SCIENCE CHINA
Information Sciences

• LETTER •

August 2017, Vol. 60 089301:1–089301:3
doi: 10.1007/s11432-016-0009-6

Multisite computation offloading in dynamic mobile cloud environments

Xiaomin JIN[1,2*], Yuanan LIU[1,2], Wenhao FAN[1,2], Fan WU[1,2] & Bihua TANG[1,2]

[1]*School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing* 100876*, China;*
[2]*Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing* 100876*, China*

Received November 23, 2016; accepted December 12, 2016; published online February 8, 2017

Citation Jin X M, Liu Y A, Fan W H, et al. Multisite computation offloading in dynamic mobile cloud environments. Sci China Inf Sci, 2017, 60(8): 089301, doi: 10.1007/s11432-016-0009-6

Dear editor,

With the development of wireless communication and computer technology, using mobile devices (MDs) has become more and more popular. However, due to a series of constraints such as heat dissipation, volume and battery capacity, MDs take much time and energy to run some applications and even cannot run heavy applications. As a result, these constraints hinder the further development of MDs. To solve the problem that MDs' performance is restricted by their own limited resources, mobile cloud computing (MCC) has been proposed to augment MDs' capabilities [1]. Compared with the client-server architecture which always executes components in the server, MCC needs to make offloading decisions that determine whether components should be offloaded or not according to the optimization objective [2].

How to make offloading decisions can be described as an application partitioning problem which is NP-complete. Compared with the traditional cloud computing, MCC has its own features [3] such as mobility, diversity of network conditions and limited channel bandwidth. These features bring changes during applications' life cycle and turn the application partitioning of MCC into the runtime application repartitioning.

In this letter, we focus on multisite computation offloading in dynamic mobile cloud environments with the consideration of environmental changes during applications' life cycle and relationships among components of an application. Multisite computation offloading can migrate components to multiple servers and it has less consumption brought by data transmission because consumption on wired networks among servers is quite small. Decision making of multisite computation offloading is described as the $(k+1)$-way application partitioning where k represents the number of servers and it has a much larger solution space than single site computation offloading, which increases algorithms' running time. In runtime application repartitioning, the key point is to propose an algorithm which can make a good tradeoff between running time and accuracy. To address this problem, we propose a runtime application repartitioning algorithm that makes a good balance between running time and accuracy with the help of historical offloading strategies based on the memory-based immigrants [4] adaptive genetic algorithm. Moreover, we explore the using of concurrent multipath transfer (CMT) (e.g., SCTP [5]) which uses multiple physical wireless interfaces to transfer data in MCC with our algorithm to combat the challenges that wireless links have limited bandwidth and low robustness.

* Corresponding author (email: jxm@bupt.edu.cn)

The authors declare that they have no conflict of interest.

Runtime application repartitioning algorithm. An application (see Appendix A for detail) can be represented by a graph $G = (V, E)$ whose vertexes represent the application's components and edges represent interactive relationships among components. Offloading strategy is represented by $X = \{x_v | 0 \leqslant x_v \leqslant k, v \in V\}$. $x_v = 0$ means that component c_v is executed in the MD and $x_v = i$ $(1 \leqslant i \leqslant k)$ means that component c_v is executed in cloud server cs_i. We define the objective function which represents the weighted total cost as

$$F(G,X) = \sum_{v\in V}\left(w_e\frac{E_v^{x_v}}{E_l(G)} + w_t\frac{T_v^{x_v}}{T_l(G)}\right) + \sum_{u\in V}\sum_{v\in V}\left(w_e\frac{E_{u,v}^{x_u,x_v}}{E_l(G)} + w_t\frac{T_{u,v}^{x_u,x_v}}{T_l(G)}\right). \quad (1)$$

w_e is the weight of energy cost and w_t is the weight of time cost. $E_l(G)$ and $T_l(G)$ respectively represent the energy and time cost of application G when it is executed locally. $E_v^{x_v}$ and $T_v^{x_v}$ respectively represent the energy and time cost when component c_v is executed according to x_v. $E_{u,v}^{x_u,x_v}$ and $T_{u,v}^{x_u,x_v}$ respectively represent the energy and time cost brought by data transmission when component c_u is executed according to x_u and component c_v is executed according to x_v.

