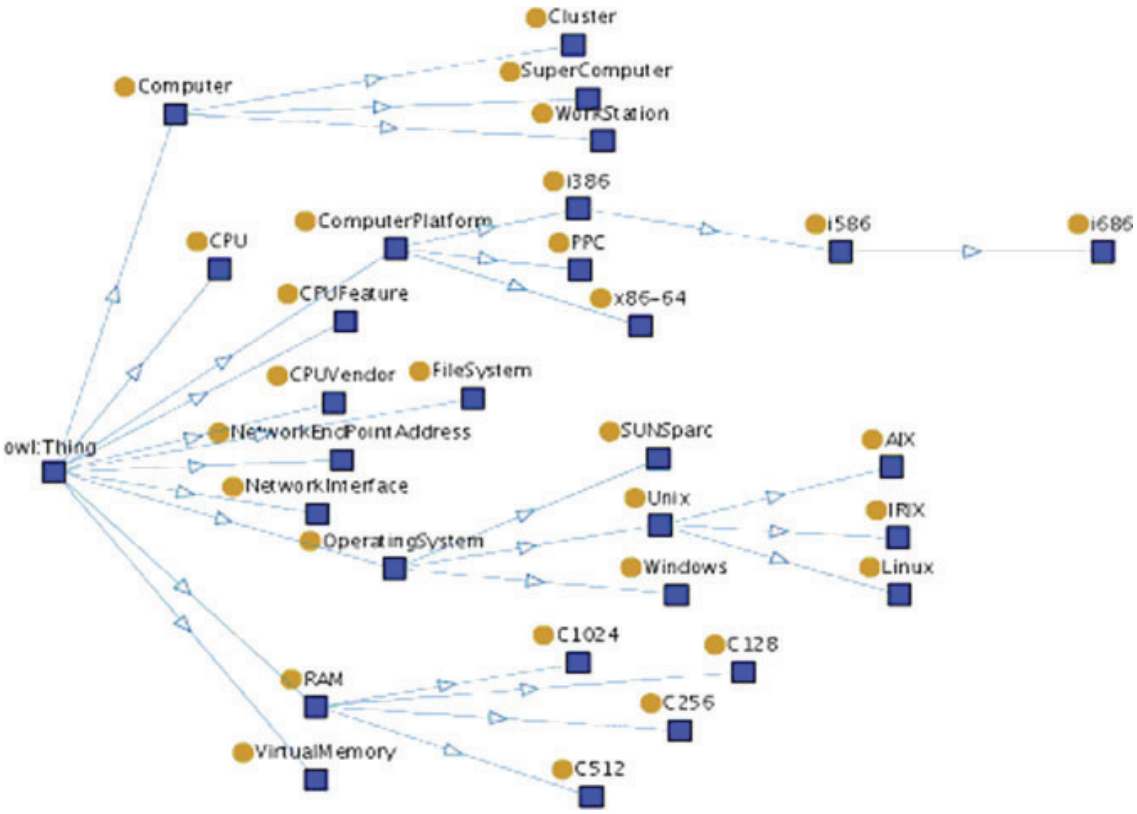

Figure 3. Portion of Ontology template.

The discovery module executes the query over the knowledge base of the Grid and obtains the resource that is matching with the user's request. It not only discovers the resource that exactly matches with that of the request, but also retrieves resources that exhibit subsumption relation when the exact match is not found. For example, if the user requests a computer with IRIX Operating system, and if the knowledge base does not possess instances of IRIX-based machine, the discovery module retrieves all Unix-based resources. This is because, the Algernon reasoner infers from the ontology template that IRIX is a subconcept of Unix that is,

$$\text{`}\leq(\text{Unix}, \text{IRIX}) = \text{true'}$$

as shown in Figure 3.

Hence, the discovery module retrieves instances of Unix as it is compatible with the IRIX Operating System.

