A Weight-based Query Forwarding Technique for Super-peer-based Grid Resource Discovery

Mahdi Mollamotalebi
Department of Computer Engineering
Buinzahra branch
Islamic Azad University
Buinzahra, Iran
mmahdi2@live.utm.my

Raheleh Maghami
Department of Computer Engineering
Buinzahra branch
Islamic Azad University
Buinzahra, Iran
maghamy@qiau.ac.ir

Abdul Samad Ismail
Faculty of Computing
Universiti Teknologi Malaysia
Skudai, Malaysia
abdsamad@utm.my

***Abstract*—Grid computing environments include heterogeneous resources shared by a large number of computers to handle data and process intensive applications. The required resources must be accessible for the grid applications on demand, which makes resource discovery a critical service. In recent years, different techniques are provided to index and discover grid resources. Response time and message load during the search process highly affect the efficiency of resource discovery. This paper proposes a technique to forward the queries based on the resource types accessible through each neighbor in super-peer-based grid resource discovery approaches. The proposed technique is simulated in GridSim and the experimental results indicated that it is able to reduce the response time and message load during the search process especially when the grid environment contains a large number of nodes.**

Keywords-grid computing; resource discovery; super-peer; weight-table; query forwarding

I. INTRODUCTION

Grid computing aims to handle data and process intensive applications. Grid environments typically include a large number of nodes each of which owns one or more resources to be shared and used by the applications [1, 2]. With regard to the continuous increasing of networks' bandwidth and resource varieties, grid computing is emerging as the next-generation computing platform in government, science, and business [3]. Resource discovery is the process that takes the execution requirements of grid applications, searches the network for required resources and returns a set of grid nodes including resources matched to the application requests [4].

Grid environments are inherently large-scale and dynamic. As a result, grid resource discovery is a time and message consuming process which affects the efficiency of the entire grid [5]. The response time and message load are two important factors for evaluating the efficiency of resource discovery approaches. The response time refers to the time between issuing a resource request and returning the resource owner addresses, and message load indicates the number of messages transferred between the grid nodes per second during the search process [6, 7]. The super-peer is a prevalent resource discovery approach in grid environments. In super-peer-based approach, the large-scale grid environment is divided into several small-scale environments in order to increase the scalability of resource discovery. Some other approaches are based on hierarchical [8-12], peer-to-peer [13, 14], agent-based [15], and centralized [16] structures. This paper proposes a weight-based technique to improve the super-peer-based resource discovery approaches in terms of the response time and message load.

In the super-peer structure, each node has the role of either regular-peer or super-peer. The super-peer nodes are connected to each other as peer-to-peer. In addition, each super-peer node acts as a central server for a set of regular-peers. A regular-peer sends the resource information and resource queries to its related super-peer node [17, 18]. When a regular-peer needs to explore the grid for some required resources, it sends a query message to its local super-peer. If the local super-peer finds the desired resources in its local index, it returns the resource reference to the requesting regular node. Otherwise, it forwards the query to its neighboring super-peer nodes. Some recent proposed super-peer-based resource discovery techniques for grid environments are described in the following paragraphs. In [19], the authors proposed a resource discovery system which uses the super-peer nodes in a framework based on Chord [13]. The Chord has a single ring to index the resources, but this technique uses multiple rings. Each node of a ring can keep the addresses of some resources matching the queries. One node in a ring does not necessarily own the real resources but it may maintain their IP addresses. A super-ring is used to keep the nodes pointing to other rings. This technique decreases the message load however it highly depends on the information existed in the cache. It also benefits from the inherent load balancing of Chord. On the other hand, it cannot control the traffic of different rings. It uses some monitoring considerations but in dynamic conditions, the monitoring potentially needs to transfer more messages.

In [12], the authors proposed a technique in which the super-peer nodes make relation between different physical organizations and maintain the local domain's resource information. Some contact peers are defined in each domain to handle the grid registrations. Different domains can have their own grid frameworks and they are communicated only through the super-peers. In this technique, the query is not forwarded to

all the neighboring nodes to traverse the network. Instead, the receiving super-peer node forwards the query to a selected number of neighbors. To this end, the super-peer nodes record the number of query-hits obtained by each neighbor in previous search attempts. In the next search processes, the super-peer node forwards the received query to the neighbors who had the highest number of query-hits in the past and by this way, it decreases the message loads.

