Layered Resource Representation in Grid Environment: An Example from VEGA Grid

Fangpeng Dong, Yili Gong, Wei Li, and Zhiwei Xu

Institute of Computing Technology of CAS
Beijing China, 100080
{fpdong, gongyili, liwei, zxu}@ict.ac.cn

Abstract. Resource discovery in wide-area distributed environments is a great challenge for the large quantity and volatility of resources. To solve this problem, mechanisms effectively representing resources are very necessary. In this paper, a resource representation scheme is proposed, which embodies the principles of VEGA Grid. This scheme is designed for routing-transferring model that is implemented as the resource discovery mechanism in VEGA Grid. The scheme is based on a three-layer model: the top user layer, on which specialized representation can be deployed according to the demand of users; the middle router layer, on which resource routers work for globally resource discovery; and the bottom provider layer for providers to publish their resources. We explain our choices of resource representations in each of these layers, and evaluate the usability, scalability and performance of the approach.

1 Introduction

The dramatic increase in demand for resource sharing and cooperating in wide-area environments motivates the emergence of new technologies, e.g., computational Grid [1] and Web Service are invited to enhance support for e-science and e-business. Although different in architectures, these technologies are confronted with resources having features of large quantity, wide distribution, mobility and multiple policies.

In VEGA Grid, we adopt a main idea of constructing Grid as a virtual computer in wide area. VEGA Grid makes a mapping between the Grid system and the traditional computer system. We use the virtual computer concept to build the VEGA Grid on legacy systems. Although the architecture adopted by Vega Grid is very simple, it can solve the problem of resource sharing and efficient cooperating. The benefit of the mapping method used here is that we can borrow many mature technologies from the traditional computer architecture design. Many important ideologies and theories are helpful to our design, but here we only introduce those related to resource discovery mechanism used by VEGA:

A *Grid processor* is an abstraction of a resource consumer that controls the activities of Grid virtual hardware [2]. A Grid processor is a logical concept, and it can be either software or special hardware.

A *Resource router* [2] is a component in Grid system bus [2]. The basic idea of the resource router is the routing-transferring method, that is, a resource router can

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2659, pp. 1133–1142, 2003.

choose a route for a resource request and forward it one by one until the request reaches a Grid virtual device that provides the proper resource.

A *Grid virtual device* is an abstraction of a resource provider in Grid. In our design, different resource providers are viewed as different virtual devices that provide various services. For each virtual device, there is a specified *virtual device driver* installed in Grid process end to provide necessary accessing interface to the device.

In an overview, VEGA Grid is constructed as Fig. 1.

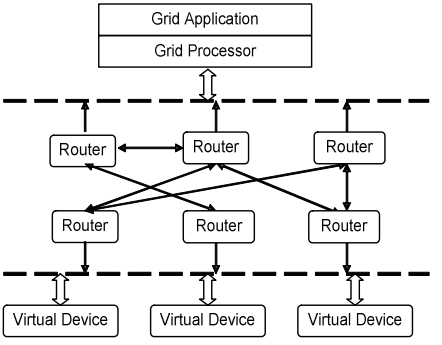

Fig. 1. The layered structure of Vega Grid virtual computer.

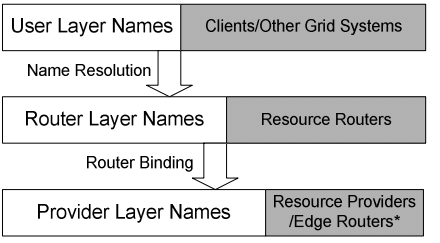

Fig. 2. The VEGA resource representation stack. *: Routers directly connected with resource providers are called edge routers.

A resource discovery process is a series of operations initiated by the top Grid Processor layer requests and completed by the bottom Virtual Device layer responses. To adapt this layered structure of VEGA Grid, we take a natural way to represent resources in three layers:

Grid applications in Grid processors are various in resource using patterns and representations, e.g., a Globus application which gets resource information from MDS [4] in LDAP is very different from a Web Service application which gets it from UDDI [12] in XML. So, if we want to integrate applications from different systems into ours, the Grid processor layer resource representations should have the capability of customizing according to application requirements. In other words, the Grid processor layer representations are not fixed but changeable. To make the definition generally understandable, we call representation in this layer user-layer name.

