A Binary Tree Based Approach to Discover Multiple Types of Resources in Grid Computing

Leyli Mohammad khanli[1], Ali Kazemi Niari [2] and Saeed Kargar[2]
[1]Assistance Professor, Cs Dept. University of Tabriz, Tabriz, Iran, [2] MS student, Islamic Azad University-Tabriz Branch, Tabriz, Iran,
{l-khanli@tabrizu.ac.ir, a.kazemi.n@gmail.com, saeed.kargar@gmail.com}

***Abstract*:** Today grid technology considered as a solution to solve complex problems. Grid included a large number of heterogeneous resources. So, a resource discovery mechanism should be able to discover these heterogeneous and dynamic resources in such environment. In many of the previous methods, there are not possible for discover multiple types of resources in a framework simultaneously. In this paper, we propose a binary tree based to discover multiple types of resources for grid environment. In this method, for discover multiple types of resources, we send a unique request. We compare our method with other methods using simulation. The experimental results show that our method for resource discovery has better performance.

***Keywords*:** Grid, Discover multiple types of resources, Tree structure.

1. Introduction

Grid presented as a way to solve special problems [1]. This environment included a large number of heterogeneous resources. Resource discovery; i.e. finding users request is the most important challenges in the grid [2].

A user request may be a combination of multiple types of resources. This challenge should be able to discover the resources in a distributed environment.

A recent decentralizes method [3], uses tree structure for resource discovery in grid environment. Also, current method has presented a resource discovery method with one resource and different qualities. Therefore, discover more resources with different qualities needed to send separate and large number requests. In our method, we use tree structure, like [4]. In our proposed method, all tree nodes use the same table for identifying the resources available in it. The table contains all information related to the resources and their attributes. Our mechanism considers a combined package in every node in the grid and every node fills in its combined package using the table.

In our method users can be found to several resources simultaneously. So, in instead of the previous methods, it would be possible for the user to access to multiple types of resources only by one request.

This paper is organized as follows: Section 2 includes related works. In section 3, we explain our proposed method. Section 4 involved experimental results and section 5 would be the conclusions.

2. Related work

Grid provided infrastructure for sharing resources with different types in environment. Resource discovery approaches is of special importance in grid environment. Many of the methods proposed for resource discovery in grid.

One of the methods presented for resource discovery problem was the so called matchmaking one to solve the Condor problems [5]. Some researchers also used method matchmaking as a new method in their works [6]-[10]. In the current method, entities advertise their requirements and characters to the environment and a matchmaking service has to find a match between advertisement and entities.

Some resource discovery methods use resource brokers to match the resources between resource consumers and resource providers to find the best resource. Resource brokers use some factors in decision making resource availability, software/hardware capabilities, network bandwidth and resource price [11],[12].

In [3], Ruay-Shiung Chang et al. proposed a resource discovery tree for grid which using bitmap. Any request by the user, should be changed into bitmap form. If a request reaches to one of the tree nodes, it will be compared with its local resource bitmap in which the data related to the local resources kept in (by AND operation). If the requested resource not found, (provided that current node is a leaf node), then the request will be sent to the parent node. Otherwise, the requests are compared with index bitmap in which the data related to children resources kept in. If the resource not found in children, too, the request will be sent to the parent node and until the requested resource be found, current process continues.

In our previous work [4], we proposed another method in resource discovery which uses a weighted tree in grid environment. In the current method, users request directed to the target node through unique paths i.e. the requests are delivered to nodes in a

determined format and the nodes send the request to their chosen children using the reserved information; but if no resource exists at the node and its children, it will be delivered to their parent node.

Our algorithm has some differences with previous ones:

In our method, every node has maximum of 2 children which makes it easier for the parent nodes to keep and manage the children.

For all nodes, we use the same table. It means that information available in tree would be accessible to all nodes only once, regarding the available resources in grid environment. Table information will not change and the nodes fill their related combined package using the table and the users only manipulate the small packages.

