Toward a Unified Elastic Computing Platform for Smartphones with Cloud Support

Weiwen Zhang and Yonggang Wen, Nanyang Technological University
Jun Wu, Tongji University
Hui Li, Sichuan University

Abstract

Application offloading has been a popular approach to alleviate a tussle between resource-constrained smartphones and resource-hungry mobile applications. In this article, for leveraging cloud computing, we propose a unified elastic computing platform that supports application offloading for mobile devices, reducing energy consumption on smartphones. The proposed computing fabric consists of an infrastructure-based cloud and an ad hoc virtual cloud formed by a cluster of smartphones. We present both an offloading policy and a mechanism under which applications are delegated to the cloud for execution. For the former, we establish a unified optimization framework to decide where each task of the application should be executed — on the standalone smartphone, in the ad hoc virtual cloud, or in the infrastructure-based cloud. For the latter, we provide implementation strategies for application offloading. The proposed elastic computing platform can enhance the scalability of smartphones, fueling a new wave of innovative mobile applications, for example, anti-virus and gaming on smartphones.

The tussle between resource-hungry applications and resource-poor smartphones is driving the evolution of mobile application platforms. Recently, there has been an explosion of mobile applications on smartphones. Many of these applications are computation-intensive, such as video streaming, data mining, and online gaming. However, these emerging applications are impeded by resource constraints on smartphones. First, smartphones are equipped with a limited battery system that has become one of the biggest complaints by users [1]. Second, smartphones still lag behind their desktop counterparts in terms of computing power and memory capacity. In addition, network connectivity is sporadic because of fading effects in the wireless channel.

Application offloading was proposed as an effective scheme to address this tussle [2]. With the advent of cloud computing [3, 4], two approaches have been investigated for application offloading. The first approach is to offload an application to an infrastructure cloud for execution. In this case, each smartphone is associated with a system-level clone or a delegated surrogate on the cloud, such as Cloudlet [5], Clone Cloud [6], and Weblet [7], which executes applications on behalf of the smartphone. However, network connectivity is not always available. In addition, with the increasing requests for application offloading to the cloud, communication at base stations or access points could become the bottleneck. The second approach is to offload the application to a group of proximal smartphones [8]. These smartphones, connected to each other by a wireless radio, cooperatively execute the application and can be viewed as a virtual cloud computing environment. However, this ad hoc virtual cloud cannot entertain computation-intensive mobile applications due to limited aggregated onboard battery systems and computing resources on smartphones. Therefore, neither of these two approaches can achieve high scalability of smartphones.

In this article, we propose a unified elastic computing platform by combining the ad hoc virtual cloud and infrastructure-based cloud for higher scalability. The infrastructure-based cloud is empowered by execution engines and cloud clones. The ad hoc virtual cloud is formed by the cooperation of smartphones within the same coverage range. With the combined fabric of the infrastructure-based cloud and the ad hoc virtual cloud in our elastic computing platform, application offloading can be conducted more efficiently.

Under the elastic computing platform, we first present the decision-making policy of application offloading (i.e., *offloading policy*). The offloading policy determines each task of the application to be executed on the standalone smartphone or offloaded to the cloud for execution. We build a directed acyclic graph model to represent the task execution of mobile application and define an optimization framework for the offloading policy. In particular, we investigate four special cases (i.e., a node, a linear chain, a tree, and a mesh in the graph) and obtain the offloading policy for each case. We also investigate two implementation strategies for an offloading mechanism, system-level and method-level offloading. In addi-

0890-8044/13/$25.00 © 2013 IEEE

tion, we study opportunities (i.e., task delegation, cloud clone peer-to-peer [P2P] network, data backup, and data staging) and challenges (i.e., performance, security and energy issues) in this platform.

The unified elastic computing platform can enhance the scalability of smartphones, fueling a new wave of innovative mobile applications (e.g., anti-virus and mobile cloud gaming).

The rest of the article is organized as follows. We present an overview of the unified elastic computing platform. We propose the offloading policy in this computing platform. We present implementation strategies for the offloading mechanism. We discuss opportunities and challenges of the computing platform. We highlight two mobile applications that can benefit from the computing platform. Finally, we summarize the article and suggest future work .

