J Supercomput (2009) 49: 4–21
DOI 10.1007/s11227-008-0214-0

Ant colony optimization inspired resource discovery in P2P Grid systems

Yuhui Deng · Frank Wang · Adrian Ciura

Published online: 28 May 2008
© Springer Science+Business Media, LLC 2008

Abstract It is a challenge for the traditional centralized or hierarchical Grid architecture to manage the large-scale and dynamic resources, while providing scalability. The Peer-to-Peer (P2P) model offers a prospect of dynamicity, scalability, and availability of a large pool of resources. By integrating the P2P philosophy and techniques into a Grid architecture, P2P Grid system is emerging as a promising platform for executing large-scale, resource intensive applications. There are two typical resource discovery approaches for a large-scale P2P system. The first one is an unstructured approach which propagates the query messages to all nodes to locate the required resources. The method does not scale well because each individual query generates a large amount of traffic and the network quickly becomes overwhelmed by the messages. The second one is a structured approach which places resources at specified locations to make subsequent queries easier to satisfy. However, the method does not support multi-attribute range queries and may not work well in the network which has an extremely transient population. This paper proposes and designs a large-scale P2P Grid system which employs an Ant Colony Optimization (ACO) algorithm to locate the required resources. The ACO method avoids a large-scale flat flooding and supports multi-attribute range query. Multiple ants can be employed to improve the parallelism of the method. A simulator is developed to evaluate the proposed resource

The initial work in this paper was conducted when Yuhui Deng was a research officer in Cranfield University. This paper was revised after Yuhui Deng joined EMC research China as a senior research scientist.

Y. Deng (✉)
EMC Research China, Beijing 100084, People's Republic of China
e-mail: deng_derek@emc.com

Y. Deng
e-mail: yuhuid@hotmail.com

F. Wang · A. Ciura
Center for Grid Computing, Cambridge-Cranfield High Performance Computing Facilities, Cranfield University Campus, Bedfordshire MK43 0AL, UK

discovery mechanism. Comprehensive simulation results validate the effectiveness of the proposed method compared with the traditional unstructured and structured approaches.

Keywords Peer-to-Peer · Grid · Resource discovery · Ant colony optimization · Complex adaptive systems

1 Introduction

Grid is a flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources [11]. The objective is to virtualize the geographically distributed resources and allow users and applications to access resources in a transparent manner. A large-scale and complex Grid application could involve hundreds or thousands of geographically distributed resources. With the growth of application requirements, a Grid system should have the ability to continue to provide satisfied capabilities including resources, performance, and fault tolerance when the system is increased in size.

Monitoring and Discovery System (MDS) [18] is a key component to provide basic mechanisms for resource discovery and monitoring in a typical Grid environment. Traditional approaches maintain a centralized MDS or a set of hierarchically organized MDSs to index resource information in the Grid. However, three reasons limit the efficiency of the traditional approaches. The first, the centralized approach, and the root node of the hierarchical method have the inherent drawback of a single point of failure. In the second, the approaches may result in system bottlenecks in a highly dynamic environment where many resources join, leave, and can change characteristics at any time. In the third, the approaches cannot scale well to a large-scale and geographically distributed system across the internet. Therefore, it is important to decentralize their functionalities to avoid the potential bottlenecks and the single point failure of the hierarchical or centralized approaches [24]. The Peer-to-Peer (P2P) model offers a prospect of robustness, scalability, and availability of a large pool of resources. The P2P philosophy and techniques could be used to implement nonhierarchical and decentralized Grid systems. 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 [3, 10, 12, 26]. The P2P Grid is emerging as a promising platform for executing large-scale, resource intensive applications.

The resources in Grid are often characterized by the dynamic, heterogeneous, and distributed features. Therefore, the description, discovery, and monitoring of resources are challenging problems. Efficient resource discovery is a crucial problem in the large-scale and geographically distributed Grid systems. The main requirements include: (1) low performance overhead; (2) fast response to query; (3) ability to locate the service provider with good Quality of Service (QoS); and (4) self-organizing capability to deal with dynamic joining and leaving of resources without centralized control [26]. Many research efforts have been invested in designing efficient resource discovery techniques for the large-scale and geographically distributed systems. Generally, the existing approaches can be classified into two categories. The first one is

unstructured approaches. The second one is structured methods. The unstructured approaches such as Gnutella are decentralized and require no precise control over network topology. The approaches propagate the query messages to all nodes in the system with a Time-To-Live (TTL) value to control the scale of search. However, the flooding based query method does not scale well because each individual query generates a large amount of traffic and the network quickly becomes overwhelmed by the messages [15]. The structured methods are decentralized and have a significant amount of structure. The topology of the system is tightly controlled and the resources are placed at specified locations which will make subsequent queries easier to satisfy [15]. Distributed Hash Table (DHT) is normally used to map contents into network addresses which can be located easily [23]. There are several projects which adopt the DHT method to locate resources [13, 21, 22]. These methods are very efficient in locating contents by one specific name or attribute, but not in supporting multi- attribute range queries [5]. Lv et al. [15] argued that the methods are completely invisible on the current network and may not work well with an extremely transient population in the network.

The universe is full of Complex Adaptive Systems (CAS) which are dynamic and highly decentralized networks consisting of many agents. The agents constantly act in parallel and react to the ambient environment in the network. Examples of these systems include human economies, the ecosystem, and the weather system. The systems are characterized by decentralized control, dynamicity, and large scale. The components of these complex adaptive systems interact with each other according to some simple local rules which result in self-organization and complex behaviors. A Grid system is by nature a complex combination of hardware, software, and network components. The geographically distributed nature, heterogeneity, dynamicity, and scalability of resources make a Grid as a CAS.

