A Resources Virtualization Approach Supporting Uniform Access to Heterogeneous Grid Resources

Cunhao Fang[1], Yaoxue Zhang[2], Song Cao[3]

[1] Tsinghua National Labatory of Inforamation Science and Technology
[2] Department of Computer Science and Technology,Tsinghua Univeristy
[3] School of Software, Tsinghua University
100084 Beijing, P. R. China
fangch@tsinghua.edu.cn

1 Introduction

Grid system has various kinds of resources such as computing resources, storage resources, instrument resources, data resources, etc. However, because of the differences of formats, descriptions, structures, and access modes of these resources, grid computing fails to access these resources uniformly and make full use of them. How to organize and manage all sorts of resources as a whole In Grid Environment, and provide the upper application with coherent description as well as uniform access interface is the problem of coherency access of different resources In Grid Environment.

Resources virtualization makes better use of the dynamic and distributed service resources under grid environment. It is a proper method to solve the problem described above.

Web service is a set of protocol warehouse defined by XML. Through protocols and standards such as SOAP，WSDL，UDDI，WSFL，BPEL4WS, it provides uniform service registry, discovery, binding, and integration mechanism facing Internet. Web service has become an important mechanism of resources mutual manipulation underlying extensive environment.

This research starts with various physical resources In Grid Environment, transforms and encapsulates various resources into services on the basis of Web service technology, and further provides upper applications with uniform service resources through uniform description and management of these services. Based on this, multi application cases with the same function body are merged, different function bodies are analyzed, and services resource view on the user function layer is extracted in support of service logic integration process on the user end. The whole virtualization architecture consists of three ***Views*** and three ***Processes***, as shown in Fig.1.

* This paper is sponsored by National Natural Science Foundation of China (NO. 90604027)

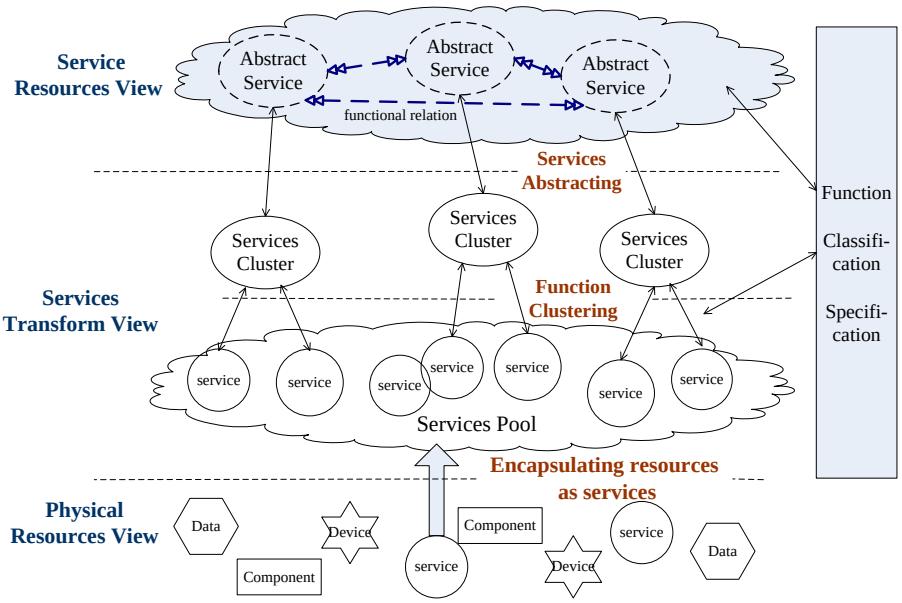

Fig. 1. Layered Architecture of the Framework

The remainder of the paper is organized as follows. Related definitions are presented in Section 2. Section 3 introduces in detail the process of resources virtualization and organization, followed by conclusion in Section 4.

2. Some Definitions

Grid system has various kinds of resources such as computing resources, storage resources, instrument resources, data resources, etc. R refer to the resources pool, all resources in the grid system are in it. Suppose there are n kinds of resources in the resources pool, they are R1， R2， …， Rn.

Definition 1:

Resource $\boldsymbol{R_i = (Category, F_i, I_i, P_i)}$

Category is the kind of resource $\boldsymbol{R_i}$;

$\boldsymbol{F_i}$ is the functional description of resource $\boldsymbol{R_i}$;

$\boldsymbol{I_i}$ is the interface of resource $\boldsymbol{R_i}$ (such as calling parameter and running results, etc.);

$\boldsymbol{P_i}$ indicates the using policy set of resource $\boldsymbol{R_i}$.