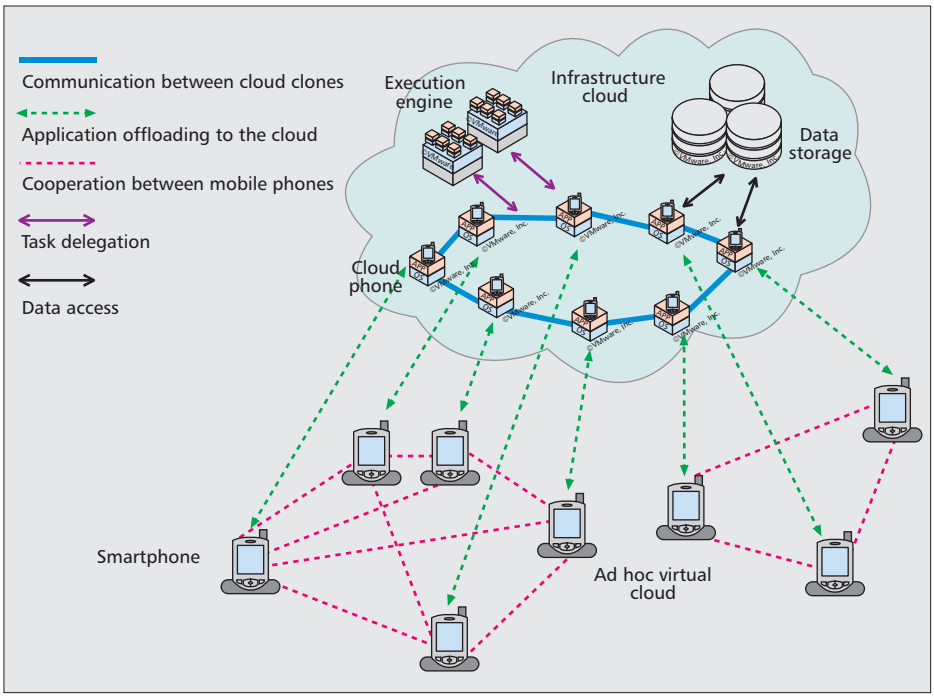

Figure 1. *Generic architecture of the elastic computing platform is composed of an infrastructure cloud and an ad hoc virtual cloud.*

An Overview of the Unified Elastic Computing Platform

This section presents an overview of the unified elastic computing platform. We first propose a generic architecture of the platform and then present execution strategies of mobile applications under the platform.

Generic Architecture of the Elastic Computing Platform

Figure 1 illustrates a generic architecture of the elastic computing platform, which consists of an ad hoc virtual cloud and an infrastructure-based cloud.

The infrastructure-based cloud is empowered by cloud clones and remote execution engines, which extends the computing power and reduces the energy consumption of smartphones. In the infrastructure-based cloud, there is an identical image of the system for each smartphone, which is referred to as the cloud clone. The cloud clone executes mobile applications on behalf of the smartphone, thus reducing application delay and energy consumption on the smartphone. The cloud clones are logically connected, forming a cloud clone P2P network. There are also execution engines and data storage in the back-end that open up more opportunities for application offloading.

The ad hoc virtual cloud is formed by a cluster of smartphones nearby that work cooperatively to accomplish application offloading. A smartphone communicates with its neighbors directly by a local wireless network interface (e.g., Bluetooth). As the smartphone moves from one environment to another, it will join a new cluster of smartphones and can still benefit from application offloading seamlessly. As a result, the ad hoc virtual cloud copes with the issue of sporadic wireless network connectivity between the smartphone and the infrastructure-based cloud.

In this elastic computing platform, the infrastructure-based cloud and the ad hoc virtual cloud complement each other to address the issues of limited battery power on the smartphone and sporadic network connectivity. Hence, the elastic computing platform enhances the scalability of smartphones.

Execution Strategies under Elastic Computing Platform

In this platform, computing resources, including any cloud clone and smartphone, are elastically allocated to the execution of mobile applications. The application execution can be made among three execution strategies, including:

- Standalone execution by the individual smartphone
- Cooperative execution by the cluster of smartphones
- Cloud execution by cloud clone or execution engine

The standalone execution requires the computation of the application to be completed on the individual smartphone. The cooperative execution and the cloud execution consume computing resources in the cloud by application offloading via a wireless network.

The Offloading Policy for the Unified Elastic Computing Platform

In this section, we propose an optimization framework for the offloading policy in the unified elastic computing platform. The offloading policy is to determine which execution strategy (i.e., standalone execution, cooperative execution, or cloud execution) should be chosen for each task of the application. We first present an optimization framework, and then investigate four special cases (i.e., a node, a linear topology, a tree, and a mesh) of offloading policy.

