Applied Soft Computing 25 (2014) 253–265

Contents lists available at ScienceDirect

Applied Soft Computing

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

Automatic offloading of mobile applications into the cloud by means of genetic programming

G. Folino*, F.S. Pisani

ICAR-CNR Istituto di Calcolo e Reti ad Alte Prestazioni, Via P. Bucci, 87036 Rende (CS), Italy

ARTICLE INFO

Article history:
Received 12 July 2013
Received in revised form 10 July 2014
Accepted 1 September 2014
Available online 2 October 2014

Keywords:
Mobile computing
Cloud computing
Genetic programming
Data mining

ABSTRACT

The limited battery life of modern mobile devices is one of the key problems limiting their use. Even if the offloading of computation onto cloud computing platforms can considerably extend battery duration, it is really hard not only to evaluate the cases where offloading guarantees real advantages on the basis of the requirements of the application in terms of data transfer, computing power needed, etc., but also to evaluate whether user requirements (i.e. the costs of using the cloud services, a determined QoS required, etc.) are satisfied. To this aim, this paper presents a framework for generating models to make automatic decisions on the offloading of mobile applications using a genetic programming (GP) approach. The GP system is designed using a taxonomy of the properties useful to the offloading process concerning the user, the network, the data and the application. The fitness function adopted permits different weights to be given to the four categories considered during the process of building the model. Experimental results, conducted on datasets representing different categories of mobile applications, permit the analysis of the behavior of our algorithm in different applicative contexts. Finally, a comparison with the state of the art of the classification algorithm establishes the goodness of the approach in modeling the offloading process.

1. Introduction

Modern smartphones boosted their capabilities due to the increasing coverage of mobile broadband networks, to the new high-performance processors, to the large-volume storage and to new different types of sensors. All these capabilities together make it possible for mobile devices to handle much more complex tasks and to execute modern mobile applications; however that consumes a lot more computing and networking resources and therefore demands much more energy, while the battery technology has not developed as fast as mobile computing technology and has not been able to satisfy the increasing energy demand. In addition, in the last few years, the introduction of larger screens further reduces the battery life of the mobile devices.

Therefore, owing to these problems and to the proliferations of mobile devices (i.e. tablets and smartphones), the interest in trying to improve the limited life of their batteries is greatly increased. A possible solution to alleviate this problem is to offload part of the application or the whole computation to remote servers, as explained in [11], where software-based techniques for reducing program power consumption are analyzed, considering both static and dynamic information in order to move the computation to remote servers.

In the last few years, the emergence of cloud computing technology and the consequent wide availability of cloud servers [3], has encouraged research into the use of offloading techniques on cloud computing platforms. A number of papers were published trying to cope with the main issues of the process of offloading, mainly oriented toward a particular problem area, i.e. wifi issues [14], network behavior [18] and network bandwidth [22], the trade-off between privacy and quality [15] and the effect of the context [1].

However, to the best of our knowledge, the works concerning this theme do not consider a model taking into account both the hardware/software issues and the user requirements, neither is there reference to an automatic and adaptive model to take the decision of performing the offloading. Indeed, the offloading technique potentially could improve both performance and energy consumption; however, it is an NP-hard problem to establish whether it is convenient to perform the migration, especially considering all the correlated problems such as network disconnections and variability, privacy and security of the data, variations of load in the server, etc.

* Corresponding author. Tel.: +39 0984831731.
E-mail addresses: folino@icar.cnr.it (G. Folino), fspisani@icar.cnr.it (F.S. Pisani).

http://dx.doi.org/10.1016/j.asoc.2014.09.016
1568-4946/

This paper presents a framework for modeling the automatic offloading of mobile applications using a genetic programming approach, which attempts to address the issues listed above; furthermore, it can be used to simulate different kinds of mobile applications and to generate rules for the offloading process in form of decision trees that can be analyzed by the user. The framework, originally introduced in [6], is made up of two parts: a module that simulates the entire offloading process, and an inference engine that builds an automatic decision model to handle the offloading process. The simulator and the inference engine both apply a taxonomy that defines four main categories concerning the offloading process: user, network, data and application. The simulator evaluates the performance of the offloading process of mobile applications on the basis of user requirements, of the conditions of the network, of the hardware/software features of the mobile device and of the characteristics of the application. The inference engine is used to generate decision tree-based models that take decisions concerning the offloading process on the basis of the parameters contained in the categories defined by the taxonomy. This is based on a GP-based tool that generates the models using the parameters defined by the taxonomy and driven by a function of fitness, giving different weights to the costs, time, energy according to the priorities assigned.