6.3. Integration with gridbus broker

The Gridbus broker has been developed by the University of Melbourne as part of the Gridbus Project. It follows a service-oriented architecture and is designed on object-oriented principles with

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

a focus on the idea of promoting simplicity, modularity, reusability, extensibility and flexibility. The broker has been designed to operate with different Grid middleware frameworks and toolkits such as Globus that primarily runs on Unix-class machines and Alchemi, which is a .NET-based Grid computing platform for Microsoft Windows enabled computers. Recently it has been extended to work with Cloud Computing technologies such as Aneka and Amazon EC2. Hence it is possible to create a cross-platform Grid implementation using the Gridbus broker. However, it does not possess its own resource monitoring and discovery mechanism and relies on external information services such as GIIS or GMD as a resource repository. Hence, the semantic component is integrated with the broker in discovering suitable resources based on the semantics of the request for executing user's jobs.

6.3.1. Job descriptor

The broker requires two files, namely the Application description file and a resource description file. In application description file, the user's job will be described which includes the location of the required input files and executable, and also the output file. In resource description file, the identity of the resource in which the user's job should be executed is mentioned. It also contains the information about the middleware such as the name of the middleware and the version. This information is needed by the broker to invoke the appropriate middleware interface for proper execution of the job submitted by the user. The user submits his requirements to the semantic component which in turn discovers suitable resource, and sends the hostname of the resource to the job descriptor. Meanwhile, the user's job and the command needed to execute the job are submitted to the descriptor. A special component called *job descriptor* is implemented which creates application description and resource description files which will be submitted to the broker to initiate scheduling of jobs (see Figure 4). Once the execution is over, the results will be collected and presented to the user.

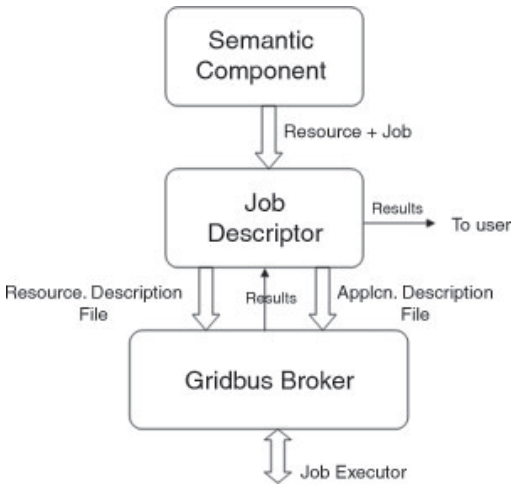

Figure 4. Semantic component with Gridbus Broker.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

6.4. Integration with Garuda

To develop the Indian eScience infrastructure and to provide Indian researchers, the seamless access of supercomputer-level processing power and knowledge resources, the Department of Information Technology (DIT), Government of India has funded its Centre for Development of Advanced Computing (CDAC) to deploy nationwide computational Grid termed as 'Garuda'. It connects and aggregates high end computational resources from 45 research laboratories spanning over 17 cities across the country [38]. In such an environment, there exists heterogeneity in the nature of resources and difference in usage policies.

In Garuda, the hardware and software configuration of all the computational resources are stored in a text file. Suitable resources are selected manually from the job submission portal. Further, Garuda can submit jobs to the selected resources in two ways, that is, either to the Moab metascheduler, or, directly to 'GRAM' component of the Globus middleware installed in the Grid resources. When the number of resources participating in the Grid is more, selecting them manually becomes highly difficult and hence a resource discovery mechanism was highly in demand.

The semantic component was modified for integration with Garuda, as the later needed only the components to construct ontology representation of Grid resources and an interface to discover the resources that matches the user's requirements.

First of all, the ontology of Garuda Grid resources was created using protégé editor. It establishes a semantic relationship between all types of computational resources in the Garuda Grid. One of the main requirements of Garuda applications are that an application compiled for lower release of the operating system can also be executed in higher releases. With this notion, the ontology template is constructed so that the higher releases were modeled as a subconcept of the lower releases.