3. Binary tree based approach to discover multiple types of resources

3.1 Request resources in different types

There are many heterogeneous resources in the grid which are geographically distributed. These resources have different types. The user may simultaneously need multiple types of resources. For example, machine (Dell PowerEdge 3250), processor type (Itanium 2, 1.5 GHz), operating system (vista), and so on. So, there should be a method in which the user may request resources in different types. Resources and their types should be known for the nodes. Managing the information of these heterogeneous resources, we propose a *resource-table* and some packages which discussed. First, we insert a list of resources with their types in the *resource-table*. Current table would be supplied to all nodes in the grid as a catalog. Figure 1 shows an example of *resource-table*. Any node has a "*Local Combined Package*" (LCP) which can extract the related code to its local resources and insert in the LCP.

Resource Identifier	Machine	Processor type	Operating system
1	Dell PowerEdge 3250	Itanium 2 1.5 GHz	vista
2	HP ProLiant BL20p G2	Xeon 3.06 GHz	xp
3	IBM pSeries 615 Model 6C3	POWER4+ 1.2 GHz	se7en
4	Sun Blade 2500	UltraSparc 1.28 GHz	linux

Figure 1. An example of resource-table

For example, in Figure 2, we show a LCP with the following resources: machine (Dell PowerEdge 3250), processor type (UltraSparc 1.28 GHz), operating system (XP). The node places the related code for their local resources on the LCP, using resource- table of Figure 1. Besides the *Local Combined Package*, the node which has children needs another combined package for children which called "*Children Combined Package*" (CCP) to determine resources available in children (Figure 3). So, the CCP contains information of collected resources from children.

When a node receive a request on resource discovery, it will be compared with LCP and then with CCP. The information of packages will be used to find the requested resources.

Machine	Processor	OS
1	4	2

Figure 2. An example of Local combined package.

Machine Left child	Machine Right child	Processor Left child	Processor Right child	OS Left child	OS Right child
10	11	35	20	84	35

Figure 3. An example of children combined package.

3.2 Initialization of our tree nodes

In this subsection, initialization means collecting data of children nodes by parent nodes during building the tree. Any node can place the received data of children in the CCP and send complete information to the parent node.

Figure 4 shows a typical grid environment with seven nodes which are connected with a tree structure. The tree is a binary one; i.e. any node can contain maximum of two children. The advantage of the tree would be easy management of children.

Here, we discuss how data collected for Machine resource. First, information of leaf nodes (D, E) which would be 2 and 3 are sent to node B and for nodes F and G (i.e. 1, 3) it will be sent to node C. Node B which received number 2 from his left child (D), insert it in left position of Machine (CCP of node B), but number 3 which is received from right child (E) are insert in right position of Machine. Node C does the same process for its children nodes (F, G); i.e. for CCP of node C, left position=1 and right position=3.

Nodes B and C should send their information to the parent node (A). We explain this process by more details. Left position of CCP=2, right position of CCP=3, LCP=2 which are in node B, are added to each other (2+3+2→7) and then send to the parent

node (A). The same process performs for node C until number 7 insert in right position of Machine for node A (1+3+3→7).

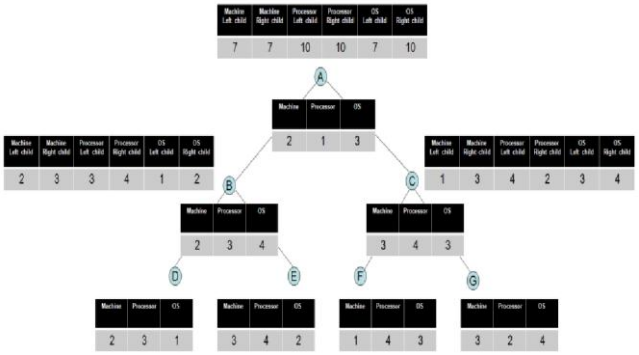


Figure 4. An example of typical grid environment.

3.2 Resource discovery in typical grid environment

In initialization, information related to the resources sent from children node to parent one. Therefore, all nodes obtain complete information of resources available in their children and descendants. Finally, the final form of tree is created. Now, the requested resources of users can discover in the current tree structure. There would be a frame which the users can record their requested resources and deliver it to a node. Our goal is that the user would be able to simultaneously request multiple types of resources. In Figure 5, the user requests the following resources: machine (HP ProLiant BL20p G2) processor type (POWER4+ 1.2 GHz), operating system (vista) as a request package. After forming the request package, the user should deliver it to one of the tree node. Figure 6 shows an example for resource discovery on a binary tree.

