Resource Discovery Based on VIRGO P2P Distributed DNS Framework

Lican Huang

Institute of Network & Distributed Computing,
Zhejiang Sci-Tech University, Hangzhou, P.R.China
huang_lican@yahoo.co.uk

Abstract. Resource discovery plays an important role on scientific workflows in large scalable network environment. This paper presents a resource discovery framework based on VIRGO P2P Distributed DNS. With the convention of resource name as the format- *functionscheme*":*global-hier-part*"/*local-name*, this framework supports flexible queries using partial keywords and wildcards, and range queries by a SQL-like query statement. The global-hier-part is managed by registers the same as DNS servers except the extension of RRs. The DNS servers construct n-tuple overlay virtual hierarchical overlay network of VIRGO. With cached addresses of DNS servers, the overload of traffic in tree structure can be avoided. The time complexity, space complexity and message-cost of lookup with this framework is $O(L)$, where L is the number of sub domains in Domain Name.

1 Introduction

Resource discovery plays an important role on scientific workflows in large scalable network environment. Before submitting a job to require a QoS performance, a suitable workflow engine host node with a set of attributes (such as CPU type, memory, operating system type) among lots of workflow engines is required to find for executing the job. Scientific workflow often requires dynamic selection of workflow routines, Web Services or workflow engines. In some case, we need to find a suitable Web Service from multiple copies of semantic Web Services to achieve better performance [1][2]. The most suitable workflow engines are required to be discovered before submitting sub workflows to these engines to execute in parallel [3]. In other cases, the Services with the same functions but with different algorithms (for example, data clustering services implemented by using the algorithms of Neural Network or SVM) may be only chosen at the just pre-execution point according to the intermediate results of workflow execution [4]. All the above cases illustrate the importance of resource discovery(the resources may be static and/or dynamic. The dynamic resources are meant both in dynamic change of attributes or dynamic exist in the network).

When there are more and more computers connected in Internet and more and more users use Internet, resources in Internet will be more and more. In this large,

decentralized, distributed resource sharing environments today or in the future, the efficient discovery of static and/or dynamic resources is a big challenge.

The resource discovery based on Client/Server is not suitable for huge amount of resources. There are many approaches for distributed resource discovery based on P2P technologies[5]. The un-structural P2P technology such as Freenet[6] using flooding way has shortage of heavy traffic and un-guaranteed search. The structural P2P technology using DHT such as Chord [7] loses semantic meanings of discovered objects. Resource discovery based on Virtual and dynamic hierarchical architecture(VDHA) [8] has shortage of high traffic load in root nodes and possible split of virtual trees. VIRGO[9][10] P2P network hybrids structural and unstructured P2P technologies by merging n-tuple replicated virtual tree structured route nodes and randomly cached un-structured route nodes(LRU and MinD) to solve the above problems.

The resources can be classified as hierarchical Domains. It is nature way to use DNS to discover resources. There are DNS-based resource discovery approached [11] by using the existing and mature DNS protocols[12]. The main advantage is the use of existing, widely accepted and consolidated DNS technologies and APIs to accomplish resource advertisement and dynamic discovery. Although DNS implementation today is scalable for translating Domain Names into IP addresses, it may encounter problem if using it to discover huge amount resources today and even more amount resources in the future. The framework proposed here based on VIRGO P2P technology is suitable for the huge amount of resources' discovery.

This paper presents a framework for distributed resource discovery based on VIRGO P2P technologies. The resources are classified as multi-layer hierarchical catalogue domains according to their semantic meaning. We then name resources as format- *functionscheme ":"global-hier-part"/"local-name* [13]. The global-hier-part part in this format is taken as Domain name. The authoritative DNS servers controlling these Domain names manage these resources. These DNS servers construct n-tuple overlay virtual hierarchical overlay network. With cached addresses of DNS servers, the overload of traffic in tree structure can be avoided. We here change some places of VIRGO to suit the distributed DNS implementation[14]. The lookup protocols of distributed DNS is similar as the protocols in VIRGO[9], which is illustrated in detail in the paper[10]. The resource discovery is effective and guaranteed. The time complexity, space complexity and message-cost of lookup protocol is $O(L)$, where L is the number of sub domains in Domain Name.