In [20], the authors proposed a resource discovery technique for superpeer-based structure using semantic and fuzzy theory. It considered delay as a major parameter of fuzzy system. The amount of semantic similarity between node services is the other parameter of the fuzzy system. Thus, it used delay, bandwidth and semantic similarity as the input parameters of fuzzy system to create semantic overlay network. The scheme consists of node grouping and resource discovery, and fuzzy theory is used in both phases. It uses a hybrid P2P structure where the nodes are divided into groups. This technique could increase the number of adequate discovered resources and reduce the response time. On the other hand, the precision of resource discovery is decreased because it emphasizes on the geographical factors to group and search the resources. In [21], the authors presented a super-peer based technique for the discovery of resources in grid environment which supports multi-attribute and range queries. The technique summarizes resource information to improve its scalability. It also exploits Routing Indices (RIs) to handle the priority of tasks and resolves the problem of access to all domains' summaries. It uses clustering to perform the summarization and builds a tree of gathered information. Then it creates a summary of the database by applying two steps, pruning and leafing. In the first step, branches are pruned at a given depth such that no branch is deeper than the chosen depth. Although its scalability, this technique is not as efficient in terms of message load and response time due to the processing overheads related to the summarization steps.

II. DESCRIPTION OF WEIGHTED-SP

The Weight-based super-peer (Weighted-SP) technique aims to reduce the message load and response time of the super-peer resource discovery by limiting the neighbor nodes receiving the forwarded queries. It is performed by adding a weight table to indexing nodes indicating the weight of their neighbors to be chosen. In each indexing node, the weight table indicates the number of resource types that are accessible through each path. By this way, when a query is received by an indexing node, it forwards the query to the limited number of neighbors with the highest weights for the requested resource type. It is not possible in super-peer structure to be aware of the resource information comprehensively during the join or update processes. This is because the super-peer nodes are related to each other as P2P without any hierarchical organizations. Also publishing all the resource join and update messages to other super-peer nodes will impose a dramatic message load on the system. Thus, Weighted-SP updates the number of resource types in the weight tables during the search process instead of the join or update processes. For this purpose, whenever a query reached the resource owner, the resource discovery process returns the found resource owner address to the requester node and then, the weight table of the super-peer nodes in the backward path of the successful query is updated. The Weighted-SP scheme is illustrated in Figure 1. Each super-peer node keeps the resource information of its local regular nodes and the weight table. The weight table includes a row for each neighbor of current super-peer. In each row, there are columns corresponding to the resource types existing in the grid. The weight tables of super-peer nodes are empty in the initial state of grid environment establishment. In this situation, super-peer nodes which have no information about the resource types in their weight table, forward the queries to all of their neighbors. But after each query forwarding, the weight tables are updated for the requested resource type.

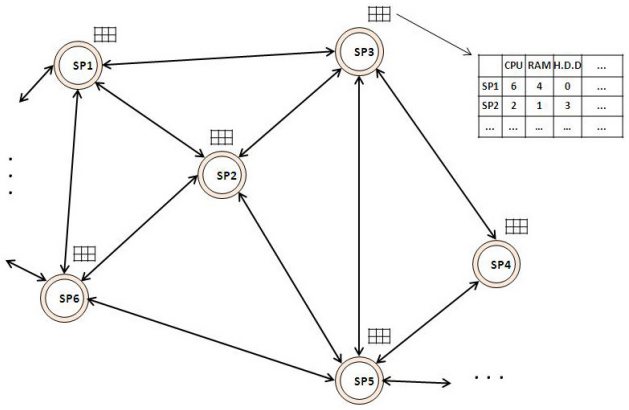

Fig. 1. The Schema of Weighted-SP Resource Discovery

The search process is initiated by regular nodes in the grid. The regular node issues its resource query and sends it to the local super-peer node. The super-peer node first investigates the local regular nodes to find the required resource. If the resource is found, the address of found resource owner will be returned to the requester node and the search process will be terminated. Otherwise, the super-peer node sorts its weight table information related to the requested resource type and sends the query to a specified percent of neighboring nodes which highest number of resource types are accessible through them. This process is continued until the required resource is found or all the super-peer nodes are investigated. By this manner, the query is forwarded to the neighbors with the highest probability of success at each step of the search process.