To locate resources required from Grid processors in a wide distributed environment cooperatively, resource routers need a global uniform mechanism to identify resources with different attributes. When the performance of routers is considered, resource representation in this layer should be stored, transferred, and identified efficiently. Here, we call resource representation in this layer router-layer name, a sequence of integral and string values.

In the bottom layer of VEGA, resource providers need a location-aware mechanism to make their resources accessible. Representation of resources at this layer is called provider-layer name. Thus, resource representation is also organized into three layers and forms a stack shown in Fig. 2. We also give the mapping between layers and typical roles working on each layer.

The rest of this paper is organized as follows: in the second section, we introduce related works. The third section explains VEGA principles, which guide our work. The fourth section gives a brief explanation for the resource router architecture, analyzes its requirements and describes our naming method in details. We report the implementation of this method in VEGA Grid in the fifth section. At last, we draw a conclusion and point out further work.

2 Related Works

In Grid environments, two strategies are mainly used to identify resources: in some cases, resources are explicitly named using either names or identifiers; some other systems provide catalogs that allow users to describe resources using attributes, and by querying these catalogs, users can select resources with particular properties [6]. The former is an appealing idea for its simpleness, but it makes the resource discovery a painful mission for its hardness: resource attributes' volatility is difficult to express [13] [14]. The latter is based on the attribute-value-matching resource discovery mechanism and it works well in an environment in which resource types are relatively stable and few, but it is difficult to define and maintain numerous types of resources and serve the matching process in a wide-area environment.

Secure Grid Naming Protocol [5] defines a scheme for location-independent logical naming of grid resources. A SGNP name is a Location-independent Object Identifier (LOID) that uniquely identifies a Grid resource. The actual position of a Grid resource is located by associating the LOID with one or more communication protocols and network endpoints. In SGNP, a LOID is hierarchically resolved, and the result of resolution is the client-to-resource binding, which means wherever a resource moves, it must connect to the same *Domain Resolver* and *Binding Resolver*.

GLOBE [9] realizes a system that assigns resources location-independent names that are resolved to contact addresses by a location-service as a means for retrieving mobile resources. In [10], the author argues the scalability problems of present object reference and symbolic names, and discusses the solutions in Globe.

In UDDI, information about business entities and services are classified in White page, Yellow page and Green page. Each data element has a globally unique primary key, for example, a business identifier can be a D&B number, tax number, or other information types via which partners will be able to uniquely identify a business.

Systems mentioned above are examples that adopt resembling ID-based resource naming and discovering in Grid environments. Name resolution is popular in these systems, and a service to locate a resource entity is required to map location-independent resource identifiers to physical resources.

Condor's Matchmaker [7] is a typical distributed resource sharing system that does not use global names for resource discovery: resource description and request are sent to a central server responsible for matching the advertisement between the resource requestor and provider. This architecture is based on the assumption that an organization will be willing to operate the central server in wide area.

Another relevant experience comes from Globus MDS, which defines for each object type an object class specification declaring the required and optional attributes of a particular object. GGF has proposed a Grid Object Specification (GOS) [8] draft to formalize object class definitions. Although MDS avoids name resolution, user-

visible name space is limited: object class names are globally defined and universal, that is, all Grid users can only see one common name space. Systems adopting other name schemes can hardly merge into MDS. Further, the information service is configured into a tree-like architecture, thus restricts both the resource representation and discovery process flexibility.

3 Naming Principles in VEGA Grid

In our designing of VEGA Grid architecture, we propose the following principles called *VEGA* to evaluate our designing work. These principles are also embodied in the resource representation and discovery approach.

Versatile Services. The Grid should be constructed as an infrastructure that provides a developing and running environment supporting various applications, using patterns, platforms and compatibilities. To achieve this goal, resource representation at application level must be flexible enough to adapt various name schemes and using patterns.

Enabling Intelligence. The Grid should have the ability to delegate users to make decisions in some circumstances. Given some polices, the Grid also should have the ability to aggregate, produce or filter information about itself. Resource router is such a component, which delegates users to find resources of interest and collects resource status in the Grid. This principle also requires support to friendly user patterns.

