Multi-Criteria Task Offloading Algorithm Based on Mobility Prediction in Ad-hoc Clouds (Post-print)

Authors: Tian Guangdong, Ge Dongyu

Date: 2018-11-29T00:00:00+00:00

Abstract

To improve the efficiency of task offloading in Ad hoc clouds, a multi-criteria task offloading algorithm based on mobility prediction is proposed to address the impact of node random mobility and resource heterogeneity on task offloading. Node departure time is predicted through time series analysis and used as a mobility metric for nodes. The Analytic Hierarchy Process (AHP) is employed to obtain the weights of CPU speed, core count, load, and mobility. Finally, task offloading is performed based on task size and the computed combined weights. Simulation results show that, compared with random task allocation algorithms and Min-Min scheduling algorithms, this algorithm can effectively reduce task execution time and energy consumption.

Full Text

Preamble

Vol. 37 No. 1
Application Research of Computers
ChinaXiv Partner Journal

Multi-Criteria Task Offloading Algorithm in Ad Hoc Cloud Based on Mobility Prediction

Tian Guangdong, Ge Dongyu
(School of Communication & Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

Abstract: To improve the efficiency of task offloading in Ad hoc clouds, this paper proposes a multi-criteria task offloading algorithm based on mobility prediction that addresses the impact of node random mobility and resource hetero-

geneity. The algorithm predicts node escape time using time series analysis and employs this metric as a measure of node mobility. By utilizing the Analytic Hierarchy Process to obtain weights for CPU speed, number of cores, workload, and mobility, tasks are offloaded according to task size and the computed combined weights. Simulation results demonstrate that compared with random task assignment and Min-Min scheduling algorithms, the proposed algorithm effectively reduces task execution time and energy consumption.

Keywords: Ad hoc cloud; multi-criteria; task offloading; time series analysis; analytic hierarchy process

0 Introduction

With the rapid proliferation of smart devices, an increasing number of mobile applications requiring intensive computational resources and high energy consumption have been developed and deployed, attracting widespread attention. Although the performance of smart devices has improved significantly in recent years, it still cannot satisfy users' explosive data processing demands in the big data era. Consequently, Mobile Cloud Computing (MCC) has emerged as a primary technology to address resource constraints in smart devices. MCC provides efficient remote computing services, enabling mobile users to offload computationally intensive tasks to remote clouds or local small-scale cloud devices for processing, with results returned upon completion. This approach reduces local processing overhead and alleviates resource scarcity issues in mobile devices. While remote clouds offer strong computational capabilities, they introduce high latency and overhead; local small-scale clouds provide robust computing power and large communication bandwidth but cannot guarantee ubiquitous access. Ad hoc clouds, as an infrastructure-less extension of MCC, enable task offloading by sharing idle resources with nearby mobile devices, offering advantages such as rapid deployment and flexible scalability.

To achieve efficient task offloading, existing research has designed various protocols and mechanisms for MCC, while studies on Ad hoc clouds remain in their infancy. To address computational-intensive application offloading, reference [6] proposed an online batch scheduling heuristic method that dynamically selects mobile devices for task offloading. Reference [7] introduced a general offloading framework for heterogeneous mobile clouds that utilizes mobile devices, nearby small clouds, and remote clouds to improve MCC service performance and availability. However, this framework does not fully consider the characteristics of Ad hoc clouds. To satisfy delay constraints of mobile devices, reference [8] proposed an energy-efficient task scheduling scheme and designed an adaptive probability scheduler for different tasks that not only meets delay constraints but also minimizes energy consumption when executing computationally intensive real-time applications, albeit with high computational complexity. To reduce energy consumption and computational costs, reference [9] presented a cloudlet-assisted

task allocation mechanism that formulates a two-stage Stackelberg game to determine the number of execution units provided by slave nodes, with master nodes determining compensation strategies accordingly. However, this mechanism does not consider the resource heterogeneity of available mobile devices during task allocation.

Although these methods can reduce task execution time or energy consumption to some extent, they fail to adequately account for the impact of node random mobility and resource heterogeneity characteristics on task offloading in Ad hoc clouds. While numerous factors influence task offloading, node random mobility and heterogeneity, as the most fundamental characteristics of Ad hoc networks, undoubtedly represent the most critical factors affecting task offloading in Ad hoc clouds. Selecting appropriate nodes and allocating resources reasonably have become urgent problems to solve. To address these issues, this paper proposes a multi-criteria task offloading algorithm based on mobility prediction that fully considers node random mobility and resource heterogeneity in Ad hoc clouds to improve task offloading efficiency.

