Computer Networks 129 (2017) 193–206

Contents lists available at ScienceDirect

Computer Networks

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

User-aware partitioning algorithm for mobile cloud computing based on maximum graph cuts

Jianwei Niu[a,*], Shihao Wang[a], Wei Niu[a], Mohammed Atiquzzaman[b]

[a] State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[b] School of Computer Science, University of Oklahoma, Norman, OK 73019-6151, USA

ARTICLE INFO

Article history:
Received 21 February 2017
Revised 14 July 2017
Accepted 18 September 2017
Available online 28 September 2017

Keywords:
Mobile cloud computing
Partitioning algorithm
Graph-based model
Maximum graph cuts
User profile

ABSTRACT

Partitioning and offloading of mobile applications have been demonstrated as a promising approach that not only enhances the performance but also extends the battery life of Smart Mobile Devices (SMDs) effectively. Researchers have proposed four factors that affect the partitioning decision: device (hardware), application (software), developer and user. However, most existing research efforts focus on the first three factors and pay little attention to the influence of user preferences on the partitioning decision. Among these factors, user preference usually affects user experience most. Moreover, previous work which took into account the other factors cannot generate adaptive partitioning results. In this work, we propose a user-aware partitioning algorithm to offer a personalized and precise partitioning plan for better user experience. Based on machine learning methods, we first propose a user profile model for characterizing the preferences of different phone users. In addition, a novel cost evaluation model (called CMET model) is proposed to evaluate the comprehensive offloading costs in terms of CPU & memory utilization, time cost and energy consumption. Finally, we propose a Max-Cuts partitioning algorithm based on Branch-and-Bound search to obtain the optimal partitioning plan. Experimental results demonstrate that for different types of phone users, our partitioning algorithm could effectively improve corresponding performances that they concern about, and achieve the most satisfactory user experience compared with state-of-the-art approaches.

1. Introduction

With the increase of computational complexity and data size of mobile applications, the latest generation of Smart Mobile Devices (SMDs) could provide users with higher quality of service. In the meantime, SMDs also expose a series of problem such as poor battery power, limited CPU & memory capacity, and long time delay when processing complicated applications [1].

Mobile Cloud Computing (MCC) is the latest practical computing paradigm that extends utility computing vision of computational clouds to resources constrained smart mobile devices [2]. In recent years, MCC has received the widespread attention because of enabling the development of computational intensive mobile applications by extending them to the powerful and resourceful cloud platform.

Offloading and partitioning are two critical components in the MCC framework. Offloading refers to the operations including transporting several compute-intensive modules of the mobile applications to the cloud platform and accelerating the execution of these modules by utilizing the powerful computing capacity of cloud (Fig. 1). Application partitioning is a technique of splitting up the application into separate components, while preserving the semantics of the original application. Partitioning is responsible for identifying the units (threads, methods, or classes) that can be processed on the cloud and minimizing the total execution cost. Thus, partitioning is a pre-phase of computation offloading in the current framework of MCC and the precision of partitioning result will directly affect the performance of the whole MCC system.

To improve the precision of a partitioning algorithm, we could optimize both the cost evaluation and partitioning models. To the best of our knowledge, most related work focuses on the improvement of partitioning models. For example, the popular models include Weight Graph Model [3–5] and Execution Tree Model [6]. Correspondingly, different algorithms are proposed for these models. Branch-and-Bound Algorithm performs well in dealing with Execution Tree Model [7] and simple Object Relation Graph [8].

* Corresponding author.
E-mail addresses: niujianwei@buaa.edu.cn, niujianwei2008@gmail.com (J. Niu), billywang0129@gmail.com, wangshihao@buaa.edu.cn (S. Wang), niuwei@buaa.edu.cn (W. Niu), atiq@ou.edu (M. Atiquzzaman).

https://doi.org/10.1016/j.comnet.2017.09.011
1389-1286/

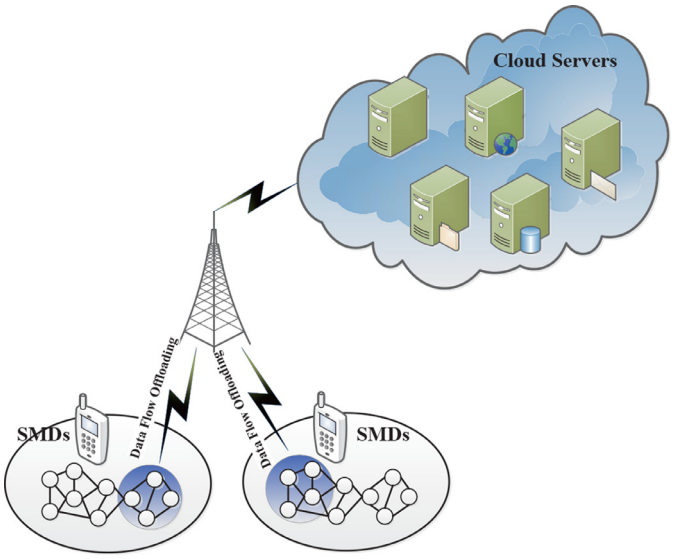

Fig. 1. Application offloading architecture.