The resource information present in the text file contains the operating system, release, location, hostname and processor Type. The semantic description module parses the resource file and extracts the resource information. These information are read sequentially and modeled as instances and appropriate slots in the ontology template. This creates knowledge base that supports the semantic retrieval of information.

The discovery module provides a user interface to submit resource request in terms of operating system, and two other variables referring to the release and the search level. The search level can be one of two values 0 and 1 that refers to the kind of search the user wants to perform while discovering suitable resource. The search level 0 refers to the direct match, that is, the resource that exactly matches with the user's requests and are to be discovered. The search level 1 retrieves resources that match closely with the requested, called Compatible match. In this case, the discovery module retrieves the resources information that matches exactly with the request as well as the ones that closely match with the request, that is, those that exhibit subsumption relations with the request. The GUI used for submitting resource request is shown in Figure 5.

7. EXPERIMENTS

Several experiments have been carried out with the semantic component to determine the performance with respect to the following aspects:

- The amount of resources retrieved for a request,
- Effect of maintaining a database to store new tags

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

- Effect of inference engine in resource discovery
- Time complexity—Comparison with matchmaking based on conventional matchmaking, matchmaking by retrieving resource information from database, and matchmaking by retrieving resource information from ontology knowledge base.

The performance of the proposed semantic-based resource discovery module has been evaluated by comparing the amount of results obtained from conventional keyword-based searching mechanisms by submitting various resource requirement requests. These requests were converted into 'queries' of the format described in Section 6.2. Semantic description module is made to run periodically across the Grid resources that contact the MDS, aggregate resource information, and create ontology knowledge base. Initially, the queries were executed through conventional matchmaking mechanism without considering the semantic present in the request (direct query). Then, the queries were all converted into Algernon query and executed over knowledge base for resource discovery. Based on the experimental results, it has been concluded that the semantic-based searching mechanism retrieved more closely matching (refer to Figure 6) resources and thus resulted in greater 'hits' than the conventional searching mechanism. This is because, even though, the resource requested by the user is not exactly matching with available ones described in the knowledge base, the semantic component retrieved resources that exhibit subsumption relationships with the request. These resources are then suggested to the users for job execution.

The second experiment was focused on increasing the efficiency of the discovery mechanism by maintaining a database to store unknown query tags and reused for the next query. It has been mentioned in the earlier section that separate query tags are proposed such as 'hasRAM' to specify requirements. This puts a minor restriction on the user forcing him to adhere to some pre-defined format. Whenever the user provides a new tag which is unknown to the discovery module, the searching process fails. Hence, the discovery mechanism is extended to support the user to use a different tag to specify a particular property which is not defined and allows the user to map the unknown label to the corresponding tag used in the ontology template and thereby update the database with new tag. The performance of this extension has been estimated by considering 'hit' and 'miss' for a particular query. The discovery module is tested with several naïve users to navigate

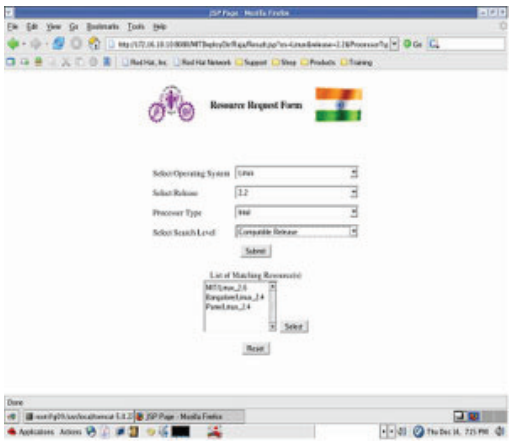

Figure 5. Semantic component in Garuda.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

