J. Cent. South Univ. Technol. (2007)04−0557−06
DOI: 10.1007/s11771−007−0107−z

Springer

An effective assemble-oriented framework for grid Web service

CHEN Zhang(陈　章)[1, 2], CHEN Zhi-gang(陈志刚)[1], DENG Xiao-heng(邓晓衡)[1], CHEN Li-xin(陈丽欣)[2]

(1. School of Information Science and Engineering, Central South University, Changsha 410083, China;
2.School of Information, Guangdong University of Business Studies, Guangzhou 510320, China)

Abstract: An effective assemble-oriented framework for grid Web service based on open grid service architecture was proposed, in which Web service semantics network constructed by software reuse was designed to enhance the locating of assemble-oriented service resources. The successful Web services assembled structure was exploited to design semantics network, the logical and the physical structure of the resource was separated in Web service, and the logical resource derived from type ID of Web service was combined with semantic structure. The organizing protocol, data structures and the mechanism of implementation in the model were presented. Experiment results show that the success ratio of Web service request comes to 100% while providing completely assembly semantics set. This model provides guarantee of the reliability of assemble Web service and establishes the foundation of Web service automatic interaction, customizing application service and dynamic service configuration.
Key words: grid; Web service; assemble-oriented; logic tree

1 Introduction

The Internet has evolved to become a commercial infrastructure of service delivery instead of merely providing host connectivity by open grid service architecture(OGSA)[1]. Facing the large amount of diverse services, how to organize and manage Web service sources has become a new research hotspot. A lot of strategies and methods were put forward, and most of them were focused on how to locate a single resource simply and quickly[2−6].

Different from domain name resources in Internet, grid resources often require appropriate collaboration, and could be duplicated and migrated in grid system. To organize grid resources, not only the location, scalability of organization architecture and load balance of resources, but also the semantic, interoperability of resources, and chain processing architecture should be considered. How to choose and assemble Web services effectively and efficiently is an important part of Web service application integration. Only this Web service resources organization can be application-oriented, process-oriented, and establishes the foundation for implementing automatic interaction, customizing application service and choosing service configuration dynamically. Traditionally Web service management is implemented with universal description, discovery and integration(UDDI) public registration warehouse, which is a signature service description mechanism and lack of clear semantic description[7−10]. Especially in open systems, signature was not adequate for assembled application. So far, a popular method is UDDI-based introduction of the concept of semantic Web[11−12], such as DAML-S[13] combined with UDDI, enhances automatic discovery capability of service. But with the increase of Web services and Web service application assembly, this method is not enough to support software reuse and service auto-discovery, the reason is that it does not consider about reusing some success applications among Web services, such as intersecting, inclusion, compatibility, substituting and applying assembled resources.

RICHARD et al[8, 14−16] firmly pointed out that software reuse could increase programming efficiency, decrease software costs, and also improve software qualities. A model based on OGSA was proposed in the paper, which improves the model proposed by CHEN[6], and transfers it from single Web service resource function to protocol-oriented and application process-oriented. With semantic network, when users apply for searching services, this architecture should be known whether there are resources to which can provide the services or be useful for interaction and chain processing between resources, and can gain these resources and related information through certain searching mechanism.

2 Assemble-oriented web service architecture

Web services' seamless assembly attracts the flow of

Foundation item: Project (60573127) supported by the National Natural Science Foundation of China; Project (06023960) supported by the Natural Science Foundation of Guangdong Province, China; Project (06JJ30032) supported by the Natural Science Foundation of Hunan Province, China
Received date: 2006−12−24; **Accepted date**: 2007−03−27
Corresponding author: CHEN Zhang, Associate professor, Doctoral candidate; Tel: +86-20-88379750; E-mail: cz_fs@263.net

B2B and enterprise application integration greatly. Because there is a great gap between the current development of grid and application requirements, grids should be highly simple, useful, seamless, and automatic. In other words, service should combine with contexts statically or dynamically. The contexts include user information, execution codes, execution speed, dependability and appropriate authentication mechanism for users. From assemble-oriented Web service, users should get not only Web service that they require, but also much more useful information for Web service assembly.

