Grid Resource Ontologies and Asymmetric Resource-Correlation*

Mumtaz Siddiqui[1], Thomas Fahringer[1], Jürgen Hofer[1], and Ioan Toma[2]

[1]Distributed and Parallel Systems Group (DPS), University of Innsbruck
Technikerstraße 21a, 6020 Innsbruck, Austria
{mumtaz, tf, juergen}@dps.uibk.ac.at}

[2] Digital Enterprise Research Institute (DERI), University of Innsbruck
Technikerstraße 21a, 6020 Innsbruck, Austria
{ioan.toma@deri.org}

Abstract:
Automatic Grid resource discovery and brokerage shields the Grid middleware complexities from the Grid users and leads towards an invisible but simple and robust Grid. Realizing this vision requires a machine understandable resource description and powerful correlation [1] mechanism. Semantic technologies like ontologies provide vocabularies with explicitly defined, unambiguously understandable and automatically machine-interpretable meanings which make the process of automatic resource brokerage possible. We propose a fully Ontology-based resource description, discovery and correlation mechanism. For the resource description model we have replaced the classical symmetric attribute based resource description model with an extensible asymmetric resource description model. This model provides foundation to our flexible and extensible discovery and correlation mechanism.

Keywords*: Grid resource brokerage, Grid resource ontology, Grid resource allocation, matchmaking, asymmetric resource matching.*

1 Introduction

With the simultaneous emergence of the Grid computing and increase in the number of Grid resources, an automatic resource management system gains importance. The automatic Grid resource management is a challenging task. It has to provide a mechanism in which the Grid resources can be advertised by the resource providers and automatically discovered and allocated by the resource requester. The discovery and allocation of resources is performed by Grid middleware components called resource broker [SF05, CFK+98, CFFK01]. Moreover, brokers often also provide negotiation mechanism between resource provider and requester.

*The work described in this paper is partially supported by the Higher Education Commission of Pakistan.
[1]Here 'correlation' means Grid resource discovery and resource matching.

A Grid resource is a Grid entity that provides capabilities to a consumer. Different resources can provide similar capabilities but with different quality of services. The resource capabilities are required to be presented in such a way that a consumer can easily discover a resource or a resource ensemble with needed capabilities. This requires some sophisticated patterns of resource discovery and negotiation. A powerful discovery mechanism builds on expressive description mechanisms. This means, it is necessary to explicitly, precisely and unambiguously describe Grid resources and specify various constraints over resource descriptions. The description should be automatically interpretable and understandable by middleware components.

Attribute based resource description, used in most state-of-the-art Grid systems, has several short comings. For example, they lack expressiveness, dynamicity and independence between the resource provider and resource consumer. The mechanism of resource description is *symmetric*, i.e. both resource provider and consumer have to agree on a certain syntax or schema.

We propose an ontology-based resource description, discovery and correlation mechanism. *Correlation* is an automatic process of creating associations between resource requests, the available resources, and their characteristics at a specific point in time. Correlation is related to resource matching in that matchmaking is a central part of it, however we use it as a more general concept. An ontology provides vocabularies with explicitly defined and machine understandable meanings. We propose Grid resource ontologies in the form of Ontology Web Language (OWL) [SvHFJ$^+$] classes and concepts for describing resources in such a way that they can be unambiguously interpreted and automatically understood by the *correlation* system. Our discovery mechanism is based on a simple but expressive request-response mechanism in which clients express requests based on OWL Query Language (OWL-QL) [FHH03].

Resource ontologies are asymmetrically extensible so that resource providers can easily augment them without loosing the semantic soundness. We do not have to agree on a certain terminology while extending semantics of generic concepts. For example, a term *Intel* defined as a class of *Processor* in our initial generic ontology of computing resources, then one can asymmetrically extend the term *Intel* to *Pentium* and *Pentium* to *Xeon*. A reasoner then can automatically and unambiguously infer that *Xeon* is a specialization of *Pentium* as well as *Intel*.

The rest of this paper is organized as follows: In Section 2 we introduce innovations in ontology languages. In Section 3 we discuss resource ontologies, requesting mechanism and correlation framework. In Section 4 we demonstrate the effectiveness of our approach. In Section 5 we review related work and compare it with our approach. In Section 6, we summarize our proposal and future extensions of Grid resource ontologies and asymmetric resource correlation.