Global Uniformity. To work corporately and find resources globally, representation at router layer must be globally uniform. Additionally, Global uniformity gives resources the ability to join the Grid anywhere, anytime and its accessibility can be promised.

Autonomous Control. The Grid should not be governed by a central administration. For the large quantity and wide distribution, resources should not be named and resolved centrally in order to gain promising performance and scalability as the growth of Grid.

4 Resource Representation Scheme

In VEGA Grid, resource routers are deployed to address the resource registry and discovery problems. Resource routers take account of the merits of resource attributes, and adopt a proper routing policy as well as an information-aggregating algorithm to overcome some performance drawbacks in the attribute matching strategy, such as the directory service. Proper representation schemes in multiple layers play an important role in promoting routers' performance, giving upper users a friendly way to access Grid resources and facilitating resource providers to make their resources accessible.

4.1 Resource Representation in User Layer

In this layer, our goal is to provide mechanisms to applications/users so that they can join VEGA and utilize resources conveniently. Guided by the principle of *Versatile Service*, it is not applicable to deploy resource name space uniform in this layer, but changeable according to the requirements of different applications. In other words, users should have the ability to customize their own resource representations in the Grid processor they connect to, so it is hard to give a generic view of user-layer representations. But some examples may be helpful to demonstrate the features of this mechanism and discuss related problems. We have defined a naming scheme for user querying computing resources that we deployed in VEGA prototype. The syntax of this scheme is as following [15]:

```
<name>::=(<class_name>|<class_id>)["?" <specification>]
<class_name>::=<string>
<class_id>::=<number>
<specification>::=(<attribute><op><value>){<op>(<attrib
ute><op><value>)}
<attribute>::=<string>
<value>::=<string>|<name>
<op>::="&gt;"|"="|"<"|"&gt;="|"/="|"<="|AND|OR|NOT
```

Using this scheme, a user can query resources he wants by giving a "name" like:

```
ComputingResource ? (CPU > "933MHz") AND (memory =
"256M") AND ((OperatingSystem ? (type = "Linux")
```

GOS [8] provides another example of user layer resource representation. An instance of GridCompute- Resource maybe looks like:

```
dn: hostname=burns.csun.edu, o=Grid
objectclass: GridComputeResource
hostname: burns.csun.edu
.....
diskDrives: /dev/dsk/dks0d4s
```

Although friendly to users, resource representation in these formations can hardly be used by resource routers because of the complexity. Moreover, it is not necessary for routers to know the meaning of each attribute-value pair. A mechanism is required to map the user-layer representation to the router-level representation. This process is similar to the domain name resolution in Internet But it is obviously too costly to deploy a specialized global "Name Resolver" for each type of representation in the Grid. Fortunately, most of representations in this layer are defined by some globally-obeyed specifications in which the meaning of each attribute is stated. This means if we understand these specifications, we can locally resolve the representations they have defined into any formation we want to. In VEGA Grid, virtual devices that want to support a particular resource representation should implement the resolver which maps the user-layer representation to the router-layer representation. These resolvers are parts of the *virtual device drivers* deployed in Grid processors. This mechanism does not exclude the adopting of the global name resolution. Resolvers like DNS can be integrated into VEGA as a special kind of services whose locations are static indicated by their virtual drivers and need not to be located via resource routers.

4.2 Resource Representation in Router Layer

In the middle layer of the architecture, resource routers work corporately to locate resources globally, receive registry from resource providers and collect resource information from neighbors. To support attribute-value based queries from users, routers construct their common name space based on resource types. This is not difficult to understand because attribute-value pairs are defined according to different resource types. This layer is the kernel of our model, for the following reasons: 1) to work globally, router-layer names are uniform in the whole Grid, that is, in this layer, the representation formation of a type of resources is immutable and can be recognized by all routers; 2) the quantity of router-layer representation formations finally determines the actual number of resource types that can be supported by the Grid, which influences the scalability of the whole Grid.

