Grid Resource Discovery Model Based on the Hierarchical Architecture and P2P Overlay Network

Fei Liu, Fanyuan Ma, Shui Yu and Minglu Li

Department of Computer Science and Engineering, Shanghai Jiaotong University,
Shanghai, P. R. China, 200030
{liufei001, fyma, yushui, mlli} @sjtu.edu.cn

Abstract. The Grid technology enables large-scale sharing and coordinated use of networked resources. The kernel of computational Grid is resource sharing and cooperating in wide area. In order to obtain better resource sharing and cooperating, discovering resource must be efficient. In this paper, we propose a Grid resource discovery model that utilizes the flat and fully decentralized P2P overlay networks and hierarchical architecture to yield good scalability and route performance. Our model adapts efficiently when individual node joins, leaves or fails. Both the theoretical analysis and the experimental results show that our model is efficient, robust and easy to implement.

1 Introduction

The kernel of computational Grid is resource sharing and cooperating in wide area. We propose a grid resource discovery model that utilizes the flat decentralized P2P overlay networks. P2P overlay networks, such as Chord [1], CAN [2] and Tapestry [3], are always used in file-sharing systems in which the discovery result has to perfectly mach the request. But resource discovery in Grid are in the absence of a naming scheme. GRIP [4] is used to access information about resource providers, while the GRRP [4] is used to notify register nodes services of the availability of this information. To deal with the problem we combine P2P and hierarchical architecture in our model. In our model nodes in Grid can be classified into two types. Register nodes are those that do not provide any resource but only manage the nodes that provide resource. This mode apply P2P architecture to register nodes, which makes the framework of register better scalable than traditional register architecture such as centralized register, hierarchical register etc. Resource nodes are the other nodes that provide resource and take on a little manage work.

2 Constructing Register and Resource Provider P2P Network

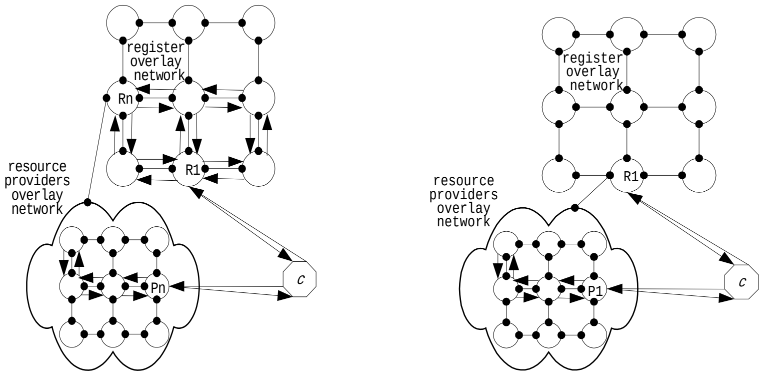
The scalability of centralized architecture is bad because the register node is its bottleneck. So in our model, we combine P2P overlay network and hierarchical architectures. There are two P2P overlay networks. One is register P2P overlay network that consists of register nodes the other is resource provider P2P overlay network that is constructed by resource provider nodes. We assume that IP is the identifier of node. We can regard IP as a point in a virtual 4-dimensional Cartesian coordinate space which is defined as S_a*={(0,0,0,0),(255,255,255,255)}*. We assume the 4 axes of S_a are *x, y, z, w*. The first register node *R1* holds space S_a. When the second register node *R2* joins, S_a is divided into two parts averagely. One parts is controlled by *R1* and the other is held by *R2*. The central point of the space controlled by *R1* is closer to *R1* than the other space. *R1* records the IP of *R2* and space controlled by *R2* and *R2* records IP of *R1* and space controlled by *R2*. In this way the neighbor relationship between *R1* and *R2* sets up. After the register overlay network contains *m* node *[R1, R2 ,…, Rm]*, the $(m+1)^{th}$ register node joins which will split the space controlled by node *Rn {1<=n<=m}* which IP is closest to IP of *Rm+1* into two parts.

We assume *P1* is a resource provider node that IP is (162.146.201.148) and it knows the register node *R1* (28.18.36.112). Then *P1* sends its GRIP data to *R1*. *R1* checks IP of *P1* and its space then transfer the GRIP data of *P1* to its neighbor. The neighbor of *R1* does the same as *R1* and Finally the GRIP data is received by *R2*. After *R2* receives the GRIP data, it records the static resource and only dynamic resource types and sends the feedback to *P1*. Owning to the dynamic resource changes over time, if *R2* holds the dynamic resource, it has to refresh dynamic resource periodically which consume much *R2* resource and result in low scalable performance. So we only store dynamic resource types in register nodes. The feedback contains the IP of *R2*, space controlled by *P1* (in 4-dimensional Cartesian coordinate space), spaces controlled by *P1*'s neighbors and static resource and dynamic resource types of *P1*'s neighbors. If there is no neighbor of *P1*, *P1* will hold the space controlled by *R2*. Then *P1* sends its GRIP data to its neighbors and its neighbors record the dynamic resource of *P1*. *P1* will send message to its neighbors to refresh the record of its dynamic resource periodically. Thus there are at most 9 nodes know the dynamic resource of *P1*. Here *P1* registers successfully and *P1* join the resource provider P2P overlay network. In this way, we can construct the resource provider P2P overlay network.