Weighted-SP uses the backward messages to be informed about the successful search results and updates the weight tables of the super-peer nodes located in the path of successful search processes. The weight tables of super-peer nodes are empty during the initial state of grid environment establishment. In this situation, super-peer nodes which have no information about the resource types in their weight table, forward the queries to all of their neighbors. After each query forwarding, the weight tables are updated for the requested resource type. The implementation design for Weighted-SP is presented in Figure 2.

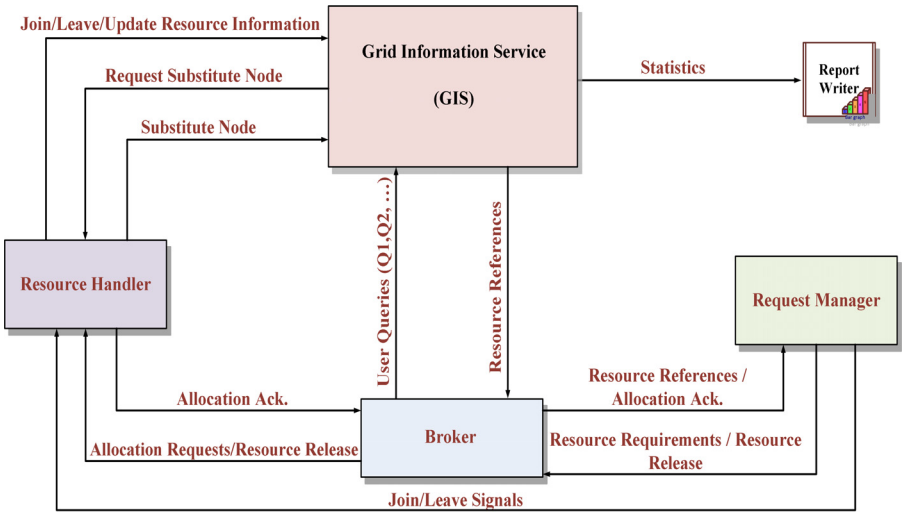

Fig. 2. Implementation Design of Weighted-SP

The Request Manager module generates the resource requests and nodes' join/leave events. The generated resource requests are assigned to users randomly and sent to the module Broker. The module Broker is responsible to schedule the resource requests and responses. It acts as a medium between the resource discovery system and grid environment. It analyzes the received resource requests according to the receive time, separates multiple attribute requests to make different independent queries, schedules all queries and forwards them in turn to the GIS. Then it waits for the resource owner address to be returned from the GIS module. When the matched resource owner address is discovered by the GIS and returned, module Broker sends the allocation request to the found resource owner. The indexing nodes and resource information are handled by the GIS. Almost all modules are related to the GIS to transfer the resource information or get the search results. When a resource is allocated to an application, its value will be updated and the new resource information is sent to the GIS. The Resource Handler module is responsible to send any resource information changes caused by the resource allocation, join, and leave events to the GIS and subsequently the GIS updates all indices related to the changed resource information. The Resource Handler also manages the allocation requests sent by the Broker. If a resource is available, then it will be allocated to the requesting application for specified period of time which is determined by the initial resource requester. The GIS is responsible to index resource information and locate the resource owners according to the received requests. After finding the resource owner addresses, the GIS returns them back to Broker to handle the allocation process. The Report Writer module receives statistical information of requested resources, transferred messages, and successful results of the search processes to analyze and make the appropriate reports according to the comparison and analysis requirements.