The pseudocode of our algorithm is shown in Appendix B. Core parts of our algorithm are illustrated as follows.

(1) Encoding and fitness function. In our algorithm, a chromosome is represented by the offloading strategy of offloadable components and its genes are encoded as integers between 0 and k. The fitness function is expressed as

$$f_h = \frac{1}{F_h^1 + F_h^2 + F(G_h, X_h)}. \quad (2)$$

F_h^1 represents the cost brought by components that have been executed at time h. F_h^2 represents the cost brought by the component that is being executed. F_h^2 is a changing value because the environment may change during its execution. $F(G_h, X_h)$ is the cost of components which are not executed at time h. Appendix C illustrates an example which is helpful to understand them.

(2) Genetic operations. Crossover and mutation probabilities are key parameters which affect the performance of genetic algorithms seriously. However, in the standard genetic algorithm, crossover and mutation probabilities are constants so that it has shortcomings of slow convergence rate and easy to fall into local optimal solutions. To avoid bad solutions are stored in the memory, we apply adaptive probabilities of crossover and mutation [6] in our algorithm.

(3) Memory. Historical offloading strategies stored in the memory (Appendix D helps to understand it) are used as the base to create immigrants to replace worst chromosomes in the current population. The memory contains $|V|$ sets of data which consists of three parts: component index, maximum and minimum parameters (environmental parameters and F_h^1), m elements. An element is made up by six parts which are component index, uplink and downlink throughput between MD and cloud servers, cost of executed components, chromosome and fitness. Similarity of parameters determines whether a new element should be added into the memory or not when updating the memory. Parameters of element e_x are normalized by

$$Y_{e_x} = \left(\frac{F_h^{1,e_x}}{F_h^{1,\max}}, \frac{t_{1,e_x}^{\mathrm{up}}}{t_{1,\max}^{\mathrm{up}}}, \frac{t_{1,e_x}^{\mathrm{down}}}{t_{1,\max}^{\mathrm{down}}}, \ldots, \frac{t_{k,e_x}^{\mathrm{up}}}{t_{k,\max}^{\mathrm{up}}}, \frac{t_{k,e_x}^{\mathrm{down}}}{t_{k,\max}^{\mathrm{down}}}\right). \quad (3)$$

Similarity between element e_i and e_j is calculated by degree of grey relation which is expressed as

$$s = \frac{1}{2k+1}\sum_{n=1}^{2k+1}\frac{\sigma + \xi\theta}{\Delta(n) + \xi\theta}, \quad (4)$$

where $\Delta(n) = \left|Y_{e_i}(n) - Y_{e_j}(n)\right|$ $(n = 1, 2, \ldots, 2k+1)$, $\theta = \max\Delta(n)$, $\sigma = \min\Delta(n)$ and $\xi = 0.5$. When updating the memory, all existing elements will be traversed. If the similarity between the new element and an existing element in the memory is larger than or equal to the similarity threshold s_{th} and the fitness of the new element is larger than that of the existing element, the new element will be added by replacing the existing element. If similarities between the new element and all existing elements are smaller than s_{th} and there is free space, the new element will be directly added.

Experiment. To simulate the dynamic mobile cloud environments, we propose a mobility model illustrated in Appendix E to generate users' trajectories and a trajectory with 30 min is used in the experiment. Figure 1 illustrates the performance of our algorithm and the improvement brought by using CMT (Wi-Fi + LTE). MIAGA represents our algorithm and others are comparison algorithms. We also evaluate our algorithm and the using of CMT through other different experiments in Appendix F.

