An Efficient Semantic Discovery of Grid Resources

Shahid Mahmood, H. Farooq Ahmed, Raihan ur Rasool, Kamran Qadir

NUST School of Electrical Engineering & Computer Science
166 A, Street No 9, Chaklala Scheme III, Rawalpindi, Pakistan
{shahid.mahmood, drfarooq, dr.raihan, kamran.qadir} @niit.edu.pk

Abstract— Grid computing provides huge number of hardware and software resources. For efficient discovery of Grid resources, it is necessary to consider resource usage policies during matchmaking process, since the discovered resources are useless if they are unavailable to use. In this paper we propose a framework for efficient discovery of Grid resources. It contains a semantic knowledge base, in the form of domain, functional and policy Ontologies, of resources description. Functional ontology assists to control operations that user want to perform with resources. We use policy ontology to enforce user specified, resource usage policies. Moreover, we also develop a policy analyzer algorithm to enforce these policies in matchmaking process. Our objective is to make prior resource selection efforts more useful by providing maximum surety that the discovered resources are available to the user by analyzing their usage policies. A request subscription knowledge base is also maintained which is used to check the request and send notification later on, if resources are not available. This framework will be evaluated using ProActive Grid Middleware.

I. INTRODUCTION

Grid Computing enables resource sharing and coordinated problem solving in dynamic multi-institutional Virtual Organizations [1]. Resource sharing facilitates to provide and utilize resources available at different geographical distributed locations. Often a scientific experiment requires high computational power and memory that cannot be provided by a single institute. Therefore by using concept of resource sharing, different institutes join together to form dynamic multi-institutional virtual organization for the availability of huge set of resources to their users.

ProActive Parallel Suite is an Open source middleware (OW2 consortium) for parallel, distributed, multi-core computing [2][14]. It supports Grid Component Model (GCM). GCM defined by the Institute on programming models of the CoreGRID project [3], defines a lightweight component model (the GCM) for the design, implementation and execution of Grid applications. The key problematics addressed by the GCM are programmability, interoperability, code reuse and efficiency. This model relies on the Fractal component model [4][5] as a basis for its specification. ProActive/GCM is a reference implementation of the GCM which provides a component framework that aims at fulfilling the needs of Grid programming [6].

The applications that run on a Grid may have certain requirements that can only be satisfied by certain type of resources with specific capabilities. User or agent selects resources according to the requirement of the application. The process of selecting resources according to the requirement of the application or job is called resource matching [7].

Currently available matchmaking system used two types of techniques to find the required resources. The first one is symmetric attribute based matchmaking as in condor matcher [8]. In symmetric attribute based matchmaking resource provider and consumer have to agree on the same name of resource attributes and values. Any change in the attributes name should be known to the consumer. Otherwise consumer can’t find resources efficiently. The exact matching and coordination between provider and consumer makes such system inflexible and difficult to extend to new characteristics and concepts. Moreover, in a heterogeneous multi-institutional environment such as the Grid, it is difficult to enforce the syntax and semantics of resource descriptions. The second one is semantic resource matchmaking [7], where ontologies are used to describe resources. Matchmaking is performed using terms defined in those ontologies instead of exact syntax matching. The loose coupling between resource and request description removes the tight coordination required between resource provider and consumers. Semantic resource matchmaking is more suitable and flexible. Also number of resources discovered by semantic matcher is greater than the symmetric attribute based matching.

However there is one thing that has got less attention in the resource matchmakers i.e. the consideration of resource usage policies, since prior selection efforts may useless if resources are not available or allowed to use.

II. RELATED WORK

An important task in the Grid is to decide what jobs to run on what type of computing resources depending on job or application requirements. Matchmaker plays an important role in this process. In [10] an online semantic matcher service for Grid is proposed that finds resources according to the requirements submitted by the user. But it does not maintain functional information for a grid service which is important in efficient discovery. It also does not provide any information about how to specify policies for resources and how decisions are made on them.