2 Ontology Languages

A commonly used definition of ontologies is that they are formal explicit specifications of a shared conceptualization [Gru93]. The level of formality used in prehending these

descriptions can be quite variable, ranging from natural language to logical formalisms, but increased formality and regularity clearly facilitates machine processing. The Web Ontology Language [SvHFJ+] is a formal standard language for representing ontologies in the Semantic Web. In OWL, an ontology is a set of definitions of classes and properties and the constraints on the way those classes and properties are employed. OWL has three variants OWL-Lite, OWL-DL and OWL-full with different levels of expressiveness.

In order to provide a powerful expressiveness and fact stating ability, OWL inherits features from RDF [RDF98] and RDF Schema [DR03] and extends them by providing new and powerful constructs. OWL can declare classes and organize them in a subsumption hierarchy, also the classes can be expressed as a logical combination of other classes. Properties can also be organized in a sub-property hierarchy. OWL provides different kinds of *restrictions* on classes and properties which are also considered as specialized concepts or classes.

In the domain of the Semantic Web, ontologies play an important role in automating processes to access information. For this, ontologies provide structured and extensible vocabularies that demonstrate the relationships between different terms allowing intelligent agents to flexibly and unambiguously interpret their semantics. For example, a computing resource ontology might include the information that the terms *Pentium* and *Celeron* are kind of *Intel* Processors, that *Intel* is not a kind of the *AMD* or *SPARC*, and that an *Intel* system is one whose processor is *Pentium* or *Celeron*. This information allows the term *"Computer with Pentium or Celeron Processor"* to be unambiguously interpreted (e.g. by a resource broker) as a specialization of the term *"Intel System"*. In the Description Logics, the fundamental reasoning of *concept expression* is subsumption [MTTK02], which checks whether one concept is a subset (or superset) of an other concept.

OWL-QL is a formal language and protocol for a query-answering dialog between intelligent agents using knowledge represented by the OWL Knowledge Base. It precisely specifies the semantic relationships among a query, a query answer and the Knowledge Base used to produce the answer. An OWL-QL query can specify which of the URIs referred to in the query pattern are to be interpreted as variables. Variables come in three forms: *must-bind*, *may-bind*, and *don't-bind*. Answers are required to provide bindings for all the must-bind variables, may provide bindings for any of the may-bind variables, and are not to provide bindings for any of the don't-bind variable. OWL-QL uses the standard notion of logical entailment: query answers can be seen as logically entailed sentences (OWL facts and Axioms) of the queried knowledge base.

3 Resource to Request Correlation

The correlation of a resource request to available resources depends on their proper and accurate description. Different languages, based on different logical formalism, could be use to write resource descriptions. One can use *Description Logics* [BCM+03], *Logical Programming* and *First order Logic*, as a logical formalism. Since in our case it is enough to check class-subclass relations between classes from resource and request descriptions we have considered a *Description Logics* based language namely OWL [SvHFJ+]. OWL

provides a rich set of modelling primitives which provides enough expressivity for our requirements. By employing OWL, we attain the following:

- Grid resource management systems make a large step towards compatibility within the Semantic Web and Grid resources descriptions become web-understandable.
- The resource provider can have maximum freedom to describe resources with different levels of complexity and completeness.
- XML-schema datatypes can be exploited for resource description.
- Complex resource matching is possible based on subsumption relationships.
- A more natural conceptual definition of resources based on the restriction over the resource attributes is possible, and a semantics level of agreement between resource provider and consumer can be achieved.
- After having a conceptual and flexible resource description, a resource broker can catagorize the resource ensembles and devise alternative options.
- Most promisingly, clients can express complex requests in a simple-human as well as machine-understandable format. Also, the system can fulfil more resource requests. For example, in a system without ontology language, a request for a *Unix* system fails if the term *Unix* is not specified. However, using an ontology this might be successfull.
- Spelling or typing errors in descriptions and requests are prevented by using a controlled vocabulary.

Another reason of selecting OWL for our solution it is the good tool support for creation and manipulation of ontologies and the availability of reasoners such as Pellet [2].

3.1 Resource Ontology

We propose a resource correlation mechanism based on physical resource ontologies, logical resource ensemble ontologies and policy ontologies defined in OWL-DL. *GridElement* is a common superclass for each concept as shown in the Figure 1. The physical resource ontology describes physical resources, whereas logical resource ontologies represent logical ensembles inferred by the resource broker based on the physical resources ontologies. For example, specific purpose *resource ensemble*s etc. Resource providers and requester will use these ontologies to annotate their resources and to described their requests in an ontological form. Furthermore, the correlation mechanism will use these ontologies in checking if a resource can correlated with a specific request.