3 the Process of Resource Discovery

If a client *c* knows any node in Grid, it can get at least one register node from that node. Then *c* sends request to register node *R1* to obtain some resource. After receiving the request *R1* checks the space controlled by it whether contains the resource *c* requesting. If the space contains the static resource *c* asking for, *R1* tells *c* that the static resource is found and sends the location of the resource to *c*. Otherwise *R1* transfers the request to its neighbors and waits for the response. If one of its neighbor has that resource, *R1* select the neighbor and sends its IP to *c*, then *c* resends request to the selected neighbor of *R1* to ask for resource. If all the neighbors of *R1* have not the resource, *R1* extends the search extent to make more register nodes check its resource until at least one register node *Rn* finds the resource and resource provider node *Pn* which belongs to the space controlled by *Rn* can provide the resource.

If *R1* has the dynamic resource *c* asking for, it randomly select a resource provider *P1* which provides the resource and maximum of the resource is not smaller than *c* requesting, then *R1* sends the IP of *P1* to *c*. After *c* receives the feedback, it sends message to *P1* to check the current load of the resource. If the free resource matches the request of *c*, *P1* accepts the request of *c* and allocate the free resource to *c*. Otherwise *P1* use experience-based+random algorithm to transfers the request of *c* to its neighbor (Fig. 1-a). The experience-based+random is as follows: nodes learn from experience by recording the requests answered by other nodes. A request is forwarded to the peer that answered similar requests previously. If no relevant experience exists, the request is forwarded to a randomly chosen node. If *R1* has not the dynamic resource *c* asking for, it do the same as the static resource discovery to find a register node *Rn* which contain the dynamic resource and the maximum of the resource is not smaller than *c* requesting (Fig. 1-b).


register overlay network
Rn
R1
resource providers overlay network
Pn
c
register overlay network
R1
resource providers overlay network
P1
c

(a) (b)

Fig. 1. The process of dynamic resource discovery

4 Experimental Results

In our experiment, we use GT-ITM models to obtain 2 groups of nodes. One group contains 5000 nodes that are used as resource providers and the other group contains 100 nodes that are regarded as registers. 20 kinds of static resources and 50 kinds of dynamic resources are in our simulator. Each kind of static resource has 10 instances and every kind of dynamic resource has 10 instances too. These resources are allocated randomly for resource providers.

In our experiment, we investigate the influence of the number of nodes to the number of hops. We activate 1000, 2000, 3000, 4000, 5000 resource providers respectively. We randomly select 40 resource providers as client to send requests. The 40 resource providers are divided into 4 groups. Fig. 2 shows that the number of hops increases slightly with the number of the computing nodes increasing. However there is still some slight disobedience in the curve because the resource which client search may be in its local node or neighbor or some near nodes. The four curves are very similar that shows our model has fine stability.

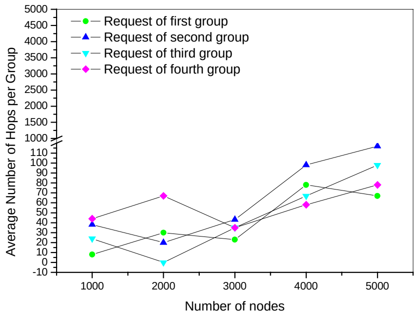


Fig. 2. Average number of hops per group for different resource provider numbers

5 Conclusions

We propose a Grid resource discovery model that utilizes the flat and fully decentralized P2P overlay networks and hierarchical architecture to yield good scalability and route performance. Register nodes are organized as P2P overlay network that removes the single-point failure and improve the performance of scalability. The register overlay network do some auxiliary manage work for resource provider overlay network, which improve the route performance. Both the theoretical analysis and the experimental results show that our model is efficient, robust and easy to implement.

Acknowledgements. This paper is supported by 973 project (No.2002CB312002) of China, and grand project of the Science and Technology Commission of Shanghai Municipality (No. 03dz15027 and No. 03dz15028)..

References

1. Ratnasamy, S.m Francis, P.m Handley, M.m Kapp, R.m And Shenker, S. A scalable content addressable network. In ACM SIGCOMM (2001)
2. Stoica, I., Morris, R., Karger, D.m Kaashoek, M., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for internet applications. In ACM SIGCOMM (2001)
3. Zhao, B.m Kubiatowicz, J.m And Joseph, A. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Tech. Rep. UCB//CSD-01-1141, U.C. Berkeley, 2001
4. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman: Grid Information Services for Distributed Resource Sharing. Proc. Of the 10th IEEE int’1 Symp. On high Performance Distributed Compting (2001)