Figure 5. An example of request package

We suppose that in Figure 6, the request delivered to node G. Regarding the current algorithm, the request first compared with LCP of node G, but no match is found, so, the request delivered to parent node (C). Node C also compares it with its LCP and again no match is found. The request is compared with left positions of CCP (because the request comes from right child) and because no information of requested resources is found, then requested package forward to the parent node. Node A compared this request package with own LCP, then compared it with left positions of own CCP.

Here, a probability match is found for the requested resource in left child of node A (node B). So, the request package is forward to node B. Operation of node B would be as follows:

Node B first compared request package with its LCP and because this LCP not equals with request package then compared it with CCP. After this operation, node B found information about requested resources in its left child i.e. node D and right child i.e. node E and send request package to these nodes. The requested package that sent by node B which reaches node E, when compared with LCP of node E, can't discover the user's requested resources in this node, but the requested resources discovered in node D. Finally, send a success message to the node G and discovered resources would be reserve for user.

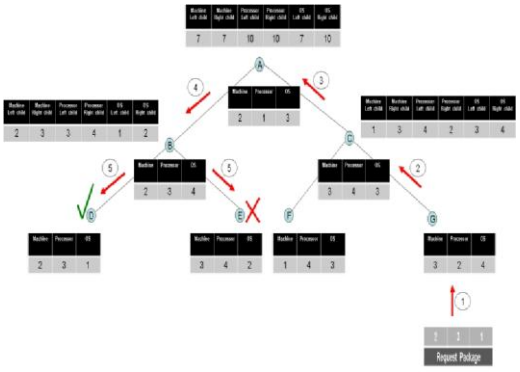


Figure 6. An example for resource discovery.

4 Experimental results

4.1 Setup

We performed the experiment at MATLAB environment and the resources and requests randomly distributed between nodes. Our experiment performed on a binary tree and compared with other approaches.

It has to be noticed that the methods which have been compared with ours, are the ones proposed for the discovery of just one resource based on a tree structure in the grid environment. Because, we didn't meet applied methods able to simultaneous discovery of multiple types of resources, so some methods recently presented for the discovery of one resource with different attributes on a tree will be compared with our method. In this process, we supposed that the current methods send the user's request separately and then discover these resources for the user.

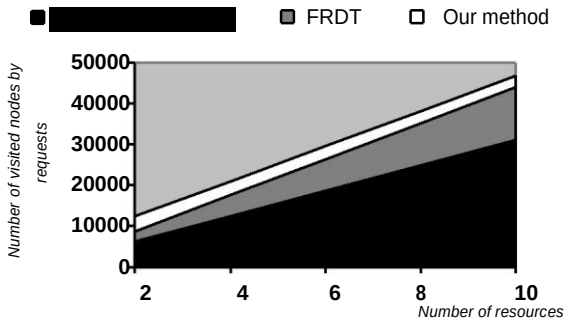
For the other methods, we suppose that after the

resources discovered and reserved for the user and before using them, the location of the discovered resources have to be compared and if all belong to one node, then they would be usable by the user. But in order to reduce the complexities available in simulation, we suppose for all simulations in other methods that all of the separated requests discover the resources from one node, but it is rarely possibility to occur.

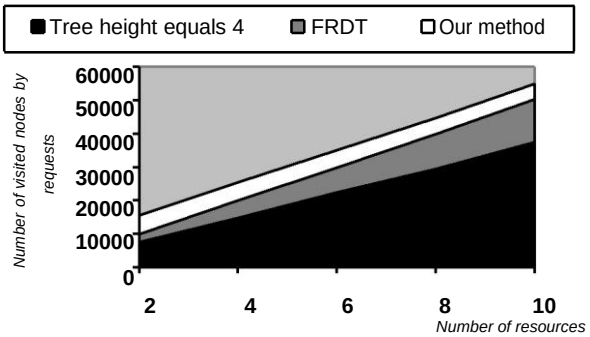
4.2 Simulation results

We conducted the experiments with different number of requested resources; i.e. one of the experiments the user requests only one resource and in some other experiments, user simultaneously requests two resources and so on. In other methods, for example, if user demands 3 resources, user should send 3 separate requests, but, in our method, just one request will be sent.

