A Trust-Based Approach for a Competitive Cloud/Grid Computing Scenario

Fabrizio Messina, Giuseppe Pappalardo, Domenico Rosaci, Corrado Santoro, and Giuseppe M.L. Sarné

Abstract. Cloud/Grid systems are composed of nodes that individually manage local resources, and when a client request is submitted to the system, it is necessary to find the most suitable nodes to satisfy that request. In a competitive scenario, each node is in competition with each other to obtain the assignment of available tasks. In such a situation, it is possible that a node, in order to obtain the assignment of a task, can lie when declaring its own capability. Therefore, lying nodes will need to require the collaboration of other nodes to complete the task and consequently the problem arises of finding the most promising collaborators. In such a context, to make effective this selection, each node should have a trust model for accurately choosing its interlocutors. In this paper, a trust-based approach is proposed to make a node capable of finding the most reliable interlocutors. This approach, in order to avoid the exploration of the whole node space, exploits a P2P resource finding approach for clouds/grids, capable of determining the admissible region of nodes to be considered for the search of the interlocutors.

1 Introduction

Nowadays, the offer of on-demand computing resources is rapidly increasing, changing the way in which users deal with special purpose computing requirements. Consequently, approaches such cloud computing [2] and grid computing [1] are becoming the most widespread and leading concepts in the field of networked

Fabrizio Messina · Giuseppe Pappalardo · Corrado Santoro
Dipartimento di Matematica e Informatica, Università di Catania, V.le Andrea Doria 6, 95125 Catania, Italy
e-mail: `{fmessina,pappalardo,santorog}@dmi.unict.it`

Domenico Rosaci · Giuseppe M.L. Sarné
DIMET, University "Mediterranea" of Reggio Calabria, Via Graziella, Feo di Vito 89122 Reggio Calabria (RC), Italy
e-mail: `{domenico.rosaci,sarne}@unirc.it`

distributed systems. These systems are composed of nodes that individually manage local resources, and when a client request is submitted to the system, it is necessary to find the most suitable nodes to satisfy that request. In a collaborative scenario the main problem is that the user request must be fulfilled as soon as possible, determining the nodes with the necessary resources, while in the case of a competitive cloud/grid system another issue must be considered. In a competitive scenario, where clients pay a price to obtain a service, each node is in competition with each other to obtain the assignment of available tasks. In such a situation, it is possible that a node can lie when declaring its own capability. A lying node will need to require the collaboration of other nodes to complete the task, paying a price, and thus the problem arises of finding the most promising collaborators. Since the nodes are in competitions, they might be fraudulent or malicious when collaborating with other nodes. Consequently, each node should have a trust model for accurately choosing its interlocutors. In the context of the e-services, trust is defined as:"the quantified belief by a truster with respect to the competence, honesty, security and dependability of a trustee within a specified context" [4]. When two agents interact with each other, one of them (the truster) assumes the role of a service requester and the other (the trustee) acts as an e-service provider. In this context, while *reliability* is a subjective measure, *reputation* is a measure of the trust that the whole community perceives with respect to a given trustee. Several reliability and reputation models have been proposed in the past for representing both reliability and reputation [4, 9, 10]. In particular, we have proposed the RRAF model [3, 8] to suitably combining these two measures into a unique, global trust measure. In RRAF, each node selects among all the nodes the most promising collaborators based on such a trust measure. However, when the size of the system becomes large, as in the case of many cloud/grid systems, this selection task becomes impracticable.

Given these premises, in this paper we propose to modify the RRAF approach to make it applicable to large cloud/grid systems. Our idea is that of introducing the possibility, for a node that has to choose its interlocutors, of limiting the search space to only those nodes that declare to have the necessary resources for realizing the collaboration. To this aim, we propose to use a technique, called SW-HYGRA (standing for Small World-HYperspace Grid Resource Allocation) [7], which organizes the servers/computing nodes, representing peers, in an overlay network featuring certain characteristics aiming at suitably making the resource finding process effective and efficient. The basic model of SW-HYGRA is to employ an overlay construction algorithm which exploits the resource status similarity, i.e. peers featuring a similar amount of resource availability tend to be interconnected - by means of the links of the overlay - thus forming clusters which, in turn, are connected together by means of few long links. Such an organization resembles the classical model of small-world networks [12]. Some experiments we have realized show that the use of SW-HYGRA in combination with a trust-based selection of the interlocutors introduce in a competitive environment a significant advantage for a node, with respect to other nodes that select their interlocutors based on the sole resource declaration.

