Dissemination of Information with Fair Load Distribution in Self-Organizing Grids

Agostino Forestiero, Carlo Mastroianni, and Giandomenico Spezzano

Institute of High Performance Computing and Networking
ICAR-CNR, Rende(CS), Italy
{forestiero,mastroianni,spezzano}@icar.cnr.it

Abstract. This paper presents an ant-inspired algorithm for building a self-organizing information system of a Grid. Ant-inspired mobile agents travel the Grid through P2P (peer-to-peer) interconnections and disseminate "descriptors", i.e., metadata information about available Grid resources. Descriptors are reorganized and spatially sorted over the Grid, thus facilitating resource management and discovery. Moreover, agents distribute descriptors so as to respect the different capabilities of Grid hosts: hosts with higher storage capacity are assigned a larger number of descriptors than low capacity hosts. The effectiveness of the presented algorithm is assessed by event-driven simulation which proves that the simple operations performed by mobile agents successfully achieve both descriptor reorganization and fair load distribution.

1 Introduction

Owing to the inherent scalability and robustness of P2P algorithms, several P2P approaches have recently been proposed for resource organization and discovery in distributed environments and specifically in Grids [1]. The main goal of these approaches is to allow users to locate Grid resources, either hardware or software, which have the required characteristics. This is reduced to the problem of finding resource *descriptors*, which are metadata documents through which it is possible to obtain information and access the resources. Descriptors are usually indexed through bit strings, or *keys* that can have a semantic meaning (for example, each bit may indicate if the resource focuses on a specific topic or provides a specific functionality) or can be obtained through a hash function. In the latter case, the hash function is often "locality preserving" [2], which assures that resources having similar characteristics are associated to similar descriptor keys.

In this paper, we present an approach for the construction of a P2P-based Grid information system, which is inspired by the behavior of some species of ants that cluster items within their environment [3] [4]. The devised algorithm is able to disseminate and reorder descriptors in order to facilitate and speed up discovery operations. Replication and relocation of descriptors are achieved by means of simple *pick* and *drop* operations performed by ant-like mobile agents. These agents probabilistically copy and relocate descriptors with a tendency to remove a descriptor that differs significantly from other descriptors in current

network neighborhood and place it where it is more similar to its neighbors. These operations allow the possible initial equilibrium (if descriptors having different keys are uniformly distributed among hosts) to be broken, and then reinforce the spatial separation and ordering of descriptors.

The ant algorithm is an improved version of the algorithm published in [5]. The enhancement presented here takes into account the storage capacity of hosts, and aims to cope with the problem of fairly distributing descriptors among hosts, which is a critical issue for the efficient operation of P2P systems. To achieve this objective, enhanced pick and drop operations are defined so as to facilitate the pick operations in hosts with low storage capacity and the drop operations in hosts with high storage capacity.

In summary, the presented algorithm concurrently achieves multiple objectives: (i) it *replicates* and *disseminates* descriptors; (ii) it *spatially sorts* them, so that descriptors with similar indexes are placed in neighbor hosts; (iii) it adapts the distribution of descriptors to the storage capabilities of different hosts. Moreover, thanks to the self-organizing nature of the ant-based approach, agent operations spontaneously adapt to the ever changing environment, for example to the joins and departs of Grid hosts and to the changing characteristics of resources. In this paper, we present the results of an event-based simulation analysis, which shows that the algorithm successfully achieves the mentioned objectives.

2 Reorganization and Fair Distribution of Descriptors

As a peer connects to the network, with a given probability it generates a mobile agent that will travel the Grid through P2P interconnections and offer its contribution to the reorganization of descriptors. Whenever an agent arrives at a new host, it operates as follows: (i) if the agent does not carry any descriptor, it evaluates the *pick probability function* for every descriptor stored in this host, so as to decide whether or not to pick this descriptor; (ii) if the agent carries some descriptors, it evaluates the *drop probability function* for each carried descriptor, so as to decide whether or not to drop it in the current host. The *pick* and *drop* operations are driven by the corresponding probability functions that are defined and discussed in the following.