For each experiment, r resource requests are issued. The resources join and leave the network and these events cause to transfer more messages. Moreover, after finding a required resource, the found resource is allocated to the requester node. The resource information changes caused by the allocation events also should be considered. The update messages that are related to the allocation events are estimated by the (1) such that s is the success rate of queries, r is the number of issued resource requests, and k is the rate of multi-attribute query issues that request two resources.

$$n_1 = 2(k+1) \cdot s \cdot r \quad (1)$$

$$n_2 = h_1 \cdot \frac{n}{N} \quad (2)$$

The messages related to join and leave events are estimated by (2) where n represents the number of regular nodes and N is the number of super-peer nodes. Assuming h_1 as the rate of join and leave events, the equation specifies the messages that are issued for the regular nodes' join and leave events. In addition to the above mentioned update messages, some messages are transferred for random change events that are applied by the system on the resources. Each grid node shares one to three resources. Equation (3) calculates these messages as well the allocation and join/leave events such that R is the maximum number of resources in each grid node and h_2 is the rate of change events on the resources.

$$m_c = n_1 + n_2 + \sum_{i=1}^{R} (h2 \cdot \frac{n}{N} \cdot i) \quad (3)$$

Considering the number of issued resource requests as r, and k as the rate of multi-attribute queries, the number of resource requests reached to an super-peer node is estimated by (4).

$$m_r = \frac{2 \cdot (k+1) \cdot r}{N} \quad (4)$$

$$m = \sum_{i=1}^{N} (m_c + m_r \cdot (1 - \frac{i \cdot P}{N})) \quad (5)$$

Considering the update and request messages at each super-peer node, and with regard to P percent of neighbors chosen to forward the queries by super-peer nodes, the overall transferred

messages can be estimated by (5). The implementation methods for Weighted-SP including the join/leave, request issue, and search processes are presented and described in the following paragraphs. Figure 3 presents the algorithm of the join events for grid resources in Weighted-SP.

```
JoinRequest()
    BEGIN
    IF FreeNodes.isNotEmpty
        BEGIN
        NewNode = FreeNodes.fetch
        Node[NewNode].isActive = true
        Resources.add(Node[NewNode].resource)
        FOR i = 0 to activated_SP.length
            IF Activated_SP[i].isNotFull
                BEGIN
                Node[NewNode].SP= Activated_SP[i]
                Activated_SP[i].Nodes.add(NewNode)
                Break
                END
        END
    END
```

Fig. 3. The Algorithm of grid Nodes' Join Events in Weighted-SP

Once the request manager issues a join event, the method JoinRequest() will be performed. This method chooses one of the free nodes to join the grid environment. With regard to the random event issues of join and leave events, usually there are some free nodes available to be chosen. After choosing a free node, its status will be changed to active and its resource is added to the existing resources of grid environment. Then a super-peer node that has free capacity to index the new received node will be found and assigned to the joint grid node. Moreover, the new node is added to the regular nodes of the found super-peer.

Figure 4 presents the pseudo code of the leave events for grid resources in Weighted-SP. The method LeaveRequest() first chooses a random node to be left. The leave event can be issued for the grid node or a resource of grid node. It is specified by the request manager through the parameter 'type' that is set to 'complete leave' if the grid node should be left from the grid. If the leave event issued for a resource of an existing grid node, a random number between one to three is chosen and then the resource will be removed from the chosen grid node. Moreover, the request manager issues events to leave the super-peer nodes. In this cases, the method LeaveRequestSP() is performed that substitutes a free super-peer for the leaving one. For this purpose, the process copies the neighbors and related nodes of the leaving super-peer node to the new one.