This multilevel multilayer Web service architecture model[6] was built on the assumption that: Web service resources can be managed and distributed by grid globally, and identified by unique resource ID. The main usage of Web service resource organization tree(WSROT) in this model is to maintain the state information of active service providers and global load information of the same type of resources. WSROT is the network node that is especially used for maintaining Web service resources.

Based on the above multilevel multilayer Web service architecture, from the point of view of Web service assembly and software reuse, an assemble-oriented Web service architecture model was proposed. This model improved the logic level of Web service proposed by CHEN et al[6], so it can maintain not only the state information of active service provider, and global load information of the same type of resources, but also the interaction among Web service resources and chain processing structure.

The Web service resource architecture in this paper is composed of four parts shown in Fig.1. The functions of the parts are as follows.

1) Client query and assemble-oriented semantic apply/register system (CQARS). It has two functions: first, after new application is assembled successfully, it transfers assemble-oriented semantic service request to each related WSROT through Web services resource register center(WSRRC), and registers assemble-oriented semantic structure to these authoritative WSROT nodes. Second, it transfers Web service searching request from users to a comprehensive format for system, and submits the request, then obtains services through the corresponding methods.

2) WSRRC. It has three main functions: first, when new application is assembled successfully, WSRRC sends assemble-oriented semantic service request to WSROT, and returns related WSROT address. Second, when a Web service resource provider (WSP) wants to add its resources to the global system, it negotiates with WSRRC first, and waits for the resource type, resource ID, and also WSROT address of resource returned from WSRRC, then WSP registers its resource information to WSROT, which is in charge of the maintenance of resources information and load information of every resource node. Third, when service applicant submits Web service request, WSRRC transmits the request to WSROT, then WSROT returns the actual address of service provider and service chain processing structure that is useful to application assembly to service applicant.

3) Web service resource organization tree (WSROT). First, Web service resources are organized to different logical resource trees by different type IDs. Each tree is in charge of maintaining state information of active service providers, global load information of the same type of resource, and assemble-oriented semantic frame. Second, it accepts assemble-oriented semantic frame service registration from client.

4) Area proxy autonomy system (APAS). It maintains the creation and disappearance of Web service in an area. APAS binds Web service resources to IP address, searches and obtains WSROT address from WSRRC, then registers resource information of the local proxy area to WSROT, and makes WSROT completely know about the state of current resources. It shares the same TTL with WSROT to maintain the resources information.

Because of the length of paper, the parts described proposed by CHEN et al[6] will not be presented here. The next part specifies details of the relations among Web services and assemble-oriented semantic architect.

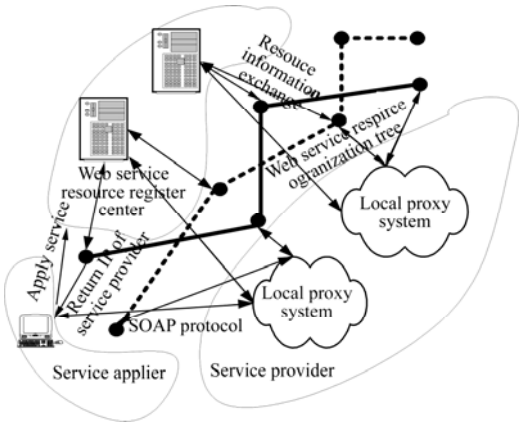

Fig.1 Grid Web service structure model

3 Relations among Web services and description of Web services assembly

Suppose Web services maintained by local proxy autonomy system are meta-services. Commonly, Web service can be noted as a two-uples: $S=<C, A>$, C is general description of service based on the standardized definition of a domain industry; A is service operating

information based on the standardized definition of a domain business. For $\forall a \in A$, define $a = <op, I, O>$, three parameters *op*, *I* and *O* separately refer to operation, input and output. As for input, $I = <p_1, \cdots, p_m>$; and output $O = <q_1, \cdots, q_n>$. Every p_i (i=1, $\cdots$, m) and q_j (j =1,$\cdots$, n), can be formatted to $<name>:<type>$.

