Distributed ACS Algorithm for Resource Discovery in Grid

Seyed Mehdi Fattahi and Nasrollah Moghadam Charkari
Computer Engineering Department, Tarbiat Modares University
Tehran, Iran
fattahi_mehdi@modares.ac.ir, moghadam@modares.ac.ir

***Abstract*— Grid is an environment that allows sharing of resources that are managed by diverse, independent administrative organizations that are geographically distributed. The main objective of grid is to enable users to solve problems using the available geographically distributed resources. To fully utilize these resources, effective discovery techniques are necessities. Grid resource discovery refers to the process of locating satisfactory resources based on user requests. In this paper, we propose a mobile agent method based on peer to peer model for the resource discovery problem that has essential characteristics for efficient, self-configuring and fault-tolerant resource discovery and is able to handle dynamic attributes. This method employs an Ant Colony System (ACS) algorithm to locate the required resources. The innovation in this algorithm is to eliminate centralized control and provide node autonomy. We compare our approach with *flooding algorithm* and *Ant System algorithm* in terms of *Average Hop count, Average Success rate* and *Querying traffic*. Our experiments show that the proposed method performs better than flooding algorithm and Ant System algorithm alike.**

***Keywords*— Resource Discovery; Grid; Ant Colony System Algorithm**

I. INTRODUCTION

A Grid is a very large scale, generalized distributed network that can scale to Internet-size environments with machines distributed across multiple organizations and administrative domains. Efficient and effective resource discovery in this environment is then critical. Grid resource discovery refers to the process of locating satisfactory resources based on user requests.

A. Resources

In Grid, resources belong to different resource classes. A resource class is a model for representing resources of the same type. Each resource class is defined by a set of attributes which specify its characteristics. Each resource has a specific value for each attribute defined by the corresponding resource class [1].

An example of resource class is "computing resource" that defines the common characteristics of computing resources. These characteristics are described by attributes such as "OS name", "CPU speed", and "Free memory". An instance of the "computing resource" class has a specific value for each attribute, for example, "OS name = Linux", "CPU speed = 1000MHz", and "Free memory = 1024MB". Table 1 lists some examples of Grid resource classes. [2].

TABLE 1
RESOURCE CLASS EXAMPLES

Resource class	Description
Computing resource	Computing capabilities provided by computers, clusters of computers, etc.
Storage resource	Storage space such as disks, external memory, etc.
Device resource	Specific devices such as instruments, sensors, etc.
Software resource	Operating systems, software packages, Web

	services, etc.
Data resource	Various kinds of data stored in file systems or databases.

The goal of resource discovery in Grids is to locate resources that satisfy a given set of requirements on their attribute values.

B. Ant Colony Algorithms

The basic idea of an ACO algorithm is from real ants which search their environment for food. In ACO, a set of software agents called *artificial ants* search for good solutions to a given optimization problem. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a weighted graph. The ants incrementally build solutions by moving on the graph. The solution construction process is stochastic and is biased by a *pheromone model*, that is, a set of parameters associated with graph edges whose values are modified at runtime by the ants [3]

1) Advantages of Ant Colony Algorithms

Ant Colony Algorithm based systems are very flexible and robust with respect to environmental constraints and disturbances which makes them very attractive for technical realizations [4]. Moreover, Ant Colony Algorithms inherits some important advantages such as:

Scalability: The number of individuals can be adapted to the network size.

Fault tolerance: Since the behavior of an ant is not controlled by a centralized entity, the loss of a few individuals does not cause catastrophic failure.

Adaptation: The ant can react to environmental changes due to the fact that each individual has the ability to adapt. This leads to a high value of flexibility.

Speed: Changes in the network can be spread very quickly among network users.

Modularity: Individuals act independently of other network layers.

Autonomy: Little or no human control is required.

Parallelism: Operations of individuals are executed in a parallel manner.

2) Main ACO Algorithms

Several special cases of the ACO metaheuristic have been proposed in the literature. Here we briefly overview the two most successful ones: ant system (AS) and ant colony system (ACS).