Recently, some research efforts have been invested in applying the natural CAS systems to tackle the resource management and the self-organization of large-scale and distributed systems. Cao [6] employed an ant-like self-organizing mechanism to distribute jobs evenly among available resources, thus achieving overall Grid load balance through a collection of very simple local interactions. A hybrid Ant Colony Optimization (ACO) algorithm is proposed and designed to select appropriate schedules in a heterogeneous computing environment [20]. Andrzejak et al. [1] proposed an adaptive resource allocation in dynamic Grid environments based on an ACO algorithm. The algorithm provides a suitable and adaptive placement of services or applications on resources, which prevents from overloading server environments and the communication infrastructure, keeps resource utilization and response times in balance, and achieves higher availability and fault-tolerance. Messor [19] implements a variation of ACO algorithm in which each ant carrying an object only drops it when it has finished wandering for a while without encountering other objects. In this instance, the ant colony tries to disperse the objects uniformly over the environment. The algorithm is suitable for distributing jobs across scalable, heterogeneous, and dynamic resources, while maintaining load balance.

In our earlier research [10], we designed a storage management architecture for the stable and trusted Grid oriented storage devices based on an existing Grid environment. A P2P model was employed to endow the architecture with dynamic scalability

and reliability. The evaluation of a small-scale system prototype demonstrated the effectiveness of the architecture. In contrast to that work, this paper attempts to explore the behaviors of a large-scale P2P Grid system. As a key function of a large-scale Grid system, the resource discovery mechanism should be able to locate a set of candidate resources for Grid users when they submit their requirements. We design a large-scale P2P Grid system which employs an ACO algorithm to locate the required resources. A simulator is developed to evaluate the ACO based resource discovery mechanism. Comprehensive simulation results give useful insights into the system behaviors.

The remainder of the paper is organized as follows. The system architecture is introduced in Sect. 2. Section 3 describes the ACO algorithm, and the design and implementation of the proposed P2P Grid system. Section 4 evaluates the system in a great detail. Section 5 discusses the topic and concludes the paper with remarks on main contributions of the paper.

2 Architecture overview

A typical Grid environment (e.g., Globus Toolkit 4) is a Virtual Organization (VO) which consists of a MDS, a Certificate Authentication (CA) center, and Grid service providers [9]. All Grid services are registered in the MDS. The interaction between the VO and Grid users is mediated through MDS which provides a virtual interface between the diverse resources and maintains a single logical view. There is typically one MDS per VO, but in a large VO, several MDSs are normally organized in a hierarchy. MDS provides service discovery, execution supervision, and monitoring of resource status information. A downstream and upstream mechanism is employed by different MDSs to exchange information automatically and efficiently. However, it is a challenge for the hierarchical architecture to manage the large-scale resources, while providing scalability. This paper organizes the MDSs of a large-scale Grid in a P2P manner instead of the traditional hierarchical architecture, and employs an ACO algorithm to discover and locate the required resources registered in the MDSs.

A scale-free network indicates that the outdegree follows a power law distribution. The power law distribution implies that the outdegrees of a small portion of the nodes are very high and the remaining nodes have low outdegrees. Scale-free networks' structure and dynamicity are independent of the network size. In other words, the networks have the same properties no matter what the number of its nodes is. Many real-world networks (e.g., computer network, social network, neural network) are scale-free networks. Many real-life P2P networks demonstrate a power law distribution which implies the networks are scale-free networks [15]. Figure 1 illustrates an example of a scale-free network which consists of 20 nodes. The topology is generated by BRITE [17]. In our simulator, the MDSs of a large-scale Grid system are organized as a scale-free network. Therefore, each node in Fig. 1 represents one MDS.

A Grid user should not see all of the complexities behind the Grid which could consist of large-scale, diverse, and geographically distributed resources. A Grid portal can offer an interface for users to launch applications which use the resources and

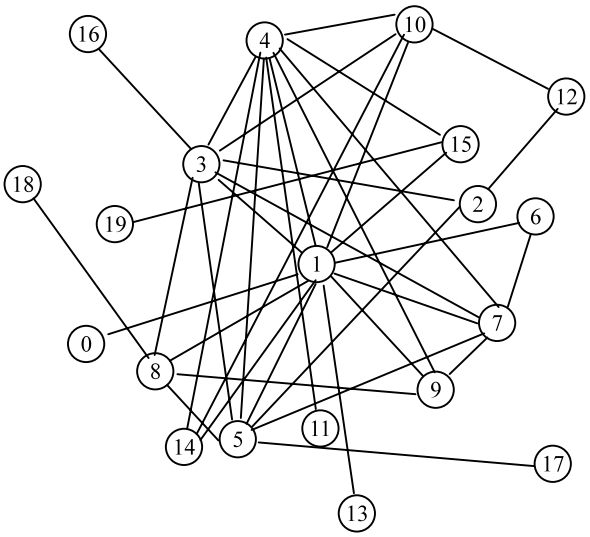


Fig. 1 An example of a scale-free network (20 nodes)

services provided by the Grid. We only focus on the resource discovery phase after the users gain an entry point into the Grid community. The entry point can be obtained by a Grid portal, out-of-band method, or other approaches. The major steps of resource discovery are labeled with the sequence number as defined in the following descriptions. (1) A request initiated by a user is redirected by the Grid portal or other methods to the nearest MDS. (2) Once the request is received, the MDS (we call it as a source MDS) searches its own resources. (3) If satisfied resources are found in the source MDS, the MDS will inform the user to communicate with the corresponding resources directly. (4) Otherwise, the source MDS will play the role of an agent to submit the request to the Grid in terms of different search algorithms. (5) If the required resource is found in a target MDS, the routing tables along the path between the source MDS and target MDS will be updated, and the user will communicate with the target MDS to request the resource.