Router-layer Resource Representation Syntax. As mentioned in Section 4.2, user-layer representations will be mapped into router-layer ones when users submit their requests to routers. Different from users, routers do not concern the meanings of attributes. In the user-layer representations, matching patterns are added to attribute-value pairs, e.g., CPU>933MHz. Routers must use these matching patterns when they decide whether or not a particular resource can satisfy a user's request. Therefore, the router-layer representations converted by resolvers in virtual device drivers at Grid processor ends should contain matching operations. The syntax in BNF is:

```
<name> ::= (<class_id>) ["/" <specification> ]
<class_id> ::= <number>
<specification> ::= [<op> <value>] {"/" <op> <value>  }
<value> ::= <string> | <name>
<op> ::= ">" |"=" | "<" | ">=" | "/=" | "<="|AND|OR|NOT
```

While representations collected from neighbors and resource providers only indicate the status of resources, which are independent from matching operations. These representations are recorded in routers' routing tables, and refreshed periodically. The syntax is:

```
<name> ::= (<class_id>) ["/" <specification> ]
(<direction>)
<class_id> ::= <number>
<specification> ::= [<value>] {"/"<value>  }
<direction> ::= <URL>
<value> ::= <string> | <name>
```

*class_id*s of each resource type are assigned by a global administration, and form a hierarchical global name space. So, it supports the naming service well both in management and in scalability. A naming service in wide-area distributed system should have the mechanism to generate a globally unique name as locally as possible. In hierarchical architecture, the namespace can be divided into different-leveled domains, each of which administers a portion of the whole namespace with local autonomy (DNS is an example of such systems). The naming service in each domain can be assigned a locally unique name. This name with the path from the root down to the domain node constitutes a globally unique name. In a hierarchical namespace, the

increase in resource types and quantity can be distributed to different domains. The probability of a single node taking all the penalty of system growth is quite low, so a bottleneck can be avoided. This brings the namespace good scalability.

Key Features of the Routing-Transfer Model. As the backbone of VEGA Grid, the resource router plays a vital role and deserves explanation in details. Here, three key features of routers will be discussed: the structure of routing tables, the updating policy and the request routing algorithm. These features bring resource routers good performance at a relatively low cost [3].

The routing table of a resource router is hashed with resources' *class_ids* used as the variable of the hashing function. From the hash table entries, aggregated resource information can be accessed according to the formation of the resource type. An important character of a routing table is that it does not have the information about all of resource instances in the Grid. This is because a router does not record the physical location of a resource instance, but records to which direction, can it find a resource instance having those attribute values it records. That is, no matter how many resource instances of the same type and equal attribute values, a router only records one of them, and usually, it is the one "nearest" to the router. Therefore, theoretically, the size of a route table is independent of the scale of resource quantity, but related to the quantity of types and instance states. For example, if there are k types of resources, each type of resource has l attributes and an attribute has m possible values, the spacial complex of the routing table will not exceed $k*(l^m)$ items.

Another factor influencing on routing tables is the updating policy. In our implementation, resource instance entries not updated after a specific interval will be regarded unavailable and removed from the routing table so that the size of the routing table will not grow too large as time goes by. To keep the performance as good as possible, when a nearer resource instance is known, the corresponding item in the table will be updated to make the direction value point to the new router.

VEGA Grid resource routers adopt a routing algorithm called SD-RT (Shortest Distance Routing-Transferring) [3]. SD-RT algorithm is designed to locate a resource as soon as possible. When a resource router receives a resource request, SD-RT algorithm will choose a router by which the resource is accessible through the shortest path from the local resource router. Especially, when the distance is 1, the resource provider is the neighbor of the router. The whole routing process is the process of mapping from the location-independent router-layer representations to location-aware provider-layer ones. Using SD-RT algorithm, a resource request can arrive at the resource provider as soon as possible. If there is more than one provider supplying the same resource, SD-RT algorithm can guarantee the request arriving at the provider nearest to resource requestors.