Definition 1: Web service behavior compatibility

For $\forall a \in S.A$, if there is an operation $a \in S'.A$ which has $a' \square a$ service, then the behavior of S' and the behavior of service S are compatible ($S'.A \square S.A$).

Definition 2: Operation compatibility

If

1) The precondition and post-condition of $a'.op$ are equal to those of $a.op$;

2) Input $a', I \square a.I$, and

3) Output $a', O \square a.O$,

then $a'.op$ and $a.op$ are compatible.

Definition 3: Substitution relation

If and only if $(a'.I \subseteq a.I) \wedge (a'.O \subseteq a.O)$, S' can be substituted by service S, that is ($S' \leqslant S$).

In grid environment, no common memory or timer exists, so the occurring sequence of two events is not assured. Therefore, the sequence of events should be partial order. In this paper, Web services are local autonomy distributed objects, which cooperate with each other to fulfill tasks assigned. To represent distribution, dependence, and concurrence of Web service events more exactly, the sequence of assembling Web service is denoted by event partial. The formalization means can ensure correctness and high efficiency of reusing assembled Web services. Following the causal dependence relation proposed by PRATT[17], the partial order is defined as: event *e* causally depends on event *f*, if event *f* does not happen, then *e* will not happen either, denoted as $f \rightarrow e$.

Definition 4: An event partial order is a quadruple(V, Σ, $\leqslant$,u)，which includes the follows:

1) An event set V, denotes all events happened.

2) A service vocabulary Σ, denotes service primitives set which Web service provides and requires.

3) A partial order relation "$\leqslant$" in V, satisfies irreflexive, antisymmetric and transitive.

4) A labeled function $u: V \rightarrow \Sigma$, assigned the notions in Σ to node in V, denotes events corresponding to service happened.

Definition 5: An assembled Web service T_c is a triple (S, N, P), where S denotes general description of service based on the standardized definition of domain industry; N denotes a set of identifiers ID of Web services who participate the task; P denotes $\bigcup_{r \subseteq N} r$. A set of regular expression in Σ, $P = P_i \bigcup P_c \bigcup P_b$, $P_i \bigcap P_c = \varnothing$, $P_i \bigcap P_b = \varnothing$, $P_b \bigcap P_c = \varnothing$, where P_i denotes initial constrain condition, P_c denotes mapping of service route, P_b denotes task behavior patterns; P_c joins supplier and applicant of the same service primitives together. P_i is a prefix of partial order events, which is a initial of T_c. P_b is behavior pattern of tasks, which points out how to complete tasks, including which events will show up, what order the events will be created on, and which events can trigger the creation and deletion of Web service instances.

Based on ontology interchange language (OIL), the relations between Web services are shown in Fig.2.

```
Class_def defined Relations among services %Definition of relations among Web services
  Slot-constraint exit %Operate compatible relation of Web services
    has value Compatibility
  Slot-constraint exit %Function substitute relation of Web services
    has value Substitution
  Slot-constraint exit %Chain using relationship exists in Web services
    has value Chain using
```

Fig.2 OIL description of Web services relations

Chain used for relations among Web services shows how a group of finely granular service resources are assembled to coarsely granular service resources.

4 Assemble-oriented resource tree

According to the method proposed by CHEN et al[6], assume that there are several resource trees in system. The authority node of resource tree exists based on Web service type ID, which is shown in Fig.3. The authority node of resource tree is used by adding three tuples storing assemble-oriented semantic resources information, viz. properties of the corresponding Web services, such as Web service sequences with compatibility, substitution, and assembly relation: Compatible_for[], substitute_for[], assemble_for[]. Compatible_for[] is a pointer array that identifies compatible Web service ID resources; substitute_for[] is a pointer array that identifies substitutable Web service ID resources; assemble_for[] is a pointer array that identifies successfully assembled Web service ID resources.

Maintenance of assemble-oriented semantic resources information consists of two parts: adding reusable resources; removing reusable information. The two processes are transferred by CQARS through WSRRC to WSROT. After an application is assembled successfully, CQARS will transfer the calling message of assemble-oriented semantic resources information

maintenance through WSRRC to WSROT, which mainly updates the arrays Compatible_for[]，substitute_for[], assemble_for[] in authority node of WSROT resource tree and resource information of nodes pointer link. When assemble-oriented semantic resource information array is full, the longest time unused resource information will be removed, according to the longest time unused algorithm.