In [11] a semantic component is introduced for Grid environment to describe and discover resources semantically. But resource usage policies are not handled. Also direct match and subsumption match are not applied together as a result of which user have less number of options to execute a job. Whenever user submits a request to semantic component, it does not give any option to the users to select resources of
their own will to execute their job. Functional information of the resources is also not maintained.

A functionality based resource discovery algorithm is proposed in [12]. Our solution is different from it in the sense that, we also consider resource usage policies besides functional information in the matchmaking process which ensures that users get only those resources for which they are permitted to use.

Our objective in applying policies with resource matchmaker is to

- Obtain resources that have maximum surety of availability to the respective user
- Enforce resource reservation according to the specified policies
- Control access to the resources
- Quality of Service through privileges to various types of user
- Control Action that user wants to perform with the help of resources.

III. ARCHITECTURE OF THE PROPOSED FRAMEWORK

We proposed a framework for efficient discovery of resources in Grid environment. Main components of this framework are Grid Resource Information Collector, Description Converter, Request Handler and Resource Matchmaker which consists of Discover Resources and Policy Analyzer components. The architecture is depicted in figure 1. We discuss these components in detail.

A. Grid Resource Information Collector

Grid Resource Information Collector (GRIC) receives resources advertisements from resource providers and also makes a subscription for them. The purpose of subscription is to get latest information of the resources (such as current CPU load, free main memory etc.) later on, if resource providers do not provide it after a specific time. GRIC sends agent to collect the updated information. If one agent is sent per node then it puts much load on GRIC in sending and receiving agents. To make the latest information collection process more efficient a single agent is used for a number of nodes. This agent has a list of nodes from which it collects latest information of resources and then gives it to GRIC component.

B. Description Converter

Description converter retrieves resources information from GRIC component. Description converter converts them into ontologies. Inference tools allow reasoning about Grid ontologies to provide powerful and flexible information. Three types of ontologies are formed form the resource description which are saved into the knowledge base. These are domain, functional and policy ontologies.

1) Domain Ontology: Domain Ontology comprises the concepts of different resources. It contains information of two different types of resources which are software (Grid Services and Softwares) and hardware (e.g. CPU, Storage, RAM).

Protégé API is used to generate the OWL DL instances of this ontology.

2) Functional Ontology: Only inputs, outputs, precondition and effects information are not enough for the efficient discovery of Grid services [12].

Functional ontology consists of functional information of Grid services i.e. what function they provide. It also helps in clearly distinguishing between services that have similar number of inputs and outputs. By using functional ontology user required resources can be easily discovered by considering required number of inputs/outputs and functionality.

![Fig. 2 Functional ontology](image)

Fig. 2 Functional ontology

![Fig. 1 Framework for efficient discovery of grid resources](image)

Fig. 1 Framework for efficient discovery of grid resources
Policy Ontology: Whenever resource providers advertise their resources they also advertise usage policies with them. Usage policies have the information of who are allowed to use the resources, when these resources will be available, under what conditions and what function users can perform with them. Policies are specified as a combination of (where, what, who, when, action)

Or
(grid.domain.cluster.computer, resources/services, VO/groups/users, time/day/week/month/year/condition, functions )

These three types of ontologies: domain, functional and policy ontologies are saved in the knowledge base. Whenever user requests for some resources reasoning is performed on this knowledge base to find the match.

C. Request Handler

Request handler filters out requirements and submits to resource matchmaker. It handler also maintains a knowledge base of user requests. Each request may have a validity period for which it will be saved in the knowledge base, so that it can be checked later on for the availability of resources if resources are not available. On receiving resources from matchmaker, request handler sends notification to resource user.

D. Resource Matchmaker

The purpose of matchmaker is to find resources according to the user requirements. We apply exact match, plug-in match, subsume match, exclusive match (when request and advertisement are in ontology hierarchy excluding child and parent) and fail match criteria in the matchmaker. Our matchmaker consists of two components, Discover Resources and Policy Analyzer.