The structure of this paper is as follows: section 2 describes VIRGO network structure for distributed DNS; section 3 presents the Convention for Universal Resource Name; section 4 discusses Distributed Resource Discovery Framework; and finally we give out conclusions.

2 VIRGO Network Structure for Distributed DNS

VIRGO [9] is a domain-related hierarchical structure hybridizing un-structural P2P and structural P2P technology. VIRGO consists of prerequisite virtual

group tree, and cached connections . Virtual group tree is similar to VDHA [8] , but with multiple gateway nodes in every group. Virtual group tree is virtually hierarchical, with one root-layer, several middle-layers, and many leaf virtual groups. Each group has N-tuple gateway nodes. In VIRGO network, random connections cached in a node's route table are maintained. These cached connections make VIRGO a distributed network, not just a virtual tree network like VDHA. With random cached connections, the net-like VIRGO avoids overload in root node in virtual tree topology, but keeps the advantage of effective message routing in tree-like network. As the change of contents in route table, VIRGO uses different lookup protocol and maintenance protocols from VDHA.

In VIRGO, user uses client to access VIRGO via access point node(called as entrance node). All users are managed by their owner nodes. Some nodes (called Gateway node)take route functions in several different layers of virtual groups. Gateway node is the node which is not only in low-layer group, but also in up-layer group.

Fig. 1. Two_tuple Virtual Hierarchical Tree Topology

Fig. 1 shows two-tuple virtual hierarchical tree topology (the nodes in different layers connected with dash line are actually the same node). In Figure 1, from the real network, three virtual overlay groups are organized. From these virtual groups two nodes per virtual group are chosen to form the upper layer virtual group.

3 Convention for Universal Resource Name

In Internet, every resource needs an uniform name. There are RFCs URI [15] ,URL[16] and URN[17]. Assuming there are so huge number of resource names, how to name these resources as to easily resolve is a big issue. Here we argue that the resources can be classified into a hierarchical structure.

Therefore, the convention for resource name is formatted as global unique hierarchical domains like domain names in DNS, and a local name to specify the local attributes of resource name. The format of convention of resource names is as follows:

functionscheme : "global – hier – part" / "local – name

Here, the functionscheme is type or function or schema for the resource; the global-hier-part is the global hierarchical name similar as domain name and the local-name is the local name for the resource.

For example,

blog : *Britney.popular.music/storyofBritney* is a resource name for a blog article about the story of Britney Spears. Whereas, *song* : *Britney.popular.music/Lonely* is a resource name for the song Lonely by Britney Spears. Here, Britney Spears is classified as *Britney.popular.music*.

The local-name part in the format can be used for the description of resource semantic meanings. It can also a URL for the other repository such as UDDI , Grimoires[18] which contains the resource metadata and/or other information.

The global-hier-part is formed as Domain Names. The resources are classified into different domains according to their semantic meaning. The top domains may be governed by International trustee such as The Internet Corporation for Assigned Names and Numbers (ICANN). Other level domains may be controlled by virtual organizations organized by the experts with specific knowledge of these domains. For example, in Domain Name *www.Madonna.popular.music*, *music* Domain is managed by music virtual organization, which reports to ICANN to approve, *popular.music* Domain is controlled by popular music virtual organization, which reports to music virtual organization to approve. The request of Domain Name *www.Madonna.popular.music* is sent to virtual organization of popular music to be approved. In Domain Name *www.Beethoven.classic.music*, *classic.music* Domain is controlled by classic music virtual organization, which reports to music virtual organization to approve. The request of *www.Beethoven.classic.music* is managed by virtual organization of classic music.

4 Distributed Resource Discovery Framework

We treat global-hier-part in the invention of resource name as domain name. We also use the similar concept of DNS zone for DNS server to manage the domain names. For example. if a DNS server's zone is *Science.Biology.Bioinformatics*, then this DNS server manages all resource information under the catalogue of *Science.Biology.Bioinformatics*. In other case, if a DNS server's zone is *Science.Biology*, then this DNS server manages all resource information under the catalogue of *Science.Biology*, which may includes sub domains of *BioInformatics*, *botany*, *zoology*, etc.