---

1 System Model

To maintain generality, assume there are N nodes in an Ad hoc cloud, where one node (master) has a set of computationally intensive tasks $\{T_i\}$ to process. Relying solely on its own resources to complete these tasks would likely exhaust its resources or introduce high latency, significantly impacting user experience. The other $N-1$ nodes (slaves) possess relatively abundant idle resources, and the master node can offload tasks to slave nodes within a certain range through high-quality wireless networks (Wi-Fi). Upon completion, results are returned to achieve resource expansion.

This paper primarily considers node mobility and heterogeneity, temporarily ignoring communication costs between nodes. To enable simultaneous multi-task execution, assume the master node allocates tasks based on processor time slots. The execution time of the i-th task can be expressed as $T_i = \sum_{s=1}^{S_i} t_s$, where the first time slot size depends on the number of tasks executed in that slot and the minimum task size, while remaining slot sizes depend on the currently executing tasks and the difference with the next smaller task size. Specifically:

$$t_1 = \frac{\min(W_i)}{C_i \times S_i}$$

$$t_s = \frac{W_i - N_i \times C_i \times \sum_{j=1}^{s-1} t_j}{C_i \times S_i}, \quad s > 1$$

where C_i and S_i represent node CPU speed and number of cores, respectively, and W_i denotes the original workload of the node when executing the i-th task.

Prediction represents a fundamental problem in random process theory. For time series prediction, the goal is to predict the value x_{t+l} at future step l from time t, obtaining the predicted value $\hat{x}_t(l)$. Research demonstrates that a stationary time series can always be expressed as a linear combination of white noise, i.e., $x_t = \sum_{j=0}^{\infty} G_j a_{t-j}$. Expanding this yields:

$$x_{t+l} = \sum_{j=0}^{\infty} G_{j+l} a_{t-j} + \sum_{j=0}^{l-1} G_j a_{t+l-j}$$

For the first part on the right side, since $\{a_t\}$ is white noise, future sequence values within the l period cannot be predicted at time t, meaning this part is incalculable and can be regarded as the prediction error $e_t(l)$. For the second part, since all sequence values $\{a_{t-j}\}$ are determined at time t, this part is calculable and can be considered the predicted value $\hat{x}_t(l)$. Thus, equation (5) can be written as $x_{t+l} = \hat{x}_t(l) + e_t(l)$.

Since $\hat{x}_t(l)$ contains all calculable terms, the variance of prediction error $e_t(l)$ can be minimized. Letting the variance of white noise a_t be σ_a^2, we have:

$$Var[e_t(l)] = \sigma_a^2 \sum_{j=0}^{l-1} G_j^2$$

Geometrically, this can be proven as an orthogonal projection in infinite-dimensional space, where $e_t(l)$ is minimized and $\hat{x}_t(l)$ completely regresses onto x_{t+l}. This represents the optimal prediction in the sense of minimizing prediction error variance, i.e., minimum mean square error prediction.

---

2 Multi-Criteria Task Offloading Algorithm

Node mobility and heterogeneity, as the most fundamental characteristics of Ad hoc networks, significantly impact efficient task offloading in Ad hoc clouds. Effective task offloading can substantially reduce task execution time and energy consumption, thereby enhancing user experience. This paper comprehensively considers node mobility and resource heterogeneity in Ad hoc clouds and proposes an efficient task offloading algorithm that effectively reduces both task execution time and energy consumption.

2.1 Mobility Prediction

In Ad hoc networks, node mobility quantification is typically represented as relative movement speed, determined by differences in received signal strength across two consecutive measurements: $v_i = \lg \frac{P_{r1}}{P_{r2}}$, where P_{r1} and P_{r2} are the received signal powers. If $v_i > 0$, the node is considered to be approaching with

relatively good mobility; otherwise, its mobility is considered poor. Although this method is computationally simple, node energy decreases during network operation in practical networks, and signal reception power is easily affected by noise, resulting in significant errors.

Time series analysis can more accurately predict node position information in Ad hoc networks. Any practical Ad hoc network must accomplish specific tasks, so node movement cannot be completely random. If node movement coordinate information is sampled as a discrete time series, time series modeling and prediction can be employed to enable nodes to predict other nodes' position information within a future time period, thereby reducing network overhead and improving resource utilization to some extent.