[2] http://www.mindswap.org/2003/pellet/index.shtml

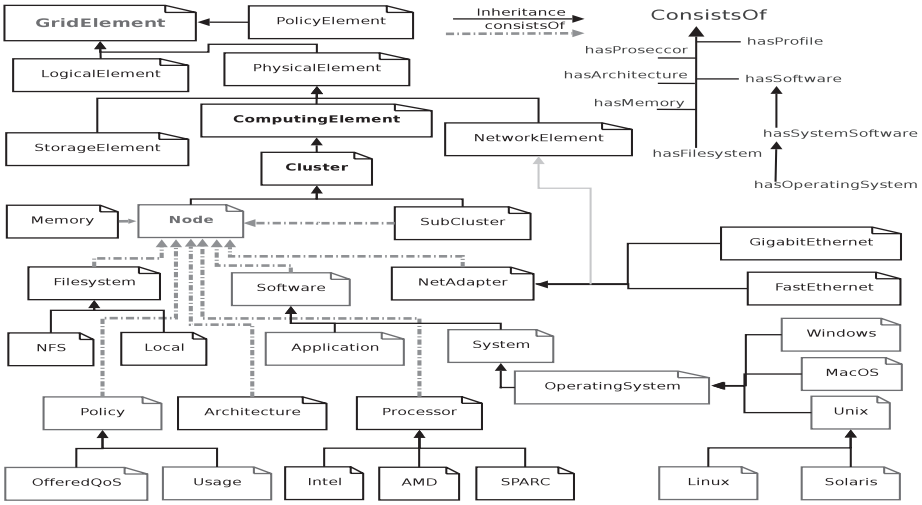

Figure 1: An incomplete class and property (top right) hierarchy of different concepts of the Grid elements and sub-elements. Solid lines show inheritance whereas dotted lines represents aggregation.

Physical Resource Ontology

This provides a rich vocabulary which enables resource providers to describe their resource(s) in a more expressive way. In this layer, a basic ontology model for the physical resources is provided which can be used or inherited to add more domain specific concepts while describing a resource.

The model: We propose to represent the concepts related to the Grid resources as OWL classes in a hierarchical way. The resource description is defined as the boolean combination of a set of constraints over the resource concepts and properties. The *constraints* can be expressed either through OWL restrictions or XML schema restrictions. The resources can be described by using different classes and properties and also by importing domain specific ontologies. In order to use the ontology model more effectively, a request for a resource or a set of resources could also be considered as a resource, which is subsummed in the latest model of the resource ontology available at the time of the request.

The key top-level ontology of the physical resources consists of classes and properties that describe *Network*, *Storage* and *Computing* elements like *Cluster*, *SubCluster* and *Node*. These classes includes *Profile*, *Policy*, *Processor*, *OperatingSystem*, *Architecture*, *Filesystem*, *Memory*, *State*, etc. A *Profile* of a resource consists of concepts like resource *Info*, *State*, *Jobs* and *Policy*. Each class defines the most general properties of the concept that it models. In order to achieve this model using OWL, each class is defined to be a subclass of a set of anonymous classes and each class restricts some of its properties. For example, the *Node* class shown in the Figure 1 is defined as subclass of several anonymous classes that each of which restricts one of the class properties such as *hasFileSystem*, *hasArchitecture*, *hasNetworkAdapter*, etc.

The vocabularies are extensible. A user can extend them asymmetrically without loosing their semantics. For example we can add a new term *NetGear* as a specialization of *GigabitEthernet* which is not compatible with *Solaris* Operating System.

The new term added independently by a user can be inferred by a reasoner that it *is-a* NetworkAdaptor. Also *integrity* of requests can be verified by the reasoner by using request satisfiability check. For example if someone requests that:

```
"I need a Computing resource with Solaris OS
    and with NetGear Ethernet adapter".
```

The reasoner can easily identify that this request cannot be fulfilled since *NetGear* is incompatible with the *Solaris* OS.

Resource Ensemble Ontology

It deals with the conceptual grouping of Grid resources. Based on the concepts and restrictions given under the physical resource ontology, a broker bequeathes different resource ensembles. The resources in a resource ensemble share some common features, or provide collectively a new and complex capability.

A resource ensemble ontology may also group resources to ensembles in which enclosed nodes collectively achieve a certain minimum number of Mflops as part of the ensemble. In this way a resource ensemble provides a required number of MFlops which is otherwise not possible by an individual node. This kind of resource ontology enables a resource broker to accept requests in a form more generic and closer to the natural language. Also, a cache of logical concepts available to other distributed discovery services can be maintained as a logical resource ensemble entity referring to the other discovery services. Figure 4 shows a sample resource ensemble ontology.