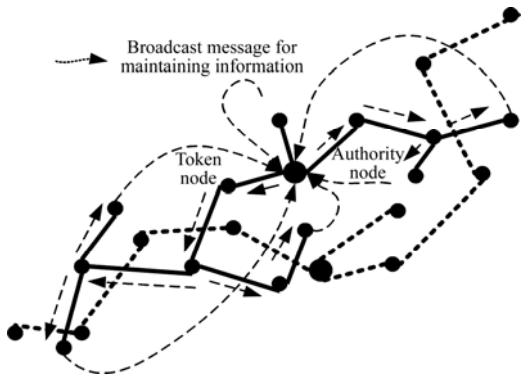

Fig.3 Assemble-oriented resource tree structure

5 Prototype system experiment and analysis

5.1 Prototype system experiment platform

Prototype system experiment platform included 12 PIV PCs with 512 MB memory. Three of these operation systems were RedHat Linux 7.3 to act as WSROT, and C language was adopted as programming language. Six of them were installed Win2000 operating system to simulate network nodes with VC6.0. The rest three of them on Win2000 simulated CQARS with VC6.0. All these PCs constituted a LAN with a 16 plugs switch.

System topology structure was created by Internet topology simulator BRITE[18], which simulated network nodes whose distribution satisfies power law and small world properties of network nodes. Because these 12 machines were connected through local networks physically, but not connected in accordance with topological graph, just the topological graphs were created, then the nodes created by different programs in these machines were taken as real nodes, then 48 hosts were gained to form 4 small worlds (or four APAS), each of them ran service proxy protocols through a SNMP network governor by assigned super cluster center node (SCCN), and others ran common host node response programs. The number of WSROT differently configured was 12. Due to the nodes were virtual, not equipped with the whole set of SOAP protocols, OGSA was installed on our real machine, so after user requested WSRRC and received IP address of resource proxy (SCCN) from WSROT, SCCN would return IP address of some real server, then executed the following service summit in the manner of real OGSA.

5.2 Protocol functions implement

In the prototype system, only a few critical functions were implemented, and simplified, so six main functions were implemented and executed on nodes.

1) The WSRRC system was configured manually, and the WSRRC self-organization protocols were designed.

2) Nodes with high congregation as SCCN(or server proxy) were set by achieving dynamic network topology information, the secondary cluster nodes were constructed if necessary, and the “cluster link” was built to be in charge of maintenance of local APAS network topology.

3) The “local Web service resource base” between resource proxies and subordinate cluster nodes was formed, resource proxies exchanged information with neighbor hosts periodically, and the information was used to refresh its own “local Web service resource base”. Resource proxies interacted with WSROT to get binding of Web service.

4) When adding resource to a resource proxy, the “adding resource algorithm” ran to interact with WSRRC and WSROT.

5) The native resource information exchanging and maintenance in WSROT system were designed, and WSROT interacting with WSRRC center was implemented to make it obtain dynamic information, load status of WSROT node itself and assemble-oriented semantic resource information.

6) The “service applicant algorithm” was designed to resolve the service request when user resource request occurred, and made user to achieve service and related resource reusing semantic information.

5.3 Instance

After installing the above system, some services were deployed in manner of OGSA standards, and then let each APAS apply resource ID from WSRRC and register to WSROT. After user system getting APAS through searching, APAS returned real IP address of a real host, then standardized Web services were accessed by the rule of SOAP and the services were obtained.

Dispose four services $S_0(a)(b，c)$, $S_1(b)(d)$, $S_2(b)(e)$, $S_3(d，e，c)(f)$ in network nodes, which can be assembled to a computing service $P(a)(f)$ shown in Fig.4. Assemble-oriented semantic service apply/register algorithm ran after service P was successfully computed and assemble-oriented semantic resource information was registered on the related WSROT resource tree. So when users apply one of the four services again, they could get not only real IP address of services, but also chain

structure of the resources.