The Optimization Framework of the Offloading Policy

We construct a general directed acyclic graph model to represent the task flow in the application, referred to as the task-flow graph. In the graph, a node represents a task, and a link connecting two nodes represents data dependence between the corresponding tasks. Data dependence indicates that a task cannot be executed until it receives some required data from its precedent tasks. In addition, each link is labeled with the cost between the adjacent nodes (e.g., energy consumption on the smartphone), which depends on the application profile and network conditions. The total cost for the application execution is the summation over the cost of each link.

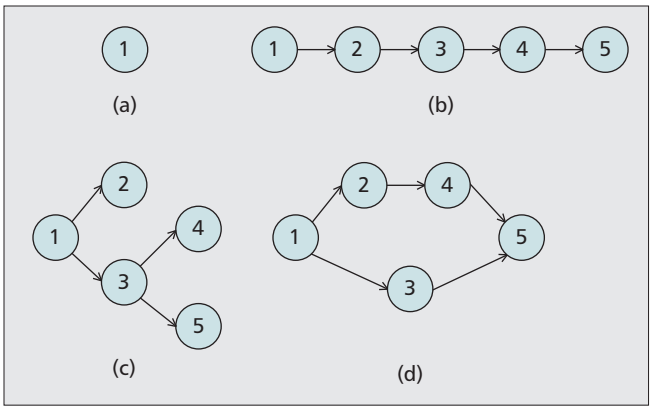

Figure 2. *Examples of task-flow graphs in different topologies: a) only one active node; b) linear chain topology; c) tree structure; d) regular mesh structure.*

We propose an optimization framework as follows. The objective is to minimize the total cost (e.g., energy consumption on the smartphone) while respecting execution constraints for all the tasks (e.g., application delay). We aim to derive optimal or near-optimal algorithms to allocate tasks into computing resources (i.e., standalone smartphone, a cluster of smartphones, and cloud clone). However, due to its combinatorial nature, the optimization problem is NP-complete. To obtain useful results and insights, we focus on a few particular cases that are computationally tractable, as shown in Fig. 2, including:

- Only one active node, representing the whole application
- A linear chain topology, representing a sequential list of tasks
- A tree structure, representing a tree-based hierarchy of tasks
- A regular mesh structure, representing a lattice-based topology of tasks

In the following subsections, we provide solutions for these particular cases.

Energy-Optimal Offloading Policy: One-Node Case

In this subsection, we consider the energy-optimal application execution policy when the task-flow graph has only one active node, as illustrated in Fig. 2a. The decision is to determine whether the entire application should be executed on the smartphone or the cloud, with an objective to minimize the energy consumption on the smartphone while meeting the application's completion deadline.[1]

We can obtain the optimal energy for both the standalone and cloud executions as presented in [9]. Consider an application profile (T, L), where the application with L bits of input data should be completed before time delay T. For standalone execution, the computation workload is completed on the smartphone by varying CPU frequency for each workload. For cloud execution, input data is transmitted to the cloud by adapting the transmission rate in response to network conditions. The minimum energy consumption of the smartphone by standalone execution and cloud execution are $\mathcal{E}_m^* = ML^3/T^2$ and $\mathcal{E}_c^* = C(n)\, L^n/T^{n-1}$, respectively. Herein, M is a constant depending on the chip architecture of the smartphone, and $C(n)$ is a function of monomial order n depending on the energy model of data transmission over the wireless channel. Comparing $\mathcal{E}_m^*$ with $\mathcal{E}_c^*$, we can determine which execution is more energy-efficient.

[1] *For ease of presentation, we focus on the analysis of cloud execution; the same rationale can be applied for cooperative execution.*

Figure 3 shows the results of optimal application execution for the one-node case. First, for $n < 3$ and $n > 3$, optimal application execution regions of the standalone and cloud executions are separated by a line. Geometrically, the slope of the line can be interpreted as a threshold,

$$R_{th} = \left[\frac{M}{C(n)}\right]^{\frac{1}{n-3}},$$

which is a constant. We define an effective data consumption rate as the ratio of input data size and delay deadline; that is, $R_e = L/T$ for the determination of the optimal execution. For example, for $n = 4$ in Fig. 3c, if the point (T, L) is below the line, cloud execution is optimal; otherwise, standalone execution is optimal. Second, for the case of $n = 3$, the energy consumption of standalone and cloud executions have the same order of L and T; thus, the decision depends on the comparison between M and $C(n)$. In the setting of this case, $M < C(n)$; hence, standalone execution is the optimal execution.

Energy-Efficient Offloading Policy: The Linear Chain Case

In this subsection, we investigate offloading policy for the task flow of a linear chain as illustrated in Fig. 2b.