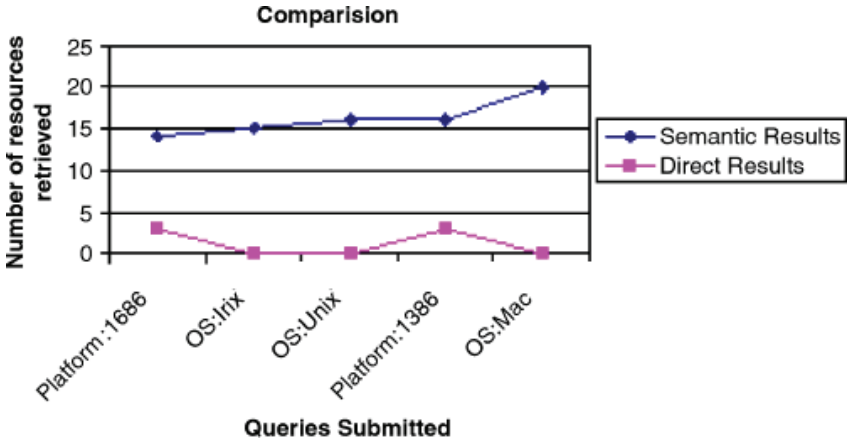

Figure 6. Direct vs semantic results.

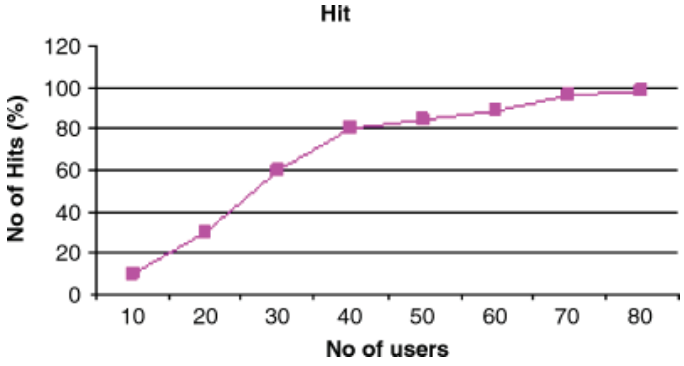

Figure 7. Effect of updating keywords in database.

the discovery portal and search for a particular resource by specifying the various characteristics of a resource. Queries were executed on the knowledge base and results were obtained. Initially, the database of synonymous words was empty and it was updated by the users whenever they provided a new label. Hence, with the number of users increasing and hence the amount of synonymous words were also increasing, the semantic search becomes more efficient as the user grows as shown in Figure 7.

The third experiment is to compare the performance of the Algernon inference engine with another popular reasoning language SPARQL query language used in S-MDS project for information retrieval from ontology knowledge base. The performance of the discovery module is heavily relying on the inference engine used for information retrieval from the knowledge base. Several open source and freely available reasoning languages were analyzed and practiced. The requirement of the discovery mechanism is to discover instances of resources that possess particular characteristics. Hence, query-based ontology reasoners were preferred and eventually Algernon and SPARQL were selected. The performance of both the languages was experimented by populating knowledge base with several numbers of resources and queries based on both the languages were tested. The time taken for executing the queries on to the knowledge base was measured. The experimental results

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

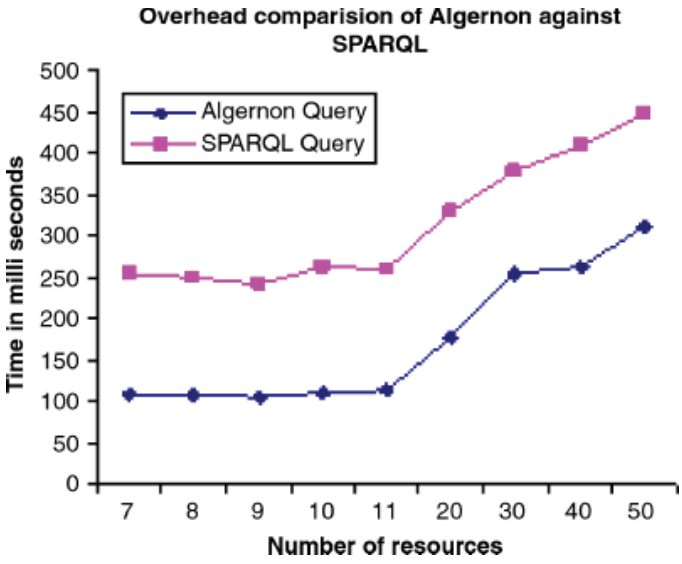

Figure 8. Algernon vs SPARQL inference engine.