The method RequestIssue() in Fig. 5 chooses a regular node as random to issue a resource request. With regard to possibility of issuing multi-attribute queries in Weighted-SP, and existence of one to three resources in each grid node, this process adds a random number of resources in the query. Considering that Weighted-SP uses the successful query messages to update the weight table of super-peers in the backward path, the super-peer of current regular node is kept in the global variable OriginSP. This variable is used subsequently in the method WTableUpdate. Then the query is forwarded to the local super-peer of current regular node.

```
LeaveRequest(type)
    BEGIN
    RandNode = Random(NumOfActivatedNodes)
    NumOfResPropsInNode = Node[RandNode].Res.length
    RelatedSP=Node[RandNode].SP
    IF type="complete_leave"
        BEGIN
        Activated_SP[RelatedSP].Nodes.remove(RandNode)
        Node[RandNode].isActive= false
        FreeNodes.add(RandNode)
        END
    ELSE
        BEGIN
        RandResToLeave = Random( NumOfResPropsInNode )
        Node[RandNode].Res[RandResToLeave].remove
        END
    END
LeaveRequestSP()
    BEGIN
    RandSP = Random(NumOfActivatedSPs)
    SubstituteSP = FreeSPs.fetch
    Activated_SP.add(SubstituteSP)
    FOR i=1 to Activated_SP[RandSP].Neighbours.length
        Activated SP[SubstituteSP].Neighbours[i]=
                Activated_SP[RandSP].Neighbours[i]
    FOR k=1 to Activated_SP[RandSP].Nodes.length
        Activated_SP[SubstituteSP].Nodes[k]=
                Activated_SP[RandSP].Nodes[i]
    Activated_SP[RandSP].Remove
    FreeSPs.add(RandSP)
    END
```

Fig. 4. The Algorithm of grid Nodes' Leave Events in Weighted-SP

```
RequestIssue()
    BEGIN
    RandNode = Random(NumOfActivatedNodes)
    MultiRes=3
        NumResReq=Random(MultiRes)
        For i=1 to NumResReq
            BEGIN
            RandRes=Random(Resources.length)
            ResReq = Resources[RandRes]
            Node[RandNode].Request.query.add(ResReq)
            END
    OriginSP= Node[RandNode].SP
    Send(Node[RandNode],
         Node[RandNode].SP,
         Node[RandNode].Request)
    END
```

Fig. 5. The Algorithm of Resource Request Issue in Weighted-SP

The ReceivedRequest()method that is shown in Figure 6 is responsible to handle the search process of requested resources. The super-peer node investigates its related regular nodes to satisfy the received query. If one of its regular nodes could satisfy the query, the super-peer performs the WTableUpdate process in order to update the weight table of super-peers in backward path and search process is terminated. If the super-peer node did not found any matched resource in its related regular nodes, it sets the number of neighbors to forward the query considering the specified percentage by the user. The weight table of current super-peer node is copied in TempWTable to prevent information changes in the original weight table during the search process.

The number of branches to forward the query is set considering the percentage of forwarding branches specified by the user. Then the weight table of current indexing node is copied to TempWTable in order to find the branches that highest number of requested resource type is accessible through them. This copy of the weight table is needed to prevent changes in the original weight table of the indexing node during the forward steps. At each step of query forwards, the neighbor with highest number of accessible resource type is found and the query is forwarded to it. The index of appropriate neighbor to forward the query is kept by the variable Hindex. In order to remove the highest value form the next forward steps, its value is set to zero in the TempWTable

```
ReceivedRequest(SenderSP,CurrentSP,Request)
    BEGIN
    Request.Visited.add(CurrentSP)
    FOR i=1 to CurrentSP.Nodes.length
        IF Request.query = CurrentSP.Nodes[i].Resource
            BEGIN
            WTableUpdate(SenderSP,Request.query,CurrentSP)
            Return CurrentSP.Nodes[i].Address
            END
    Count=percentage * CurrentSP.NumOfNeighbours
    TempWTable=CurrentSP.WTable
    FOR i=1 to Count
        BEGIN
        index=1
        Highest=TempWTable.fetch(CurrentSP.Neighbours[index],
                                    Request.query.ResType)
        Hindex=index
        WHILE index < CurrentSP.NumOfNeighbours
            BEGIN
            index=index+1
            TempVal=TempWTable.fetch(CurrentSP.Neighbours[index],
                                        Request.query.ResType)
            IF Highest < TempVal
                BEGIN
                Highest= TempVal
                Hindex=index
                END
            END
        TempWTable.set(CurrentSP.Neighbours[Hindex],
                        Request.query.ResType, 0)
        Send(CurrentSP,CurrentSP.Neighbours[Hindex],Request)
        END
    END
```