Policy Ontologies

include vocabularies for the description of different conventions and agreements on resource usage, user's requirements, community authorization etc. For example, we can describe that a specific SMP-node will only execute MPI application. A resource provider and/or a middleware authorization service can specify which community users or groups can (or cannot) access specific resources. Also a resource provider and user can specify offered and required quality of service, respectively.

3.2 Querying on the Grid

We propose OWL-QL querying mechanism for requesting Grid resources. OWL-QL can be used to describe a resource request in a simple and very expressive format by using a collection of OWL *facts and axioms*. An OWL-QL query includes a query pattern that is a collection of OWL sentences, a list of *must-bind*, *may-bind* and *don't-bind* variables. These

kinds of variable bindings distinguish OWL-QL from other query languages and help in accessing alternative or close matches. Also, a query optionally includes a query premise, an answer pattern and a reference to Answer Knowledge Base (AKB). For example a client can request for nodes providing the following query pattern:

```
Query: ("Which node has 64bit Solaris operating system?")
  Query Pattern: {(hasOperatingSystem ?node ?os)
                  (type ?os Solaris)
                  (forArchitecture ?os 64bit)}
  Must-Bind Variables List: (?node)
  May-bind Varables List  : ()
  Don't-bind Variable List: ()
  Answer Pattern: {(?node)}
Answer: ("mulle.dps.uibk.ac.at" "quirl.dps.uibk.ac.at")
         Answer pattern instance: {("mulle.dps..." ...)}
```

OWL-QL is designed for answering queries of the form *"What URIrefs and literals from the AKB and OWL denote objects that make the query pattern true?"* [FHH03]. A query premise can be added as shown in the example below:

```
Query: "if Node mulle.dps.uibk.ac.at has Solaris
        operating system and DpsNFS filesystem is
        mounted on it then what is the available
        Disk space on DpsNFS?"
  Premise: {(type mulle.dps.uibk.ac.at Node)
            (hasFileSystem mulle.dps.uibk.ac.at DpsNFS)}
  Query pattern: {(available DpsNFS ?size)}
  Must-bind Variable List: (?size)
  ....
Answer: (mulle.dps.uibk.ac.at has 32GB available disk on
         DpsNFS filesystem)
```

In OWL-QL queries URLs of answering servers and references to answer knowledge bases can be specified. We propose to exploit this in making a distributed resource correlation framework. For example we can install resource ontology and usage policy KB, on different nodes and a requester can specify them dynamically while making a request.

The use of human readable surface syntax for queries and answers are very useful in a Semantic Grid context. It would be possible to devise a translation mechanism in which a simple request in the form of a formal natural language syntax could be translated into an OWL-QL query pattern. For instance in the first example the *modal verbs* can be replaced with a *must-bind* variable (*which node* with *?node*), objects with OWL classes, etc.

3.3 Correlation Mechanism

Resource correlation is based on the subsumption mechanism of concepts and roles represented by the Description Logics formalisms. *Description Logics* techniques are employed to classify the Grid resource descriptions. Resource descriptions are maintained in a hierarchy of concepts or classes. Classes are linked through roles established by using the restrictions on classes and properties. Each resource description is embedded in the hierarchy persistently once it is satisfied. A request for the resource(s) could also be made similar to the resource description and subsummed in the taxonomy.

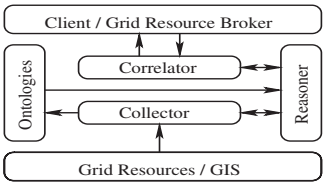

Figure 2: The Correlation Framework.

This mechanism enables the *Grid resource broker* to calculate an exact or a close match. Based on the subsumption of the request in the resource taxonomy, the broker service can easily suggest or offer the requester some other possible options that can be exercised if an exact match can not be found. This kind of offering is not possible without an ontological model. The resource subsumption in the taxonomy and request correlation is performed by considering the matching concepts based on the following propositions:

- $Request \sqsubseteq Resource | Request \equiv Resource$: This shows that a *request* concept is a sub concept or an equivalent concept of the resource classes, which means an exact and preferable match satisfying all necessary and sufficient conditions.
- $Request \sqsupseteq Resource$: Represents *request* as a super concept of the *resource*. Resources belonging to the super-concept do not fully fulfil all constraints set by the request, but are considered as the best closest match and used as an alternative.
- $\neg (Request \sqcap Resource \sqsubseteq \bot)$: Intersection of both resource and request concepts is satisfiable and considered as a least suitable option as an alternative.
- $(Request \sqcap Resource \sqsubseteq \bot)$: This means that no match is found.

These concept matching propositions clearly organize the relationships in a well defined discrete scale. This provides a solid ground for a *Description Logics* reasoner to be used for the classification of *Request* to compute its subsumption relationship against all registered resources.