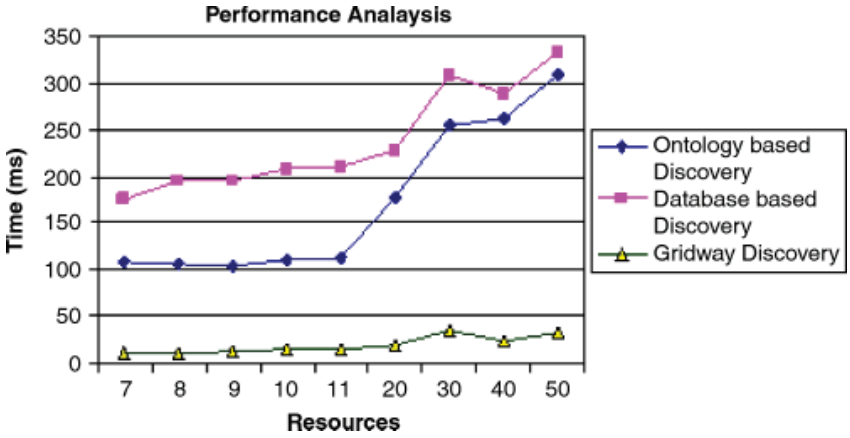

Figure 9. Performance comparison of matchmaking mechanism in Gridway scheduler, semantic-based matchmaking and database-based matchmaking.

showed that Algernon-based queries were executed quicker than the SPARQL queries as shown in Figure 8.

The last experiment was carried out to analyze time overhead in various matchmaking systems. Every Grid scheduler uses its own matchmaking strategy for discovering suitable resources. For example, Gridway, a Globus-based Grid metascheduler, aggregates the available resource information through the information manager and stores them in hostpool, and then compared with the requirements expressed in the job request. In gLite middleware, Berkeley database is maintained for storing resource information. In such cases, the discovery module retrieves information from the database and performs matchmaking. In semantic discovery mechanism, information is retrieved from the ontology knowledge base and matchmaking is carried out. In this experiment, the time

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

complexity occurs during matchmaking in all the abovementioned techniques. It is observed that the *object*-based gives the best results with respect to the time taken and semantic-based discovery is better than the matchmaking carried out by retrieving information from the database. This experiment can also be interpreted such that the time taken while retrieving information from the knowledge base is less than retrieving information from the database. The results are shown in Figure 9.

8. CONCLUSION

The proposed knowledge layer components support the ontology representation of Grid resources which in turn enable the discovery of suitable Grid resources based on the semantics of the request. This feature complements the Grid scheduler to find out closely related resource and can suggest to the users a flexible way of scheduling application to the resources. Integration of semantic component with Grid schedulers requires a knowledge of the operational flow of that scheduler. Appropriate interface need to be developed to communicate the resources discovered by the semantic component to the Grid scheduler for deployment of applications.

Further, ontology can also be used to represent the resource usage policies and in such cases, the discovery mechanism needs to be extended to discover the resources to consider the policies before making scheduling decision. Protégé-OWL libraries allow to modify the structure of ontology template dynamically and hence it is possible to extend the template when new 'concepts' are to be included. Exploitation of the rule-based inference can further improve the performance of resource discovery mechanism.

ACKNOWLEDGEMENTS

The authors sincerely thank the Ministry of Communication and Information Technology, Government of India, for financially supporting the Centre for Advanced Computing Research and Education of Anna University Chennai, India. This work is also supported by the Australian Department of Innovation, Industry, Science and Research.

