Future Generation Computer Systems 48 (2015) 28–38

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

AntiCheetah: Trustworthy computing in an outsourced (cheating) environment

Roberto Di Pietro [a], Flavio Lombardi [b], Fabio Martinelli [c], Daniele Sgandurra [d,*]

[a] Cybersecurity Research Department, Bell Labs, Paris, France
[b] Istituto Per le Applicazioni del Calcolo "Mauro Picone" (IAC) – National Research Council of Italy, Rome, Italy
[c] Institute for Informatics and Telematics (IIT), National Research Council of Italy, Pisa, Italy
[d] Department of Computing, Imperial College London, United Kingdom

HIGHLIGHTS

- We discuss an approach to enforce trustworthy computing on cheating environments.
- We provide a model for autonomic, multi-round, distributed cloud computations.
- The approach optimizes cost while detecting cheaters within a confidence threshold.
- We evaluate the approach through extensive simulations.

ARTICLE INFO

Article history:
Received 18 September 2014
Received in revised form
14 January 2015
Accepted 10 February 2015
Available online 18 February 2015

Keywords:
Trustworthy computation in cloud
Cheaters detection
Autonomous cloud computing

ABSTRACT

The increasing need for performing expensive computations has motivated outsourced computing, as in crowdsourced applications leveraging worker cloud nodes. However, these outsourced computing nodes can potentially misbehave or fail. Exploiting the redundancy of nodes can help guaranteeing correctness and availability of results. This entails that reliable distributed computing can be achieved at the expense of convenience.

In this paper, we provide a solution for a generic class of problems that distribute a parallel computation over a set of nodes where trustworthiness of the outsourced computation is important. In particular, we discuss AntiCheetah, an approach modeling the assignment of input elements to cloud nodes as a multi-round system. AntiCheetah is resilient to node cheating, even in scenarios where smart cheaters return the same fake values. To this end, cost-efficient redundancy is used to detect and correct anomalies. Furthermore, we discuss the benefits and pitfalls of the proposed approach over different scenarios, especially with respect to cheaters' behavior. Extensive experimental results are analyzed, showing the effectiveness and viability of our approach.

1. Introduction

In the realm of cloud computing, the term *Computing-as-a-Service* refers a form of totally-outsourced computing offered at different layers (IaaS, PaaS, SaaS) by many alternative cloud providers (Amazon, Microsoft and Google, among the others). Software-as-a-Service (SaaS), in particular, is increasingly widespread thanks to reduced licensing and management costs. In general *High Performance Computing-as-a-Service* [1,2] allows system administrators to avoid the setup and management costs due to the creation and software configuration maintenance of computing nodes. Furthermore, the Cloud offers cheap and scalable pay-as-you-go resources where splitting and offloading computation of parallel algorithms is feasible and convenient. However, many remote computing nodes have historically been proven to misbehave, especially if they are rented with a pay-per-use approach [3]. As an example, remote computing nodes can save their energy and space resources by faking computation (i.e. pretending to compute) and returning erroneous results. One possible example is returning a random result instead of calculating a computationally-intensive function.

* Corresponding author.
E-mail addresses: roberto.di_pietro@alcatel-lucent.com (R. Di Pietro), flavio.lombardi@cnr.it (F. Lombardi), fabio.martinelli@iit.cnr.it (F. Martinelli), d.sgandurra@imperial.ac.uk (D. Sgandurra).

http://dx.doi.org/10.1016/j.future.2015.02.004
0167-739X/

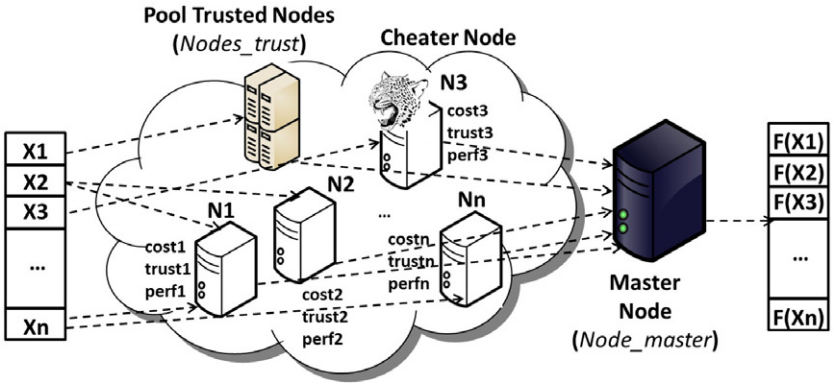

Fig. 1. Use case: Node N_3 is a cheater (the Cheetah).