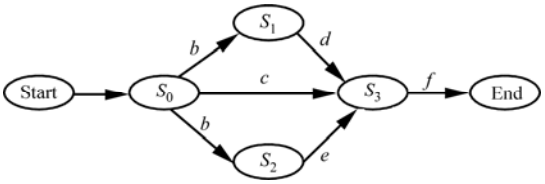

Fig.4 Web service chain structure example

5.4 Experiment result analysis

So far in most of papers precision ratio and recall ratio are adopted as evaluation meter of Web service discovery mechanism[19].

Precision ratio:

$$P_{\mathrm{WS}} = \frac{A \cap B}{B}$$

Recall ratio:

$$R_{\mathrm{WS}} = \frac{A \cap B}{A}$$

where A is standard result set, and B is return result set. This method, using metrics in information searching, takes Web service as an isolated resource, but still should be improved in the aspect of evaluating assemble-oriented Web services application. In this paper assemble-oriented Web services request success ratio was introduced to show the validity of this organization model supporting assemble-oriented Web services location.

Supposing that there are several Web resources waiting for assembly, and in these T_{re} service requests, S_{re} services are requested successfully, and assembled successfully, then assemble-oriented Web services request success ratio R_{su} is defined as

$$R_{\mathrm{su}} = \frac{S_{\mathrm{re}}}{T_{\mathrm{re}}}$$

If an assembled Web service includes n Web services, and each Web service in grid environment is supplied by m_i ($1 \leqslant i \leqslant n$) service agents, in which there are k_i ($0 \leqslant k_i \leqslant m_i$) service agents that can supplied satisfied services. Then, the success ratio of these n Web services resources can be achieved by using the model proposed by FOSTER[2−6].

$$R_{\mathrm{su}} = \prod_{i=1}^{n} \frac{k_t}{m_i}$$

However, the success ratio by using the model of this paper satisfies

$$1 \geqslant R_{\mathrm{su}} \geqslant \prod_{i=1}^{n} \frac{k_i}{m_i}$$

Obviously, by the method of this paper, at the beginning of system running, when assembled semantic warehouse still has not formed, $R_{\mathrm{su}} = \prod_{i=1}^{n} \frac{k_i}{m_i}$. With the system running normally, and assembled semantic warehouse growing completely, the success ratio is driven close to 1.

A scheme is designed for comparing the simulation experiments, where framework A takes the framework introduced in this paper, and framework B takes organization architecture proposed by CHEN et al[6], and taking R_{su} as metrics, the comparison result of the two frameworks is shown in Fig. 5. In the experiments, the same 20 Web services W_1, W_2, ⋯,W_{20} in the two frameworks were disposed and these Web services were divided into 6 groups, and Web services in the same group could substitute each other. The system created an assemble service request randomly and iteratively. At the beginning, success ratios of the two frameworks are almost the same, but after a while, as assembled semantic warehouse in framework A growing completely, service request success ratio will increase gradually, driven close to 1. But for framework B, the success ratio will not increase as time passes.

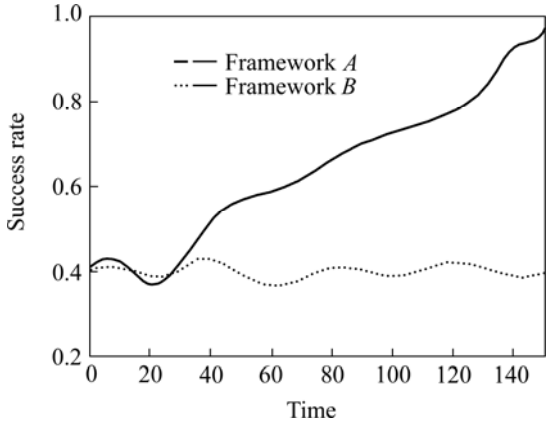

Fig.5 Success rate in different framework

6 Conclusions

1) A method to enhance the location of assemble-oriented resource in Web service organization structure level was presented.

2) An assemble-oriented framework for grid Web service based on OGSA was proposed, in which the successful service assembled and application process-oriented were reused.