Ant system (AS) was the first ACO algorithm to be proposed in the literature (Dorigo et al. 1991, Dorigo 1992, Dorigo et al. 1996). Its main characteristic is that the pheromone values are updated by all the ants that have completed the solution.

The first major improvement over the original ant system to be proposed was ant colony system (ACS), introduced by Dorigo and Gambardella (1997). The first important difference between ACS and AS is the form of the perform pheromone update. ACS pheromone update is performed only by the best ant, that is, only edges that were visited by the best ant are updated.

II. RELATED WORK

There are many projects undergoing to address the issue of service discovery in the grid. Traditionally resource discovery in grids is mainly based on centralized or hierarchical models. These models are not scalable and could be the potential bottleneck of performance and security and single point of failure.

Condor is a resource management system for compute-intensive jobs [5].Condor adopts a centralized scheduling model.

Recently, several research projects have investigated techniques for P2P Grid systems which let us share resources (computers, databases, instruments, and so forth) distributed across multiple organizations.

The P2P Grid is emerging as a promising platform for executing large-scale, resource intensive applications. Iamnitchi et al. propose resource discovery approach in [6] based on an unstructured network similar to Gnutella combined with more sophisticated query forwarding strategies taken from the Freenet overlay network. Several systems exploiting DHT-based P2P approaches for resource discovery in Grids have recently been proposed [7], [8].

Recently, some research efforts have been invested in applying Ant Colony Optimization systems to tackle the resource management and the self-organization of large-scale and distributed systems. Cao [9] employed an ant-like self-organizing mechanism to distribute jobs evenly among available resources. A hybrid Ant Colony Optimization (ACO) algorithm is proposed and designed to select appropriate schedules in a heterogeneous computing environment [10].

III. DISCOVERY OF RESOURCES

A. Overview

A typical Grid environment is a Virtual Organization (VO) which consists of Monitoring and Discovery System (MDS) and Grid resource providers [11]. All Grid resources are registered in the MDS. In this paper, we organize the MDSs of a large-scale Grid in a P2P manner instead of the traditional hierarchical architecture, and employ a Distributed ACS algorithm to discover the required resources registered in the MDSs. In fact each node in our Grid topology is a MDS and in this paper we use "node" to refer to MDS.

Many real-world Grids are scale-free networks. Scale-free networks' structure and dynamicity are independent of the network size [12]. Due to the dynamicity and scalability of Grid. illustrates an example of a scale-free network.

A scale-free network is a network whose degree distribution follows a power law, at least asymptotically. That is, the fraction *P(k)* of nodes in the network having *k* connections to other nodes goes for large values of *k* as $P(k) \sim k^{-\gamma}$ where γ is a constant whose value is typically in the range $2<\gamma<3$, although sometimes it may lie outside these bounds[13].

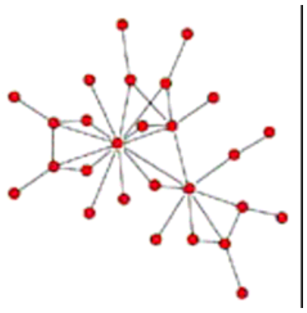

Figure 1-Scale free network

A simple model for scale-free networks was proposed by Barabasi and Albert [14]. In their model, the network starts with m_0 connected nodes and grows by the addition of one node at a time. When a node is added, m new connections from the new node to already existing nodes are made, and each already existing node can be chosen to receive a connection with probability proportional to its degree k.

In this paper, we used the Barabasi and Albert model to generate scale-free network topologies.

B. Proposed Algorithm

In our approach, we assume the Grid is directed graph G(V,E) that have n nodes and m edges. Each edge e(i, j) connect two node i, j where:

$$e\,(i,j)\ \varepsilon\ E,\ i, j\ \varepsilon\ V,\ i \leq n,\ j \leq n$$

Each node in the Grid has a number of resources. Here we assumed that each resource is identified with resource type attribute. And value of pheromone for each type of resource on each edge e(i,j) is maintained by τ_{ij}^{k} .