3 Ant colony inspired Grid resource discovery

In this section, we will explain the ACO algorithm and discuss how to design and implement a P2P Grid by leveraging the ACO algorithm. In the following discussion, the nests and ants in the ACO algorithm represent nodes and user requests, respectively.

3.1 Ant colony optimization

The basic idea of an ACO algorithm is from real ants which search their environment for food [7]. Ants start from their nests and wander randomly. The ants which found food will return to their nests in terms of their memory and drop pheromone on trails. Other ants which come across such a trail will follow the trail to check the

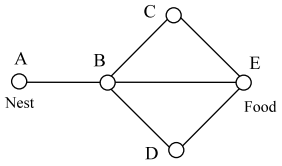


Fig. 2 An example of ant colony optimization

food instead of wandering randomly. If they find the food, they will return home and reinforce the pheromone on the trail. Ants make local decisions based on their owner observations and the local environment information modified by other ants instead of direct communication with each other. This kind of indirect communication between ants is called stigmergy. A key point is that the pheromone evaporates over time. The more time it takes for an ant to travel back to its nest, the more pheromone will be evaporated. When an ant reaches an intersection, the ant has to decide which branch to take. The ants which take a short branch march faster than those which take a long branch. Therefore, the pheromone density on the short branch remains higher. Other ants will more likely choose the branch in terms of the pheromone density. Eventually, all the ants which go to get the food will take the shortest branch. Figure 2 illustrates a simple example of the above scenario. Initially, the ants may take the paths of A→B→C→E, A→B→E, or A→B→D→E. After the initial stage, most of the ants will take the shortest path A→B→E.

We assume that the Grid is a directed graph $G = (V, E)$ which has n nodes and m edges. Each edge $e(i, j)$ of the graph connects two nodes i and j, where $e(i, j) \in E$, $i, j \in V, i \leq n, j \leq n$. The pheromone on the edge $e(i, j)$ is α_{ij} which can be changed by ants when they visit the edge and can be evaporated over time. An ant located in node i employs the pheromone α_{ij} to calculate the probability of choosing node j as the next node. We have the probability (that is, transition probability) as a function of time t:

$$P_{ij}(t) = f\left(\alpha_{ij}(t)\right) = \frac{\alpha_{ij}}{\sum_q \alpha_{iq}}, \tag{1}$$

where $q \in J_i$. J_i denotes the set of neighbor nodes where the ant located in node i can move to. The transition probability satisfies the constraint: $\sum_{j \in J} P_{ij} = 1$, where $J = J_i, \forall i \in V$. As time goes by, the pheromone on the edge $e(i, j)$ can be computed as follows:

$$\alpha_{ij}(t + 1) = (1 - \lambda)\alpha_{ij}(t) + \Delta\alpha(t), \quad \lambda \in (0, 1], \tag{2}$$

where $(1 - \lambda)\alpha_{ij}(t)$ indicates that the pheromone on the edge $e(i, j)$ evaporates with time, and $\Delta\alpha(t)$ denotes the amount of pheromone dropped by an ant on the edge $e(i, j)$.

3.2 Design and implementation

The system is designed as an overlay network which is built on the top of the existing network. Nodes in the system can be regarded as being connected by virtual or logical links. Each virtual or logical link corresponds to one or multiple physical links in the underlying network. Each node in the system has a unique identifier and is equal in

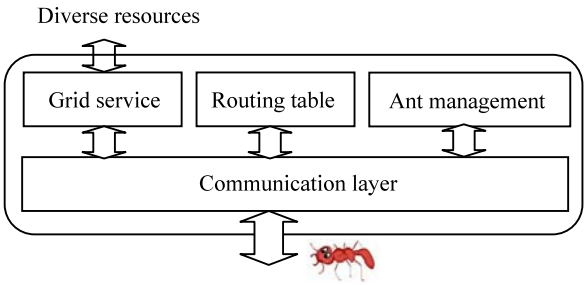


Fig. 3 Architecture of an ant nest

functions and capabilities. Any node can communicate with anyone else by taking advantage of the routing table. All resources have different resource ID which can be used by users to locate the required resources. The resources are randomly distributed across all the nodes. Each resource is allocated to at least one node. Some nodes may have multiple resources. Please note that the resources in our system are represented by resource IDs. In a real Grid system, users locate the Grid services instead of real resources.

Each node in the system is configured as one nest which can issue ants. A nest is composed of a Grid service module, a routing table module, an ant management module, and a communication module which are illustrated in Fig. 3. Service is becoming a basic application pattern of Grid because the service offers a standard means of interoperating between different applications running on a variety of platforms. The distributed and heterogeneous resources should be wrapped into Grid services to provide transparent accesses. Because a key function of a traditional MDS is to monitor the registered Grid services, to minimize the modification of a traditional Grid architecture, the Grid service module in a nest is employed to monitor and manage the registered Grid services which will be located by ants. The routing table module maintains the routing information used by ants to travel in the system. Flash crowd is a common phenomenon on the internet. It means that an unexpected and overloading surge of traffic is incurred by a large number of people who visit a website which catches their attention [2]. Due to the flash crowd phenomenon, the resources in a Grid system can be divided into hot resources and cold resources. We use two fixed Least Recently Used (LRU) lists including a hot list and a recent list to track the most frequently and recently accessed resources (hot resources). When a routing table is updated by an ant, the corresponding routing information (an entry) will be recorded on the recent list. If the entry on the recent list is updated by another ant again in a short period, the entry will be promoted to the hot list. If the promoted entry is already on the hot list, the entry will be moved to the head of the hot list. If the hot list is full, the last entry on the hot list will be degraded to the recent list. If the recent list is full, the last entry on the recent list will be discarded. A dynamic data replication mechanism which automatically places the data replicas where they are needed can tackle the problem of flash crowd [8]. Our simulation only focuses on the resource discovery rather than the performance optimization. The ant management module in a nest is responsible for generating ants in terms of user requests and killing ants when the ants which reach their termination time visit the nest. The communication layer manages the network topology and the movement of ants between

nests. Please note that a P2P network is highly dynamic. The communication layer is able to identify the new neighbors that joined the system and the neighbors that left the system.

