Computer Communications 81 (2016) 1–11

Contents lists available at ScienceDirect

Computer Communications

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

Reducing your local footprint with anyrun computing

Alan Ferrari*, Silvia Giordano, Daniele Puccinelli

Institute for Information Systems and Networking, University of Applied Sciences of Southern Switzerland (SUPSI), Switzerland

ARTICLE INFO

Article history:
Received 2 July 2015
Revised 14 December 2015
Accepted 22 January 2016
Available online 1 February 2016

Keywords:
Code offloading
Computation offloading
Opportunistic computing
Bayesian network
Software profiling

ABSTRACT

Computational offloading is the standard approach to running computationally intensive tasks on resource-limited smart devices, while reducing the *local footprint*, *i.e.*, the local resource consumption. The natural candidate for computational offloading is the cloud, but recent results point out the hidden costs of cloud reliance in terms of latency and energy. Strategies that rely on local computing power have been proposed that enable fine-grained energy-aware code offloading from a mobile device to a nearby piece of infrastructure. Even state-of-the-art cloud-free solutions are centralized and suffer from a lack of flexibility, because computational offloading is tied to the presence of a specific piece of computing infrastructure. We propose AnyRun Computing (ARC), a system to dynamically select the most adequate piece of local computing infrastructure. With ARC, code can run anywhere and be offloaded not only to nearby dedicated devices, as in existing approaches, but also to peer devices. We present a detailed system description and a thorough evaluation of ARC under a wide variety of conditions. We show that ARC matches the performance of the state-of-the-art solution (MAUI), in reducing the local footprint with stationary network topology conditions and outperforms it by up to one order of magnitude under more realistic topological conditions.

1. Introduction

Though mobile smart devices are becoming increasingly powerful, resource-intensive tasks are still well beyond their reach. Smartphones and tablets are still no match for complex tasks such as pattern matching or face recognition algorithms. Computational offloading is the standard approach to reduce the *local footprint*, *i.e.*, the local resource consumption of smart devices. With computational offloading, a set of programmatic instructions or even an entire program are run onto a remote device. The cloud appears to be the natural place to offload to, as it offers virtually unlimited resources and computing power. Internet connectivity is becoming truly ubiquitous, and at the same time the Internet is becoming increasingly cloud-centric. It is far more cost-effective to outsource resource-intensive tasks to a powerful, dedicated, high-performance computing infrastructure than to run them locally. Nevertheless, in spite of its undeniable benefits, cloud access is still highly inefficient for the offloading device in terms of latency and energy consumption, as shown by recent research that quantifies the local footprint of cloud access [1]. Though large corporations with a stake in cloud computing label it as green, cloud computing is far from green when one accounts for the cost of the information transfer between the cloud and client devices [2]. Cloud computing does save computing energy, but such savings are generally offset by the energy cost of offloading experienced by end devices. This is critical if the end device that interacts with the cloud is a mobile smart device, where battery lifetime and CPU usage are key concern.

In recent years, several approaches to (more or less) cloud-free offloading [1,3,4] have been proposed, and their details are provided in Section 2. Though these cloud-free offloading schemes are extremely valuable, they are not very flexible because they all require dedicated computing resources to run the offloaded code. Because using a remote machine over a WAN results in increased latency and suboptimal energy consumption, as shown in [1], for best results the dedicated computing hardware should be within the same LAN as the devices that need to offload their code. Thus, state-of-the-art cloud-free offloading generally lacks flexibility because dedicated resources need to be known a priori, and the offloading device is bound to share a LAN link with such resources. Using an analogy with the unicast addressing methodology, we refer to this flavor of computational offloading as *unirun computing.*

The goal of any offloading device is to minimize its local footprint, or at least reduce it compared to local execution. For the sake of flexibility, rather than having to offload to a specific high-end resource and being confined to a specific LAN, it would be

* Corresponding author. Tel.: +41 58 666 65 83.
E-mail addresses: alan.ferrari@supsi.ch, alan.ferrari@gmail.com (A. Ferrari), silvia.giordano@supsi.ch (S. Giordano), daniele.puccinelli@supsi.ch (D. Puccinelli).

http://dx.doi.org/10.1016/j.comcom.2016.01.006
0140-3664/© 2016 Elsevier B.V. All rights reserved.

preferable for a smart device to be able to offload to another smart device with enough resources.

To overcome the limitations of state-of-the-art computational offloading strategies and dynamically leverage any suitable device, in this paper we propose (ARC), a system for the reduction of the local footprint of a mobile smart device. The terminology *anyrun computing* is chosen to draw an analogy with anycast addressing.

With ARC, code is offloaded whenever possible not only to dedicated (fixed) devices, but also to peer devices. The ability and flexibility to offload to the best resources available is of paramount importance in a world of heterogeneous devices with varying levels of computing power and resources.

With ARC, any encountered peer can play the role of the offloading device and offloading decisions are made based on Bayesian statistics, whose simplicity makes them particularly suitable to relatively limited resources of mobile smart devices.