In this approach, some ants are generated for each request and each ant within each node must be using proposed algorithm to perform required actions. The proposed algorithm has two main functions:

- Transition Probability
- Update Pheromone

In this algorithm nodes and ants function independently and there is no central control on Grid. These functions are explained below:

1) Transition Probability

An ant which located in i-th node and follows a resource of k type, first search in local resources and if satisfying the demanded resource backward to initiated node otherwise it computes probability of selecting j node as the next node using the rate of pheromone on each edge of e(i,j) for a k type resources $P_{i,i}^{k}$. We defined the transition probability function to the next node as follows:

$$P_{ij}^{k} = \frac{\tau_{ij}^{k}}{\sum_{q} \tau_{iq}^{k}}, \quad q \in J_i \qquad (1)$$

Where, J_i represents the neighbors of the node and this ant in node i can be transmitted to them. By using the above function, transition probability values are computed for all neighbors of current node. Then, the ant selects its next destination according to these probabilities.

2) Update pheremones

The next step is Update pheromones for each resource demand on graph edges. In the proposed algorithm, after all generated ants for each request returned to initiated node, the best ant (the ant which find the best resource based on distance) is selected and the value of pheromones for demanded resource type on edges which been traversed by the ant update based on the following function:

$$\tau_{ij}^{k} = (1\text{-}\rho)\,\tau_{ij}^{k} + \rho\Delta_{ij}^{best} \;, \rho \in (0, 1] \qquad (2)$$
$$\Delta_{ij}^{best} = \frac{1}{L_{best}}$$

Where, ρ is the pheromone evaporation rate and Δ_{ij}^{best} is the amount of the pheromone which is set on the route by best ant and is computed based on the length of the shortest traversed path route by the request's ants.

IV. EXPERIMENTAL RESULTS

In our simulator, we use NS2 as the simulation tool. Our Grid is organized as a scale-free network and its topology is generated by Inet 2.0 [15]. Inet is a topology generator whose primary feature is that of emulating power-law relations for edge degrees. We use Inet results to configure Grid in NS2 tool.

In our simulation, number of nodes is 5000. Inet 2.0 used to generate a random topology to connect the nodes by using the Barabasi probability model.

2000 resources with 15 various types were generated and distributed across the network for all the performance measurements that examinant in this paper.

Twenty requests are generated simultaneously each Iteration.

For each request several ant (4 for this experiment) generate and The TTL of ants were set as 128. So the *maximum propagation depth* in *flooding based algorithm* was set to 128.

A. Compared approaches

We compare our approach with flooding-based algorithm and Ant System-based algorithm in terms of Average Hop count, Average Success ratio and Querying traffic.

With *flooding-based*, node N that searches for a resource R checks its resources list, and if the resource is not found there, N contacts all its neighbors. In turn, N's neighbors check their resources lists and if the resource is not found locally, they propagate the search message to all their neighbors. The method ends when either the resource is found or a maximum of *t* steps is reached.

AS-based algorithm performs like ACS algorithm. But with a difference in AS-based algorithm, all ants that find demand resource (regardless of being best ant) update the pheromone on traversed routes. This process is performed during returning to initiator node.

B. Performance measurements

We employ *Average Success ratio*, *Average Hop Count* and *Querying Traffic* as metrics to measure the performance of our approach.

1) Average Success ratio

The Average Success ratio represents the percentage of requests which find their required resources before reaching their ants TTL.

Figure 2 show the Average Success ratio of the Distributed ACS algorithm, the AS-based algorithm and flooding based algorithm. The system topology consisted of 5000 nodes. Each request produced four ants.

According to Figure 2, the average success ratio of the Distributed ACS algorithm, the AS- based algorithm and flooding based algorithm are 0.9467, 0.8362 and 0.8781, respectively.