Fig. 6. The Algorithm of Resource Discovery in Weighted-SP

The WTableUpdate process shown in Figure 7 is responsible to update the weight table of superpeer nodes in backward path upon finding the required resource. Its first parameter is the address of super-peer node that should update its weight table. The query is send as the second parameter to inform the receiving super-peer that this update is related to which resource type. The third parameter is the address of current super-peer. The receiving super-peer adds the value of resource type that is accessible through its neighbor. Then it fetches the previous super-peer in order to perform the update process. The update process is repeated until reaching the original super-peer node related to the initial requester regular node.

```
WTableUpdate(SP,Query,NeighbourSP)
    BEGIN
    SP.WTable.add(NeighbourSP,Query.ResType)
    PreSP=Request.Visited.fetch
    IF PreSP <> OriginSP
        WTableUpdate(PreSP,Request.query,SP)
    END
```

Fig. 7. The Algorithm of Weight Table Update in Weighted-SP

III. SIMULATION RESULTS AND EVALUATION

In order to perform the simulation experiments for the proposed technique, it is implemented in Java under the simulation toolkit GridSim version 5.2. The hardware platform includes CPU Intel Core 2 Duo 2.93 GHz and 4 GB of RAM. The experiments are performed by different number of grid nodes i.e. 1000, 5000, 10000, and 20000. Also, to investigate the impacts of the forwarded neighbors, three values i.e. 25, 50, and 75 of the forward percentages are considered for the indexing nodes. In addition, to have the reliable results, 1000 resource requests are issued per experiment and the average results of the message load and response times are calculated. In order to compare and evaluate the results, an existing super-peer-based technique [21] is simulated in the same hardware and software platform. This technique is referred as SP in the rest of this paper.

The message load in the performed experiments is measured by dividing the number of messages transferred during the search process to the time spent for issuing all the requests. Also, the response time is taken by calculating the average value of all the requests' response time. The response time of each request is the duration between issuing the request and finding the result. The experimental results of Weighted-SP are presented in the following paragraphs. Table I**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** presents the message load of Weighted-SP for different number of grid nodes and forwarding percentage values. Also, the reduction rates of the message load for Weighted-SP relative to SP are presented in Table II.

TABLE I. THE MESSAGE LOAD OF WEIGHTED-SP FOR DIFFERENT NUMBER OF GRID NODES AND FORWARDING PERCENTAGE VALUES

	Number of grid Nodes			
	1000	**5000**	**10000**	**20000**
Pr =25%	4.6	13.6	30.6	61.1
Pr =50%	6.1	17.58	40.52	80.1
Pr =75%	7.39	22.34	51.4	104.73
SP	8.41	25.74	60.49	124.97

The reduction rates of message load indicate that choosing the lower percentages of neighbors to forward the queries results to the higher rates of reductions in the message load. The reduction rate of message load is not equal to the excluded neighbors since the weight tables are used only to limit the query forwards. The update messages between super-peer nodes and their related regular nodes are transferred regardless to the weight tables. **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** presents the average response time of

SP and Weighted-SP for different number of grid nodes. The response time of resource discovery is affected by the weight-based forwards of the queries because the neighbors with lower accessibility of required resource types are excluded from the search process. Therefore, the average response time for the issued queries is reduced.

TABLE II. THE MESSAGE LOAD REDUCTION RATES OF WEIGHTED-SP FOR DIFFERENT FORWARDING PERCENTAGES

	Number of grid Nodes				Overall reduction rate
	1000	**5000**	**10000**	**20000**	
Pr =25%	45.2%	47%	49.3%	51.1%	48.15%
Pr =50%	27.4%	31.7%	33%	35.9%	32%
Pr =75%	12%	13.2%	15%	16.1%	14%

TABLE III. THE AVERAGE RESPONSE TIME OF SP AND WEIGHTED-SP FOR DIFFERENT NUMBER OF GRID NODES

	Number of grid Nodes			
	1000	**5000**	**10000**	**20000**
Weighted-SP	0.19	0.32	0.45	0.59
SP	0.2	0.35	0.51	0.67
Reduction Rate	5%	8.5%	11.5%	11.9%