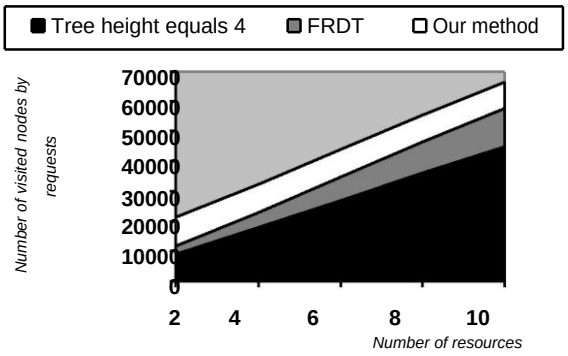
In the first experimental tests, the number of nodes which the requests send in tree method and FRDT with height 4, compared with our method. We supposed 300 users that everyone requested different number of resources (Figure 7 (a, b, c, d)).



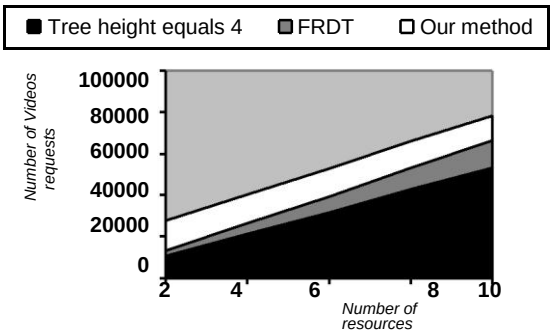
(a) # nodes= 85



(b) # nodes=156



(c) # nodes=259



(d) #nodes=400

Figure 7. The number of visited nodes for 300 requests.

In the second experimental tests, the average number of nodes in which the requests are sent, shows for methods flooding-based algorithm, MMO [13],[14] and our method. The results presented in Figure 8. In this test 300 users requesting one resource.

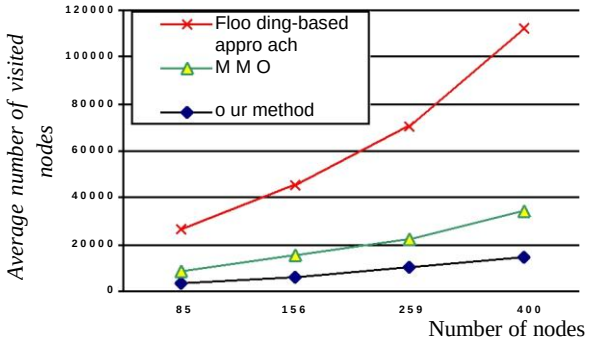


Figure 8. Average number of nodes that equests are forwarded using different approach.

In the last experimental tests, number of visited nodes during resource discovery and updating are show in tree and FRDT methods and also in our method. In the first environment, we supposed 100

users that everyone requested five resources (Figure 9) and in the second environment we supposed 100 users that everyone requested ten resources, each time (Figure 10). As observed in Figure 9 and Figure 10, if the number of user increases and every user request more resources, each time, our method would be more efficient comparing other methods.

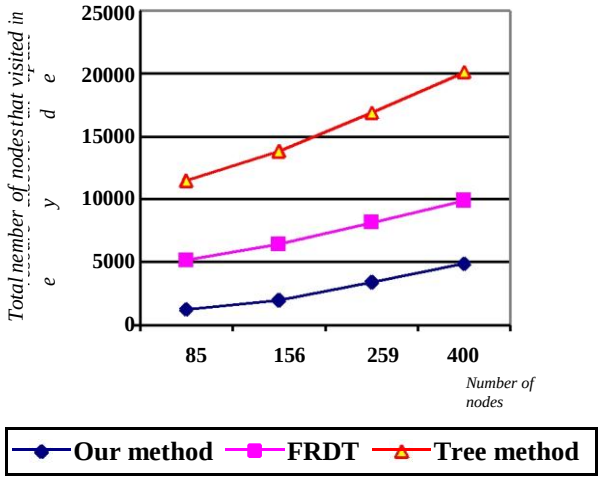


Figure 9. Total number of visited nodes by requests for 100 users each user requests five resources**.**

