Simulation Modelling Practice and Theory 50 (2015) 109–129

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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

Discrete-event simulation for efficient and stable resource allocation in collaborative mobile cloudlets

Steven Bohez [a,*], Tim Verbelen [a], Pieter Simoens [a,b], Bart Dhoedt [a]

[a] Department of Information Technology, Internet Based Communication Networks and Services (IBCN), Ghent University – iMinds, Gaston Crommenlaan 8/201, B-9050 Ghent, Belgium
[b] Department of Industrial Technology and Construction, Ghent University College, Valentin Vaerwyckweg 1, B-9000 Ghent, Belgium

ARTICLE INFO

Article history:
Available online 14 June 2014

Keywords:
Mobile cloud computing
Cloudlet
Collaborative applications
Discrete-event simulation

ABSTRACT

The deployment of highly interactive, media-rich applications on mobile devices is hindered by the inherent limitations on compute power, memory and battery capacity of these hand-held platforms. The cloudlet concept, opportunistically offloading computation to nearby devices, has proven to be a viable solution in offering resource-intensive applications on mobile devices. In this paper, we propose to extend the cloudlet concept with collaborative scenarios, in which not only hardware resources for processing are shared between all cloudlet users, but also the data computed.

In a cloudlet, the resource demand should be spread over all available cloudlet nodes. User mobility and fluctuations in wireless bandwidth will cause the optimal resource allocation to vary over time. The cloudlet middleware must continuously balance the performance gain of reallocating components with the operational costs in terms of user experience and management complexity. In this paper, we formulate this optimization problem based on a theoretical cloudlet model capturing the infrastructure, application structure and user behavior.

In order to solve this problem, two heuristic allocation algorithms based on Steepest Descent (SD) and Simulated Annealing (SA) are described. Besides optimality of the found solution, it is also important to limit the number of reallocations at runtime. To evaluate the performance and stability of the algorithms, we propose a discrete-event model for cloudlet simulation. For multiple application scenarios, we observe that SD performs 4 times less reallocations than SA. By introducing hysteresis, the number of reallocations by SA can be nearly halved without any significant degradation of application performance.

1. Introduction

Mobile devices such as smartphones and tablets have become more popular than ever. Gartner [1] has estimated the total number of smartphones exceeding 1.8 billion items sold in 2013. Recently, even smaller mobile devices and wearable computers have become available on the consumer market. Manufacturers such as Samsung [2] and Sony [3] have introduced smart watches acting as easily-accessible dashboards for their smartphones. Meanwhile, Google [4] has introduced a pair of nearly-autonomous smart glasses with video-capture and voice-recognition functionality.

* Corresponding author. Tel.: +32 93314940; fax: +32 93314899.
E-mail addresses: steven.bohez@intec.ugent.be (S. Bohez), tim.verbelen@intec.ugent.be (T. Verbelen), pieter.simoens@intec.ugent.be (P. Simoens), bart.dhoedt@intec.ugent.be (B. Dhoedt).

http://dx.doi.org/10.1016/j.simpat.2014.05.006
1569-190X/

The popularity of these mobile devices has multiple reasons. Not only are they portable and always-connected, but they have access to hundreds of thousands of easy-to-install apps. Due to the high number of sensors (such as cameras and microphones, inertial sensors, location sensors and so on) and high-resolution touchscreens, mobile devices are prime candidates for highly-interactive and media-rich experiences, such as immersive games and Augmented Reality (AR). Even though continuous advances in mobile processors and battery technology have made mobile devices more powerful, they still struggle to execute this kind of applications due to their inherent limitations on resources such as processing power, memory and battery capacity. These limitations are even more pressing on the newest generation of wearable computers.

To offset resource limitations, offloading computation and/or storage to infrastructure in the network has become a necessity [5]. Due to the large and variable network delay, offloading to a distant cloud is however infeasible for applications involving highly responsive user interaction. To offload such applications, resources need to be available closer to the user, e.g. co-located with the wireless access point or at the base station [6]. This is the concept of cyber foraging [7]. As the infrastructure is placed in the vicinity of the mobile users, the network delay is limited to the delay incurred in the single-hop wireless access network.