3.4 Framework

The resource correlation mechanism is part of the discovery service of the Askalon's Grid resource management system [Fah]. It consists of a *Correlator*, a resource information *collector*, a *reasoner* and ontology model or knowledge base(s). The architecture is shown in the Figure 2.

Collector or Observer: This component interacts directly with the physical resources and Grid Information Services like MDS [CFFK01], collects resource data, optionally performs translation and updates the resource ontology. Before including a resource description in the underlying knowledge base, the satisfaction of all concepts is checked, realization of a non satisfiable resource is not possible. When the resource description is accepted it becomes a set of new concepts within the subsumption tree. A concept within the tree

under an ontology resource model like *Node* represents the whole resource advertisement. The *collector* component can collect resource description directly from the resource or it can collect aggregated resource descriptions from a Grid information service.

Reasoner: Different Description Logics reasoners such as RACER [VR01] are available which perform validation of concepts, check integrity of ontologies, classify the taxonomy, check entailment and answer OWL-QL queries. Some open-source OWL DL reasoners can be used in conjunction with OWL API libraries and other ontology frameworks for building semantic applications.

Correlator: The Correlator is the heart of the resource correlation framework. It accepts requests from the client and optionally transforms into the required format by using resource ontology model. It checks satisfiability of the concepts in the request and subsumption of the request in the resource ontology. Based on the subsumption created by the reasoner, the *Correlator* performs resource matching using propositions as described in Section 3.3.

4 Examples

In this section we demonstrate the effectiveness of the proposed resource ontology and correlation mechanism for the Grid resources. We have created a subset of Grid Resource Ontology by using Protege Ontology Editor with OWL plugin[3]. We have used OWQL Server[4] to query for resources, and RACER reasoner[5] to see the classification of a resource request in the Grid resource KB. The subsumption of resource requests in the model proves propositions given in Section 3.3 and matches with OWL-QL query result. The following example contains a query to search for nodes with *Solaris* operating system and with network file system called as *ZidNFS*.

```
Query: ("Which node has Solaris OperatingSystem with
         ZidNFS FileSystem?")
  Query Pattern: {(hasOperatingSystem ?node ?os)
                  (type ?os Solaris)
                  (hasFileSystem ?node ZidNFS)}
  Must-Bind Variables List: (?node)
  May-bind Varables List  : (?os)
  Don't-bind Variable List: ()
  Answer Pattern: {(?node)}
Answer: ("Node4, Node5, Node1")
```

This query is performed at the time when the resource ontology model as shown in the Figure 3 with solid lines was available. The may-bind *?os* variable indicates that the answering agent or server can return not only the exact match but close matches as well. As a result *node4*, *node5* and *Node1* are returned. The sequence indicates that the *Node4* could be an exact match whereas *Node1* could be a close match or an alternative option.

The example ontology shown in the Figure 3 explains how the resource correlation is achieved. At the time of request we have a subtree of the main subsumption tree in the

[3] http://protege.stanford.edu/plugins/owl/
[4] http://onto.stanford.edu:8080/owql/FrontEnd/
[5] http://www.sts.tu-harburg.de/ r.f.moeller/racer/

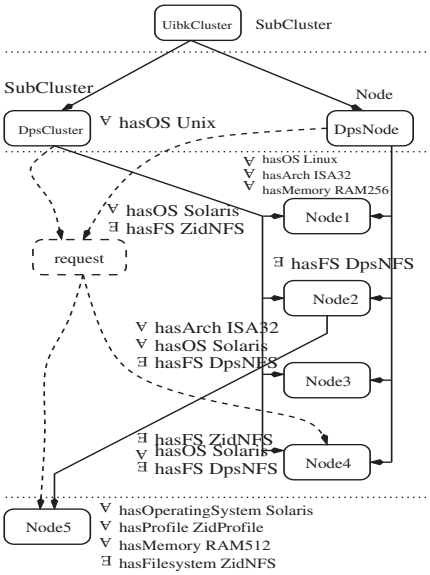

Figure 3: Subsumption of a *Request* in the existing resource taxonomy.

system with seven Grid resources. Resources are shown in a hierarchy in which resource description in each lower layer is a specialization of the resource concept given in the upper layers. We added the request to the KB as follows:

```
<owl:Class rdf:ID="Query">
 <owl:equivalentClass>
  <owl:Class>
   <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
     <owl:allValuesFrom rdf:resource="#Solaris"/>
     <owl:onProperty>
      <owl:ObjectProperty
       rdf:about="#hasOperatingSystem"/>
     </owl:onProperty>
    </owl:Restriction>
    <owl:Restriction>
     <owl:someValuesFrom rdf:resource="#ZidNFS"/>
     <owl:onProperty>
      <owl:ObjectProperty
       rdf:about="#hasFileSystem"/>
     </owl:onProperty>
    </owl:Restriction>
    <owl:Class rdf:about="#Node"/>
   </owl:intersectionOf>
  </owl:Class>
 </owl:equivalentClass>
</owl:Class>
```