The pick probability function, as well as the drop probability function discussed later, is defined starting from the similarity function $f(\bar{d}, R)$ reported in formula (1). This function measures the average similarity of a given descriptor $\overline{d}$ with all the descriptors d located in the *visibility region* R. The visibility region includes all the hosts that are reachable from the current host with one hop. In formula (1), N_d is the overall number of descriptors maintained in the region R, and $H(d, \bar{d})$ is the Hamming distance between d and $\overline{d}$. B is the number of bits that are contained in descriptor keys. The parameter α defines the similarity scale [6]; here it is set to $B/2$, which is half the value of the maximum Hamming distance between binary vectors having B bits. The value of $f(\bar{d}, R)$ assumes values ranging between -1 and 1, but negative values are truncated to 0.

$$f(\bar{d}, R) = \frac{1}{N_d} \cdot \sum_{d \epsilon R} (1 - \frac{H(d, \bar{d})}{\alpha}) \quad (1)$$

The probability of picking a descriptor $\overline{d}$ from the current host must satisfy two basic requirements:

(i) it must be inversely proportional to the average similarity $f(\bar{d}, R)$, thus obtaining the effect of averting a descriptor from co-located dissimilar descriptors. As soon as the possible initial equilibrium is broken (i.e., descriptors having different keys begin to be accumulated in different Grid regions), a further reorganization of descriptors is increasingly driven, because the probability of picking a dissimilar descriptor increases.

(ii) it must be inversely proportional to the storage capacity of the current host. This assures that in steady conditions high capacity hosts store more descriptors than low capacity hosts, thus respecting the different characteristics of the hosts in a Grid. To cope with this requirement, each host must estimate the average capacity of a Grid host or, more specifically, the average amount of storage space that is offered by a host to store resource descriptors. Estimation is achieved through an exchange of messages with neighbor hosts. Each host assigns a reference value of 1 to the estimated average value and assigns itself a storage index that is proportional to the amount of storage space that this host offers to the network. For example, a host assigns itself a capacity index equal to 5 if it offers an amount of storage space that is 5 times the estimated average storage space offered by a generic host.

The pick probability function P_{pick}, reported in formula (2), is evaluated by an agent for each descriptor $\bar{d}$ stored in the local host, to probabilistically decide whether or not to pick this descriptor. The value of P_{pick} is obtained as the product of two factors. The first factor is inversely proportional to $f(\bar{d}, R)$, the average similarity of the descriptor under consideration with all the other descriptors stored in the visibility region R. The second factor takes into account the capacity of the current peer L_{peer}, and the average capacity of a generic peer $\overline{L}$, which is set to 1 as discussed before. The formula assures that it is more probable to pick a descriptor if it is an outlier in the local region and/or if the local host has less storage capacity then the estimated average.

$$P_{pick} = \left(\frac{k_p}{k_p + f(\bar{d}, R)} \right)^2 \cdot \left(\frac{k_{pl}}{k_{pl} + \frac{L_{peer} - \overline{L}}{\overline{L}}} \right)^2 \quad (2)$$

In the P_{pick} formula, the parameters k_p and k_{pl} can be tuned to modulate the relative impact of the two factors. In particular, the parameter k_p assumes a value between 0 and 1 and can be used to tune the degree of similarity among descriptors. In fact, the first factor of the pick probability function approaches 1 when $f(\bar{d}, R)$ is much lower than k_p (meaning that $\overline{d}$ is extremely dissimilar from the other descriptors) and 0 when $f(\bar{d}, R)$ is much larger than k_p (meaning that $\overline{d}$ is very similar to others descriptors). Here k_p is set to 0.1, as in [3]. Conversely, the value of k_{pl} assumes a value greater than 1, to assure that the denominator

of the second factor is greater than 0. In the case that the value of P_{pick} value exceeds 1, it is truncated to 1, which corresponds to having a 100% probability of picking a very dissimilar descriptor and/or of picking a descriptor from a host with very low capacity.

The pick operation can be performed with two different modes, *copy* and *move*. If the *copy* mode is used the agent, when executing a pick operation, leaves the descriptor on the current host, *generates a replica* of it, and carries the new descriptor until it drops it into another host. Conversely, with the *move* mode, an agent picks the descriptor and *removes* it from the current host. Each agent first operates in the copy mode, than it switches to the move mode, in order to prevent an excessive proliferation of replicas, which would hinder the correct spatial sorting of descriptors. This mechanism is better described in [5].