For random time series modeling, reliability typically requires statistical properties such as stationarity, normality, and zero-mean characteristics. Therefore, after obtaining sample sequences, these properties must first be verified. If not satisfied, preprocessing is required to obtain time series meeting the requirements. Stationarity testing can employ parametric or non-parametric methods. For non-stationary time series, multiple differencing operations are typically applied for stabilization, with restoration performed after subsequent processing. Normality testing, also known as pure randomness testing, treats pure random sequences (white noise following normal distribution) as having no analytical value. In practical Ad hoc networks, node movement always serves specific purposes, so node coordinate sequences cannot be pure random sequences, allowing the normality testing process to be omitted. Zero-mean testing examines whether the mean of the random process described by the time series is zero. Here, time series describe node coordinates, making it difficult to guarantee zero mean for moving node coordinates, thus requiring zero-mean processing of original sequences.

After preprocessing sample sequences, model selection can proceed. In time series analysis and forecasting, the most commonly used models are ARMA (Auto Regressive Moving Average Model), which currently represent the most effective prediction models. After model selection, parameter estimation and model testing are required. ARMA model parameter estimation methods can be broadly classified into three categories: first, parameter estimation methods developed based on time series theory itself; second, optimization parameter estimation methods based on optimization theory iterative computation; and third, parameter estimation methods based on difference models in control theory. This work primarily uses the generalized least squares estimation from the third category. The ARMA modeling process is a dynamic correction process; after completing parameter estimation, model validity must be tested. The fundamental condition for ARMA model establishment is that $\{a_t\}$ must be white noise, so practical model testing methods all revolve around this requirement.

Based on the above steps and specific Ad hoc network node mobility scenarios, the prediction process is as follows: (a) initialize global parameters; (b) sample node position information to obtain sample sequences; (c) test sequence

stationarity—if stationary, proceed to the next step, otherwise perform differencing and repeat step (c); (d) select an appropriate time series prediction model; (e) perform parameter estimation according to the selected prediction model; (f) conduct model testing to determine if prediction errors are white noise—if yes, proceed to the next step, otherwise return to step (d); (g) perform final minimum mean square error prediction based on the above steps and output results.

As previously stated, nodes with completely random movement do not exist in practical Ad hoc networks. Therefore, based on time series prediction and historical position coordinates of nodes, their velocity at the next moment can be estimated. Combined with predicted next-moment position coordinates, the time required for each node to escape the master node' s communication range can be estimated. The longer the escape time, the higher the probability of successful task execution and the better the mobility. Nodes in closer proximity are often considered more suitable for task offloading. Assume a node' s movement trajectory at a certain moment is as shown in [Figure 1: see original paper], with the master node coordinate as the center and communication distance R as the radius to construct its communication range. In the figure, (x_i, y_i) represents node position coordinates, and v_x and v_y are the horizontal and vertical components of current velocity, respectively. The next-moment node coordinates are $(x_i + v_x, y_i + v_y)$. The escape time can be calculated as:

$$T_{pre,i} = \frac{\sqrt{(x_i - x)^2 + (y_i - y)^2} - R}{\sqrt{v_x^2 + v_y^2}}$$

2.2 Weight Determination

The primary heterogeneous factors affecting task execution are CPU speed, number of cores, and workload. Based on these factors and node mobility, the weight formula is constructed as:

$$f_i = \omega_1 s_i + \omega_2 c_i + \omega_3 d_i + \omega_4 t_i$$

where ω_1, ω_2, ω_3, and ω_4 are weight factors; s_i, c_i, d_i, and t_i represent node CPU speed, number of cores, remaining idle workload, and escape time, respectively.

This paper employs the Analytic Hierarchy Process (AHP) to determine weight factors. AHP is a simple, flexible, and convenient multi-criteria decision-making method proposed by operations research expert Professor Saaty. The importance scales and their meanings in this method are shown in .

Importance Scale and Their Definitions

Scale Value	Definition
1	Two factors are equally important
3	One factor is slightly more important
5	One factor is obviously more important
7	One factor is strongly more important
9	One factor is extremely more important
2, 4, 6, 8	Intermediate values between adjacent judgments

Most public cloud service providers can offer unlimited resources to users. In Ad hoc clouds, however, available resources highly depend on node quantity and hardware specifications. Node mobility is another factor requiring consideration for task offloading decisions—nodes with longer predicted residence times are often considered more suitable for task offloading. The comparison of various factors is shown in , where node mobility can be measured using the predicted escape time obtained previously. The data in the table can be adjusted according to actual conditions.