In this paper we offer the following research contributions:

- a detailed system description of ARC;
- a thorough evaluation of ARC in a wide variety of conditions on a custom testbed;
- a comprehensive overview of the benefits of anyrun computing compared to unirun computing.

This paradigm drastically differs to standard device-cloud(let) architecture (e.g. the one we found in MAUI [1]) because it leverages any possible devices thus it breaks down the physical link to cloud(let).

As further motivation for ARC, we offer two examples of use cases where the benefits of ARC are made clear.

Home gaming. Though higher-end devices are generally available to smartphone users while at home, smartphones enable gaming experiences that cannot be achieved on desktop machines, laptops, or even tablets thanks to the availability of sensing devices and touch displays. While modern smartphones are equipped with HD screens and GPU to augment the visualization and the 3D rendering quality, their capabilities are much more limited than laptops. As an example, Google's Nexus 5 uses the Adreno 330 graphic board, which is over ten times slower than the Mac Book Pro's HD Graphics 4000 graphics card. The performance gap is an inevitable byproduct of the different form factor and power draw requirements of smartphones. With ARC, GPU computations can be dynamically offloaded to any available higher-end machine; in a home gaming scenario, heavy computational activities (e.g. 3D rendering) can be offloaded to the smartphone's user higher-end devices to drastically improve the gaming experience.

Computation as a service. Just like wireless communication (WiFi) is offered as a service by many businesses (especially franchised chains), communication could also be offered as a service to dynamically augment the computing capabilities of smartphones. Offloading computationally intensive tasks from smartphones to locally available computing infrastructure would contribute to keeping customers on the premises for longer periods of time, contributing to higher sales volumes. With ARC, the offloading would be carried out dynamically to maximize the quality of experience of the user so that desktop- and laptop-grade computation can be accessed on smartphones without the energy and latency penalty of cloud access and with no need for prior knowledge of the high-end computing resources available. Moreover, ARC would also enable the provision of computation as a service by means of the resources of other users, which would be feasible within a specific community of subscribers. Much like users of ridesharing services (such as Uber [5]) can access transportation as a service using resources of other users, users of ARC could achieve something similar for computation as a service. (We view the existence of a specific community of subscribers as a prerequisite for the viability of this model to enable compensation schemes for users lending their own computing resources.)

2. Related work

The ubiquitous and pervasive computing vision is finally becoming a reality as portable devices continue to become more and more widespread and powerful [6,7]. The latest generations of portable smart devices are extremely resource-rich, but they cannot compete with higher-end computing devices when it comes to computationally intensive tasks [8]. Cloud computing is now viewed as a natural solution to overcoming the limitations of mobile devices. Computational offloading (*a.k.a.* code offloading) is a solution to augment the capabilities of mobile systems by migrating computation to more resourceful devices (such as cloud servers) [9]. With the uptake of mobile smart devices, computational offloading is no longer restricted to the cloud, but can also target resource-rich(er) devices.

Given that smart devices generate a huge amount of heterogeneous sensory data that require plenty of processing power, it has been suggested that a mobile phone sensing architecture should rely on the mobile computing cloud [10], so that a smartphone can outsource resource-intensive tasks to a remote high-performance computing system reachable over the Internet. On the one hand, the idea of remote execution [11] and cyber-foraging [12] are first-class citizens in the world of pervasive computing, and the mobile computing cloud appears to be the natural choice [10], because portable smart devices will always be relatively resource-constrained compared to their fixed counterparts. As an example, the CloneCloud system [13] leverages execution migration techniques to clone a smartphone's state to the cloud so that computationally-intensive applications are run on a virtual smartphone clone within the cloud before reintegrating the results from the cloud back into the actual smartphone. On the other hand, the high-performance computing resources that form the computing cloud are typically available at a remote location, and the energy footprint of the data transfer may be significant [14].

To address the inherent resource-poverty of mobile terminals along with the setbacks of relying on distant clouds, the *cloudlet* model [3] has been proposed by Satyanarayanan et al., who empirically show the limitations of WAN-based cloud solutions and propose a novel approach based on accessing high-resource devices located in close proximity. The ThinkAir framework from Kosta et al. [4] proposes a novel computational offloading architecture based on smartphone's virtualization in the cloud. The authors provide method level computational offloading. The offloading strategy is chosen based on a method's energy footprint and device status in terms of resource usage and network connectivity. The authors show that the offloading gain in terms of energy consumption is one order of magnitude greater compared to local execution.

The Mobile Assistance Using Infrastructure (MAUI) system [1] has been recently proposed by Cuervo et al. to enable the fine-grained energy-aware offload of code from a mobile device to a MAUI node, *i.e.*, a nearby piece of infrastructure connected to the mobile device by a high-performance WLAN link. MAUI aims to reduce the energy footprint of mobile devices by delegating code execution to remote devices; it dynamically selects the function to be offloaded depending on the expected transmission costs of the network and provides an easy way for the developers to use the framework in their code. mobile terminals can leverage cloudlets of nearby infrastructure that can be accessed over Wi-Fi. This is certainly a promising strategy, especially given the recent results on the advantages of augmenting 3G with Wi-Fi [15]. It is shown