REFERENCES

1. Foster I, Kesselman C (eds). *The Grid*: *Blueprint for a New Computing Infrastructure*. Morgan Kaufmann: Los Altos, 1999; 259–278.
2. Foster I, Kesselman C, Tuecke S. The anatomy of the grid: Enabling virtual organizations. *International Journal of High Performance Computing Applications* 2001; **15**(3):200–222.
3. Foster I, Kesselman X, Nick J, Tuecke S. The physiology of the grid: An open grid services architecture for distributed systems integration. Open Grid Service Infrastructure WG, Global Grid Forum, 22 June 2002; 1–5.
4. Foster I, Kesselman C. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications* 1997; **11**(2):115–128. Globus Middleware. Available at: http://www.globus.org [18 September 2005].
5. gLite Middleware. Available at: http://www.gLite.org [5 October 2007].
6. Erwin D. UNICORE—A grid computing environment. *Concurrency*, *Practice and Experience Journal* 2002; **14**:1395–1410. Unicore Middleware. Available at: http://www.unicore.eu [19 October 2007].
7. De Roure D, Jennings NR, Shadbolt NR. The semantic grid: A future e-science infrastructure. *Grid Computing—Making the Global Infrastructure a Reality*. Wiley: New York, 2003; 437–470.
8. Saha GK. Web Ontology Language (OWL) and Semantic Web. ACM Ubiquity, September 2007; 1.
9. Algernon Inference Engine. Available at: http://Algernon-j.sourceforge.net/doc/overview.html [5 December 2005].
10. Moab Metascheduler. Available at: http://www.clusterresources.com/resources/documentation.php [21 January 2006].