The remaining of the paper is organized as follows. In Section 2 we introduce the competitive cloud/grid scenario we deal with. Section 3 describes our reliability and reputation model, while 4 introduces the technique for finding suitable nodes with SW-Hygra. An experimental evaluation of the proposed approach is provided in Section 5. Finally, in Section 6 we draw our conclusions and discuss possible developments of our ongoing research.

2 The Competitive Cloud/Grid Scenario

Our approach deals with a cloud/grid system, composed by nodes that provide a set of services to clients, and that are in competitions to obtain the tasks requested by the clients. In order to model such a competition, we suppose that when a client needs a service, he sends a request to a *Task Allocator* TA, which assigns the request to a node. For sake of simplicity, the assignment of the client's request to a node is performed by TA at pre-determined temporal steps. When a new step begins, TA examines all the service requests submitted by clients, and assigns each request to the node considered the most suitable based on the effectiveness shown by it in the past. When the assignment is done, the client must pay a given *service price sp* to the selected node to obtain the service. During each temporal step, the nodes of the system can interact with each others, in order to exchange information. The interactions among nodes follow this protocol:

- A node a, in order to provide a client with a service requiring a resource amount $\overline{q}$, may decide to search the collaboration of another node b, belonging to the admissible region $S(\overline{q})$ (see Section 4). Before requiring this collaboration, a could ask a third node c for a recommendation about the expertise of b. This recommendation is an evaluation of the *Quality of Service* (QoS) generally associated with the services provided by b. The node c can accept or refuse to give the recommendation. If it accepts, then a must pay a given *reputation price rp* to b. The obtained recommendations can be used by a for updating its internal *reputation model* (see Section 3). In words, a uses the gossips coming from the other nodes in order to understand what is the reputation of b in the community.
- At the end of the step, TA *i)* directly asks a feedback to the client about his evaluation of the quality of each service that a provided him during the step and *ii)* provides this client's feedback to a. The feedback informs a about the quality of the contributions given by the contacted nodes. This way, a can use the feedback to update its internal *trust model*.

3 The Reliability-Reputation Model

This section presents a trust model dealing with the previously introduced scenario. Moreover, we present a methodology for computing the trust measures involved in this scenario: reliability and reputation.

We denote by $\mathscr{A}$ the list containing all the nodes belonging to the cloud/grid system, and by a_i the i-th element of $\mathscr{A}$. A set of five mappings, denoted by SR_i, RR_i, R_i, β_i, and P_i is associated with each node a_i, where each mapping receives a node j as input and yields as output a different trust measure that the node a_i assigns to the node a_j. Each trust measure is represented by a real number belonging to the interval $[0,1]$, where 0 (1) is the minimum (maximum) value of trust. In particular:

$SR_i(j)$ represents the *service reliability* that the node a_i assigns to the services provided by the node a_j. We recall that the reliability represents the subjective measure of the trust that a node has in another node. The value 0 (1) means complete unreliability (reliability).

$RR_i(j)$ represents the *recommendation reliability* that a_i assigns to the *recommendations* provided by a_j. In other words, $RR_i(j)$ is a measure of how much the node a_i considers as reliable the suggestions coming from the nodes j about other nodes, i.e. it represents the reliability of the gossip coming from a_j.

$R_i(j,c)$ represents the *reputation* that the node a_i assigns to the node a_j, based on some recommendations coming from nodes of the community. Although the reputation is not based on a subjective evaluation of the node, it is not an objective measure, since each node a computes the reputation of another node b independently of how the other nodes compute it. Thus, we can say that the value $R_i(j)$ represents how the node a_i perceives the reputation of a_j in the community. The value 0 (1) means minimum (maximum) reputation.

