Reputation-based Cooperation in the Clouds *

Alessandro Celestini[1], Alberto Lluch Lafuente[2], Philip Mayer[3], Stefano Sebastio[2], and Francesco Tiezzi[2]

[1] Istituto per le Applicazioni del Calcolo, IAC-CNR, Rome, Italy
[2] IMT Institute for Advanced Studies, Lucca, Italy
[3] Ludwig-Maximilians-Universität München, Germany

Abstract. The popularity of the cloud computing paradigm is opening new opportunities for collaborative computing. In this paper we tackle a fundamental problem in open-ended cloud-based distributed computing platforms, i.e., the quest for potential collaborators. We assume that cloud participants are willing to share their computational resources for shared distributed computing problems, but they are not willing to disclose the details of their resources. Lacking such information, we advocate to rely on reputation scores obtained by evaluating the interactions among participants. More specifically, we propose a methodology to assess, at design time, the impact of different (reputation-based) collaborator selection strategies on the system performance. The evaluation is performed through statistical analysis on a volunteer cloud simulator.

1 Introduction

Cloud computing has gained huge popularity in recent years. This is mainly due to the progress in virtualization technologies and the transfer of data centers to low-cost locations. This emergent paradigm meets many of today's requirements like the need of elaborating big volumes of data or the necessity of executing applications of which only the front-end is able to run on a mobile device. Next to the presence of traditional cloud computing platforms built running in proprietary data centers, another trend that is gaining popularity is the use of volunteer resources offered by institutions or ordinary people for, e.g., scientific computations. These collaborative environments can effectively be seen as cloud computing platforms where participants take advantage of virtualization techniques to share their computational resources for distributed computing applications, like the execution of tasks. Differently from grid computing, we cannot expect volunteer participants to guarantee a certain level of performance in terms of shared resources or online availability. On the other hand, volunteer clouds offer the unique opportunity of letting participants find their collaborators in the entire volunteer network. The quest for collaborators is one of the key aspects in such platforms.

* Research partially supported by the EU through the FP7-ICT Integrated Project 257414 ASCEns, STReP project 600708 QUANTICOL, and the Italian PRIN 2010LHT4KM CINA.

The contribution of the paper is twofold: (1) a reputation-based approach to the collaborator selection problem, and (2) a methodology to assess, at design time, the impact of the selection strategies on the system performance. We focus on a peer-to-peer cooperative environment on top of which a cloud platform offers a task execution service. The aim of the platform is to maximize the number of successfully executed tasks. We consider a cloud platform with an integrated reputation system, where a reputation score is associated to each node denoting the trustworthiness of the node. Reputation scores are computed on the basis of the rating values released by other nodes. These ratings evaluate the behavior of the node in past interactions. Specifically, we exploit the concept of reputation as an indicator of the likelihood that a node will successfully execute the task, i.e., the higher the reputation the higher the probability that the task will be successfully executed. We assume that tasks have an associated Quality of Service (QoS) requirement given by a deadline, after which the task execution is considered unsatisfactory.

The reputation-based node selection strategies provide *loose coupling* and *self-adaptivity*, since the nodes take their decisions based on the reputation learning mechanism. Overall, the system is able to autonomously adapt the load of nodes during system execution while avoiding to interact with nodes to check their current status. This is in particular useful in platforms that are *dynamic*, where nodes can join and leave the system continuously over time, and *heterogeneous*, since participants with different computational resources are rated with the same mechanism but can customize their strategies according to their needs.

As a reference case study for experimenting with the proposed reputation-based approach, we have used the Science Cloud [10,3]. In particular, we have modeled this cloud platform with DEUS [1] and carried out a number of experiments considering different configuration scenarios. The obtained results show the benefit of the use of reputation-based approaches. Our experimental analysis shows that a probabilistic reputation-based strategy (compared to both reputation based and random approaches) is more robust to the workload variation, offering the best performance at a reasonable communication overhead.