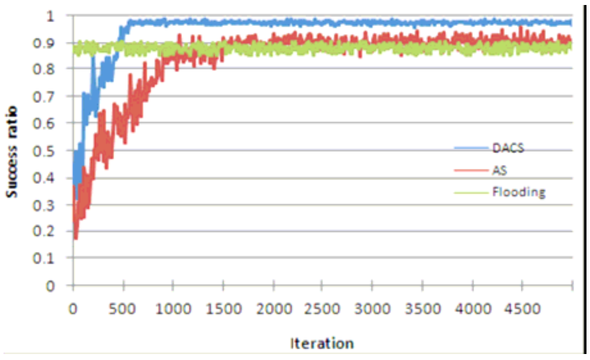

Figure 2- Average success ratio

Figure 2 indicates that DACS and AS based have learning phase. When the learning phase is completed, the system demonstrates very good success ratio.

2) Average Hop Count

The *Hop Count* indicates the network distance which the requests have to travel to discover the demanded resources. For this measurement, we averaged *Hop Count* for successful requests.

In our proposed algorithm, *Hop Count* is edges count that best ant travel to discover demanded resource. Likewise, in *AS based Algorithm* this edge count is *Hop Count*. And in *flooding based algorithm*, propagation depth is considered for this measurement.

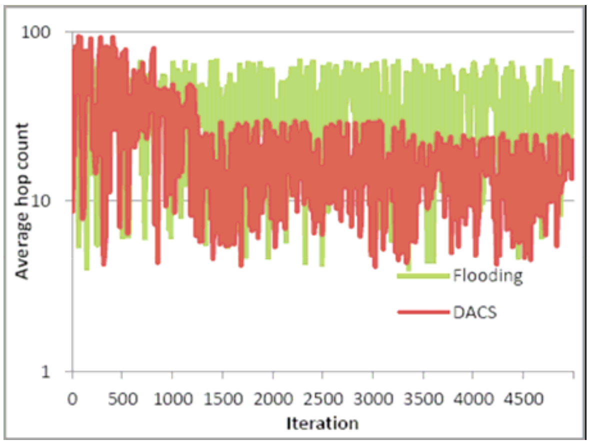

Figure 3- Average Hop Count DACS, Flooding

The average hop of DACS in Figure 3 illustrates a convergence trend with the increase of iteration, which passes the learning phase. Flooding based algorithm has a flat trend, because it isn't learning phase.

Note that the *Y* axis of Figure 3 and Figure 4 is in logarithmic scale.

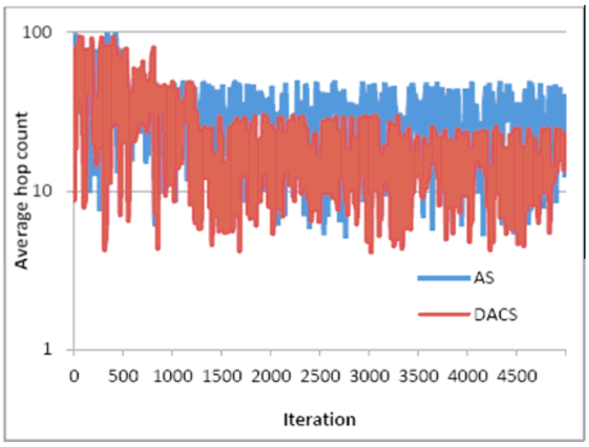

Figure 4- Average Hop Count DACS, AS

The proposed algorithm perform better than AS-based algorithm because only best ants change pheromone in it; whereas in AS-based algorithm all ants influence pheromones and cause other requests' ants to select their routs based on those pheromones which leads to an increase in hop count of found resources.

3) Querying Traffic

In this section we compare proposed algorithm, AS based algorithm and Flooding based algorithm in terms of Querying Traffic measurement.