After picking some descriptors, an agent must decide whether or not to drop them in the hosts through which it passes. For each carried descriptor $\bar{d}$, the agent evaluates the drop probability function P_{drop}, which as opposed to the pick probability, must be: (i) directly proportional to the similarity function $f(\bar{d}, R)$, i.e., to the average similarity of $\bar{d}$ with the descriptors maintained in the visibility region; (ii) directly proportional to the storage capacity of the current host. P_{drop} is defined in formula (3), which satisfies the two mentioned requirements. In (3), the parameter k_d is set to a higher value than k_p, specifically to 0.5, in order to limit the frequency of drop operations. This is useful to let the agents carry descriptors to appropriate Grid regions, without dropping them too early. In the same fashion as k_{pl} in formula (2), k_{dl} must be given a value higher than 1.

$$P_{drop} = \left(\frac{f(\bar{d}, R)}{k_d + f(\bar{d}, R)} \right)^2 \cdot \left(\frac{k_{dl}}{k_{dl} + \frac{\overline{L} - L_{peer}}{L_{peer}}} \right)^2 \tag{3}$$

3 Performance Evaluation

The performance of the ant algorithm was evaluated with an event-based simulator. A Grid network having a number of hosts N_p equal to 2500 is considered. Hosts are linked through P2P interconnections, and each host is connected to 4 peers on average. The topology of the network is built using the scale-free algorithm defined by Albert and Barabasi [7], which incorporates the characteristic of preferential attachment (the more connected a node is, the more likely it is to receive new links) that was proved to exist widely in real networks.

Peers can go down and reconnect. The *average connection time* of a peer is generated according to a Gamma probability function, with an average value set to 100,000 seconds. To maintain a stable number of agents, the lifecycle of agents is correlated to the lifecycle of peers. When joining the Grid, a host generates an agent with a probability P_{gen}, and sets the life-time of this agent to its own average connection time. This setting assures that the overall number of agents is nearly equal to the number of peers times P_{gen}. In our experiments, P_{gen} is set to 0.5, therefore the number of agents is about half the number of peers. Every

time a peer disconnects from the Grid, it discards the descriptors previously deposited by agents, thus contributing to the removal of obsolete descriptors. The average time T_{mov} between two successive agent movements is set to 60 s, whereas the maximum number of P2P hops that are performed in a single agent movement is set to 3. The number of resources published by each host is obtained with a Gamma stochastic function with an average value equal to 15. Resource descriptors are indexed with bit keys having B bits. Descriptor keys are obtained through the application of a locality preserving hash function [2]. This guarantees that similar keys are given to descriptors of similar resources.

The effectiveness of the ant algorithm is evaluated through the spatial *homogeneity function* H. Specifically, for each peer p, the average homogeneity H_p of the descriptors located in the visibility region of p, R_p, is calculated. This is obtained, as shown in formula (4), by averaging the Hamming distance between every couple of descriptors in R_p and then subtracting the obtained value from B, which is the maximum Hamming distance. Thereafter, the value of H_p is averaged over the whole Grid, as formalized in formula (5).

$$H_p = B - AVG_{\{d_1,d_2 \epsilon R_p\}}(H(d_1, d_2)) \tag{4}$$

$$H = \frac{1}{N_p} \cdot \sum_{p \epsilon Grid} H_p \tag{5}$$

The objective is to increase the homogeneity function as much as possible, because it would mean that similar descriptors are actually mapped and aggregated into neighbor hosts, and therefore an effective sorting of descriptors is achieved. In [5], several performance results concerning the basic version of the algorithm are discussed. The present work, however, focuses on the ability of the enhanced version of the algorithm of achieving a satisfactory load distribution among hosts that have different storage capabilities.