The Science Cloud [10,3] is a volunteer P2P Platform-as-a-Service (PaaS) system developed within the European project ASCENS [2] with the aim of creating a decentralized platform for sharing computational resources in scientific communities. Participants contribute with their desktops, mobile devices, servers, or virtual machines by running platform nodes on them. Nodes may be *heterogeneous*, i.e., they may offer different virtual resources (CPU, disk, memory) and also highly *dynamic*, i.e., they may enter or leave the system at any time, and their load as well as their resources may change. The Science Cloud provides distributed application execution as its main functionality. Applications may range from batch tasks to more sophisticated human-interactive applications. We focus here on task-based scenarios where, for each task, one *initiator* node is chosen as being responsible for processing the task (not necessarily executing the task itself). This node needs to be secure against failures, i.e., if it goes down another node needs to take its place. The initiator may choose to execute

the task itself, but may also choose to delegate to a collaborator node. Whoever finally executes the task is called the *executor*. We assume that a deadline is associated with each task and that a task is successfully executed if the deadline is met. Moreover, each task requires only one executor node. Since the SCIENCE CLOUD is a cooperative environment, we consider scenarios without malicious nodes: nodes accept a task only if they satisfy the resource requirements of the task and if they estimate that they are able to execute it. These estimations assume that the node will remain always connected. However, nodes are not aware of their online/offline times and thus, it may happen that a node accepts a task but, before finishing its execution, goes offline. When a node goes offline it loses all the tasks in its queue regardless of the time in which it will return online. An offline node that returns online will maintain its identifier; this is indispensable to describe the node behavior. Our goal in such scenarios is to maximize the overall number of tasks executed, which is to be achieved by selecting, for each new task, the node most likely to successfully execute the task.

2 Node Selection

We suggest and discuss here some reputation-based strategies based on the Beta reputation system [9] for addressing the node selection problem. Each strategy consists of a *node ranking schema* and a *node selection strategy*. First, we briefly describe (see Fig. 1) the underlying architecture and the protocol that nodes follow to implement the reputation-based strategies: (1) the initiator node sends a request to the reputation system asking for a list of potential executor nodes ordered according to the node selection strategy; (2) the reputation system provides the desired answer to the initiator node; (3) the initiator node starts contacting the potential executor nodes using the obtained list; (4) the contacted nodes send their response to the initiator, either rejecting the request or accepting it (and eventually communicating the completion of the task execution); (5) the initiator node provides feedback to rate its interaction with the contacted executor nodes.

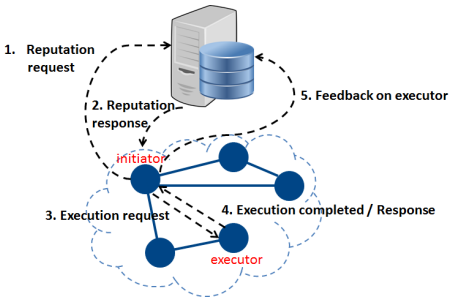

Fig. 1. Finding a collaborator

Reputation Systems. A reputation system associates a reputation score to each node, denoting the trustworthiness of the node, i.e., the higher the reputation, the more trustworthy the node. The reputation score of each node is computed on the basis of the rating values released by other nodes. Such ratings correspond to evaluations of the behavior of the nodes in past interactions, which in our case can have only two possible outcomes: 'satisfactory' (i.e., the task was executed and its QoS was satisfied) or 'unsatisfactory' (the task was not executed or

its QoS was not satisfied). In other words, we consider *binary* ratings. In our approach, the reputation of a node is an indicator of the probability that the node will successfully execute the task. We consider the termination deadline as the QoS parameter and we assume that missing a deadline makes the task completion useless.

In this work, we focus on *probabilistic trust* systems [7,8] which use probability distributions to model the behavior of a node. The goal of such systems is to provide an estimation of the distribution's parameters modeling the node behavior on the basis of past interaction outcomes, i.e., the ratings. This estimation is indeed the reputation score of the node and is used to compute the probability of future interaction outcomes with it.