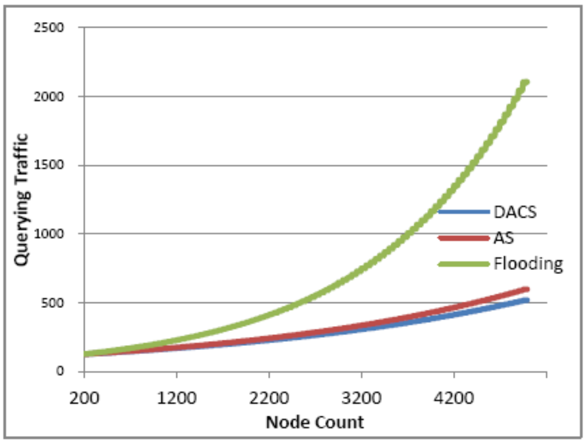

Figure 5- Querying Traffic

In this experiment, number of iteration is 3000 and algorithms are comparing in terms of querying traffic by number of nodes.

Figure 5 demonstrates that network traffic in proposed algorithm and AS based algorithm is fewer than flooding based algorithm.

V. CONCLUSION AND FUTURE WORKS

This paper, offers a solution for resource discovery in grid which can support range query. The proposed approach is completely decentralized and provides node autonomy. Other characteristics of proposed algorithm include:

- No need to central control
- Simplicity of implementation
- Low complexity in agents (ants)

One of the main restrictions of this approach is the simplicity of query which is defined with only one attribute. Therefore finding a suitable solution to support multi attribute query is a priority.

REFERENCES

[1] Domenico Talia, Paolo Trunfio and Jingdi Zeng, Peer-to-Peer Models for Resource Discovery in Large-Scale Grids: A Scalable Architecture,

[2] Andreozzi S., Burke S., Field L., Fisher S., Konya B., ambelli M., Schopf J., Viljoen M. and Wilson A., GLUE Schema Specification Version 1.2, Final Specification - 3 Dec 05. http://infnforge.cnaf.infn.it/glueinfomodel/index.php/Spec/ 12.

[3] Colorni A, Dorigo M and Maniezzo V, Distributed optimization by ant colonies, Proc of the first European conference on artificial life, pp 134–142, 1992.

[4] Roth M and Wicker S, Termite: ad-hoc networking with stigmergy, IEEE global telecommunications conference (GLOBECOM 03), San Francisco, 1–5 December 2003

[5] M. Litzkow and M. Livny, Experience with the Condor Distributed Batch System, Proc. IEEE Workshop on Experimental Distributed Systems, 1990.

[6] A. Iamnitchi and I.T.Foster, On fully decentralized resource discovery in grid environments, proceedings of the Second International Workshop on Grid Computing, London, UK, Springer-verlag,2001. 51-62.

[7] Andrzejak, A., Xu, Z., Scalable, Efficient Range Queries for Grid Information Services, Proc. of 2nd IEEE Int. Conf. on Peer-to-peer Computing (P2P'02), Link¨oping, Sweden, 2002.

[8] Cai M., Frank M., Chen J. and Szekely, P., MAAN: A Multi-Attribute Addressable Network for Grid Information Services, Journal of Grid Computing, vol. 2 n. 1 pp:3-14, 2004.

[9] Cao J, Self-organizing agents for Grid load balancing, Proc of the fifth IEEE/ACM international workshop on Grid computing, pp 388–395, 2004.

[10] Ritchie G and Levine J, A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments, Proc of the 23rd workshop of the UK

planning and scheduling special interest group, 2004.

[11] Deng Y and Wang F, A heterogeneous storage Grid enabled by Grid service, ACM SIGOPS Oper Syst Rev 41(1):7–13. Special issue: File and storage systems, 2007.

[12] Lv Q, Cao P, Cohen E, Li K and Shenker S, Search and replication in unstructured peer-to-peer networks, Proc of the 16th ACM int conf on supercomputing (ICS'02), New York, pp 84–95, 2002.

[13] ALBERT BARABASI and ERIC BONABEAU, Scale-free networks, SCIENTIFIC AMERICAN, 2003.

[14] A.-L. Barab´asi and R. Albert. Emergence of scaling in random networks. Science, 286:509, 1999.

[15] Cheng Jin, Qian Chen and Sugih Jamin, Inet: Internet Topology Generator, Department of EECS, University of Michigan.