Cyber foraging can be realized using VM-based cloudlets [8], which are personalized Virtual Machines (VMs) on a nearby trusted server that execute (parts of) the mobile application. The concept of VM-based cloudlets has recently evolved to component-based cloudlets [9,10]. These are systems consisting of a group of computing nodes, both fixed and mobile, that are sharing resources with one another. Mobile applications consist of several loosely-coupled components that can quickly be redeployed at runtime in order to offload parts of the application to other devices. Using smaller software components instead of VMs as a unit of deployment, allows for both faster and more flexible offloading.

Another type of applications that are gaining popularity, are collaborative applications. Also called groupware, these are applications where multiple users work together in a shared context to accomplish a common goal, possibly in an interactive and real-time fashion. Niantic Labs' Ingress [11] is an example of an interactive and collaborative AR game with over half a million active users. These applications face additional challenges when executed in a mobile environment. Not only does the group of users change over time, with new users arriving and others leaving, the wireless connectivity may cause users to become temporarily disconnected.

The resource-sharing concept of cloudlets offers an interesting opportunity for collaborative scenarios: besides sharing computing resources, users may also share data such as processing results or context information. They could even share a software component that holds the same user data to reduce resource consumption. For example, in a location-based collaborative game, users in close proximity of each other could share the same virtual space and interact with the same virtual objects.

To realize interactive collaboration in a way that is transparent to the user and easy to implement by the application developer, we propose to use a middleware platform that offers support for collaboration through state sharing. In [12], we extended the architecture of the component-based cloudlet middleware from [13] with support for collaborative scenarios to create a collaborative cloudlet middleware. The basic mechanisms facilitating collaboration are shared offloading, whereby a component instance is shared between multiple users, and state synchronization, whereby state is shared through the active exchange of state updates. These mechanisms are described in more detail in Section 3 of this paper.

An important aspect of any cloudlet middleware is the autonomous deployment and configuration of all application components currently executed in the cloudlet. The cloudlet middleware should optimize the user experience while coping with the processing and network limitations of the cloudlet. Collaborative scenarios bring additional configuration complexity since some components are used by multiple users. This optimization problem is formally described in Section 4 based on a theoretical model of both the cloudlet infrastructure and the behavior and resource usage of the application components. In Section 5, two heuristic allocation algorithms, based on the search heuristics Simulated Annealing (SA) and Steepest Descent (SD), are discussed that optimize the allocation of components over the different nodes in the cloudlet while taking into account all necessary state sharing.

Evaluating the actual performance of such allocation algorithms is a difficult task. For a suitable allocation, it is not only important that (i) the resource consumption is optimized, (ii) the device and infrastructure constraints are satisfied, but also that (iii) the application deployment remains stable over time, i.e. unnecessary reallocations are avoided. In [12], the SA heuristic is evaluated in the static case by comparing the heuristic outcome for a number of randomly generated problems to the optimal solution obtained by exhaustive search. While this may give an indication of the performance of the applied heuristics, the evaluation in this previous paper did not take into account runtime parameters. In practice, allocation algorithms are used in control loops, where the differences in input parameter values are typically minor between different runs. This is an important consideration when evaluating time-related performance metrics of the algorithm such as stability. Preferably, the full impact of cloudlet system dynamics on the allocation result should be evaluated, such as arriving and departing users, variations in computational and network demand, etc.

Setting up real-life experiments is however costly and time-consuming, especially for cloudlets of realistic proportions. Not only does the infrastructure need to be correctly configured, it is nearly impossible to exactly reproduce experimental conditions with different algorithms or algorithm configurations. This is however required for a fair comparison. Therefore, in Section 6, a discrete-event simulation model is proposed that captures the dynamic behavior of the cloudlet on a method-level granularity. Using simulations, different allocation algorithms can be compared under the same circumstances and a wider range of experimental settings can be explored. In Section 7, different configurations of SA and SD are compared for different scenarios. Finally, Section 8 presents our main conclusions and ideas for future work.