For the definition of our strategies we exploit the Beta reputation system [9]. The name of this system is due to the use of the Beta distribution to estimate the posterior probabilities of binary events. In the Beta system, the behavior of each node is modeled as a Bernoulli distribution with success probability $\theta \in [0, 1]$. This means that, when interacting with a party whose behavior is (determined by) a given θ, the estimated probability that a next interaction will be satisfactory is θ. The reputation computed by the system is then an estimation $\tilde{\theta}$ of the node's behavior θ. Specifically, to compute the reputation of a given node, the Beta reputation system takes as input the number α of past satisfactory interactions with the node and the number β of past unsatisfactory interactions. The reputation $\tilde{\theta}$ of the node is given by the expected value of a random variable ϑ distributed according to the Beta distribution $Beta(\alpha + 1, \beta + 1)$, with $\alpha \geq 0$, $\beta \geq 0$, that is defined as $\tilde{\theta} = \mathrm{E}[\vartheta] = \frac{\alpha+1}{\alpha+\beta+2}$.

Summing up, in our case the reputation of a node denotes the likelihood that the node, if selected, will not disconnect before completing the task and that it will accept the task because it is not overloaded, i.e., it can meet the deadline. Thus, nodes with high reputation should be able to successfully execute a task with higher probability.

Node Ranking Schema. The interactions we aim at evaluating in our systems are: (i) **accept**, the selected node accepts the task; (ii) **reject**, the selected node rejects the task, since it cannot meet the deadline; (iii) **complete**, the selected node successfully completes the task execution, i.e., it meets the deadline and does not go offline during the execution; (iv) **fail**, the selected node fails in executing the task because it goes offline during the execution.

Notably, we assume that the executor nodes are truthful: they are able to accurately predict the task completion time and accept a new task *iff* they are principally able to execute it within the task deadline. However, nodes do not know their online-offline cycles a priori. It is thus possible that a node misses a task it has accepted by going offline.

Each action can be evaluated by the nodes as satisfactory (+), unsatisfactory (–) or nothing (0), which corresponds to giving a positive, negative or no rating, respectively. The ranking schema is defined by the value assigned to each individual interaction which in our case is **accept** (0), **reject** (–), **complete** (+) and **fail** (–). Notably, the negative rating assigned to the action **fail** is given for

each task whose execution was not successful. Notice also that no rating is given in case of task acceptance.

Node Selection Strategies. Reputation scores are used by the initiator node for the selection of an executor. We consider the following node selection strategies: **Random (`R`)**: a node is chosen randomly using an uniform probability distribution over the node pool (i.e., reputation is not taken into account). **Reputation-based (`RB`)**: the node with the highest reputation score is chosen. If more than one node exists with the same score, the choice is arbitrary (i.e., random). **Probabilistic reputation-based (`PRB`)**: a node is chosen randomly using a probability distribution over the node pool. Such a distribution assigns a probability to each node that is proportional to the reputation score of the node, i.e., the higher the node reputation, the higher the probability the node is selected. The idea is to introduce some randomness to avoid congesting nodes with good reputation, and also some fairness by giving nodes with low scores the chance to achieve a higher ranking (again). The probability that a given node i will be selected among l nodes (the node-i's neighbors) according this strategy is defined as $P(select_i) = \frac{\tilde{\theta_i}}{\sum_{j=1}^{l} \tilde{\theta_j}}$.

3 Validation

We present here an experimental validation of the proposed approach based on simulations and their analysis. Our simulation model is implemented in DEUS [1], an open-source discrete event simulation tool developed in Java. Our statistical analysis has been performed with MultiVeStA [12,15], a distributed statistical analysis tool that can be integrated with any discrete event simulator. MultiVeStA provides a language (MultiQuaTEx) to express the system properties of interest in a compact fashion, and performs independent distributed DEUS simulation runs until these properties are evaluated with the required accuracy.

The simulator implements the basic machinery to suitably model the scenarios under consideration. In the following we discuss some parameters of the configurations of the simulator that can be taken into account to set up the desired volunteer cloud scenarios.

Tasks are generated by initiators according to some parametric process that determines frequency of task generation, their duration (expressed as CPU cycles) and memory occupation. Tasks are defined by their duration and their deadline. If the deadline expires the task execution is considered to be useless. The deadline offset is defined as 20% beyond the ideal task duration. Thus, if a task that requires `t_exec` arrives at time `t_arrival`, the task execution is considered useful if it is completed within time: `t_arrival + t_exec + task_exec*`20%.

When a node accepts a task execution request coming from another node a communication overhead is evaluated to the simple yet realistic network models described by Saino *et al.* [13].