4.3 Provider-Layer Names

The router, which a resource is registered to, needs the location information for client-to-resource binding. So, when a provider registers its resource to a neighbor router, the location information should also be delivered. That is, the resource providers and the routers, to which providers register their resources, need a location-aware mechanism to represent location information. When a resource is replicated to

different addresses, it is also required to distinguish these replicas. Location information can be a key attribute in the resource description [8]. In our model, we put it in the resource identifier to form the bottom layer name. The reason is, for some special resources that can be accessed only via IDs, e.g., a particular file that has only one copy in the Grid, we can leave their attributes null to omit router processing. Representations in this layer are formed by binding the location information, such as URLs, to the resources' router-layer. The syntax in BNF is:

```
<name> ::= (<class_id>) ["/" <specification> ]
(<location>)
<class_id> ::= <number>
<specification> ::= [<value>] {"/" <value>  }
<location> ::= <URL>
<value> ::= <string> | <name>
```

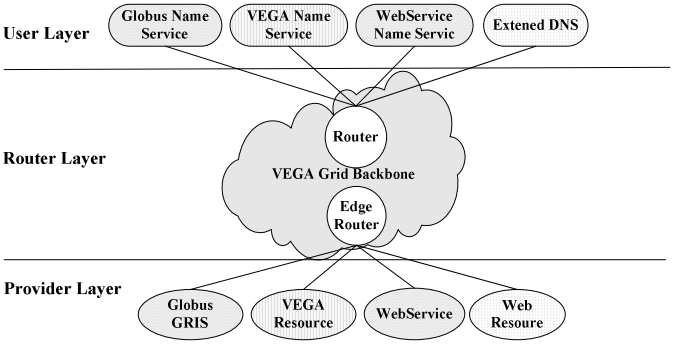

Fig. 3. Various resources integrated in VEGA Grid.

5 Evaluation

Resource representation is a very important topic in Grid environments. It has vital effect on both the systems and users. In this section, our approach is evaluated from three respects: usability mainly concerned by resource consumers, scalability mainly concerned by resource provider, and performance concerned by the whole system.

Usability: The explicit separation of the user layer from the router layer facilities the deploying of the variable user customized representations. By implementing specialized name resolvers plugged into virtual device drivers, user-layer names can carry enough information to describe users' requirements. This improves the usability, and also makes other Grid systems merge into VEGA easily. When another Grid system having its own naming scheme wants to join VEGA Grid, it only needs to deploy a new name resovler that converts its local names to VEGA's router-layer representations. Because the original name space is reserved, applications based on the former system can be easily replanted in VEGA, which accommodates VEGA's versatile service principle. Fig. 3 gives a view of other Grid systems integrated with VEGA. UDDI in Web Service and GIIS (Grid Information Index Service) in Globus

can be replaced with resource routers. By installing VEGA virtual device drivers, Globus and Web Service applications can run on VEGA Grid infrastructure.

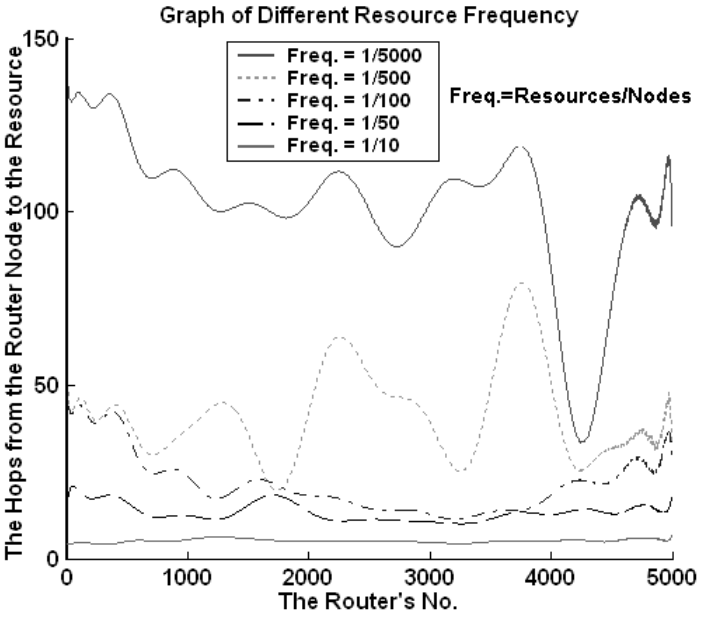

Fig. 4. The x Axis gives router IDs from 1 to 5000, and y Axis shows the hops needed to access a particular type of resources from each router under different resource frequencies .