In the literature, the storage capacity of hosts is often assumed to be distributed according to some statistical distribution, for example, the Pareto distribution. Here we assume a simpler distribution, which facilitates a more accurate and assessable analysis of our algorithm. Specifically, Grid hosts are divided into two classes: low capacity and high capacity hosts. They can correspond to ordinary personal computers and high capacity servers, respectively. It is assumed that half the load of the system is equally shared among the hosts of each class, and we vary the percentage of hosts that belong to the two classes. The following notation is adopted: the pattern $\{H : L\}$ means that $H\%$ ($L\%$) of the hosts are high (low) capacity ones, and that the load of each host is obtained by sharing half the overall capacity of the system among the hosts of each class. Following this notation, we analyzed the behavior of the algorithm with patterns $\{10 : 90\}$, $\{20 : 80\}$, $\{30 : 70\}$, $\{40 : 60\}$, and $\{50 : 50\}$. Of course, the last case corresponds to a scenario, used for comparison purposes, in which all the hosts have approximately the same capacity.

A set of simulation experiments were performed to assess the distribution of load obtained with our algorithm. In these experiments, the number of bits B in

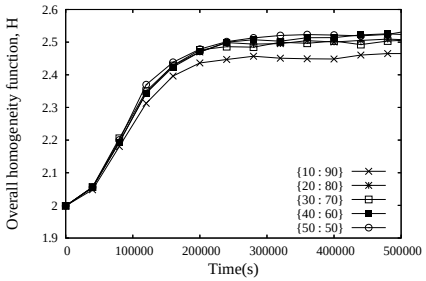

Fig. 1. Overall homogeneity function vs. time, with different patterns of capacity distribution.

resource descriptor indexes is equal to 4. Figure 1 shows that the work of agents makes the value of the spatial homogeneity function H increase from about $B/2$ to much higher values. After a transient phase, the value of H becomes stable: it means that the system reaches an equilibrium state despite the fact that peers go down and reconnect, agents die and others are generated, etcetera. In other words, the algorithm adapts to the varying conditions of the network and is robust with respect to them. Figure 1 also shows that the trend of the homogeneity function is similar for all the tested patterns of capacity distribution. Therefore, the load distribution feature of the algorithm does not affect the accumulation and reorganization of descriptors, which of course is a positive outcome.

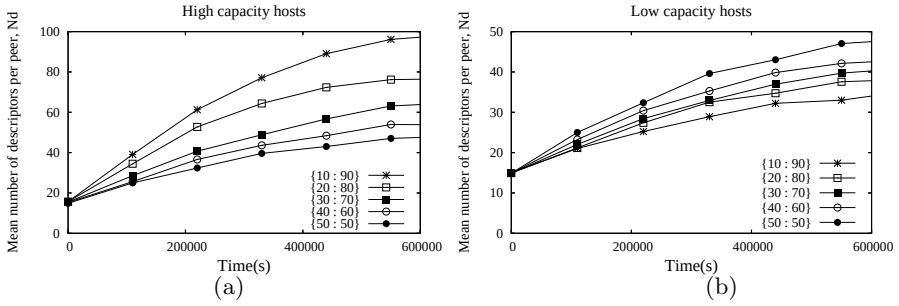

Fig. 2. Average number of descriptors that are stored in high (a) and low (b) capacity hosts, with different patterns of capacity distribution. The factors k_{pl} and k_{dl} of pick and drop probability functions are both set to 3.

The effectiveness of the load distribution approach is confirmed by Figure 2, which shows the average number of descriptors stored in high and low capacity hosts, with different capacity distribution patterns. The factors k_{pl} and k_{dl} of the pick and drop probability functions (see Section 2) are both set to 3. Figure 2(a) shows that, as the percentage of high capacity hosts decreases, and consequently,

the average capacity of such hosts increases (because half the system load is divided among a lower number of hosts), these hosts are actually assigned a larger number of descriptors. For example, the average number of descriptors stored by high capacity hosts, in steady conditions, is about 100 with the pattern $\{10 : 90\}$, whereas it is less than 60 with the pattern $\{40 : 60\}$. The opposite effect is observed for low capacity hosts, as can be observed in Figure 2(b). Note also that the trend corresponding to the pattern $\{50 : 50\}$ is comparable in the two figures, since high and low capacity hosts coincide in this case. Therefore, the objective of assigning more descriptors to hosts that have better storage capabilities, and at the same time of alleviating the load of ordinary hosts, is successfully achieved.