The main contribution of the framework can be summarized as follows:

- The proposal of a taxonomy of the main properties of a mobile application useful for the task of offloading, divided into four categories: user, network, data and application.
- The design of an automatic framework that drives the process of offloading, by building and validating models for that process.
- A GP-based tool for building the decision tree-based model, which will decide whether it is convenient to perform the offloading of a mobile application, is adopted.
- The system permits the analysis of the behavior of our algorithm for different categories of mobile applications, presenting different distributions of the main properties.
- A simulation tool integrating both the cloud and the mobile part.

The rest of the paper is structured as follow: Section 2 surveys existing works; Section 3 describes the software architecture of the system and the different modules composing the framework; Section 4 presents the results of the method using different experimental setups; Section 5, finally, concludes this paper by giving a discussion of the approach and some final considerations.

2. Background and related works

All the issues involved in the offloading decision, such as network disconnections and variability, data privacy and security, variations in load of the server, etc. need to be evaluated carefully and that makes it difficult to design an automatic system for this purpose. In fact, analyzing the works in the literature concerning the offloading of mobile applications, the problem of finding an automatic methodology to perform offloading is not much explored.

A paper introducing general arguments on the process of offloading was written by Kumar and Lee [13]. The authors analyze the main problems derived from offloading mobile applications onto the cloud such as privacy, costs, energy consumption and show which applications can benefit from this approach. They introduce a simple model for deciding whether it is convenient to perform the offloading and they try to apply the technique only to computationally expensive functions while computing other tasks locally.

Other papers are devoted to the utility of performing offloading, basing their considerations on some criteria, i.e. energy consumption, costs, network use, etc. For instance, in [16] the decision to perform the offloading is based on the difference between the energy used when the task was executed locally on the mobile device or remotely on the cloud servers. The power consumption of the local execution is estimated by counting the CPU cycles while, as for the remote execution, it is calculated only considering the network use (data transfer). Our model is more sophisticated, as it considers also the hardware components that are used during computation and the issues concerning the transfer of the data (i.e. CPU, wifi, 3g, display, system, etc.).

In [11] a two-step method is used. First, a database of application power use is built through standard profiling techniques. Then, the authors exploit the property stated in the paper that, for a limited class of applications (i.e. applications in which the cost depends only on the scale of the input), the algorithmic complexity combined with the profiling can be used to predict the energy cost of the execution of the application itself. Unfortunately, real-world applications can be hardly modeled considering only their input.

Many papers are devoted to techniques and strategies to alleviate the process of offloading analyzing the code of the application or optimizing some energy-consuming processes, i.e. the acquisition of the GPS signal. For instance, Saarinen et al. [20] analyze the application source code and identify methods presenting hardware and/or software constraints, which do not permit the offloading. In addition, they also consider traffic patterns and power saving modes. However, the work does not consider network conditions and user requirements. Note that these approaches are orthogonal to our work and can be adopted in order to optimize some phases of the offloading process.

Spectra [5] is a remote execution system that monitors application resource use and the availability of the resources on the device and dynamically chooses how and where to execute application tasks. The framework provides APIs to developers to build application suitable to the defined architecture. X-ray [19] is an automatic system, which profiles an application and decides what offloading computation is useful and when. The X-ray profiling stage observes application and system events (Gui, sensor, GPS, memory, CPU) and identifies "remotable" methods. If a method does not use local resources then it is remotable. Differently from these two systems, our framework is not method-based, but considers the entire application and the decision to perform the offloading is based not only on the application characteristics (size of data, privacy concern) but also on the system status (battery, 3g or wifi connection) and on some constraints requested by the user.

Gu et al. [10] extend an offloading system with an offloading inference engine (OLIE), mainly used to overcome the memory limitations of a mobile device. OLIE solves two problems: first, it decides when to trigger the offloading action; second, it selects a partitioning policy that decides which objects should be offloaded and which pulled back during an offloading action. The decision of performing the offload is based on the available memory and on the network conditions (i.e. bandwidth, delay). To achieve a more powerful triggering system, OLIE uses a fuzzy control-based model with rules specified by the system and by the application developers. For instance, a simple rule for making adaptive offloading triggering decisions can be specified as follows: *"if (AvailMem is low) and (AvailBW is high) then NewMemSize is low"* and the fuzzy system simply uses a linguistic approach to assign a specific value to the linguist value of low (i.e. 500 k). In comparison to simple threshold-based offloading triggering, the fuzzy control model allows OLIE to implement more expressive and configurable triggering conditions. This approach is orthogonal to our and could be used to improve the thresholds decided by our tool, but our decision tree-based model is more expressive and powerful, as it considers also issues related