In Figure 1(a), results of MIAGA are smaller than that of other algorithms, which means that MIAGA improves MDs' performance better than

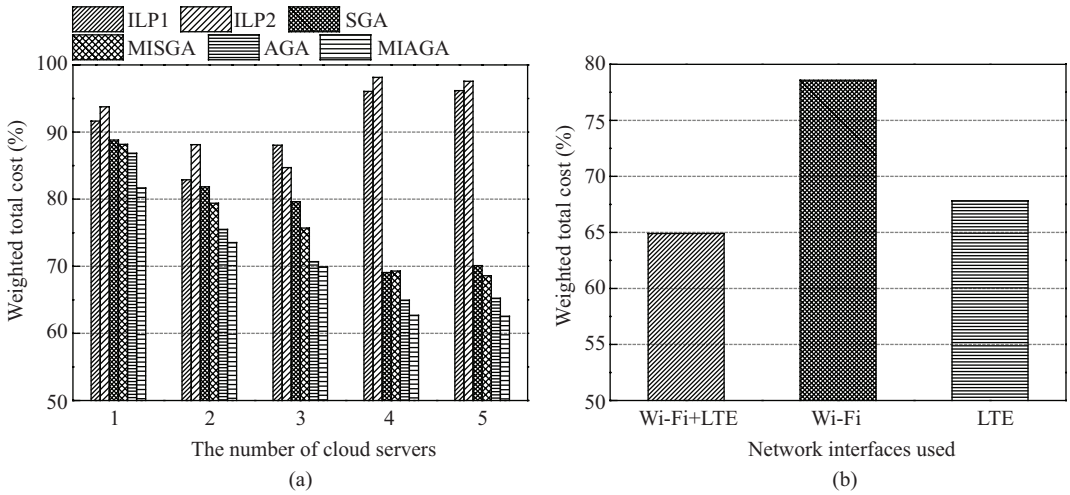

Figure 1 Experimental results. (a) Weighted total cost with different number of cloud servers ($w_e = 0.5$, $w_t = 0.5$); (b) weighted total cost with different interfaces ($w_e = 0.5$, $w_t = 0.5$).

other algorithms. MIAGA can get excellent solutions by costing little extra running time because it just needs a simple calculation of adaptive crossover and mutation probabilities, which makes MIAGA store the historical excellent offloading strategies. These historical offloading strategies help MIAGA improve its convergence rate and its ability to adapt to environmental changes. It can be observed that results of MIAGA decrease (from 81.66% to 62.54%) with the increase in the number of cloud servers, which highlights the advantage of multisite computation offloading.

From Figure 1(b), it can be seen that the result of CMT (64.90%) is smaller than that of Wi-Fi (78.58%) and LTE (67.82%). When using CMT, MIAGA determines whether to offload a component or not and selects the optimal interface to transfer data. MIAGA can utilize advantages of Wi-Fi and LTE to enhance the performance of computation offloading in MCC.

Conclusion. In this letter, a runtime application repartitioning algorithm based on memory-based immigrants adaptive genetic algorithm is proposed and it performs better than other algorithms in dynamic mobile cloud environments. We explore the using of CMT with our algorithm to combat the challenges that wireless links have limited bandwidth and low robustness. The experimental results show that using CMT with our algorithm improves the performance of computation offloading in MCC.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant No. 61502050), YangFan Innovative & Entrepreneurial Research Team Project of Guangdong Province, Civil Aerospace Science and Technology Project and Fundamental Research Funds for the Central Universities.

Supporting information Appendixes A–F. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Kumar K, Liu J B, Lu Y H, et al. A survey of computation offloading for mobile systems. Mobile Netw Appl, 2013, 18: 129–140
2 Kumar K, Lu Y H. Cloud computing for mobile users: can offloading computation save energy? Computer, 2010, 4: 51–56
3 Qi H, Gani A. Research on mobile cloud computing: review, trend and perspectives. In: Proceedings of the 2nd International Conference on Digital Information and Communication Technology and It's Applications, Bangkok, 2012. 195–202
4 Yang S. Memory-based immigrants for genetic algorithms in dynamic environments. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, Washington, 2005. 1115–1122
5 Iyengar J R, Amer P D, Stewart R. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. IEEE ACM Trans Netw, 2006, 14: 951–964
6 Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans Syst Man Cybern, 1994, 24: 656–667