Resource names are published in their authoritative DNS server. The extension RRs in DNS server contains the parts of the functionscheme and local-part of conventions of resource name. The local-part of resource can be a URL for

Fig. 2. Distributed Resource Discovery Framework

the other repository such as UDDI, Grimoires[18] which contains the resource metadata and/or other information.

Every DNS server is the same but some coexist in more than one layer. Every DNS Server maintains a route table and RRs related to its authoritative domain zone. Route table includes addresses of Foreign Name Servers which are prerequisite for Virtual Hierarchical Overlay Network and cached addresses of Foreign Name Servers which are refreshed by TTL rule, etc.

The DNS servers join VIRGO network by placing themselves in virtual groups according to their authoritative domain names(The only root domain- "root" is anonymous placed at the top point of all the domains). When a DNS server joins, its route table will add some other nodes' entities, and some other nodes will update their route table by adding the joining node's entity. The detail of the algorithm can be in the Internet draft[14].

The query process is as the following(see figure 2): User program sends QUERY MESSAGE, which contains SQL-like statement in the following subsection, to Local Name Server. If Local Name Server is the authoritative Domain Name Server, then the Local Name Server will check its RRs to resolve the request resource. Otherwise, The Local Name Server will route to the Foreign Name Server which is closer to the authoritative Domain Name Server by calculating theoretical hops. Then the Foreign Name Server routes to the even closer Foreign Domain Name Server. Repeat this process, until the authoritative Domain Name Server has been found. Finally, the authoritative Domain Name Server resolves request resource by check its RR record, and responses to the Local Name Server. The latter will forward the response to the User Program. The details of the algorithm can be found in [14].

For example, to discover resource chloroP, we first locate the DNS authoritative server for domain-Science.Biology.Bioinformatics, then in this server, we check its extension RRs for the resource, which gives URL of other repository.

Finally, we find resource information such as URL, metadata ,etc.in the item of localdirectore/chloroP/ in the repository.

4.1 SQL-Like Query Language and Maintenance Language

The local-part part in resource convention can be formed as entities with attributes in the domain defined by global-hier-part. We use SQL-like query language to query resources. The SQL-like query language is formed as the following:

```
query ::= SELECT{SCHEMA|SUBDOMAIN|ENTITY|ATTRIBUTE|*
|expr[[AS]c_alias]{,expr[[AS]c_alias]...}}
FROM domainref[ WHERE search_condition];
```

Here, SCHEMA is for querying functionschema for a Domain; SUBDOMAIN is for querying subdomain for a given super Domain; ENTITY is for querying all entities for a given search condition ; ATTRIBUTE is for querying all attributes for a given search condition ; expr is for the attribute queried, which may be a set of attributes; [AS] c_alias is the alias name for expr; domainref is for Domain Name, in which we can use * to indicate all sub domains; search_condition is similar to SQL statement in Database.

In the following, there are several examples of query operations.

Operation 1

The form "?:global-hier-part" is to query all functionschema of the global-hier-part domain. For example, "?:Britney.popular.music" queries all functionschema of Britney.popular.music. We can use the following Statement:

```
SELECT SCHEMA FROM Britney.popular.music;
```

Operation 2

The form "functionschema:global-hier-part/?" is to query all resources with the functionschema in the global-hier-parts domain. For example, "song:Britney.popular.music/?" queries all songs by Britney Spears. We can use the following Statement:

```
SELECT * FROM Britney.popular.music WHERE SCHEMA ='song';
```

Operation 3

Suppose that global-hier-part can be expressed as leafDomain.superDomain. The form "functionschema:?.superDomain/" is to query all leafDomains. This is possible because of the hierarchical structure and the protocols. For example, "song:?.popular.music" queries all popular singers. We can use the following Statement:

```
SELECT SUBDOMAIN FROM *.popular.music WHERE SCHEMA ='song';
```

