Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 23 (2007) 904–912
FGCS
FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

Sabotage-tolerance and trust management in desktop grid computing

Patricio Domingues[a], Bruno Sousa[b], Luis Moura Silva[b,*]

[a] *School of Technology and Management, Polytechnic Institute of Leiria, Portugal*
[b] *Dep. Engenharia Informática, University of Coimbra, Polo II, 3030-Coimbra, Portugal*

Received 19 April 2006; received in revised form 30 November 2006; accepted 5 December 2006
Available online 23 January 2007

Abstract

The success of grid computing in open environments like the Internet is highly dependent on the adoption of mechanisms to detect failures and malicious sabotage attempts. It is also required to maintain a trust management system that permits one to distinguish the trustable from the non-trustable participants in a global computation. Without these mechanisms, users with data-critical applications will never rely on desktop grids, and will rather prefer to support higher costs to run their computations in closed and secure computing systems.

This paper discusses the topics of sabotage-tolerance and trust management. After reviewing the state-of-the-art, we present two novel techniques: a mechanism for sabotage detection and a protocol for distributed trust management. The proposed techniques are targeted at the paradigm of volunteer-based computing commonly used on desktop grids.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Desktop grids; Sabotage-tolerance; Trust management; Dependability

1. Introduction

In the past years, several initiatives of desktop grid computing have shown the potential opportunity for exploiting the idle CPU cycles that can be found in millions of Internet computers. Sound examples include SETI@home, Climateprediction.net, Einstein@home, among several others [1]. To support such global computations, there have been some notable advances in desktop grid middleware, with the emergence of open source platforms such as BOINC [2] and XtremWeb [3].

The verification of results is an important issue that needs to be addressed in any volunteer computation. Indeed, hardware and software mishaps as well as malicious volunteers can falsify the outcome of computations, rendering the results useless. Thus, a major concern of middleware tools supporting volunteer computation is to provide results validation and sabotage tolerance mechanisms. Since computations are run in open and non-trustable environments, it is necessary to protect the integrity of data and to validate the computation results. Without a sabotage detection mechanism, a malicious user can potentially undermine a computation that may have been executing for weeks or even months. Therefore, it is no surprise that users with computationally demanding applications do not easily trust open environments, rather preferring to have their applications executed over more controlled clusters which offer some reliability and trustability. This means that sabotage-tolerance is a mandatory issue in desktop grids in order to make them trustable and dependable. In this paper, we discuss the existing contributions and we present initial ideas for a new sabotage-tolerance mechanism targeted at real desktop grid initiatives.

Along with sabotage-tolerance techniques, it is crucial to devise protocols for trust management in desktop grids. For this purpose, low-level techniques are employed to gather valuable information for the creation and maintenance of local reputation lists. On top of that, higher level protocols are needed for globally sharing and maintaining an updated view of the participants' reputation. Some trust management systems have already been proposed in the area of Grid, like the Grid EigenTrust framework [4] and the EigenTrust system for P2P networks [5], among some other proposals [6]. However, these trust management systems do not properly exploit the computational paradigm of volunteer-based computing. In this paper, we propose an invitation-based protocol

* Corresponding author.
E-mail addresses: patricio@estg.ipleiria.pt (P. Domingues), bmsousa@dei.uc.pt (B. Sousa), luis@dei.uc.pt (L. Moura Silva).

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.12.001

for trust management targeted at volunteer desktop grids. The protocol establishes and updates the reputation of the participants according to their relationship in the volunteer-chain, using underlying sabotage-tolerance mechanisms to detect sabotage attempts to undermine the computations, or simply, computation errors due to faulty hardware.

The rest of this paper is organized as follows: Section 2 describes the state-of-the-art for sabotage-tolerance. Section 3 presents a novel mechanism to detect sabotage attempts, based on checkpoint comparison. Section 4 describes the state-of-the-art of trust management in distributed and P2P systems, while Section 5 outlines our protocol for a reputation system in desktop grids and explains some of its novelties. Finally, Section 6 concludes the paper and presents some insights about future work.