Scalability: Three factors enhance scalability of our scheme. 1) Location independence: The resource representations in the upper two layers are not bound to resources' physical locations, and it is the routers' responsibility to locate the proper resources. Therefore, resources can transfer and be replicated in wide area without imposing additional restrictions to users and providers. 2) Resource router features: As analyzed in 4.3, the size of routing table is independent of the quantity of resource instances in the Grid. This means the scalability in resource instance quantity will not be limited by resource routers' capacity. Additionally, the resource type name space is constructed hierarchically, which also benefits resource type's scalability. 3) The ability of top-layer namespace partition relieves the burden of name resolution and allows VEGA to expand by merging other Grid systems.

Performance: We do beneficial work in both user layer and router layer to promote the performance of resource discovery. In the user layer, remote name resolution, which is usually a bottleneck of wide-area distributed systems, is avoided. Local resolvers based on resource representation specifications are implemented in virtual device drivers. In the router layer, resource information routing-transferring model provides good performance to locate resources in environments that have large scale of resources and nodes. Fig. 4 shows the experiment results [3] in an environment consisting of 5000 nodes, where resource frequency varies from 1/5000.to 1/10

6 Conclusion and Future Work

In this paper, we analyze the problems of resource representation under the guidance of VEGA Grid's principles, and introduce a three-layer representation scheme adapting the resource information routing-transfer model. We also give examples of syntaxes adopted by each layer. The scheme is accordant with VEGA's principles of versatile service, enabling intelligence, global uniformity and autonomous control. Additionally, it has good usability, scalability and performance.

Our future work includes efforts in further verifying the efficiency of our approach in larger and more complex environments, as well as improving resource routers' performance. Research on the representation syntax of each layer is also an important content of our work.

References

1. I. Foster and C. Kesselman (Eds): The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, 1998.
2. Wei Li, Zhiwei Xu, Bingchen Li and Yili Gong: The Vega Personal Grid: A Lightweight Grid Architecture, PDCS 2002.
3. Wei Li, Zhiwei Xu, Fangpeng Dong and Jun Zhang: Grid Resource Discovery Based on a Routing-Transferring Model, 3rd International Workshop on Grid Computing (Grid 2002).
4. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman: Grid Information Services for Distributed Resource Sharing, Proc. of HPDC-10, IEEE Press, August 2001.
5. Joshua Apgar, Andrew Grimshaw, Steven Harris, Marty Humphrey and Anh Nguyen-Tuong: Secure Grid Naming Protocol (SGNP), Avaki Corporation, University of Virginia, GGF Working Group Chairs and Steering Group, January, 2002.
6. Ann Chervenak (ISI/USC): Naming and Information Management in Grid Systems, http://www.sdsc.edu/GridForum/RemoteData/Papers/chervenak.pdf.
7. R. Raman, M. Livny and M. Solomon, Matchmaking: Distributed Resource Management for High Throughput Computing, Proc. of IEEE Intl. Symp. High Performance Distributed Computing, July 1998.
8. Steven M. Fitzgerald, Gregor von Laszewski and Martin Swany: GOSv2: A Data Definition Language for Grid Information Services, Feb. 2001.
9. Maarten Van Steen, Philip Homburg and Andrew S. Tanenbaum: Globe: A Wide-Area Distributed System, IEEE Concurrency (1999), pp. 70–78.
10. G. Ballintijn, M. van Steen and A.S. Tanenbaum: Scalable Naming in Global Middleware, Proc. 13th Int'l Conf. on Parallel and Distributed Computing Systems.
11. M. van Steen, F. Hauck, P. Homburg and A.S. Tanenbaum: Locating Objects in Wide-Area Systems, IEEE Commun. Mag., vol.36, nr.1, pp.104–108, Jan. 1998.
12. UDDI Technical White Paper, http://www.uddi.org.
13. Adriana Iamnitchi and Ian Foster: On Fully Decentralized Resource Discovery in Grid Environments, GRID 2001, 2nd International Workshop on Grid Computing.
14. Amin Vahdat, Michael Dahlin, Thomas Anderson and Amit Aggarwal: Active Names: Flexible Location and Transport of Wide-Area Resources, Proceedings of the 2nd USENIX Symposium on Internet Technologies & Systems, October 11–14, 1999.
15. Yili Gong, Fangpeng Dong, Wei Li and Zhiwei Xu: A Dynamic Resource Discovery Framework in Distributed Environments, Proc. of International Workshop on Grid and Cooperative Computing, (GCC 2002).