The request is then classified in the subsumption tree by the RACER reasoner. This request is shown in the subsumption tree as *Request* surrounded by the dotted rectangle and its relation with other nodes as dotted lines. It is shown that nodes *Node4* and *Node5* are specialized concepts of the *Request* and as such are marked as exact matches. There is no equivalent concept. If we look for the super concepts of the *Request* up to the root,

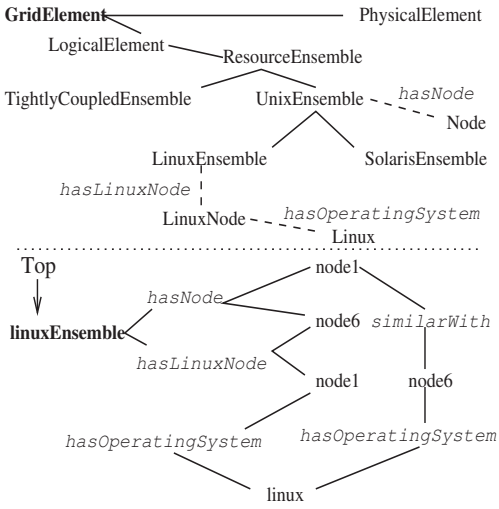

Figure 4: *ResourceEnsemble class hierarchy (above) and graph of the instances relationships (below).*

then sub clusters *DpsCluster* and *DpsNode* would be marked as a match. By employing the third proposition as specified in Section 3.3, node *Node1* is found compatible with the request. All other nodes are declared inconsistent as the restrictions over the properties are not consistent.

Resource ensemble ontologies can provide provisioning of aggregated power of available resources. For example, a resource ensemble with aggregated compute power of processors of a set of nodes.

The following request returns a *Linux Resource Ensemble* with *Linux* Nodes as given in the ontology of *LinuxEnsemble* in Figure 4.

```
Query: ("What resource ensamble consists of Linux Nodes?")
  Query Pattern: {(consistsOf ?ensemble ?node)
                  (type ?node Node)
                  (hasOperatingSystem ?node ?os)
                  (type ?os Linux)}
  Must-Bind Variables List: (?ensemble, ?node)
  May-bind Varables List  : ()
  Don't-bind Variable List: ()
  Answer Pattern: {(?ensemble with nodes ?node)}
Answer: ("linuxResourceEnsemble with nodes Node1, Node6")
```

We can demonstrate that a request for a node with Windows OS and with SPARC architecture cannot be fulfilled as both *Windows* OS and *SPARC* are described as *incompatible* in the underlying resource ontology.

```
Query: ("Which node has Windows OS and SPARC architecture?")
  Query Pattern: {(hasOperatingSystem  ?node Windows)
                  (hasPlatform ?node SPARC)
  Must-Bind Variables List: (?node)
  May-bind Varables List  : ()
  Don't-bind Variable List: ()
Answer: ("Incompatible")
```

In another example we can show how a requester can set preferences for the response by specifying different variable bindings.

```
Query: ("Which resource ensemble has at least 8 processors
         with 64bit architecture and Intel platform?")
  Query Pattern: {(consistsOf  ?ensemble ?node)
                  (type ?nodes Node)
                  (hasProcessor ?node ?processor)
                  (type ?processor Processor)
                  (hasArchitecture ?processor ?arch)
                  (hasPlatform ?processor ?platform)
                  (type ?arch ISA64)(type ?platform Intel)}
  Must-Bind Variables List: (?node, ?platform)
  May-bind Varables List  : (?arch)
  Don't-bind Variable List: (?processor, ?ensemble)
Answer: ("... ...")
```

This query example describes that the answering agent should return names of nodes in an ensemble. The ensemble should have processors with *Intel* platform and preferably with *64bit* architecture. The Grid resource ontology terms and concepts used in the examples are extensible as shown in Figure 1. No coordination between resource requesters and providers is required for this purpose.

5 Related Work

The most prominent information systems approaches in the Grid and Web communities are the Meta-computing Directory Service (MDS) [CFFK01] and UDDI [OAS]. They support a simple query language for the resource and service selection but lack expressive description capabilities. Thereby, there is no sophisticated resource correlation mechanism available. In the traditional Grid resource management systems, different symmetric, attributed-based resource matching mechanisms are used. These mechanisms provide expressiveness to some extent but the drawback is that they still require symmetric attribute-based descriptions and constraint mechanisms.