We use deputy mechanism to demonstrate different kinds of services through service uniformly. With regard to services, we have the following definition:

Definition 2:

Service $\boldsymbol{S_i = (Name_i, BaseInfo_i, Interface_i, Binding_i, Function_i, AD_i)}$

$Name_i$ is service identification;

$BaseInfo_i$ is basic non-behavior information of the service including name, author, version, manufacture, etc;

Interface$_i$ is the interface of service $\boldsymbol{S_i}$ (such as calling parameter and running results, etc.);

Binding$_i$ is binding protocol set of the service;

Function$_i$ is function description of the service;

AD_i is application domain of the service.

3. Resources virtualization and organization process

As Fig. 1 shows, resources virtualization and organization composes of three ***Views*** and three ***Processes***:

Three ***Views:***

1) Physical Resources View: this view refers to various Grid resources and corresponding supporting tools (e.g. resource deploying tools).

2) Services Transform View: this view aims to implement the uniform resources transform (to Web Services format) and to organize the structure of transformed ***Services Pool*** by analyzing, categorizing and managing these service resources according to ***Function Classification Specification (FCS)***.

3) Service Resource View: this view aims to provide users with well-organized services resource views on the application level through further mining the semantic relations of services, to support service active discovery, service composition on demand and other upper-layer applications.

Three ***Processes:***

1) Service instancing process of Resources (Encapsulating resources as services)***:***

As described above, Web service based on Internet protocol provides us an important mechanism to realize resources mutual manipulation underlying extensive environment. Thus, the first step of resources virtualization and organization process is to transform all resources into services and pack them up in the service format. Through uniform description and management of these services, uniform service resources are provided to upper application.

In this process, we adopt service component deputy mechanism. During the active service implementation process, we view various resource formats such as components, data and equipments as service component deputies. Transferring between these resources is converted into service requests and responds between service component deputies (based on message mechanism).

Resource $\boldsymbol{R_i}$ is transformed and encapsulated as a web services (denoted as $\boldsymbol{S_i}$) by mapping the information of $\boldsymbol{R_i}$ to Web Services Specification.(e.g. mapping the $\boldsymbol{R_i.F_i}$ to the UDDI registration information of service $\boldsymbol{S_i}$, mapping the $\boldsymbol{R_i.I_i}$ to the WSDL description information, mapping the $\boldsymbol{R_i.P_i}$ to the WSDL binding information, etc.). After that, a ***Service Agent*** is initiated to delegate the "virtual service $\boldsymbol{S_i}$". the ***Service Agent*** first deploy the $\boldsymbol{R_i}$ entity into the grid environment according to the resource kind denoted by $\boldsymbol{R_i.Category}$, when a calling request comes, the ***Service Agent*** calls resource $\boldsymbol{R_i}$ and encapsulate the calling results of resource entity to the message style of services $\boldsymbol{S_i}$. Through the description mapping and ***Service Agent*** mechanism, resource $\boldsymbol{R_i}$ is completely transformed to service $\boldsymbol{S_i}$.

We present the general process of resources service instancing as follows. With regard to service resources, we directly deploy them into the grid environment; with regard to other resources, we can firstly deploy them into the grid environment through deputy mechanism, and then pack them up into services.

$\forall r_i \in R_{set.} 1 \le i \le n$

If $r_i \in WS$ then $r_i \rightarrow R_{set}$

Else

$GridServices(r_i \vee deploy(r_i)) \rightarrow R_{set}$

Endif

2) Function Clustering Process of Services:

Service function clustering is to extract the functions of all service components, and to constitute a function categorization norm according to the relationship between different functions.

Based on the function categorization norm, a set of services with similar functions are merged upon this function, and is demonstrated as an abstract service. There are some related definitions:

(1) ***Func(Si)*** is the function extraction function, indicating the extracted function of service ***Si***.

(2) ***FS*** is the function categorization norm. Initially, it is empty. According to each service, through function extraction function, extracted function key words are made one item of FS. Users are required to define relationships between different function key words.

The service clustering flow is described as followings. (a) Services pool directory according to the ***FCS*** is setup. Each service will advertise its function in the directory. (b) Based on the Services pool directory, a set of services with similar functions are merged upon a directory item, and are demonstrated as a service cluster. (c) The whole service cluster forms a net-shaped organization structure, which sets the foundation for service discovery and further organization.

The general process of service function clustering is described as follows:

$\forall s_i \in S_{set.} 1 \le i \le n$

If $(\exists funcItem_k \in FS, map(Func(s_i), funcItem_k) = TRUE)$

Then

$ServiceCluster(s_i, funcItem_k)$;

Else

$New(funcItem, m+1)$;

$funcItem_{m+1} = Func(s_i)$;

$ServiceCluster(s_i, funcItem_k)$

Endif

3) Service Abstracting and Functional Relations linking Process:

After the above two processes, a set of services with similar functions are linked to the same function item with identical transferring interfaces and semantics transferring relationships.

The relationships between services can be combination, inheritance, inclusion, equation (replacement), calling, etc. Combinations compose of gradation, filiations, coalition, recursion, etc. No matter the relationship is combination, inheritance, inclusion, equation, or transferring, there are different semantic relationships between these services.