Operation 4 The form "functionscheme :global-hier-part/local-name" is to query specific resource. For example, "song:Britney.popular.music/Lonely" queries the information of the song-Lonely. We can use the following Statement:

```
SELECT Lonely FROM Britney.popular.music WHERE SCHEMA ='song';
```

Operation 5

The form "functionscheme :global-hier-part/expression" is to query the resources which satisfy the conditions of the expression indicated. Expression can

be scope, maximum, minimum, where condition like SQL statements. For example, "song:Britney.popular.music/where year between 2006 and 2007" queries all songs by Britney Spears which are produced between 2006 and 2007. We can use the following Statement:

```
SELECT * FROM Britney.popular.music WHERE SCHEMA = 'song' AND
year ≥ 2006 AND year ≤ 2007;
```

Another example is the case for selecting an optimal service. We first use Ganglia to get performance attributes of all machines in a given domain, and select an optimal machine by calculating maximum value of the formula –"CPU speed*CPUNumber/CPUload"; we can use the following statement:

```
SELECT ServiceAddress FROM cluster.computer.IT WHERE SCHEMA
= 'hardware' AND max(ganglia.CPUspeed*ganglia.CPUNumber/
ganglia.CPUload);
```

The resources are maintained by the SQL-like maintenance language. Insert statement registers new resource into the domain. Delete statement drops the resource registration from the domain. Update statement changes the information registered in the domain. The formats of these statements are as the following:

```
DELETE FROM domainref [WHERE search_condition];
```

```
INSERT INTO domainref[(entity.attribute){, entity.attribute...}]]
VALUES(expr|NULL{, expr|NULL...});
```

```
UPDATE domainref SET entity.attribute = expr|NULL{,
entity.attribute = expr|NULL...};
```

For example, the following statement deletes all songs by Britney Spears which are produced between 2006 and 2007.

```
DELETE FROM Britney.popular.music WHERE SCHEMA = 'song' AND
year ≥ 2006 AND year ≤ 2007;
```

4.2 Complexity

Because the DNS server nodes are virtually organized as a tuple virtual tree, every DNS server has a route table which includes prerequisite DNS servers' IP addresses for Tree Paths (TREE portion) and cached DNS servers' IP addresses (CACHED portion).

Because the message is routed according to the minimum of theoretical distance from destination node, and the route table contains TREE portion, every hop reduces the distance from destination node by at least one hop, Therefore, $\text{hops}(a, b) < \text{length}(a) + \text{length}(b) - 1$ (1)

Where, $\text{hops}(a, b)$ is for the hops from node a to node b; $\text{length}(a), \text{length}(b)$ are for node a domain name lengths and node b domain name lengths respectively.

For example, the length of www.nic.fr is 3. So,

$\text{timecomplexity} = O(L)$ (2)

$message_cost = O(L)(3)$,

where L is the length of domain name.

Because the route table of the virtual gateway nodes virtually existed from root layer to bottom layer groups has the maximum route items of nodes's information, we have:

$$MaxItems = L * N_tuple * nvg + Max_Cached \quad (4),$$

where L is the length of domain name., N_tuple is multiplicity of gateway nodes for virtual tree, nvg is number of virtual groups, Max_Cached is the maximum number of cached records in the route table

Therefore,

$$SpaceComplexity = O(L) \quad (5)$$

5 Conclusion

Resource discovery plays an important role on large scalable scientific workflows. We here presents a resource discovery framework based on a novel distributed DNS. The proposed distributed DNS framework is based on VIRGO P2P technologies. The resources are classified into hierarchical domains, which can be managed by registers the same as DNS servers except the extension of RRs. The DNS servers construct n-tuple overlay virtual hierarchical overlay network of VIRGO. With cached addresses of DNS servers, the overload of traffic in tree structure can be avoided. The time complexity, space complexity and message-cost of lookup with this framework is $O(L)$, where L is the length of domain name. With the convention of resource name as the format-*functionscheme ":" global-hier-part"/"local-name*, this framework supports flexible queries using partial keywords and wildcards, and range queries. The framework presented here uses the similar message of DNS. Therefore, we can use existing, widely accepted and consolidated DNS technologies , APIs and open source codes to implement the framework. Meanwhile, the framework with VIRGO P2P technology is scalable for discovery of huge amount of resources today and even more amount resources in the future. We plan to implement this framework by merging the source codes of VIRGO project[19] and dnsjava[20].