$\beta_i(j,c)$, called *reliability preference*, represents the *preference* that a_i assigns to the usage of the reliability with respect to the reputation in evaluating a_j. In other words, when a_i computes the overall trust score to assign to a_j, it considers both the contributions of service reliability $SR_i(j)$ and reputation $R_i(j)$. The percentage of importance to give to the service reliability is represented by the value $\beta_i(j)$, while the percentage to assign to the reputation is 1-$\beta_i(j)$. In our framework, the mapping β_i is arbitrarily chosen by a_i following its personal strategy.

$P_i(j,c)$ represents the overall preference that ta_i assigns to a_j, based on both the reliability and reputation perceived by a_i.

Besides the five mappings described above, we define a mapping denoted by $RECC_i$, in order to represent the *recommendations* that the node a_i has obtained by the other nodes. Formally, $RECC_i$ is a mapping that receives two nodes a_j and a_k as input, and yields as output a recommendation $RECC_i(j,k)$ representing the recommendation that a_j provided to a_i about a_k (i.e., a real value ranging in [0,1]).

Then, the mappings are updated by a_i at each step, as follows:

- **Phase 1: Reception of the Recommendations.** a_i receives, at the current step, some recommendations by the other nodes, in response to previous recommendation requests. These recommendations are stored in the *RECC* mapping.
- **Phase 2: Computation of SR mapping.** The TA sends to a_i the feedbacks for each service s provided in the past step, where the contributions given by other nodes to a_i are evaluated. These feedbacks are contained in a mapping *FEED*,

where each feedback $FEED(s,j)$ is a real number belonging to $[0,1]$, representing the quality of the collaboration that the node a_j provided to the node a_i concerning the service s. A feedback equal to 0 (1) means minimum (maximum) quality of the service. Basing on these feedbacks, a_i updates its mappings SR and RR. Specifically, we choose to compute the current reliability shown by a_j in its collaboration with a_i by averaging all the feedbacks concerning a_j. Therefore, denoting by $Services(j)$ the set of services provided by a_i with the collaboration of a_j at the previous step, the current service reliability $sr(j)$ shown by a_j is computed as

$$sr(j) = \frac{\sum_{s \in Services(j)} FEED(s,j)}{Services(j)}$$

At each new step, this current reliability is taken into account for updating the element SR_i. We choose to compute this value by averaging the value of SR_i at the previous step and the current reliability computed at the new step. Thus:

$$SR_i(j) = \alpha \cdot SR_i(j) + (1-\alpha) \cdot sr(j)$$

where α is a real value belonging to $[0,1]$ and representing the relevance that a_i gives to the past evaluations of the reliability with respect to the current evaluation. In other words, α measures the importance given to the *memory* with respect to the current time. In an analogous way, the feedbacks are used to update the recommendation reliability RR. The current recommendation reliability of a_j at a given step is computed by averaging all the errors made by a_j in providing a recommendation. In words, if a_j recommended to a_i the node a_k with a recommendation $RECC(j,k)$, and the feedback for a_k concerning a service s is $FEED(s,k)$, the error made by a_j by its recommendation is $|RECC(j,k) - FEED(s,k)|$. By averaging all the errors concerning services of the category c, we obtain an evaluation of the current precision of a_j with respect to the recommendations relating to a_k, that is $(\sum_{s \in Services(k,c)} |RECC(j,k) - FEED(s,k)|)/Services(k)$. Finally, by averaging this precision on the set $Nodes(j)$ of all the nodes a_k evaluated by a_j in the previous step, we obtain the current recommendation reliability $rr(k)$:

$$rr(k) = \frac{1}{|Nodes(j)|} \sum_{k \in Nodes(j)} \frac{\sum_{s \in Services(k)} |RECC(j,k) - FEED(s,k)|}{|Services(k)|}$$

Now, to update the element $RREL_i(j)$, we use a weighted mean between the value of $RREL_i(j)$ at the previous step and the current recommendation reliability:

$$RREL_i(j) = \alpha \cdot RREL_i(j) + (1-\alpha) \cdot rr(k)$$

where α has the same meaning than for the case of the service reliability.