Various Factors Comparison Levels

Factor	CPU Speed	Core Count	Workload	Mobility
CPU Speed	1	5	5	7
Core Count	1/5	1	3	5
Workload	1/5	1/3	1	4
Mobility	1/7	1/5	1/4	1

From , the judgment matrix is obtained as:

$$A = \begin{pmatrix} 1 & 5 & 5 & 7 \\ 1/5 & 1 & 3 & 5 \\ 1/5 & 1/3 & 1 & 4 \\ 1/7 & 1/5 & 1/4 & 1 \end{pmatrix}$$

After normalization and calculation, the weight factors are:

$$\omega_1 = 0.548, \quad \omega_2 = 0.194, \quad \omega_3 = 0.206, \quad \omega_4 = 0.062$$

2.3 Algorithm Description

As previously described, the primary factors affecting task offloading are node CPU speed, number of cores, workload, and mobility. Considering these factors comprehensively and combining time series analysis with AHP, a multi-criteria task offloading algorithm is proposed and described below.

Algorithm 1: Multi-Criteria Task Offloading Algorithm Based on Mobility Prediction

Input: N, $\{s_n\}$, $\{c_n\}$, $\{d_n\}$, $\{t_n\}$, R, $\{T_i\}$

1. Sort tasks in descending order based on size
2. For each node i from 1 to N:
   - Calculate predicted escape time $T_{pre,i}$
   - Calculate combined weight $f(s_i, c_i, d_i, t_i)$ using equation (11)
3. While task list is not NULL:
   - Map each task to the node with maximum combined weight
   - Update node expected workload
   - Remove completed tasks from list, reassign uncompleted tasks
4. Repeat until task list or node list is empty

Output: Task allocation mapping

---

3 Simulation and Analysis

This section presents MATLAB-based simulations comparing the proposed WCST (Workload, Cores, Speed, Time) algorithm with Random Mapping (RM) and Min-Min scheduling algorithms. Random task assignment allocates tasks completely randomly to available nodes, while Min-Min is a classic task scheduling algorithm in grid computing that first identifies the minimum execution time among all tasks, then selects the task with the smallest execution time. The algorithm assigns tasks to resources that yield the minimum completion time, repeating this process until all tasks are scheduled. Min-Min' s limitation lies in its priority scheduling of smaller tasks using high-computation-capability resources, resulting in suboptimal scheduling and poor load balancing when the number of small tasks exceeds large tasks.

The simulation adopts the widely used Random Waypoint mobility model in Ad hoc networks, with four nodes configured with CPUs of: quad-core 1300 MIPS, single-core 1008 MIPS, dual-core 1600 MIPS, and octa-core 1200 MIPS (Million Instructions Per Second). The tasks to be offloaded are configured as shown in , where each task comprises five subtasks. The effectiveness of the proposed algorithm is verified through the completion of six tasks, with task execution time and energy consumption as evaluation metrics.

Task Size Configuration

Task ID	Subtask Size (Instructions, unit: 1.0e+11)
1	0.5, 0.3, 0.4, 0.6, 0.5
2	0.4, 0.5, 0.3, 0.4, 0.5
3	0.8, 0.7, 0.9, 0.6, 0.8
4	1.0, 0.9, 1.1, 0.8, 1.0

Task ID	Subtask Size (Instructions, unit: 1.0e+11)
5	1.2, 1.1, 1.0, 1.3, 1.1
6	1.5, 1.3, 1.4, 1.2, 1.5

Task execution time is the most critical factor affecting user experience. [Figure 2: see original paper] compares the execution times of six tasks using RM, Min-Min, and WCST algorithms. The results show that WCST significantly reduces task execution time compared to RM because RM allocates tasks completely randomly, while WCST fully considers device heterogeneity (CPU speed, core count, and workload) and mobility, performing task allocation based on combined weights of various factors to ensure successful task execution probability and achieve efficient offloading. Compared to Min-Min, WCST consumes slightly longer execution time for tasks 1 and 2 because these tasks are smaller and thus prioritized by Min-Min using high-capability resources. For remaining tasks, WCST more effectively reduces execution time and achieves better load balancing across tasks through more reasonable allocation.

Energy consumption is another crucial factor affecting user experience. [Figure 3: see original paper] compares energy consumption across six tasks for the three algorithms. The results indicate that WCST effectively reduces energy consumption compared to RM because when smart devices have excessive load or poor CPU performance, executing computationally intensive tasks is inefficient and consumes more energy. WCST considers these factors to enable more efficient task offloading. Compared to Min-Min, WCST also reduces energy consumption to some extent and achieves load balancing.