The min-cut based algorithms (including the traditional min-cut [9], the max-flow-min-cut [10] and stoer-wagner algorithm [11]) show a better performance in time complexity when handling complicated Weight Garph Models. As for the cost evaluation, most work considers one single objective such as minimizing time or energy consumption [12]. Nevertheless, in practical situations, partitioning decision is affected by many factors (memory & CPU capacity of SMDs and bandwidth latency [13]), and the weights of these factors also depend on user preferences. The *aim* of this work is to develop a partitioning algorithm which is more adaptive and context-aware as compared to previous work; we achieve this by taking into account multiple cost evaluation dimensions as well as considering user preferences.

Generally, researchers divide these factors that influence the partitioning decisions into four categories: (1) Application factors. It mainly contains the type of an application (e.g. multimedia and gaming applications), the input data size, occupied memory space, the quality of service [14]. (2) SMDs factors, which could be classified into the internal and the external. The internal factors includes the current CPU load and the memory capacity; the external factors includes the quality of network communication and the cache capacity of the server in the cloud side. (3) Developers factors. Such factors are usually used in the static partitioning, which relies on the partitioning strategies made by the developers (e.g. programming triggers). (4) User factors. During the practical use of SMDs, the user's behavior patterns and their preferences will affect the choice of the optimal scheme. For example, the users who emphasize the rendering quality of a mobile phone game would most like to sacrifice some power in return for high-quality gaming experience. Others who are bothered by the frequent battery charging may prefer to lowering the application performance to gain more stand-by time. Thus the user preferences serve as a crucial factor when making a precise partitioning plan. However, most of the existing work fails to consider the influence of user preferences on the individual partitioning strategy when they design partitioning models, which will cause the performance degradation.

In this paper, we *propose* a novel partitioning model which takes the user factors into consideration during formulating the partitioning strategy for mobile applications. We also propose a novel multi-parameters cost evaluation model for the calculation of edge weights of our partitioning model. In the weight calculation, both the system and application parameters are imported in order to construct our Max-Cut Graph precisely. Our partitioning model is dynamic and it could generate the real-time optimal partitioning plan according to the context. The following are our main *contributions*:

- We quantify the influence of user preferences on the partition decision-making for the first time. We adopt a machine learning-based method to train a user profile model which could generate the personalized coefficients of different parameters according to the user profile. Thus our model is able to generate different partitioning plans for different users.
- We adopt the Max-Cut Graph to model the mobile application that needs to be partitioned. Since the sum of weights can be regarded as a constant for a single graph, we transfer the application partitioning problem into a maximum graph cuts problem and propose a Max-Cut Partitioning algorithm to solve the segmentation of Max-Cut Graph. The segmentation corresponds to the optimal partitioning plan.
- We propose a cost evaluation model called CMET (CPU-Memory-Energy-Time) to compute the edge weights of the Max-Cut Graph. In CMET model, we conduct a comprehensive analysis of CPU & memory utilization, time cost and energy consumption for each module (or function), which is more comprehensive than previous work.

The rest of this paper is organized into the following sections: Section 2 discusses the related partitioning algorithms, including their cost evaluation models and partition models. The framework of our partitioning algorithm and the training of user profile model is shown in Section 3. Section 4 elaborates our cost evaluation model and Section 5 highlights our partitioning algorithm based on maximum graph cuts. In Section 6, experiments are conducted for the performance comparison between our user-aware partitioning algorithm and other state-of-the-art baselines. The experiments results and a brief error analysis is also discussed in Section 6. Section 7 is the discussion and conclusion.

2. Related work

Generally, partitioning algorithms can be divided into the static methods [15–21] and the dynamic methods [4,11,22–24], which depends on whether the partitioning process happens in the run-time of the application. The static partitioning could be realized by programmer's annotations [20] and function classification [21]. MAUI enables programmers to provide an method-level partitioning of application via annotating the 'local' and 'remote' components of the application. Misco classifies all the functions as 'Map function' and 'Reduce function' separately according to the distributed system rules. However, the weaknesses of static methods are overt. First, it is inflexible and usually misses the optimal partitioning plan. Second, the necessity of annotating the methods causes heavy workload when the partitioning model becomes complex with more module nodes.

Most contemporary offloading processing frameworks use dynamic partitioning in which additional computing resources are occurred in run-time application profiling and partitioning [25]. For example, when the application executes, varieties of resource profiles begin to work for monitoring and recording different real-time system parameters. There are usually three types of profile. Hardware profile monitors the consumed energy of CPU, screen and signal interfaces. Software profile records thread CPU time, number of method calls, thread memory allocation and etc. Network profile collects the data of Round Trip Time (RTT), network bandwidth and other parameters of 3G and WiFi interfaces. In [5], the profile also records the past invocation data like former execution time and energy consumed in order to help future decision. For most dynamic methods, they will abstract their