- **Phase 3: Computation of R and β.** The recommendations contained in $RECC_i$ are used by a_i to compute the reputations of the other nodes of the community. In particular, a_i computes the reputation of another node a_j as a weighted mean of all the recommendations received by the other nodes concerning a_j, where the weight of each recommendation value is the recommendation reliability. Thus:

$$R_i(j) = \frac{\sum_{k \in AS, k \neq i} RECC_i(k,j) \cdot RR_i(k,j)}{\sum_{k \in AS, k \neq i} RR_i(k,j)}$$

The β coefficient associated to the agent a_i is recorded in the mapping β_i.

- **Phase 4: Computation of P.** The node a_i finally computes the overall preference measure $P_i(j)$ in the node a_j by considering both the service reliability $SR_i(j)$ and the reputation $R_i(j)$. In particular, the value of the mapping $\beta_i(j)$ is used to weight the importance of the service reliability with respect the reputation:

$$P_i(j) = \beta_i(j) \cdot SR_i(j) + (1 - \beta_i(j)) \cdot R_i(j)$$

At each step, the node a_i exploits the mapping P to select, among the nodes belonging to the admissible region $S(\overline{q})$, the most suitable candidates to require a collaboration.

4 Finding Suitable Nodes with SW-HYGRA

According to the schema reported in the Sections above, a node receiving a job request, which cannot directly fulfill, searches for another node (or a set of nodes) exposing an adequate amount of resources and, by exploiting reputation data, establishes whether it is best suited to perform the job. Since we are considering an environment model composed of a *huge* number of nodes[1], to support such a search process we adopt a *peer-to-peer* approach since it is known to be more efficient and scalable than a centralized solution [5, 11]. The schema adopted in this paper is derived from SW-HYGRA (*Small-World HYperspace-based Grid Resource Allocation*), a P2P resource finding approach for clouds/grids developed and studied by the authors in the recent years [6, 7]. SW-HYGRA is based on organizing the nodes in an *overlay network*, exploiting the links to surf the network, from node to node, until the one able to offer the required resource is found. The key aspect of SW-HYGRA is the algorithm adopted to construct the overlay network, which is also the basis for an efficient resource finding process. SW-HYGRA uses a geometric abstraction by modeling the entire system as a $n-$dimensional space where each *resource type* represents a *coordinate* whose value is the *available quantity* of the considered resource. Each node, on the basis of its resource availability, is represented as a *point* in the hyperspace, therefore its position changes each time a new job is allocated on it, or a running job terminates, freeing the resources no more used. A *metric* is introduced to measure the "distance" between two nodes, i.e. how much two nodes are "far" in terms of resource availability. To this aim, we exploited the Euclidean distance computed using node's coordinates.

[1] In an order which ranges from 10K to 1M.

4.1 Overlay Construction Algorithm

Overlay construction is the key for a fast and effective resource finding. It is performed by means of a decentralized algorithm which runs on each node of the network executing the following steps (say n^* a generic node):

1. node n^* contacts its linked/neighbor nodes in order to obtain, in turn, their linked nodes; this operation allows a node to obtain the set of linked nodes at *2-hops*;
2. the set is ordered by using the Euclidean distance of each node from n^*;
3. on the basis of some threshold parameters k and h, node n^* rearranges its links, interconnecting itself with at most k near nodes and at least h far nodes.

As reported in [6], which details algorithm performances obtained by means of some simulation measurements, the effect of the said steps is to create some *clusters of nodes* featuring a short intra-cluster distance, while keeping *long links* between clusters. Since the Euclidean distance is a measure of resource availability similarity, such clusters are characterized by nodes with a resource status very close to each other. By exploiting short links, a fast navigation inside the cluster is possible to e.g. refine a resource finding process, while, by using long links, it is possible to fast reach the region (i.e. the cluster) in the hyperspace where the nodes offering the requested resources reside. As proved in [7], the overlay network obtained features a structure quite similar to a *small-world* [12]; as it is known, such networks exhibit a high clustering degree and a very low average path length, characteristics which are very important to make resource finding effective. Some steps in the construction of an overlay network by exploiting the proposed algorithm are depicted in Figure 1.