Given the numerous parameters in the algorithm (CPU speed, core count, workload, predicted escape time, etc.), sensitivity analysis is performed focusing on workload and predicted escape time. First, simulations are conducted with the load factor removed from WCST, with task execution time as the metric, shown in [Figure 4: see original paper]. Then, simulations are performed with the predicted escape time factor removed, shown in [Figure 5: see original paper].

As illustrated in [Figure 4: see original paper] and [Figure 5: see original paper], removing the load factor from the original algorithm significantly increases task execution time, while removing the predicted escape time factor also increases execution time, albeit to a lesser degree. This demonstrates that each factor in the algorithm directly affects task execution efficiency, with higher-weight factors having greater impact.

[Figure 2: see original paper] Task Execution Time Comparison Chart
[Figure 3: see original paper] Task Execution Energy Consumption Comparison Chart
[Figure 4: see original paper] Task Execution Time Comparison Chart (Without Load Factor)
[Figure 5: see original paper] Task Execution Time Comparison Chart (Without

Mobility Factor)

4 Conclusion

As an infrastructure-less extension of mobile cloud computing, Ad hoc clouds hold broad application prospects. However, node random mobility and heterogeneity in Ad hoc networks are important factors affecting their application. This paper proposes a multi-criteria task offloading algorithm based on mobility prediction to address the impact of node mobility and heterogeneity on task offloading in Ad hoc clouds. The algorithm predicts node escape time using time series analysis as a mobility metric, fully considers node heterogeneity characteristics, employs AHP to obtain factor weights, and performs task offloading based on combined weights. Simulation results show that although the proposed algorithm has slightly higher complexity, it effectively reduces task execution time and energy consumption compared to random task assignment and Min-Min algorithms, thereby improving user experience.

References

[1] Muraleedharan R. Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture [C]// Computers and Communications. Piscataway, NJ: IEEE Press, 2012: 000059-000066.

[2] Dinh H T, Lee C, Niyato D, et al. A survey of mobile cloud computing: architecture, applications, and approaches [J]. Wireless Communications & Mobile Computing, 2013, 13(18): 1587-1611.

[3] Liu Yanchen, Lee M, Zheng Yanyan. Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system [J]. IEEE Trans on Mobile Computing, 2016, 15(10): 2398-2410.

[4] Yaqoob I, Ahmed E, Gani A, et al. Mobile Ad hoc cloud: A survey [J]. Wireless Communications & Mobile Computing, 2016, 16(16): 2572-2589.

[5] Zhou Bowen, Dastjerdi A V, Calheiros R N, et al. A context sensitive offloading scheme for mobile cloud computing service [C]// Proc of IEEE International Conference on Cloud Computing. New York, USA: IEEE Press, 2015: 869-876.

[6] Li Bo, Pei Yijian, Wu Hao, et al. Heuristics to allocate high-performance cloudlets for computation offloading in mobile Ad hoc clouds [J]. Journal of Supercomputing, 2015, 71(8): 3009-3036.

[7] Zhou Bowen, Dastjerdi A V, Calheiros R, et al. mCloud: A context-aware offloading framework for heterogeneous mobile cloud [J]. IEEE Trans on Services Computing, 2017, PP(99): 1-1.

[8] Shi Ting, Yang Mei, Li Xiang, et al. An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds [J]. Pervasive & Mobile Computing, 2016, 27(C): 90-105.

[9] Guo Xijuan, Liu Liqing, Chang Zheng, et al. Data offloading and task allocation for cloudlet-assisted Ad hoc mobile clouds [J]. Wireless Networks, 2018: 1-10.

[10] Rojas I, Pomares H. Time series analysis and forecasting [J]. Contributions to Statistics, 2016.

[11] Sun Lu. A min-max optimization approach for weight determination in analytic hierarchy process [J]. Journal of Southeast University (English Edition), 2012, 28(2).

[12] Azar H, Majma M R. Using a multi criteria decision making model for managing computational resources at mobile ad-hoc cloud computing environment [C]// Proc of International Conference on Engineering and Technology. Piscataway, NJ: IEEE Press, 2017: 1-5.

[13] Patel G, Mehta R, Bhoi U, et al. Enhanced load balanced Min-min algorithm for static meta task scheduling in cloud computing [J]. Procedia Computer Science, 2015: 545-553.

Note: Figure translations are in progress. See original paper for figures.