In order to identify the appropriate values of the forwards percentage, the success rates of the search processes are measured. Because the weight-based query forwarding excludes some neighbors form the search domain, it affects negatively the success rate of the search process. Table IV presents the success rate of Weighted-SP for different number of grid nodes and query forwarding percentages. The success rate of Weighted-SP is less than SP in all cases. It is because Weighted-SP excludes some neighbors from the investigation during the search process. But the success rate does not increase significantly when the queries are forwarded to more neighbors. It is because the success rate of Weighted-SP depends to the information of weight tables more than the forwarding percentage. If the weight tables do not contain useful information about the resource types, this technique cannot forward the queries to the appropriate grid nodes. In Weighted-SP, the super-peer nodes are not informed about the resource information of other ones, and the weight tables are updated by using the results of resource queries. Thus more number of requests enriches the information of weight tables. In order to investigate the impacts of number of issued queries on the success rate of Weighted-SP, the experiments are repeated with three more values i.e. 5000, 10000, and 20000 of the issued resource queries.

Table V presents the success rate of Weighted-SP for 1000 grid nodes and different number of issued queries. The results indicate that issuing more number of queries increases the success rate of Weighted-SP. It is because the weight table's information of super-peer nodes in Weighted-SP depends on the search results of issued queries. The higher number of issued queries enriches the information of weight tables, and subsequently the queries can be forwarded more efficiently toward the appropriate neighbors. As a result, Weighted-SP could reduce the message load and response time of the search process by excluding some neighbors form the search domain based on the specified percentage of query forwards. Because the results of the search processes are used to update the weight table of super-peer nodes, more number of issued queries enriches the weight tables of Weighted-SP and subsequently it increases the success rate of this technique.

TABLE IV. THE SUCCESS RATE OF WEIGHTED-SP FOR DIFFERENT NUMBER OF GRID NODES AND QUERY FORWARDING PERCENTAGES

	Number of grid Nodes			
	1000	**5000**	**10000**	**20000**
Pr =25%	44.8%	45%	45.2%	45.5%
Pr =50%	47.2%	47.6%	47.9%	48.1%
Pr =75%	49.7%	51%	51.7%	52.3%

TABLE V. THE IMPACT OF NUMBER OF ISSUED QUERIES ON THE SUCCESS RATE OF WEIGHTED-SP

	Number of Requests			
	1000	**5000**	**10000**	**20000**
Pr =25%	44.8%	47.1%	51.7%	57.8%
Pr =50%	47.2%	58.8%	64.1%	74.9%
Pr =75%	49.7%	69.1%	75.3%	79.6%
Average of Success Rates	47%	58%	63%	71%

IV. CONCLUSION

The resource discovery service responsible to find the resources required by grid applications has an important role in grid computing systems. This paper proposes a weight-based technique to improve the super-peer-based grid resource discovery solutions in terms of message load and response time. In the proposed technique, each indexing node keeps a weight table consisting of its neighbors and the number of different resource types that are accessible through each neighbor. In super-peer structure, the resource information of other super-peer nodes cannot be collected during the join or update processes, and such information is collected by using the successful search results. The experimental results of the proposed technique indicated that it reduces the message load and response time during the search process. With regard to the contents of the weight tables, choosing higher percentage of neighbors to forward the queries causes the system to act similarly to conventional super-peer-based systems. On the other hand, choosing a very low percent of neighbors excludes many resource owners from the search domain. The results showed that choosing more than 75% of neighbors could result to suitable values of the message load and response time while keeping the success rates high. In addition, higher number of resource requests enriches the weight tables of the proposed technique in the super-peer-based structure and causes it to act more efficiently.