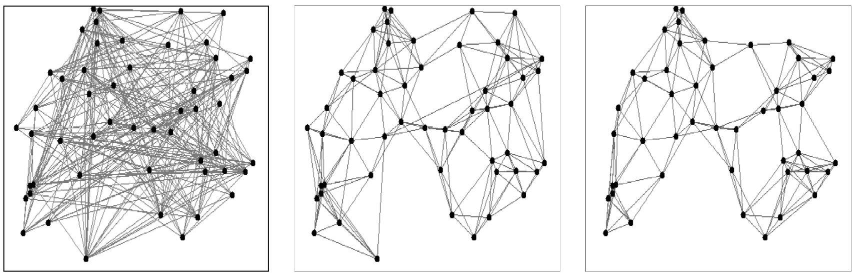

Fig. 1 The overlay network construction

4.2 Resource Finding

According to the said hyperspace abstraction, not only a node can be viewed as a point in the metric space but also a *resource request* can be represented in the same way. Indeed, requesting to execute a job implies to ask the system to allocate a certain amount of provided resources, such as *at least* a certain quantity of RAM, a certain number of CPU or cores, etc. Since such a request in general carries the specific quantities of resources and, since resources are the coordinates of our hyperspace, the request denotes a *point* representing the lower-left corner of a *region* or *semi-space* whose internal nodes are those offering adequate resources for the given request. Such a semi-space is called in SW-HYGRA as the *admissible region* S, for the specific request and its discovery is the aim of the resource finding algorithm detailed below. As usual in P2P systems, also resource finding exploits a decentralized approach which is based on the following *check-and-forward* model:

1. A node receiving the request checks if it is able to fulfill it; if this is the case, the node belongs to the admissible region, so we reached the target and can continue with step 4;
2. if the node does not have sufficient resources, it contacts its neighbors and, on the basis of their resource status, forwards the request to one of them, selected using an appropriate heuristics; such an heuristics is chosen in order to help the request to reach the admissible region as soon as possible;
3. the algorithm keeps track of all the nodes visited (this set is carried together with the request), if all the possible nodes to jump onto are already analyzed, the system does not include a node suitable to host the request, so the algorithm terminates with failure;
4. when we found a node belonging to the admissible region, by suitably navigating through links the algorithm can reach other valid nodes, in order to build the set needed by the reputation schema.

A study on effectiveness, correctness and performances of this resource finding algorithm can be found in [6], as well as the evaluation of some forwarding heuristics. Results proved that, above all in presence of high overall system load (low resource availability), the algorithm described performs very well, ensuring to find the target node in less than 10 steps (on average) with a network size in the order of 10^5 nodes.

5 Experiments

In order to prove the effectiveness of the proposed approach, we performed a set of simulations with the same C-based simulation tool we used for the experimental analysis of the SW-Hygra system [6]. To this aim, we extended the basic test-bed by introducing the reputation model into the SW-HYGRA system, as explained below.

The role of *Task Allocator* has been defined in the following way: once it receives a request, it is forwarded to the most suitable node (see Section2), which in turn will look for a collaboration with another node through the SW-HYGRA algorithm explained in Section 4.2. We modeled the capacity of the single node to provide a certain level of quality of service by the ratio $\frac{q*}{q}$, where q^* is the actual amount of resources offered by the single node, and q is the amount of declared resources. Since in our experimental test-bed the coordinates of the nodes are based on the value of q (see Section 4.1), according to the above considerations, the nodes within the admissible region might not have enough resources to provide the maximum level of service, i.e. to fully satisfy the received request. We categorized the nodes into two subsets: T (with Trust model) includes the nodes which use the reputation model whenever they have to select their collaborator; *WT* (Without Trust model) includes nodes which do not use the reputation model nor relative information.

The simulation has been performed in a test-bed of 10^5 nodes. We studied *i)* the effect of the different values of $\frac{q^*}{q}$ on the average QoS provided to the clients[2], and *ii)* the effect of the different values of the ratio $\frac{|WT|}{N}$, where N is the total number of nodes. The results, reported into Figures 2a and 2b, show that the integration of the competitive approach described into Sections 2 and 3 has makes a difference for the level of QoS provided by the whole system.