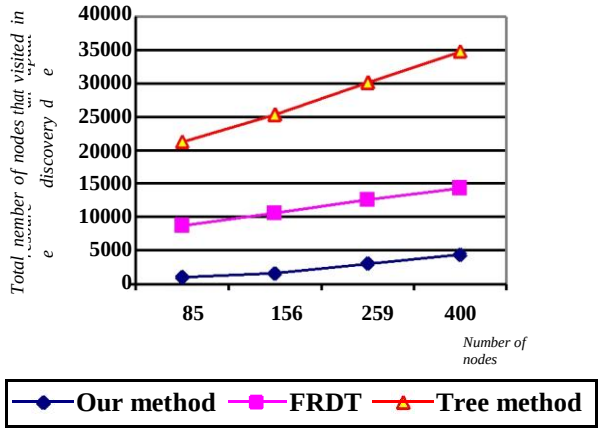


Figure 10. Total number of visited nodes by requests for 100 users each user requests ten resources**.**

5 Conclusions and Future Work

In this paper, we proposed a resource discovery mechanism which is on the basis of binary tree. So, every node has maximum of 2 children which makes it easier for the parent nodes to keep and manage the children. In our method, the users can send multiple requests in the form of a unique request.

In the future, if we can find a method that by using it, we discovered several resources on a free tree mode (not merely binary), we can improve the resource discovery mechanism significantly.

References

[1] Ian Foster, Carl Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2003.

[2] YiLi, G., FangPeng, D., Wei, L., ZhiWei, X.:VEGA Infrastructure for Resource Discovery in Grids. J. Comput. Sci. & Technol, pp.413-422 (2003)

[3] Shiung Chang, R., Shuo Hu, M.: A resource discovery tree using bitmap for grids. Future Generation Computer Systems (2009)

[4] L.M. Khanli, S. Kargar, FRDT: Footprint Resource Discovery Tree for grids, Future Gener. Comput. Syst. 27 (2011) 148–156.

[5] R. Raman, M. Livny, M. Solomon, Matchmaking: distributed resource management for high throughput computing, hpdc, in: Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC-7'98), 1998, p. 140.

[6] K.I. Karaoglanoglou, H.D. Karatza, Resource Discovery in a dynamical grid based on Re-routing Tables, Simulation Modelling Practice and Theory 16 (2008) 704–720.

[7] Rajesh. Raman, Matchmaking Frameworks for Distributed Resource Management, University of Wisconsin-Maddison, 2001.

[8] Ye Zhu, Junzhou Luo, Teng Ma, Dividing Grid Service Discovery into 2-stage matchmaking, ISPA 2004, LNCS, vol. 3358, 2004, pp. 372–381.

[9] S .Tangpongprasit, T .Katagiri, H .Honda, T .Yuba, A time-to-live based reservation algorithm on fully decentralized resource discovery in grid computing, Parallel Computing 31)6( )2005(529-543.

[10] Muthucumaru Maheswaran, Klaus Krauter, A parameter-based approach to Resource Discovery in Grid Computing Systems, GRID, 2000.

[11] Bradley, A., Curran, K., Parr, G., 2006. Discovering resource in computational GRID environments. The Journal of Supercomputing, 35, 27–49.

[12] D. Lacks, T. Kocak, Developing reusable simulation core code for networking: The grid resource discovery example, The Journal of Systems and Software 82 (2009) 89-100.

[13] M. Marzolla, M. Mordacchini, S. Orlando, Resource discovery in a dynamic environment, in: Proceedings of the 16th International Workshop on Database and Expert Systems Applications, DEXA'05, September 3_7, 2005, pp. 356_360.

[14] M. Marzolla, M. Mordacchini, S. Orlando, Peer-to-peer systems for discovering resources in a dynamic grid, Parallel Computing 33 (4_5) (2007) 339_358.