A Framework for Semantic Grid Service Discovery

Fei Liu, Fan-yuan Ma, Ming-ju Li, and Lin-peng Huang

Department of Computer Science and Engineering, Shanghai Jiaotong University,
Shanghai, P. R. China, 200030
{fliu, ma-fy, li-ml, huang-lp}@cs.sjtu.edu.cn

Abstract. Grid computing provides key infrastructure for distributed problem solving in dynamic virtual organizations. It has been adopted by many scientific projects, and industrial interest is rising rapidly. Compared with traditional single computer system, effective service locating in Grid is difficult because of huge amount and wide-area distribution of dynamical services. This paper presents a new Grid framework that discovery services through the overlay network based on Grid service description semantic vectors generated by Latent Semantic Indexing (LSI) [1]. This framework uses WSRF as its base architecture and regards each Grid node as a register node to publish service that enables it has better scalability. Comparing with other Grid services discovery techniques those are based on simple keyword matching, our framework has better accuracy for it considers the advanced relevance among Grid descriptions.

1 Introduction

Grid computing technologies enable wide-spread sharing and coordinated use of networked resources [2]. Grid evolve around sharing of networks, computers, and other resources. They provide support for services, which are distributed over multiple networked computers known as clusters. Because Grid computing is service oriented computing, service discovery is important for Grid. A service in this context may be defined as a behavior that is provided by a component for use by any other component based on a network-addressable interface contract (generally identifying some capability provided by the service). A service stresses interoperability and may be dynamically discovered and used. According to [3], the service abstraction may be used to specify access to computational resources, storage resources, and networks, in a unified way. How the actual service is implemented is hidden from the user through the service interface, hence, a compute service may be implemented on a single or multi-processor machine. However, these details may not be directly exposed in the service contract. The granularity of a service can vary, and a service can be hosted on a single machine, or it may be distributed.

Owning to these characteristics, traditional centralized service discovery services are difficult to be used in Grid. So we propose a new Grid framework that discovery services through the flat, fully decentralized overlay network. Most of Grid service discoveries are based on simple keyword matching, ignoring advanced relevance between queries and Grid descriptions. The service discovery in this paper is based on Grid description and queries semantics vectors generated by Latent Semantic
Indexing (LSI). A semantic discovery process relies on Grid service description, containing high-level abstract descriptions of service requirements and behavior.

The rest of this paper is as follows. Section 2 discusses related works. Section 3 introduces the Latent Semantic Index and Section 4 proposes the Grid service discovery framework based on semantic overlay networks. We conclude in Section 5 with lessons learned and future work.

2 Related Works

Service discovery has played a crucial role in the evolution and deployment of distributed systems. Early distributed systems comprised collections of components (e.g. client/server or object-oriented) that were implicitly linked through function names, or linked through TCP/IP-based host and port addresses. Federated domain name servers (DNS) simplified and abstracted the use of these numeric addresses by providing a registry-based mechanism for locating the hosts. JINI [5] used a similar approach as part of its Java-based distributed infrastructure. Classes exposed and published their interfaces as proxy objects with a JINI discovery service. By searching for a given class-name, matching proxy objects could then be retrieved and invoked, which would in turn invoke the remote service. Whilst providing a mechanism whereby services could easily be added, removed or replaced within a system, this approach was based on an assumption that there was a shared agreement about what a given service type was called (i.e. its class name) and that there was an agreed and well defined interface. Other distributed technologies support similar principles, including DCOM, Corba, XPCOM, etc.

Web Services extend the idea of JINI services by relaxing several assumptions. Built upon web technologies, Web Services are declared in XML and utilized Web-based protocols to publish and retrieve XML documents. The Simple Object Access Protocol (SOAP) [6] provides a transport mechanism to shuttle XML content between services or applications. The Web Services Description Language (WSDL) [7] explicitly defines the interface of a service. By adhering to these definitions, services can be produced that automatically publish WSDL descriptions that in turn are used to define the content of SOAP messages, and thus simplifying the development of interoperable components.

However, in order to utilize these published services, developers must first locate them. Unlike JINI, Web Services do not belong within well defined class hierarchies, and thus it is not feasible to locate services through class labels. Instead, the UDDI service directory [8] provides a mechanism whereby service providers can register information about themselves, such as physical address, contact phone number, etc; the types of business they are involved in; and references to the service descriptions. UDDI registries provide this information in response to white-pages queries (i.e. given the name of a service provider, what are its detail) and yellow-pages queries (i.e. what service providers provide services that belong to a given pre-defined service type). Based on a set of queries, developers are able to browse through a list of service descriptions to locate a desired service. However, little support is provided for searching for a service based on a capability or user defined data.

Technical Models (tModels) support the specification of additional attributes that can be associated with objects stored in the UDDI repository. In their most common