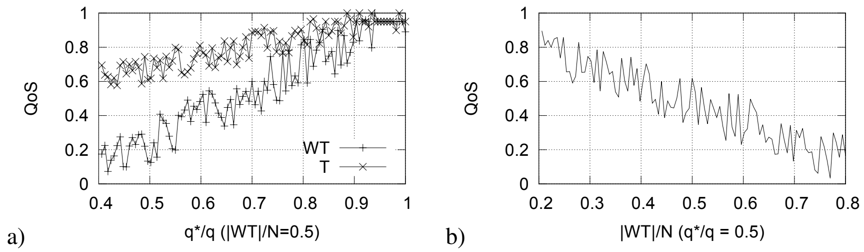

Fig. 2 Results of the: a) Average QoS for the sets WT and T; b) Average QoS vs ratio $\frac{|WT|}{N}$

Indeed, we observed that nodes in T provide, in average, a better level of satisfaction (QoS) than the others in WT (Figure 2a), and whenever the size of set WT grows over the 50% of the total, the QoS of the system drastically decreases (Figure 2b).

6 Conclusions

In this paper, we have presented a trust-based approach to support task allocation in a competitive cloud/grid system. Our approach, similarly to other similar techniques developed in the past for competitive agent systems, allows a node to choose the most promising interlocutors to obtain a collaboration, based on both a direct trust measure (reliability) and a reputation measure derived from the recommendations of the other nodes. However, differently from the aforementioned approaches, that generally explore the whole agent space for selecting the interlocutors, our technique exploits the SW-HYGRA Grid Resource Allocation for organizing the servers/computing nodes in an overlay network featuring certain characteristics. This way, our approach suitably makes the resource finding process efficient in a competitive cloud/grid system, since a node that has to find some interlocutor for a collaboration using its trust model can limit its search to an admissible region previously discovered by SW-HYGRA. The basic model of SW-HYGRA is to employ an overlay construction algorithm which exploits the resource status similarity, i.e. peers featuring a similar amount of resource availability tend to be interconnected by means

[2] It is collected in form of feedback according to Section 2.

of the links of the overlay thus forming clusters which, in turn, are connected together by means of few long links. Some experiments we have realized show that nodes using our trust-based approach perform significantly better than nodes that do not use any trust model to select their interlocutors. As for our ongoing research, we plan to extend our approach in order to consider nodes that dynamically change in time their resource capabilities. In this future scenario, the advantage of using a trust-based model should be even more significant.

References

1. Foster, I., Kesselman, C.: The grid: blueprint for a new computing infrastructure. Morgan Kaufmann (2004)
2. Furht, B., Escalante, A.: Handbook of Cloud Computing, 1st edn. Springer Publishing Company, Incorporated (2010)
3. Garruzzo, S., Rosaci, D.: The Roles of Reliability and Reputation in Competitive Multi Agent Systems. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 326–339. Springer, Heidelberg (2010)
4. Grandison, T., Sloman, M.: Trust Management Tools for Internet Applications. In: Nixon, P., Terzis, S. (eds.) iTrust 2003. LNCS, vol. 2692, pp. 91–107. Springer, Heidelberg (2003)
5. Iamnitchi, A., Foster, I.: A Peer-to-Peer Approach to Resource Location in Grid Environments. In: Grid Resource Management. Kluwer Pub. (2003)
6. Messina, F., Pappalardo, G., Santoro, C.: Decentralised Resource Finding in Cloud/Grid Computing Environments: a Performance Evaluation. In: Proc. of the 21th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, Toulouse, France
7. Messina, F., Pappalardo, G., Santoro, C.: Exploiting the Small-World Effect for Resource Finding in P2P Grids/Clouds
8. Rosaci, D.: Trust measures for competitive agents. Know.-Based Syst. 28, 38–46 (2012)
9. Sabater, J., Sierra, C.: Review on Computational Trust and Reputation Models. Artificial Intell. Review 24(1), 33–60 (2005)
10. Sabater-Mir, J., Paolucci, M.: On Representation and Aggregation of Social Evaluations in Computational Trust and Reputation Models. Int. J. Approx. Reasoning 46(3), 458–483 (2007)
11. Talia, D., Trunfio, P.: Toward a Synergy Between P2P and Grids. IEEE Internet Computing 7(4), 94–96 (2003)
12. Watts, D., Strogatz, S.J.: Collective Dynamics of 'Small-World' Networks. Nature 393(6684), 440–442 (1998)