3) The simulation experiment results show the model is an effective organization solution for assemble-oriented Web service. Provided with a completely assembled semantic warehouse, the success ratio of assemble-oriented Web service request will be 1. This model greatly improves the efficiency of traditional Web resource organization architecture on supporting assemble-oriented Web service

resources, and establishes the foundation of implementing automatic interaction, customizing application service and dynamic service configuration.

References

[1] FOSTER I, KESSELMAN C, NICK J. The physiology of the grid: an open grid services architecture for distributed systems integration. [EB/OL]. [2005−01]http://www.globus.org/research/papers/ogsa.pdf.

[2] FOSTER I, KESSELMAN C. The globus project: A status report[C]// Proc of the IPPS/SPDP'98 Heterogeneous Computing Workshop. Orlando, Florida: IEEE Computer Society Press, 1998: 4−18.

[3] LITZKOW M J, LIVNY M, LUTKA M W. Condor-a hunter of idle workstations[C]// Proc of 8th Int'l Conf on Distributed Computing Systems. Washington: IEEE Computer Society Press, 1988: 104−110.

[4] LI Wei, XU Zhi-wei, BU Guan-yin, et al. An effective resource locating algorithm in grid environments[J]. Chinese Journal of Computers, 2003, 26(11): 1546−1549. (in Chinese)

[5] RAJASEKAR A, MOORE R. Dada and metadata collections for scientific application[C]// Proc of 9th International Conference on High-Performance Computing. Amsterdam, Holland: Lecture Notes in Computer Science, 2001, 2110: 72−80.

[6] CHEN Zhi-gang, LIU An-feng, XIONG Ce, et al. An effective loading-balancing framework for grid web service[J]. Chinese Journal of Computers, 2005, 28(4): 458−466. (in Chinese)

[7] ETZKOM L H, DAVIS C G. Automatically identifying reusable components in object-oriented legacy code[J]. IEEE Computer, 1997, 30(10): 66−71.

[8] RICHARD T D. The economics of component-based development[J]. Information Systems Management, 2000, 17(1): 92−95.

[9] ZHU Shu-ren, DENG Ting-ting. Fuzzy matching routing filter in content-based publish/subscribe[J]. Journal of Central South University of Technology, 2007, 38(1): 138−142.(in Chinese)

[10] ZHU Shu-ren, LI Wei-qin. Design and implementation of self-protection agent for network-based intrusion detection system[J]. Journal of Central South University of Technology, 2003, 10(1): 69−73.

[11] SIVASHANMUGAM K, VERMA K, SHETH A, et al. Adding semantics to Web services standards[C]// Proceedings of the 1st International Conference on Web Services (ICWS'03). Las Vegas, Nevada: 2003: 395−401.

[12] MCLLRAITH S, SON T, ZENG H. Semantic Web services[J]. IEEE Intelligent Systems (Special Issue on the Semantic Web), 2001, 16(2): 46−53.

[13] ANKOLEKAR A, BURSTEIN M, HOBBS J R, et al. DAML-S: Semantic markup for Web services[C]// Proceedings of SWWS'01. California: Stanford, 2001: 411−430.

[14] HOOPER J W, CHESTER R O. Software Reuse: Guidelines and Methods[M]. New York: Plenum Press, 1991.

[15] KIM Y, STOHR E A. Software reuse: Survey and research directions[J]. Journal of Management Information Systems, 1998, 14(4): 113−147.

[16] VITHARANA P, JAIN H. Research issues in testing business components[J]. Information and Management, 2000, 37(6): 297−309.

[17] PRATT V. Modeling concurrency with partial orders[J]. Journal of Parallel Program, 1986, 15(1): 33−71.

[18] BU T, TOWSLEY D. On distinguishing between Internet power low topology generators[C]// The IEEE INFOCOM 2002. Washington, 2002: 638−647.

[19] CHEN De-wei, XU bin, CAI Yue-ru, et al. A p2p based Web service discovery mechanism with bounding deployment and publication[J]. Chinese Journal of Computers, 2005, 28(4): 615−626. (in Chinese)

(Edited by YANG Hua)