Acknowledgements. This paper is funded by "Qianjiang Rencai" funds by Government of Zhejiang Province(No:2007R10013).

References

1. Huang, L., Akram, A., Allan, R.J., Walker, D.W., Rana, O.F., Huang, Y.: A Workflow Portal Supporting Multi-Language Interoperation and Optimisation. *Journal Concurrency and Computation: Practice and Experience* 19(12), 1583–1595 (2007)
2. Walker, D.W., Huang, L., Rana, O.F., Huang, Y.: Dynamic Service Selection in Workflows Using Performance Data. *Journal of Scientific Programming* 15(4), 235–247 (2007)

3. Huang, L.: Framework for Workflow Parallel Execution in Grid Environment. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4489, pp. 228–235. Springer, Heidelberg (2007)
4. Huang, L.: Production Rule Based Selection Decision for Dynamic Flexible Workflow. In: 3rd IEEE International Conference on e-Science and Grid Computing, pp. 627–632. IEEE Press, New York (2007)
5. Foster, I., Iamnitchi, A.: On Fully Decentralized Resource Discovery in Grid Environments. In: Lee, C.A. (ed.) GRID 2001. LNCS, vol. 2242, pp. 51–62. Springer, Heidelberg (2001)
6. Clarke, I., Sandberg, O., Wiley, B., Theodore, W., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 46–66. Springer, Heidelberg (2001)
7. Stoica, I., Morris, R., Karger, D., Kaashoek, F.M., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. In: 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149–160. ACM Press, New York (2001)
8. Huang, L., Wu, Z., Pan, Y.: Virtual and Dynamic Hierarchical Architecture for e-Science Grid. International Journal of High Performance Computing Applications 17(3), 329–347 (2003)
9. Huang, L.: VIRGO: Virtual Hierarchical Overlay Network for Scalable Grid Computing. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 911–921. Springer, Heidelberg (2005)
10. Huang, L.: A P2P service discovery strategy based on content catalogues. Data Science Journal 6, S492–S499 (2007)
11. Giordano, M.: DNS-Based Discovery System in Service Oriented Programming. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 840–850. Springer, Heidelberg (2005)
12. Mockapetris, P.: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. Specification, RFC1035 (1987), <http://www.ietf.org/rfc/rfc1035.txt>
13. Huang, L.: Distributed Universal Resource Name Resolution based on Distributed DNS (Internet Draft) (2008), <http://www.tools.ietf.org/html/draft-licanhuang-dnsop-urnresolution-01>
14. Huang, L.: Distributed DNS Implementation in IPv6 (Internet Draft) (2007), <http://tools.ietf.org/id/draft-licanhuang-dnsop-distributeddns-02.txt>
15. Berners-Lee, T., Fielding, R., Irvine, U.C., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax, RFC 2396 (1998), <http://www.ietf.org/rfc/rfc2396.txt>
16. Berners-Lee, T., Masinter, L., McCahill, M.: Uniform Resource Locators (URL), RFC 1738 (1994), <http://www.faqs.org/rfcs/rfc1738>
17. Daigle, L., Gulik, D. v., Iannella, R., Faltstrom, P.: URN Namespace Definition Mechanisms, RFC 2611 (1999), <http://www.ietf.org/rfc/rfc2611.txt>
18. Wong, S.C., Tan, V., Fang, W., Miles, S., Moreau, L.: Grimoires: Grid Registry with Metadata Oriented Interface: Robustness, Efficiency, Security. IEEE Distributed Systems Online 6(10) (2005)
19. VIRGO: HomePage of VIRGO, <http://virgo.sourceforge.net/>
20. DNSJava: HomePage of DNSJava, <http://www.dnsjava.org/>