An ant mainly consists of two components. The first one is a life cycle management module. The second one is a memory management module. The life cycle management module maintains a TTL value to manage ants. In contrast to the TTL in the IP packages, the TTL in our simulation specifies how many hops an ant can travel before being discarded or returned. When an ant is generated by a nest, the TTL is initialized by a default value. The value is decreased with the increase in number of nests which the ant has visited. If an ant finds the required resources before reaching its TTL, the ant will reset the TTL and travel back to its original nest. Otherwise, the ant will be killed by a nest due to reaching its life termination. The memory management module in the system is used by ants to store the information of the required resources and the nests which they have visited. After finding the required resources, the ants return on the same route and update the routing information of the visited nests with the information they collected on the way. The memory module is designed as queue in terms of LRU policy. The length of the queue is the same as that of TTL.

When a user request looking for resource I is redirected by a Grid portal or other methods to the nearest MDS J, the MDS will issue one ant or several ants to locate the resources in terms of the user's requirement. The initial TTL of the ant is set as the default value. The following major steps can be described by the pseudo code as follows.

```
While (TTL of the ant >0)
{
    If (resource I is registered in MDS J)
    { The ant will travel back to its original nest with the same route in terms of its memory;
      The routing tables of the nests revisited by the ant will be updated; }
    else
    { The ant randomly selects an entry from the routing table of the MDS J; The ant records
      the entry information in its memory; The ant moves to another MDS K in terms of the
      entry; J=K; TTL=TTL-1;
      if (TTL= =0) The ant will be destroyed; }
}
```

Other ants which look for the same resource will take a route in terms of the routing information modified by the previous ants.

4 Experimental evaluation

A real implementation of the comprehensive and complicated system would be very difficult and take a long time. Simulation is a principal approach to evaluate the effectiveness of our proposed method, because it is much easier to change parameters and

configurations compared with a real system implementation. By using a simulator, we can evaluate the method in different environments and compare the method with a variety of other approaches.

4.1 Evaluation environment

In order to validate the proposed method, we developed a simulator from scratch. The simulator can analyze the network topology which has a specific format and design each node in the topology as an ant nest. BRITE [17] is a network topology generator which integrates several network models including a Waxman model [25], a BarabasiAlbert model [4], etc. The Waxman model can generate a random topology to connect the nodes by using the Waxman's probability model [25]. The BarabasiAlbert model employs incremental growth and preferential connectivity to generate a network topology which demonstrates a power law in the frequency of outdegrees (scale-free networks). As discussed in Sect. 2, the scale-free networks' structure and dynamicity are independent of the network size. Due to the dynamicity and scalability of Grid, we adopted the BarabasiAlbert model to generate scale-free network topologies. Figure 1 illustrates an example of the scale-free network consisting of 20 nodes. BRITE can assign bandwidths to links in terms of different possible distributions. In our simulation, we assume that each link has the same bandwidth.

The minimal time unit of the simulator is defined as one simulation cycle. Ten user requests are generated simultaneously every 5 simulation cycles. In the topology generated by BRITE, after each edge of the graph G is initialized with some pheromone, a small number of ants run for a large number of iterations. One iteration is defined as a period in which each ant chooses a path from its start point to its destination by using the transition probability in terms of (1). The network distance in the simulator is measured by hops. One hop is the minimal network distance between two nodes. One thousand resources were generated and distributed across the network for all the performances reported in this paper. The TTL of ants were set as 256.

4.2 Performance evaluation

Choosing appropriate metrics to evaluate system performance is very important. We employ average hop and average hit ratio as two metrics to measure the performance of the ACO based resource discovery. The average hop indicates the network distance which the ants have to travel to find the required resources. The average hit ratio represents the percentage of ants which find their required resources before reaching their TTL. Because ten user requests are generated simultaneously every 5 simulation cycles, we calculate the average hop and the average hit ratio of the ten user requests. Therefore, one iteration in Figs. 4 and 5 is equal to 5 simulation cycles. We performed 4,000 iterations. Because the hash algorithm is very effective in locating resources by one specific name or attribute, the algorithm is developed and integrated in the simulator as a baseline system. We compare the performance of the ACO based resource discovery against the baseline system to evaluate the effectiveness of the proposed method by using the two metrics. In order to explore the impact of the user behaviors on the system performance, we adopt two different mathematical distributions including a discrete uniform distribution and a normal distribution to generate