We also calculated the variance and the coefficient of variation CV of the number of descriptors stored by the high and low capacity hosts, to verify how the load is distributed among the hosts of the same class. We found that the the value of CV ranges from about 0.73 to about 0.82 for low capacity hosts and from 0.82 to 0.98 for high capacity hosts. These results reveal that the distribution of load within a class of hosts is not highly affected by the pattern of capacity distribution. Moreover, the value of CV decreases as the number of hosts of the class under consideration increases: therefore, the highest values of CV are obtained with pattern $\{50 : 50\}$ for high capacity hosts and with pattern $\{10 : 90\}$ for low capacity hosts.

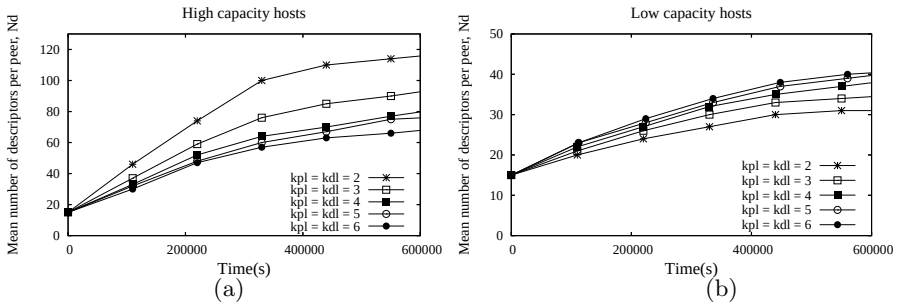

Fig. 3. Average number of descriptors that are stored in high (a) and low (b) capacity hosts, with different values of the factors k_{pl} and k_{dl}. The pattern of capacity distribution is set to $\{10 : 90\}$.

It is also possible to regulate the fraction of load assigned to high and low capacity hosts by tuning the values of the factors k_{pl} and k_{dl}. To analyze this point, a set of experiments were made with the distribution pattern $\{10 : 90\}$ and different values of those factors. Figure 3 shows that the number of descriptors stored in high (low) capacity hosts is inversely (directly) proportional to the value of the factors k_{pl} and k_{dl}. For example, the average number of descriptors stored in high capacity hosts, in steady conditions, is almost 120 if the factors

are set to 2, while it decreases to much lower values for larger values of k_{pl} and k_{dl}. Therefore these factors can be used to balance the load among high and low capacity hosts, according to network and host requirements.

4 Conclusions

In this paper we introduced and evaluated an ant-inspired algorithm for building a P2P information system of a Grid. Grid resources are described by metadata documents, or "descriptors", which are indexed by binary keys. Ant-inspired mobile agents exploit probability functions to replicate descriptors, pick them from some hosts and drop them into other hosts. The objective is to reorganize descriptors and spatially sort them on the network. Moreover, descriptors are distributed by agents respecting the different capabilities of Grid hosts. Simulation analysis confirmed the effectiveness of the algorithm both in the spatially sorting of descriptors and in the achievement of a fair distribution of load among hosts having high and low storage capabilities. Indeed, hosts with higher storage capacity are assigned more descriptors than low capacity hosts. Currently, we are designing a discovery algorithm that exploits the characteristics of the obtained information system. According to this algorithm, query messages may be driven to hosts that have a large number of useful descriptors. Preliminary results are confirming that the performance of discovery operations is indeed improved thanks to the relocation and reorganization of information performed by mobile agents.

References

1. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)
2. Cai, M., Frank, M., Chen, J., Szekely, P.: Maan: A multi-attribute addressable network for grid information services. In: Proc. of GRID '03, 4th International Workshop on Grid Computing, Washington, DC, USA (2003)
3. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems. Oxford University Press, New York, NY, USA (1999)
4. Forestiero, A., Mastroianni, C., Spezzano, G.: So-Grid: A self-organizing grid featuring bio-inspired algorithms. ACM Transactions on Autonomous and Adaptive Systems **3**(2) (2008)
5. Forestiero, A., Mastroianni, C., Spezzano, G.: Antares: an ant-inspired P2P information system for a self-structured grid. In: Proc. of Bionetics '07, 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems, Budapest, Hungary (2007)
6. Lumer, E.D., Faieta, B.: Diversity and adaptation in populations of clustering ants. In: Proc. of SAB94, 3rd International Conference on Simulation of Adaptive Behavior: from animals to animats, Cambridge, MA, USA (1994)
7. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439) (1999)