REFERENCES

[1] D. Puppin, S. Moncelli, R. Baraglia, N. Tonellotto, F. Silvestri, "A grid Information Service Based on Peer-to-Peer", 11th International Euro-Par Conference, Lisbon, Portugal, August 30-September 2, 2005

[2] M. A. Arafah, H. S. Al-Harbi, S. H. Bakry, "grid computing: a STOPE view", International Journal of Network Management, Vol. 17, pp. 295-305, 2007

[3] M. R. Islam, M. T. Hasan, G. Ashaduzzaman, "An architecture and a dynamic scheduling algorithm of grid for providing security for real‐time data‐intensive applications", International Journal of Network Management, Vol. 21, pp. 402-413, 2011

[4] M. Hauswirth, R. Schmidt, "An Overlay Network for Resource Discovery in grids", Sixteenth International Workshop on Database and Expert Systems Applications, pp. 343-348, 2005

[5] A. Hameurlain, D. Cokuslu, K. Erciyes, "Resource discovery in grid systems: a survey", Int. J. Metadata Semant. Ontologies, Vol. 5, pp. 251-263, 2010

[6] B. Beverly Yang, H. Garcia-Molina, "Designing a super-peer network", 19th International Conference on Data Engineering, pp. 49-60, 2003

[7] D. T. P. Trunfio, P. Fragopoulou, H. Papadakis, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, S. Haridi, "Peer-to-Peer Models for Resource Discovery on grids", Future Generation Computer Systems, Vol. 23, No. 7, pp. 864–878, 2007

[8] A. Padmanabhan, S. Ghosh, S. Wang, "A Self-Organized Grouping (SOG) Framework for Efficient grid Resource Discovery", Journal of grid Computing, Vol. 8, pp. 365-389, 2010

[9] Y. Gong, F. Dong, W. Li, Z. Xu, "VEGA infrastructure for resource discovery in grids", Journal of Computer Science and Technology, Vol. 18, pp. 413-422, 2003

[10] Y. Ma, B. Gong, L. Zou, "Resource discovery algorithm based on small-world cluster in hierarchical grid computing environment", Seventh International Conference on grid and Cooperative Computing, pp. 110-116, 2008

[11] R.-S. Chang, M.-S. Hu, "A resource discovery tree using bitmap for grids", Future Generation Computer Systems, Vol. 26, pp. 29-37, 2010

[12] C. Mastroianni, D. Talia, O. Verta, "Designing an information system for grids: Comparing hierarchical, decentralized P2P and super-peer models", Parallel Comput., Vol. 34, pp. 593-611, 2008

[13] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications", SIGCOMM Comput. Commun. Rev., Vol. 31, pp. 149-160, 2001

[14] P. Merz, K. Gorunova, "Fault-tolerant Resource Discovery in Peer-to-peer grids", Journal of grid Computing, Vol. 5, pp. 319-335, 2007

[15] M. Marzolla, M. Mordacchini, S. Orlando, "Resource Discovery in a Dynamic grid Environment", Sixteenth International Workshop on Database and Expert Systems Applications, pp. 356-360, 2005

[16] D. Cokuslu, A. Hameurlain, K. Erciyes, "grid resource discovery based on centralized and hierarchical architectures", International Journal for Infonomics, Vol. 3, pp. 227-233, 2010

[17] C. Mastroianni, D. Talia, O. Verta, "A super-peer model for resource discovery services in large-scale grids", Future Generation Computer Systems, Vol. 21, pp. 1235-1248, 2005

[18] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, S. Haridi, "Peer-to-Peer resource discovery in grids: Models and systems", Future Generation Computer Systems, Vol. 23, pp. 864-878, 2007

[19] J. Salter, N. Antonopoulos, "An optimized two-tier P2P architecture for contextualized keyword searches", Future Generation Computer Systems, Vol. 23, pp. 241-251, 2007

[20] S. Javanmardi, S. Shariatmadari, M. Mosleh, "A novel decentralized fuzzy based approach for grid resource discovery", International Journal of Innovative Computing, Vol. 3, No. 1, pp. 23-32, 2013

[21] A. C. Caminero, A. Robles-Gomez, S. Ros, R. Hernandez, L. Tobarra, "P2P-based resource discovery in dynamic grids allowing multi-attribute and range queries", Parallel Computing, Vol. 39, pp. 615-637, 2013