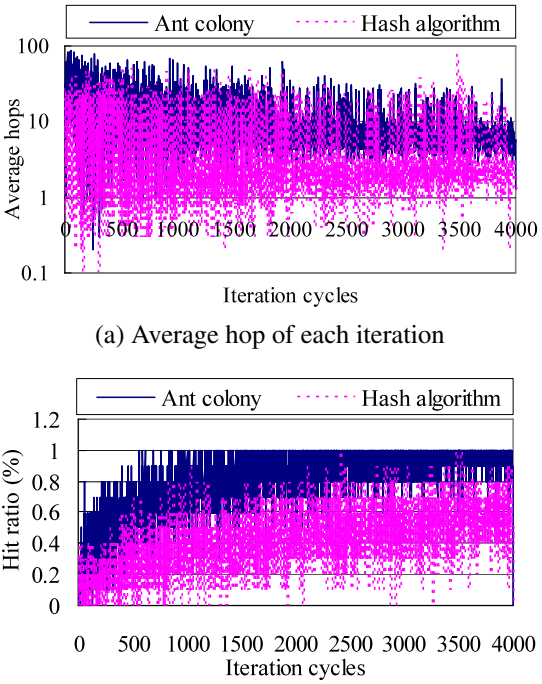


Fig. 4 System performance (the requested resources are generated in terms of a uniform distribution)

the requested resources for users. The discrete uniform distribution indicates that all resources are requested with equal possibility. The normal distribution implies that a portion of resources are accessed with high possibility. We believe that the normal distribution can simulate the flash crowd phenomenon and mimic the behavior of a real system more accurately.

Figures 4 and 5 show the performance of the ACO based resource discovery and the baseline system. Please note that the Y axis of Figs. 4(a) and 5(a) are both in logarithmic scale. The system topology consisted of 1,000 nests. The capacity of routing table of each nest was set as 256 entries. Each user request produced one ant. According to Fig. 4, the average hops of the ACO based system and the baseline system are 11.55 and 5.03, respectively. It indicates that the ACO based system takes longer network distance to find the required resources than the baseline system. However, the hit ratio of the ACO based system (80.1%) is much higher than that of the baseline system (42%). The reason is that the ACO based system has a learning phase. When the learning phase is completed, the system demonstrates very good performance including high hit ratio and low hop. The average hop of the ACO based system in Fig. 4 illustrates a convergence trend with the growth of simulation cycles, which confirms the learning phase. The hit ratio of the ACO based system in Fig. 4 gradually approaches 100% with the increase in number of simulation cycles. On the contrary, the average hop and the average hit ratio of the baseline system in Fig. 4 both demonstrate much stronger oscillating than the ACO based system. Figure 5 depicts a similar performance trend to Fig. 4 such as the learning phase and

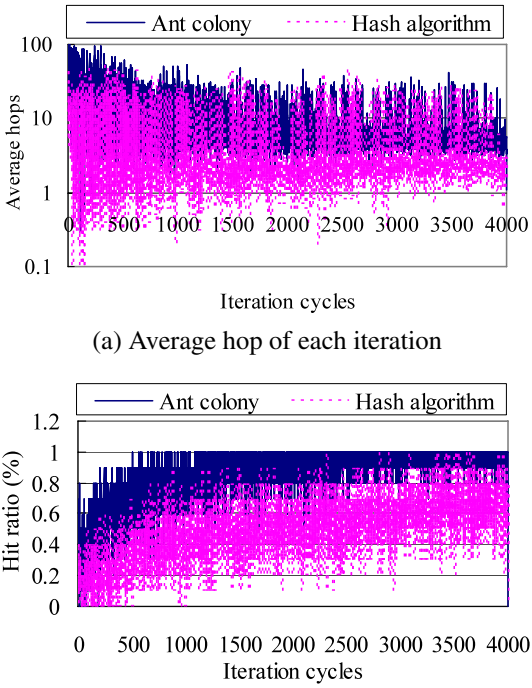
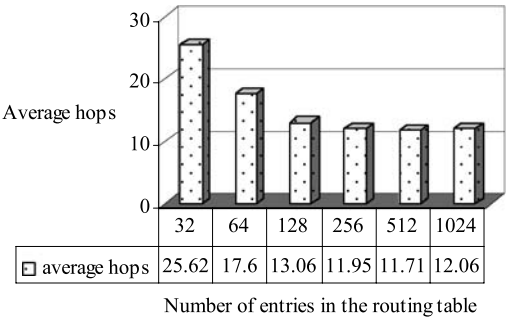


Fig. 5 System performance (the requested resources are generated in terms of a normal distribution)

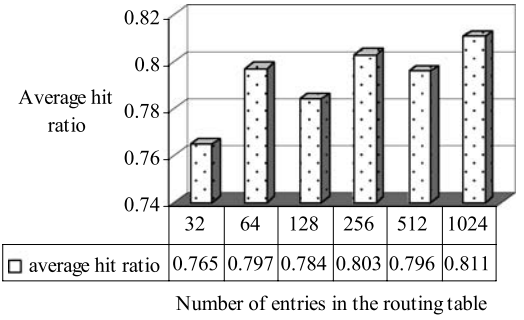
the convergence trend. However, due to the different distributions of the requested resources, the performance in Fig. 5 shows different behavior. The average hops of the ACO based system and the baseline system are 10.47 and 4.96, respectively. The hit ratios of the two systems are 85.2% and 49.2%, respectively. As expected, Fig. 5 illustrates that the average hops of the two systems are decreased, and the hit ratios of the two systems are both improved. The reason is that a portion of the resources are hot resources which are requested by users frequently, which results in less updating of the routing tables in the system. The spikes in the hop curves of the ACO based system denote that the information of the requested resources are not listed on the routing tables and the system has to launch a new learning phase which incurs a long network distance.