2. Sabotage-tolerance techniques

The master–worker model is the common paradigm for computing over desktop grids. Under this model, an application is broken into a large set of individual tasks, with tasks being distributed for computation by the master (also referred to as the *supervisor*) to request workers. After having processed a task, a worker sends the computed results to the supervisor. In an open environment like the Internet, it is necessary to assess the integrity and correctness of the results, since any host can run a worker.

The taxonomy of the sabotage-tolerance techniques can be classified in three distinct groups: (a) replication and voting; (b) sampling; and (c) checkpoint-based techniques. Next, we review each of these groups.

2.1. Replication and voting

The replication technique is also known as double-check [7] or as majority voting [8]. It was first deployed on a wide-scale by the SETI@home project to cope with erroneous results provoked by faulty hardware and malicious users eager to claim credits for work not performed [2]. The technique is based on the replication of individual tasks to different and preferably non-related workers. When completed, the results of the N replicas are compared and a majority voting is applied. The results that do not agree with the majority are marked as erroneous. If no majority can be determined (e.g. all results disagree), results are classified as erroneous and the task needs to be re-executed. N corresponds to the replication factor, and should be at least equal to two. The error rate of the replication method is determined by the replication factor N and by the percentage of erroneous/malicious volunteers. High levels of redundancy augment the resiliency at the cost of higher impact in the overall performance. For instance, the Einstein@home [9] project diminished its replication factor from 3 to 2 when it switched to a more computational demanding stage (S5), an evidence that replication can significantly consume computing resources. The main benefits of the replication approach are its support for generic computation and its simplicity, which eases its implementation — the technique is supported by the main desktop grid middlewares, and employed by all major public computing projects. On the contrary, a major weakness lies in the wasting of resources, since to complete a task, at least N instances need to be effectively computed. Furthermore, in computations that produce results sensible to hardware and software specificities, some further restrictions might be needed to support replication. For instance, some applications are extremely susceptible to floating-point implementations, and the same task run over different machines can yield different numerical results. A viable workaround is *homogeneous redundancy*, upon which replicas of a task are only assigned to homogeneous systems [11]. Regarding sabotage, the replication technique can be bypassed by smart colluding saboteurs as long as they manage to control a majority of replicas of a task. A more subtle limitation of replication-based validation for public computing environments is the potentially long interval that might elapse between the completion of the first result and the existence of enough results for majority voting. This is relevant in credit-based projects, where the effort of volunteers is rewarded through virtual credits. Indeed, credit assignment for a given task is only performed after the result has been validated, that is, after a majority of results matched and a so called *canonical result* exists. This means that the worker of the first result might wait a significant amount of time for receiving its due credits. Although this might be perceived as an irrelevant issue, credits and the associated tops, where users are ranked according to their earned credits, are major motivation factors for volunteers to participate in projects and thus everything related to credits should be treated carefully to avoid disgruntled volunteers [23].

2.2. Sampling techniques

Sampling techniques were developed to overcome the limitations of replication, namely its inefficient usage of resources. Sampling techniques are proposed under four different approaches: (a) naïve; (b) quizzes; (c) spot checks with black lists; and (d) ringers.

(a) *Naïve*. The naïve sample is a simple technique which uses probes to test the trustworthiness of participants [7]. Basically, the supervisor sends some test samples to the participants and then checks the results sent back by the assessed workers. However, the technique can be easily compromised by malicious workers if they are able to distinguish test samples from real application tasks. Indeed, a malicious worker can compute correctly the test samples, only faking application tasks, with its dual behavior possibly going unnoticed. The fact that test samples are computationally less demanding than real tasks makes the identification of test samples relatively easy and thus seriously compromises the usefulness of the technique. Furthermore, if the test samples are sent separately from the batch of real tasks, the detection of samples is even easier and the technique becomes almost useless in a hostile environment, as occurred in early versions of SETI@home [2].

Du et al. [7] extend the naïve sample technique by proposing the *commitment-based sampling* (CBS) approach for strictly