The Condor [Tea] system is one example in which a symmetric attribute based matchmaking is performed for the resource allocation in a distributed Grid infrastructure. For this purpose a classified advertisement-like matchmaking framework is developed. In this framework, resources and requests are described in the form of attribute *name-value* pairs and resource consumers and providers specify their matching constraints. These constraints are then evaluated to determine a match for each *request* with available resources. This matchmaking works only if both request and resource use the same attribute names and agree upon attribute values. A *Request* ClassAd example is shown below:

```
Request JobClassAd:
[ Type = "Job"; Owner="mumtaz"; Constraint = type == "Machine" &&
          Arch == "Intel" && Disk >= 20000; OpSys == "Linux260"; ]
```

This example shows a request for a machine with *Intel* architecture, *Linux* operating system and at least 20GB disk space. *Resource ClassAds* are described in the same way by using

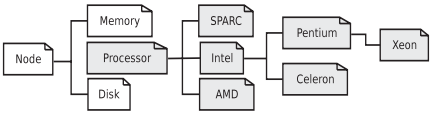

Figure 5: Class hierarchy of Processor.

the same syntax. The *constraint* clause of request *classAd* is matched with the *constraint* clauses of resource *classAds* to find the match. The disadvantage of this system is that both the resource provider and the requester have to agree on the same syntax and they cannot extend the terms or concepts without coordination with each other. This kind of asymmetric extension of concepts is possible in our proposed system. For example, as shown in Figure 5, the term *Intel* can be extended to *Pentium* and *Xeon* without coordination between parties, and the correlation system automatically understands the semantics of the new terms.

The work described in this paper [TDK03] is the first effort for ontology-based Grid resource matching devised based on semantic web technology. In this paper an asymmetric description of resources and requests are modelled and described separately. Instead of a syntax-based resource matching, a semantics based matchmaking is proposed. Due to the *asymmetric description*, no coordination between resource providers and consumers is required before a new description vocabulary is added. A simple example for Job Request is given below:

```
JobRequest.Name     "request1"
JobRequest.Owner    "mumtaz"
JobRequest.JobType  "MPI"
JobRequest.RequestResource.ResourceType       "ComputerSystem"
JobRequest.RequestResource.RequiredOS.OSType  "Unix"
```

Based on domain background knowledge and rules specified in TRIPLE [FHH+02], the matchmaker concludes that Linux and SunOS can be used as a Unix operating system, and the requested MPI job can run on any tightly-coupled machine like Linux cluster or a shared memory system. The system is based on RDF-Schema for ontology description. Domain background knowledge and matching rules described in the TRIPLE rule language are explicitly required to perform resource matching. TRIPLE rules are first compiled into XSB rules, which are then further compiled into instructions for XSB virtual machine. TRIPLE/XSB evaluates rules and finds the best match for the request with the help of background knowledge and ontologies. The disadvantage of this system is the overhead of explicit rules definition when the semantics vocabulary increases. Also the recursive rules and two phase compilation is time consuming. The underlying ontology language of this system i.e. RDFS, supports a smaller set of semantics vocabulary as compared to the OWL which is the language of our proposed system. An extension to the standard RDFS is possible but it leads to the requirement of a non-standard specialized reasoner and query constructs. Our proposed system has no such non-standard requirements.

6 Conclusion

In this paper we have proposed an ontology-based resource description, discovery and correlation mechanism which is required for an automatic brokerage service of the Grid resource management system. We described the prowess of OWL and showed that it is a powerful language that quite well suits our needs. We have also demonstrated that OWL-QL can be used as the request response mechanism to discover exact or close matches. The proposed correlation mechanism can be used to describe resources asymmetrically. We demonstrated the effectiveness and highlighted the advantages of the proposed system by providing several examples.

We plan to integrate this correlation mechanism in our previous work about Grid resource management and brokerage (GridARM) [SF05]. This will make the process more sophisticated and will allow the broker to perform resource allocation by semantically interacting with other middleware services. Furthermore, we plan to extend ontologies to cover logical Grid resources and integrate correlation framework in our work about on-demand provision of software components and services, that is *GLARE* [SVHF05] and *Otho* toolkit [HVSF05].