The capacity of routing table in each nest could have a significant impact on the system performance, because more entries in the routing table, more useful information can be maintained by a nest. Figure 6 illustrates the system performance as a function of the number of entries in the routing table of each nest. The topology used for test was composed of 6,000 nests. The user requests were produced in terms of a normal distribution. Each user request issued one ant. The simulation period was 4,000 iterations. Figure 6(a) plots the average hops of the all tests with 32, 64, 128, 256, 512, and 1,024 entries allocated for the routing table, respectively. It shows that the average hop gradually decreases when the number of entries is increased from 32 to 256. However, when the number of entries goes beyond 256, the average hop shows a negligible decrease. We even can observe a slight increase when the number

Fig. 6 System performance



(a) Average hop of the overall test



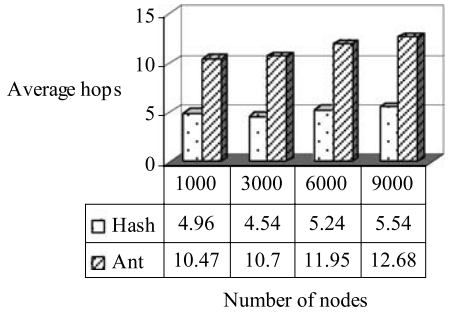
(b) Average hit ratio of the overall test

of entries reaches 1024. The reason is that each nest has limited number of neighbors. When all the neighbors are on the list of its routing table, further increase of the routing table's capacity will not improve the performance. On the contrary, it could produce some performance degradation because large capacity of the routing table incurs search overhead. Figure 6(b) demonstrates that the trend of hit ratio grows with the increase in number of entries in the routing table. There are two bars which seem disharmonic, but the variation is negligible. According to the above measurements, we believe that the optimal number of entries is 256 for the simulated system.

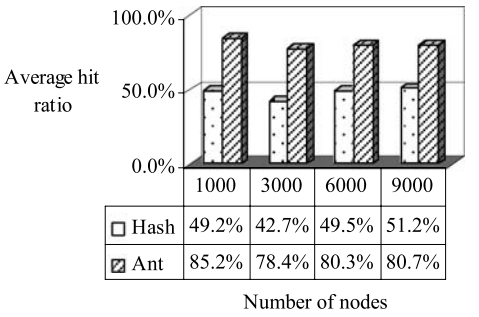
4.3 Scalability evaluation

Figure 7 demonstrates the performance of the system which consists of 1,000, 3,000, 6,000, and 9,000 nests, respectively. The capacity of routing table of each nest was configured as 256 entries. A normal distribution was employed to generate user requests. Each user request issued one ant. The simulation period was 4,000 iterations. The metrics used to measure the performance in Fig. 7 are different with that of Figs. 4 and 5. We plotted the average hop and average hit ratio of the overall test as a function of the number of nests. Figure 7(a) shows that the average hop grows with the increase in the number of nests. When the system scale is expanded from 1,000 nests to 9,000 nests, the average hops of the ACO based system and the baseline system (hash algorithm based) are increased by 21.1% and 11.7%, respectively. The observed phenomenon is normal because it takes more time to locate a resource

Fig. 7 System performance



(a) Average hop of the overall test



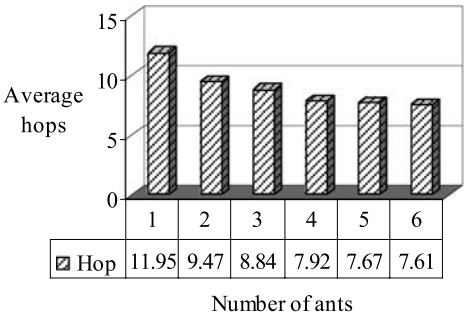
(b) Average hit ratio of the overall test

in a large system than a small system. Figure 7(b) depicts the average hit ratios. We expected to see a significant decrease of hit ratio with the growth of the system scale. However, it is interesting to observe that there is only a slight variation of the hit ratio with the increase in number of nests. One possible reason might be that the structure and dynamicity of a scale-free network are independent of its network size. According to the above measurements, we believe that the ACO based system and the baseline system are both scalable because the systems do not show significant performance degradation when the system scale is increased by a factor of 9.

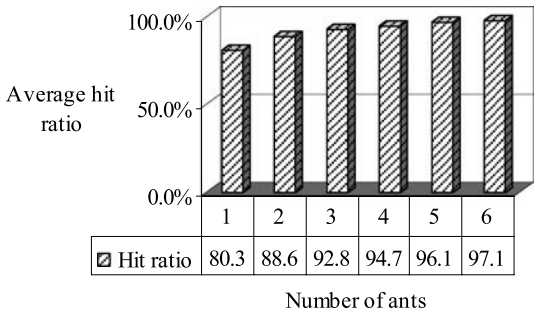
4.4 Parallelism evaluation

A very important feature of the ACO based method is the parallelism. Several ants can be launched by one user request to search the required resources simultaneously. We measured the average hop and the average hit ratio of the overall test as a function of the number of ants issued by each user request. The topology was fixed as 6,000 nests. The capacity of the routing table of each nest was configured as 256. The user requests were generated by using a normal distribution. The simulation period was 4,000 iterations which indicate 40,000 user requests are issued in the system. Figure 8 shows the system performance as a function of different number of ants. The test results are achieved by launching 1, 2, 3, 4, 5, and 6 ants per user request. As expected, the average hop gradually decreases with the increase in number of ants per request, and the average hit ratio grows steadily with the increase in number of ants per request. Figure 8 also indicates that 3 ants per request can achieve the best

Fig. 8 System performance



(a) Average hop of the overall test



(b) Average hit ratio of the overall test

performance because more ants per request can only obtain a negligible performance improvement.