In all these cases, tenants would like to have some guarantees that service level agreements (SLAs) regarding reliability are met. The problem of secure and reliable outsourced computation is not novel, especially as regards the correctness and the timeliness of returned results. The novelty of the problem, when contextualized in the in cloud scenario, is that it is now economically feasible for tenants to dynamically rent a large number of computing resources (possibly from heterogeneous sources) at the same time. An example is a service like Amazon Mechanical Turk [4], where requester nodes offload computation to crowdsourced workers rented on the Cloud. Hence, many computations could be offloaded to potentially untrusted nodes.

This paper discusses effective approaches enforcing trustworthy computing in a possibly cheating environment. The goal of the proposed approach is to enable efficient reliable computation of parallel functions of a large number m of elements over n nodes [5]. In particular, we discuss AntiCheetah, a novel solution for efficient, reliable, distributed computing [6]. We aim to guarantee that the distributed computation outcome is correct (within a reasonable confidence interval percentage) even when cheating nodes are present. The discussed threat model includes rational adversaries whose objective is to reduce their computing effort. The proposed approach minimizes the cost of selected outsourced resources while keeping the computing time below a given threshold. In addition, AntiCheetah also minimizes management costs, as the system self-configures and adapts to configuration changes.

In detail, AntiCheetah uses a reputation model to dynamically create a new assignment matrix at every protocol iteration (round). This allows the system to be efficient (with respect to computing and communication costs and computing time) and effective against smart cheaters.

Contribution of the paper. The multi-fold contribution of this paper includes:

- discussing and modeling trustworthy distributed computing on a possibly cheating environment;
- detailing AntiCheetah, a multi-round approach for effectively distributing the workload over a large number of computing nodes;
- evaluating AntiCheetah through extensive simulations, showing that it achieves reliable workload distribution while guaranteeing reduced cost and timeliness of results;
- showing how, once the system is fed with its configuration parameters, e.g. the cost and performance of the nodes, it updates and adjusts the nodes' parameters at each iteration, e.g. to take into account their trustworthiness, without the need of any external intervention;
- discussing the convenience of AntiCheetah over different scenarios and conditions.

The paper is organized as follows. Section 2 describes the problem by introducing a use case where the proposed framework can be deployed. It also presents the threat model and a taxonomy of adversaries. Section 3 details the introduced framework, called AntiCheetah. Section 4 discusses the features of the ad-hoc built simulation engine and provides a rich set of results collected over some real use-case examples. In Section 5 we provide some insights on the effectiveness and convenience of the proposed approach. Section 6 surveys relevant related work. Finally, Section 7 draws conclusions and introduces some hints for future extensions.

2. Problem statement

To provide the general context targeted by the proposed framework, we first discuss a simple example scenario and then detail the threat model.

Suppose that an Information Systems Manager (*ISM*) is willing to perform a computationally-intensive task over a data set on a cloud. In this scenario, there might be two main possible goals for the *ISM* [6]:

- cost optimization: the *ISM* has to ensure that the computation ends reliably within a given time frame and she is willing to spend as little as possible.
- time optimization: the *ISM* has a maximum budget and she has to reliably compute the function by minimizing the required time.

In both scenarios, the *ISM* seeks some guarantee that the computation is performed correctly. In the first scenario in particular, the *ISM* would like to have guarantees that the computation is correct (within a certain confidence level) and to minimize the cost involved with renting cloud computing resources by keeping the timeliness of the results before a predefined threshold.

Fig. 1 shows such generic scenario, where the *ISM* needs to compute a function f on a vector X of length m, using a set of nodes chosen from available cloud nodes, some of which may be cheaters [5]. The output is itself a vector of length m where the jth element is $f(j)$. The *ISM* would be interested to know what is the amount of cloud resources required to satisfy the above requirements, i.e. correctness and cost-efficiency with a maximum time ($T_{\max}$) to receive the results. X has to be sent to n cloud nodes (*VMs*), where each node VM_i has an associated unitary cost per operation c_i and the time to perform an unitary operation is t_i.[1] Given that we assume some of the nodes are cheaters, where the percentage of cheaters (*CheaterRate*) is $\frac{k}{n}$, where k is the number

[1] We assume that the application of f has the same cost and complexity for each input.