The nodes realize an exclusive task execution environment where the whole Virtual Machine (VM) is assigned to only one task at a time. Its behavior is

modeled by a $M/G/1/+\infty$ queue using the Kendall's notation [4], i.e., Poisson arrival process, general service time distribution with only one VM and infinity queue capacity. A task is accepted by a node only if the node is able to satisfy the requested task deadline, taking into account the tasks already on its queue but without knowing its departure time (i.e., the point in time when it goes offline). Thus, it is possible that a node accepts a task since it is able to satisfy the QoS constraint, but after a while it leaves the network losing the task execution results until that point. In this case the task is lost.

There are executor nodes of two classes: *stable* and *unstable*. Stable nodes are always online. Unstable nodes have two possible states (online and offline) and two transitions (from online to offline and back). Their change of state obeys some parametric, periodic or stochastic model. There are n stable nodes and m unstable nodes created in the initial simulation stage. During the simulation, unstable nodes can leave the network (causing a miss for all the tasks on their execution queue) and reconnect subsequently according to a parameterizable process. When a node comes back online it retains its identifier; in this way the behavior history of unstable nodes is preserved. Nodes are heterogeneous. Disregarding of their class they have computational resources (CPU, RAM) randomly selected in some range (uniformly distributed). The node RAM constitutes a constraint on the task that can be accepted by the node.

In the following we refer to the average results obtained after reaching a 95% confidence interval, with a radius of 0.05, evaluated with the Student's t-test [4]. To evaluate the performance of the proposed strategies we have considered four different measurements to be relevant: the hit rate perceived, the messages spread in the network (total and refused messages), the QoS (Quality of Service) perceived by the task initiators (through the waiting and sojourn times) and the algorithm fairness (considering how well the followed approach is able to equally distribute the task load). For sake of brevity not all the results are reported here. We refer the interested reader to our Technical Report [5].

In our scenarios the arrival processes are Markovian, i.e., the inter-arrival time between two consecutive tasks can be modeled as an exponential random variable with a mean value equal to 750 ms or 1000 ms in the comparing workload. Also, the unstable node departure and reconnection times are modeled with Markovian processes with a mean value equal to 72 seconds. The simulated time is 7 hours, and the temporal analysis considers a granularity of 200 sec. The tasks are described by a deadline of 20% of its duration, a task duration uniformly distributed in a range of [0...24] minutes and a memory requirement in the range of [0...512] MB.

In the low-load situation, analyzing the hit rate (H+R) (Fig. 2) almost all the approaches behave similarly. When the node load increases, it is the `PRB` approach that obtains the best performance, since it is able to spread the load on more nodes in comparison to the `RB` and the `R` approaches. At the same time, the `PRB` approach is able to take into account the information gained on the evaluation of the node's behavior through the reputation scores, and does not stop on a local minimum. From the rate of refused requests (not shown here),

we have observed that both reputation-based approaches (RB and PRB) are able to identify the stable nodes and redirect the load towards them. The strategies that implement a random choice (i.e., R and PRB) are able to spread the load more uniformly among the nodes.

Fig. 2 shows the benefits of using different node selection strategies. It is worth observing that the RB approach is more sensitive to the increase in system load; indeed its performance on the $H+R$ rate decreases quickly. The PRB approach instead is the more stable under the change of workload. The task distribution among nodes shows that the RB approach tends to direct the load to few nodes.

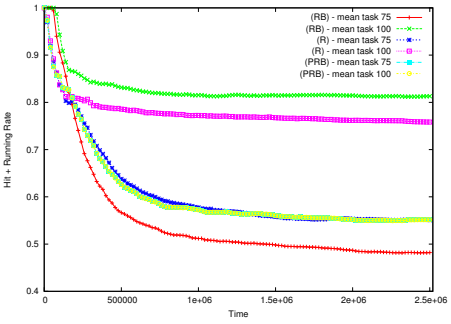

Fig. 2. $H+R$ rate

Concluding, the node selection process done through a reputation-based mechanism can be an effective way to select an executor node. Using only the reputation score, it is possible to observe a degradation on the performance when the task load is high, since the reputation initially leads to a redirection of all tasks to a few nodes that soon get overloaded and consequentially lose their score due to task rejections. The mix of the two approaches, realized in the PRB approach, seems to be the more effective way to use the knowledge acquired with the reputation scores, and at the same time avoids getting stuck in a performance local minimum. This is because the PRB approach allows some degree of exploration of the nodes that do not currently have high scores.