4.5 Network traffic evaluation

Network traffic is a very important metric used to measure the effectiveness of the resource discovery method in a large-scale system. Too much traffic could have significant impact on the overall system performance. For example, the flooding method can quickly saturate the available network resources. For the ACO based approach, each ant in the system represents one message. Figure 9 plots the number of ants in the system with different system configurations. The simulated system consisted of 6,000 nests. Each nest had 256 entries in its routing table. The user requests were generated by using a normal distribution. The measurement lasted for 20,000 simulation cycles. The total number of ants in the system as a function of simulation cycle is illustrated in Fig. 9(a). According to the figure, at the beginning of the simulation, there are a large number of ants in the system. After a certain period (about 1,000 simulation cycles), the number of ants decreases sharply and gradually stabilizes. This phenomenon confirms the learning phase discussed in Sect. 4.2. Different curves in Fig. 9(a) represent the different number of ants launched by one user request. As expected, the total number of ants grows with the increase in number of ants per user request. Figure 9(b) illustrates a comparison of the total number of ants and the number of survived ants in the system. In this measurement, each user request

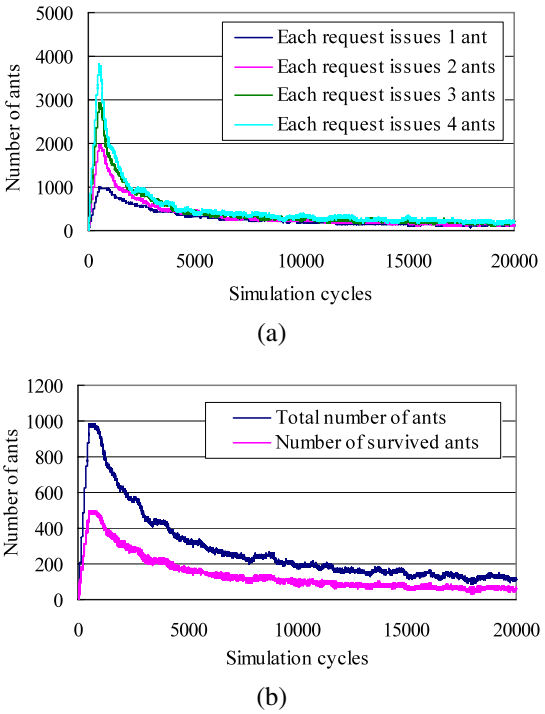


Fig. 9 Number of ants in the system with different system configurations

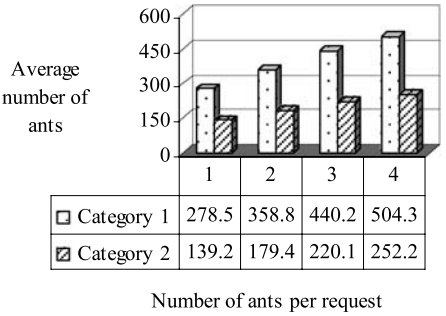


Fig. 10 Average network traffic in the system

issues one ant. The survived ants denote the ants which find the required resources before reaching their TTL. The two curves in Fig. 9(b) show a similar trend with the growth of simulation cycles, and the number of the survived ants in the system is much smaller than that of the total number of ants.

Because the number of ants issued by each user request has a significant impact on the network traffic, we investigated the average number of ants through the overall test as a function of the number of ants per request. Figure 10 shows the network traffic in the system with different system configurations. The measurement lasted for 4,000 iterations. In Fig. 10, Category 1 means the average number of ants which traveled in the system. Category 2 denotes the average number of the survived ants which find their resources before reaching their TTL. Figure 10 implies that the average number

of ants grows with the increase in number of ants per request, and the number of total ants is much higher than the number of the survived ants. The test results are harmonious with the measurements of Figs. 8 and 9.

5 Discussion and conclusion

As discussed in Sect. 1, the structured method is very efficient in locating contents by one specific name or attribute in P2P system. However, the method does not support multi-attribute range queries. It also has difficulty to implement on the current network, especially for the network which has an extremely transient population. For the unstructured method (e.g., Gnutella-style flooding), the corresponding node sends a query message to its all neighbors which in turn forward the message to their own neighbors. If a node possesses the requested resource, it sends a query hit message that will follow the same path back to the requesting node [16]. The method is effective for locating highly replicated items and is resilient to peers joining and leaving the system. However, it is poorly suited for locating rare items, and it is not scalable as the load on each node grows linearly with the total number of queries and the system size [14].

This paper proposes and designs a large-scale P2P Grid system which employs an ACO algorithm to locate the required resources. We evaluated the system with comprehensive simulation. The ACO based resource discovery mechanism is different from the unstructured method because it is based on probability when it chooses a neighbor to send a message. Initially, the ants (messages) walk randomly from nest to nest to locate resources. If the ants find the required resources, they will take the same path to return to their original nest and update the routing information on the path in terms of their memory. The other ants which are looking for the same resources will travel in the system according to the routing information. Therefore, most of the ants are most likely to choose the shortest path to travel in the system. This method avoids a large-scale flat flooding of the unstructured method, thus saving the network resource consumption. On the other hand, the searching efficiency can also be improved by employing multiple ants which can work in parallel. Compared with the structured method, our simulation demonstrated that the ACO based method takes longer network distance than the hash method. However, it has much higher hit ratio (80.1% against 42%). The ants in the ACO method can carry a large amount of information in their memory when it is required. Therefore, multiple user requirements can be stored in the memory, thus supporting multi-attribute range query. This feature is very important for a Grid system, because the Grid users should be able to locate resources with multiple requirements. According to the above analysis, we conclude that the ACO based method is a good choice for the resource discovery in a large-scale Grid system.

Acknowledgements We would like to thank the anonymous reviewers for helping us refine this paper. Their constructive comments and suggestions have significantly improved the presentation of the paper. In addition, we are grateful to Prof. Christophe Cerin, Prof. Jean-Luc Gaudiot, and Prof. Kuan Ching Li for giving us the opportunity to clarify our thoughts.