1) Discover Resources: Discover Resources. This component takes query from request handler component and performs reasoning on the knowledge base. Pellet reasoning tool is used for the reasoning. In discovery process functional information is first searched in functional ontology and then domain information is retrieved from domain ontology. After this policies of the discovered resources are retrieved with respect to user from the policy ontology. The discovered resources with their policies are then given to the Policy Analyzer. Resources returned by the policy analyzer are then ranked by the Discover Resources component.

2) Policy Analyzer: Resource providers specify usage policies on their resources. These usage policies may be different for different type of users or groups. A Policy Analyzer Algorithm takes discovered resources and their usage policies information from Discover Resources component. After that it analyzes and enforces provider specified policies with respect to user. This policy analyzer algorithm realizes all our objectives that we have mentioned in related work section. It returns only those resources that provide user maximum surety that they are available to use. This helps to save the time wasted in submitting job to the resources that are not available or allowed to the user. Policy Analyzer also assists resource reservations. It does not return those resources that are reserved for other people at that time. Another benefit of the policy analyzer is that it also facilitates in controlling the action that a user can perform with the help of grid service/resource. A grid service/resource may provide different functionality and provider can specify policies for its different function for different users. Policy analyzer enforces them and does not return a resource for which user required action is not permitted. Pseudo code for policy analyzer algorithm is shown in figure 4.

![Fig. 3 Policy ontology](image)

![Fig. 4 Policy analyzer algorithm](image)
Where DR denotes Discovered Resources and UP denotes Usage Policies. PolicyAnalyzer function takes resource and checks its each usage policy; if user satisfies all of them then it returns available i.e. 1 otherwise unavailable i.e. 0. The end result of the formula in equation (1) is the total number of resources, in the discovered resources, that are unavailable for a particular user.

IV. SYSTEM EVALUATION

This framework is developed for ProActive Grid middleware. As a case in point two types of users are considered. User U1 is using Semantic matchmaker where as U2 is using our system (Semantic with Usage Policies), which not only considers semantic information in the matchmaking process but also resource usage policies that are specified by the providers. Both users submit same type of requests to their matchmakers. They submit 12 different queries for finding hardware resources to their matchmakers. The result of queries in terms of retrieved and relevant node is shown in the following graph. SB stands for Semantic Based and SUB stands for Semantic with Usage Policy.

![Fig. 5 Retrieved and relevant node in SB and SUP systems](image)

Semantic Based system always have less number of relevant resources in the retrieved resources because some of them are unavailable to use (due to their provider’s specified usage policies) and they are not considered in the relevancy results. Whereas our system (Semantic with Usage Policies) has more relevant resources since it not only uses semantics but also enforces resource usage policies and quality of service in matchmaking process.

The Precision [14] in Semantic Based and Semantic with Usage Policies can be calculated with the help of following formula

\[
Precision = \frac{\text{Relevant Resources} \cap \text{Retrieved Resources}}{\text{Retrieved Resources}} - -(2)
\]

Therefore precision for existing semantic based and proposed Semantic with Usage Policy are as follows

<table>
<thead>
<tr>
<th>Precision (Semantic Based)</th>
<th>Precision (Semantic with Usage Policy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6588 %</td>
<td>0.8709 %</td>
</tr>
</tbody>
</table>

This shows that Semantic Matchmaker with Usage Policies has more precision than non usage policy based Semantic matchmaker. The reason is, semantic matchmaker with usage policies returns only those resources that satisfy provider specified resource usage policies and are available to use

V. CONCLUSION AND FUTURE WORK

In this paper we presented a framework for efficient semantic resource discovery in Grid which is different from the conventional system in the sense that it is also enforcing resource usage policies. Using our system discovered resources has maximum surety to the user to their availability which prevents wastage of time (as by using conventional system) in submission of job to the resources that are not available or allowed to the user. As a future work we intend to add the feature of direct submission of user job with discovered resources to the ProActive [14] middleware.

REFERENCES