4 Concluding remarks

In this paper, we have investigated the problem of task distribution in voluntary, peer-to-peer cloud computing environments where nodes are willing to share their resources to other nodes. We have proposed a solution based on (1) a reputation-based approach to the collaborator selection problem, and (2) a methodology to assess, at design time, the impact of the selection strategies on the system performance.

Some similar works to ours are the trust management framework proposed by Mishra *et al.* [11] for the sake of trustworthy load balancing in cluster environments, and the reputation-based approach to discovery and selection of reliable resources in P2P Gnutella-like environments, proposed by Damiani *et al.* [6].

We have shown that reputation-based systems can be beneficial in cases where available node resources are unknown, or where nodes deliberately do not want to disclose their status (e.g., current load) or their resources (e.g., CPU, memory). In our experiments, the reputation score calculated through the evaluation of node interactions has been used as the main criteria for selecting

nodes for task execution. Our simulation results shows how the task performance parameters are affected by the use of three different strategies.

Currently, we are calculating reputation scores by considering all aspects of the behavior of a node in a uniform way, i.e., all (satisfactory or unsatisfactory) ratings have the same weight. We plan to extend our analysis with more sophisticated reputation-based approaches, where separate behavioral aspects of a node (e.g., capacity or online/offline period) are rated differently and where further aspects may be taken into account. In this way, we can tune the selection strategies according the specific needs of a given cloud application, which for example may privilege node availability with respect to other features. Furthermore, we are investigating the implementation of reputation-based node selection strategies in the SCIENCE CLOUD platform to validate the simulation results with experiments on a real-world cloud platform. We are also evaluating an ACO-based technique for volunteer clouds [14] and we are investigating how to combine it with the node *pre*-selection strategies proposed in this paper.

References

1. M. Amoretti, M. Picone, F. Zanichelli, and G. Ferrari. Simulating mobile and distributed systems with DEUS and ns-3. In *HPCS*, 2013.
2. European integrated project ASCENS. `http://www.ascens-ist.eu/`.
3. The science cloud platform. `http://svn.pst.ifi.lmu.de/trac/scp/`.
4. G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing Networks and Markov Chains*. Wiley, 2 edition, 2006.
5. A. Celestini, A. Lluch-Lafuente, P. Mayer, S. Sebastio, and F. Tiezzi. Reputation-based cooperation in the clouds. `http://eprints.imtlucca.it/2191/`. Technical report, IMT Lucca, 2014.
6. E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS*, 2002.
7. Z. Despotovic and K.Aberer. A Probabilistic Approach to Predict Peers' Performance in P2P Networks. In *Cooperative Information Agents (CIA)*, 2004.
8. D. Gambetta. *Trust: Making and Breaking Cooperative Relations*, chapter 13: Can We Trust Trust?, pages 213–237. Basil Blackwell, 1988.
9. A. Jøsang and R. Ismail. The beta reputation system. In *the 15th bled electronic commerce conference*, 2002.
10. P. Mayer, A. Klarl, R. Hennicker, M. Puviani, F. Tiezzi, R. Pugliese, J. Keznikl, and T. Bureš. The Autonomic Cloud: A Vision of Voluntary, Peer-2-Peer Cloud Computing. In *SASO AWARENESS Workshop*, pages 1–6, 2013.
11. S. Mishra, D. Kushwaha, and A. Misra. A cooperative trust management framework for load balancing in cluster based distributed systems. In *Recent Trends in Information, Telecommunication and Computing (ITC)*, pages 121–125, 2010.
12. MultiVeStA website. `http://code.google.com/p/multivesta/`.
13. L. Saino, C. Cocora, and G. Pavlou. A toolchain for simplifying network simulation setup. In *SIMUTOOLS*, 2013.
14. S. Sebastio, M. Amoretti, and A. Lluch-Lafuente. A computational field framework for collaborative task execution in volunteer clouds. In *SEAMS*, 2014.
15. S. Sebastio and A. Vandin. MultiVeStA: Statistical model checking for discrete event simulators. In *VALUETOOLS*, 2013.