References

1. Andrzejak A, Graupner S, Kotov V, Trinks H (2004) Algorithms for self-organization and adaptive service placement in dynamic distributed systems. Technical report HPL-2002-259. HP Laboratories Palo Alto
2. Arlitt M, Jin T (1999) Workload characterization of the 1998 world cup web site. Technical report HPL-1999-35 (R.1)
3. Basu S, Banerjee S, Sharma P, Lee S (2005) NodeWiz: peer-to-peer resource discovery for Grids. In: Proc of IEEE int symposium on domain computing and the Grid (CCGrid 2005), vol 1, pp 213–220
4. Barabási A, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
5. Cai M, Frank M, Chen J, Szekely P (2003) MAAN: a multi-attribute addressable network for Grid information services. In: Proc 4th int workshop on Grid computing, pp 184–192
6. Cao J (2004) Self-organizing agents for Grid load balancing. In: Proc of the fifth IEEE/ACM international workshop on Grid computing, pp 388–395
7. Colorni A, Dorigo M, Maniezzo V (1992) Distributed optimization by ant colonies. In: Proc of the first European conference on artificial life, pp 134–142
8. Deng Y, Wang F (2007) Opportunities and challenges of storage Grid enabled by Grid service. ACM SIGOPS Oper Syst Rev 41(4):79–82
9. Deng Y, Wang F (2007) A heterogeneous storage Grid enabled by Grid service. ACM SIGOPS Oper Syst Rev 41(1):7–13. Special issue: File and storage systems
10. Deng Y, Wang F, Helian N, Wu S, Liao C (2008) Dynamic and scalable storage management architecture for Grid oriented storage devices. Parallel Comput 34(1):17–31
11. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the Grid: enabling scalable virtual organizations. Int J High Perform Comput Appl 15(3):200–222
12. Iamnitchi A, Foster I, Nurmi D (2002) A peer-to-peer approach to resource discovery in Grid environments. In: Proc of the 11th symposium on high performance distributed computing, Edinburgh, UK, August 2002, pp 419–434
13. Kubiatowicz J, Bindel D, Chen Y et al (2000) OceanStore: an architecture for global-scale persistent storage. In: Proc of the 9th int conf on architectural support for programming languages and operating systems (ASPLOS 2000), pp 190–201
14. Lua E, Crowcroft J, Pias M, Sharma R, Lim S (2005) A survey and comparison of peer-to-peer overlay network schemes. IEEE Commun Surv Tutor 7(2):72–93
15. Lv Q, Cao P, Cohen E, Li K, Shenker S (2002) Search and replication in unstructured peer-to-peer networks. In: Proc of the 16th ACM int conf on supercomputing (ICS'02), New York, pp 84–95
16. Mastroianni C, Talia D, Verta O (2005) A P2P approach for membership management and resource discovery in Grids. In: Proc of the int conf on information technology (ITCC05), pp 168–174
17. Medina A, Lakhina A, Matta I, Byers J (2001) BRITE: an approach to universal topology generation. In: Proc of the international workshop on modeling, analysis and simulation of computer and telecommunications systems (MASCOTS'01), pp 346–353
18. Monitoring and discovery system. http://www.globus.org/toolkit/mds/
19. Montresor A, Meling H, Montresor A (2002) Messor: load-balancing through a swarm of autonomous agents. Technical report UBLCS-2002-11, Dept of Computer Science, University of Bologna
20. Ritchie G, Levine J (2004) A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. In: Proc of the 23rd workshop of the UK planning and scheduling special interest group
21. Rowstron A, Druschel P (2001) Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In: Proc of the 18th ACM symposium on operating systems principles (SOSP '01), pp 188–201
22. Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 ACM SIGCOMM conference, pp 149–160
23. Sylvia R, Scott S, Ion S (2002) Routing algorithms for DHTs: some open questions. In: Proc of the 1st int workshop on peer-to-peer systems (IPTPS'02). Springer, Heidelberg, pp 45–52
24. Talia D, Trunfio P (2003) Towards a synergy between P2P and Grids. IEEE Internet Comput 7(4): 94–96
25. Waxman B (1998) Routing of multipoint connections. IEEE J Sel Areas Commun 6(9):1617–1622
26. Zhu C, Liu Z, Zhang W, Xiao W, Yang D (2003) Analysis on greedy-search based service location in P2P service Grid. In: Proc of peer-to-peer computing, pp 110–117

Yuhui Deng received his Ph.D. degree in computer architecture from Huazhong University of Science and Technology in 2004. From February 2005 to January 2008, he was a research officer in Cambridge-Cranfield High Performance Computing Facilities, Cranfield University. He joined EMC Research China as a senior research scientist in 2008. He has authored and co-authored one book chapter, more than 10 international journal papers, and several leading conference papers. He is on the editorial board of International Journal of Grid and High Performance Computing and a book titled Encyclopedia of Grid Computing Technologies and Applications. He has served as committee members for several professional conferences in the field. He is also a reviewer of several international journals. His research interests cover data storage, computer architecture, Grid Computing, performance evaluation, etc.

Frank Wang is the director of Centre for Grid Computing, Cambridge-Cranfield High Performance Computing Facility (CCHPCF), Cranfield University. He is Chair in e-Science and Grid Computing. Professor Wang is on the editorial board of IEEE Distributed Systems Online, International Journal of Grid and Utility Computing, International Journal of High Performance Computing and Networking, and International Journal on Multiagent and Grid Systems. He is on the High End Computing Panel for the Science Foundation Ireland (SFI). Prof. Wang is the Chair (UK & Republic of Ireland Chapter) of the IEEE Computer Society.

Adrian Ciura is an M.Sc. student of Centre for Grid Computing, Cranfield University. His research interests mainly focus on grid computing and e-engineering.