The resource ontology that we proposed will be further refined and developed in the future. Based on real resource matching scenarios tested in Grid environments we will decide if the expressivity of OWL [SvHFJ$^+$], the language we currently use, is sufficient for our needs. A possible future candidate for the ontology language that we will consider is WSML [FB02, dBLK$^+$05]. WSML is a family of language representations for ontologies and services which address the shortcomings of OWL [dBPLF05]. A thorough analysis of WSML will reveal whether it fits our particular needs compared to OWL.

References

[BCM$^+$03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[CFFK01] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid Information Services for Distributed Resource Sharing. In *Tenth IEEE International Symposium on High-Performance Distributed Computing(HPDC-10)*. IEEE Press, 2001.

[CFK$^+$98] Karl Czajkowski, Ian Foster, Nick Karonis, Stuart Martin, Warren Smith, and Steven Tuecke. A Resource Management Architecture for Metacomputing Systems. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 62–82. Springer Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459.

[dBLK$^+$05] Jos de Bruijn, Holger Lausen, Reto Krummenacher, Axel Polleres, Livia Predoiu, Michael Kifer, and Dieter Fensel. The WSML Family of Representation Languages. Working draft, Digital Enterprise Research Insitute (DERI), March 2005. Available from `http://www.wsmo.org/TR/d16/d16.1/v0.2/`.

[dBPLF05] Jos de Bruijn, Axel Polleres, Rubén Lara, and Dieter Fensel. OWL-. Working draft, Digital Enterprise Research Insitute (DERI), May 2005. Available from `http://www.wsmo.org/2004/d20/d20.1/v0.2/`.

[DR03] Brickley D. and Guha R.V. RDF vocabulary description language 1.0. W3C Proposed Recommendation. Available from http://www.w3.org/TR/rdf-schema/., December 2003.

[Fah] Thomas. Fahringer. ASKALON - A Programming Environment and Tool Set for Cluster and Grid Computing. Available from http://dps.uibk.ac.at/askalon, Institute for Computer Science, University of Innsbruck.

[FB02] Dieter Fensel and Christoph Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[FHH+02] Richard Fikes, Patrick Hayes, Ian Horrocks, Stefan Decker, and Michael Sintek. Triple - a query, inference, and transformation language for the semantic web. In *13th Int. Semantic Web Conf. (ISWC 2002)*, number 3342, 2002.

[FHH03] Richard Fikes, Pat Hayes, and Ian Horrocks. OWL-QL- A Language for Deductive Query Answering on the Semantic Web. Technical report ksl 03-14, stanford university, stanford, ca., 2003.

[Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Aquisition, 1993, pages 199-220*, 5:199–220, 1993.

[HVSF05] Juergen Hofer, Alex Villazon, Mumtaz Siddiqui, and Thomas Fahringer. The Otho Toolkit: Generating Tailor-made Scientific Grid Application Wrappers. In *Proceedings of 2nd International Conference on Grid Service Engineering and Management (GSEM'05)*, Erfurt, Germany, September 19-22 2005.

[MTTK02] Paolucci M., Kawamura T., Payne T., and Sycara K. Semantic matching of web services capabilities. In *1st International Semantic Web Conference (ISWC 2002)*, pages 333–347. Springer, 2002.

[OAS] OASIS. UDDI: The Universal Description, Discovery and Integration. Avaibalbe from http://www.uddi.org/about.html.

[RDF98] Resource Description Framework. http://www.w3.org/TR/WD-rdf-syntax, 1998.

[SF05] Mumtaz Siddiqui and Thomas Fahringer. GridARM: Askalon's Grid Resource Management System. In *European Grid Conference (EGC 2005)*, Lecture Notes in Computer Science. Springer Verlag, February 2005.

[SVHF05] Mumtaz Siddiqui, Alex Villazon, Juergen Hofer, and Thomas Fahringer. GLARE: A Grid Activity Registration, deployment and provisioning framework. In *Proceedings of the International Conference for High Performance Computing, Networking and Storage (Supercomputing 2005)*, Washington, USA, November 12-18 2005.

[SvHFJ+] Bechhofer S., van Harmelen F., Hendler J., Horrocks I., McGuinness D.L., Patel-Schneider P.F., and Stein L.A. OWL web ontology language 1.0 reference, W3C Proposed Recommendation. Available from http://www.w3.org/TR/owl-ref/.

[TDK03] H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based Resource Matching in the Grid–The Grid meets the Semantic Web. In *Second International Semantic Web Conference, Sanibel-Captiva Islands, Florida*. October 2003.

[Tea] The Condor Team. http://www.cs.wisc.edu/condor/.

[VR01] Haarslev V. and Moller R. RACER system description. In *Int. Joint Conf. on Automated Reasoning (IJCAR01)*, volume 2083, pages 701–705. Lecture Notes in Artificial Intelligence, 2001.