In order to better virtualize service resources, we organize up the semantic relationships between services to form up a hierarchical function relationship network, namely service function semantics Ontology and form the services resources view on the application semantics level. There are some related definitions.

(1)Abstract function

$$AS(\ funcItem_k\ ) = \bigcup s_i, \forall s_i, map(Func(s_i),\ funcItem_k\ ) = TRUE$$

(2)Semantic relationships between abstract services

Let Si and Sj be two different sub services in A1, the relationships between them R has the same logic relationship with function sub set defined in Def 4.

(3) Service function semantics Ontology

Service function semantics ontology composes of three parts: a set of abstract services, integration combination relationships between services, and information sequences of integration.

Ontology = （$A_{AS\text{-}subset}$， $A_{Message}$， $A_{Scenario}$）

• $A_{AS\text{-}subet}$ =(AS_1, AS_2, … , AS_n) It is the abstract services subset cited in the ultimate active service combination layer. This active service is composed of services in this subset.

• $A_{Message}$ =(M_1, M_2, … , M_n) is a finite set of messages. One message is composed of message type, message dispatcher, message receiver, and message body. Message body includes message parameters and protocol-related data. Definitions of this part is the same as the message definition of WSDL.

• $A_{Scenario}$ is the composition scheme of active service defined on abstract service component subsets AS_1 and AS_2. It describes sequence relationships and controlling relationships of each abstract service in AS.

On the bases of the above definitions, the definition process of semantics relationships between abstract services are defined as follows:

$\forall as_i \in AS_{set.} \exists funcItem_i, as_i = AS(\ funcItem_i), 1 \le i \le n,$

$\forall as_i, as_j \in AS_{set.}$其中$1 \le i, j \le n$

If （ $Relation(as_i, as_j) = TRUE$ ）

Then

Add（$as_i, as_i, Relation(as_i, as_i), Message(as_i, as_i)$）to ***Ontology***）

Endif

With the help of service function semantics Ontology, we can further record users' requirement analysis and re-composition results after functions decomposition. We use service function semantics Ontology to define the relationships between this the sub-functions of decomposed service.

4. Conclusion

In this paper, we present a virtualization architecture consisting of three views and three processes to organize and manage all sorts of resources as a whole In Grid Environment, and provide the upper application with coherent description as well as uniform access interface.

First, various physical resources In Grid Environment are transformed and encapsulated into services on the basis of Web service technology, and uniform service resources are provided for upper applications through uniform description and management of these services. Then by analyzing, identifying, classifying, organizing, storing, and managing these transformed services (i.e. Function Clustering Process of Services), multi application cases with the same function body are merged, and then the semantic relationships between services are further mined to better virtualize service resources. At last, we organize the semantic relations among services to establish a hierarchical function relationship network, and form the Services Resources View on the application semantics level.

References

1. Zhang yaoxue, Fang cunhao. Active services: Concepts, Architecture and Implementation [M]. Thomson Learning, 2005 July.
2. Borenstein, N., and Freed, N. (1993). MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC 1521 September.
3. Chan, S.W. K., and Franklin, J. (2003). Dynamic context generation for natural language understanding: a multifaceted knowledge approach. IEEE Transactions on Systems, Man and Cybernetics, Part A, 3(31), 23–41.
4. Hristidis,V., Papakonstantinou, Y., and Balmin, A. (2003). Keyword proximity search on XML graphs. Proc. of the 19th International Conference on Data Engineering, 5–8 March, (pp. 367–378).
5. Lawrence, S., Giles, C. L., and Fong, S. (2000). Natural language grammatical inference with recurrent neural networks. IEEE Transactions on Knowledge and Data Engineering, 12(1), 126–140.
6. Meuller, A., Mundt, T., and Lindner, W. (2001). Using XML to semi-automatically derive user interfaces. Second International Workshop on User Interfaces to Data Intensive Systems,May 31 to June 01.

7. Nakauchi, K., Ishikawa, Y., Morikawa, H., and Aoyama, T. (2003). Peer-to-peer keyword search using keyword relationship. Proc. of the CCGrid 2003. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 12–15 May, (pp. 359–366).
8. Hai, Z. G. (2000). A problem-oriented and rule-based component repository. The Journal of Systems and Software, 50, 201–208.
9. Fang, C. H., Zhang, Y. X., and Xu, K. G. (2003). An XML-based data communication solution for program mining. Intelligent Data Engineering and Automated Learning, 4th International Conference (pp. 569–575). Hong Kong, Berlin Heidelberg: Springer-Verlag.
10. Zhang, Y. X., Fang, C. H., and Wang, Y. (2004). A feedback-driven online scheduler for processes with imprecise computing. Journal of Software, 15(4), 616–623.