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

11. Huedo E, Montero RS, Llorente IM. The gridway framework for adaptive scheduling and execution on grids. *Scalable Computing—Practice and Experience* 2005; **6**(3):1–8. Gridway Metascheduler. Available at: http://www.Gridway.org [13 November 2006].
12. Czajkowski K, Fitzgerald S, Foster I, Kesselman C. Grid information services for distributed resource sharing. *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing* (*HPDC-10*). IEEE Press: New York, 2001; 181.
13. Imamagic E, Radic B, Dobrenic D. An approach to grid scheduling by using condor-G matchmaking mechanism. *Journal of Computing and Information Technology* 2006; **14**:329–336.
14. Venugopal S, Buyya R, Winton L. A grid service broker for scheduling e-science applications on global data grids. *Concurrency and Computation*: *Practice and Experience* 2006; **18**(6):685–699.
15. Monitoring and Discovery Service. Available at: http://www.globus.org/toolkit/mds [3 October 2005].
16. Yu J, Venugopal S, Buyya R. A market-oriented grid directory service for publication and discovery of grid service providers and their services. *The Journal of Supercomputing* 2006; **36**(1):17–31.
17. Corcho O, Alper P, Kotsiopoulos L, Missier P, Bechhofer S, Goble1 C. An overview of S-OGSA: A reference semantic grid architecture. *Journal of Web Semantics* 2006; **4**(2):102–115.
18. Wroe C, Goble C, Greenwood M, Lord P, Miles S, Papay J, Payne T, Moreau L. Automating experiments using semantic data on a bioinformatics grid. *IEEE Intelligent Systems* (Special issue on e-Science Jan/Feb) 2004; **19**:48–55.
19. Murphy MJ, Dick M, Fischer T. Towards the semantic grid: A state of the art survey of semantic web services and their applicability to collaborative design, engineering, and procurement. *Communications of the IIMA* 2008; **8**(3):11–24.
20. Chen L, Shadbolt NR, Goble CA, Tao F, Cox SJ, Puleston C, Smart PR. Towards a knowledge-based approach to semantic service composition. *Second International Semantic Web Conference*, Sanibel Island, FL, U.S.A., 20–24 October 2003.
21. He H, Tangmunarunkit H, Decker S, Kesselman C. A Semantic Matchmaker Service on the Grid. *WWW2004*, 17–22 May 2004; 326–327.
22. Said M, Kojima I. S-MDS: Semantic monitoring and discovery system. *Journal of Grid Computing* 2009; **7**:205–224.
23. Pahlevi Said M, Kojima I. Semantic grid resource monitoring and discovery with rule processing based on the time-series statistical data. *Proceedings of 9th International Conference on Grid Computing*, Tsukuba, Japan, 2008; 358–360.
24. Pernas AM, Dantas MAR. Using ontology for description of grid resources. *Proceedings of 19th International Symposium on High Performance Computing Systems and Applications* (*HPCS'2005*), 2005; 223–229.
25. Tangmunarunkit H, Decker S, Kesselman C. Ontology-based resource matching in the grid—The grid meets the semantic web. *Proceedings of 1st Workshop of Semantics in Peer to Peer and Grid Computing in Conjunction with 12th W3C*, Budapest, 2003; 706–721.
26. Zhang WY, Wang Y. Towards building a semantic grid for e-government applications. *WSEAS Transactions on Computer Research* 2008; **3**(4):273–282.
27. Yin JW, Zhang WY, Cai M. Weaving an agent-based semantic grid for distributed collaborative manufacturing. *International Journal of Production Research* 2009; **47**:3079–3095.
28. Hartung M, Loebe F, Herre H, Rahm E. A platform for collaborative management of semantic grid metadata. *Book Chapter from Intelligent Distributed Computing*, *Systems*, *and Applications*, vol. 162. Springer: Berlin/Heidelberg, 2008; 115–225.
29. Selvi T, Balachandar RA, Kandasamy V, Buyya R, Raman R, Mohanram N, Varun S. Semantic-based grid resource discovery and its integration with the grid service broker. *Proceedings of the 14th International Conference on Advanced Computing and Communications*, Surathkal Karnataka, India, 20–23 December 2006.
30. Berners-Lee T, Hendler J, Lassila O. The Semantic Web. Scientific American, May 2001.
31. Resource Description Framework. Available at: http://www.w3.org/RDF.
32. Horridge M, Knublauch H, Rector A, Stevens R, Wroe C. A practical guide to building OWL ontologies using the protege-OWL plugin and CO-ODETools edition 1.0. The University of Manchester, Stanford University, August 2004; 11–50.
33. Semantic Web Rule Language. Available at: http://www.w3.org/Submission/SWRL/.
34. Roure D, Jennings N, Shadbolt N. The semantic grid: A future e-science infrastructure. *Grid Computing—Making the Global Infrastructure a Reality*. Wiley: New York, 2003.
35. Stumme G, Ehrig M, Handschuh S, Hotho A, Maedche A, Motik B, Oberle D, Schmitz C, Staab S, Stojanovic L, Stojanovic N, Studer R, Sure Y, Volz R, Zacharias V. The Karlsruhe view on ontologies. *Technical Report*, University of Karlsruhe, Institute AIFB, 2003; 3–5.
36. Davies J, Studer R, Sure Y, Warren PW. Next generation knowledge management. *BT Technology Journal* 2005; **23**(3):175–190.
37. Thamarai Selvi S, Balachandar RA, Vijayakumar K, Mohanram N, Vandana M, Rajagopalan R. Semantic description and discovery of grid services using functionality based matchmaking algorithm. *Proceedings from International Conference on Web Intelligence*, Hong Kong, China, December 2006; 170–173.
38. Prahlada Rao BB, Ramakrishnan S, Raja Gopalan MR, Subrata C, Mangala N, Sridharan R. e-Infrastructures in IT: A case study on Indian national grid computing initiative—GARUDA. *Computer Science—Research and Development*, vol. 23(3–4). Springer: Berlin, Heidelberg, June 2009; 283–290.

Copyright © 2009 John Wiley & Sons, Ltd.

DOI: 10.1002/spe