We first construct a directed acyclic graph to model task execution in a linear chain in Fig. 4b. Two dummy nodes, S and D, are introduced for application initiation and termination. Node k indicates that task k has been completed on the smartphone, while node $\underline{k}$ represents that task k has been completed by the cloud clone, where $k = 1, 2, \ldots, n,$ and n is the total number of tasks in the application. A link between the adjacent nodes represents data dependence between the tasks. In this case, data dependence requires that task k can only be started after the completion of task $k - 1$, since the output data of task $k - 1$ is input data of task k. Also, each link is associated with a nonnegative cost to complete the corresponding task (e.g., energy consumption on the smartphone or completion time). In addition, if task k accesses local resources (e.g., GPS and sensors), it should be executed on the smartphone, and hence the cost of the link connecting with node $\underline{k}$ is infinite.

Based on the graph in Fig. 4, we then formulate the offloading policy as a constrained shortest path problem as presented in [10]. The objective is to find the shortest path in terms of energy consumption between S and D in the graph, subject to the constraint that total completion time of that path should be less than or equal to time deadline T_d. A path p is feasible if total completion time satisfies the delay constraint. A feasible path p^* with minimum energy consumption is the optimal solution among all the feasible paths. Mathematically, we have

$$\min_{p\in\mathcal{P}} \quad \mathbb{E}[e(p)] = \mathbb{E}\left\{\sum_{(u,v)\in p} e_{u,v}\right\}$$

$$s.t. \quad \mathbb{E}[d(p)] = \mathbb{E}\left\{\sum_{(u,v)\in p} d_{u,v}\right\} \le T_d,$$

where $\mathcal{P}$ is the set of all possible paths, and $e_{u,v}$ and $d_{u,v}$ are energy consumption and completion time between any adja-

cent nodes u and v on the path p, respectively. Note that the expectation is taken over the channel state due to fading effects in wireless networks. This constrained optimization problem, however, is NP-complete.

To solve this optimization problem efficiently, we adapt a canonical algorithm, Lagrangian Relaxation Based Aggregated Cost (LARAC). We define the aggregated cost function, $L(\lambda) = \mathbb{E}[e(p)+\lambda d(p)] - \lambda T_d$, where λ is a Lagrange multiplier. By the Lagrange duality principle, we have $L(\lambda) \le \mathbb{E}[e(p^*)]$, which gives a lower bound for the optimal solution of offloading policy. More details of the algorithm can be found in [10], and are here omitted due to page limits.

Figure 5 illustrates an example of offloading policy for the linear chain topology under a stochastic channel model where data transmission rate is independent and identically distributed. We observe that the decision depends on the ratio between data size and workload. For example, for task 1, its input data size is large while workload is small; hence, it is executed on the smartphone. For task 3, its input data size is small and workload is large; hence, it is efficient to offload task 3 to the cloud clone for execution.

Offloading Policy by Parallel Execution: The Tree and Mesh Cases

In this subsection, we study offloading policy by parallel execution for the task-flow graphs of tree and mesh. We observe that loose data dependence of tree and mesh and multiple wireless interfaces on the smartphone provide us opportunities to offload tasks more efficiently.

First, tree and mesh structures have looser data dependence on tasks than linear chain topology. Hence, tasks without data dependence can be executed in parallel. For example, in Fig. 2c, tasks 2 and 3, at the same level of the tree, can be simultaneously executed by the cloud clone in the infrastructure-based cloud or the cluster of smartphones in the ad hoc virtual cloud.

Second, using multiple wireless interfaces on the smartphone can accelerate data transmission before parallel execution of tasks on the cloud. Input and output data of tasks can be concurrently transmitted via multiple wireless network interfaces (e.g., Bluetooth and 3G) to reduce data transmission time. For example, in Fig. 2d, output data of task 1 can be transmitted to the cloud clone and smartphones nearby simultaneously by 3G and Bluetooth before the execution of tasks 2 and 3, respectively.

Based on these two opportunities, the offloading policy for tree and mesh task-flow graphs is to jointly allocate computing resources for the execution of all the tasks and choose wireless network interfaces for data transmission. As an example, we consider the mesh structure in Fig. 2d, aiming to minimize application completion time. It is equivalent to minimize the absolute value of difference between completion time of the upper and lower paths between tasks 1 and 5 in the graph.

The Offloading Mechanism of the Unified Elastic Computing Platform

There are two alternative implementation strategies of offloading mechanism: the method-level offloading approach (e.g., MAUI [11]) and system-level offloading approach (e.g., Cloudlet [5] and CloneCloud [6]). In this section, we discuss the offloading mechanism of the infrastructure-based cloud and ad hoc virtual cloud, respectively.

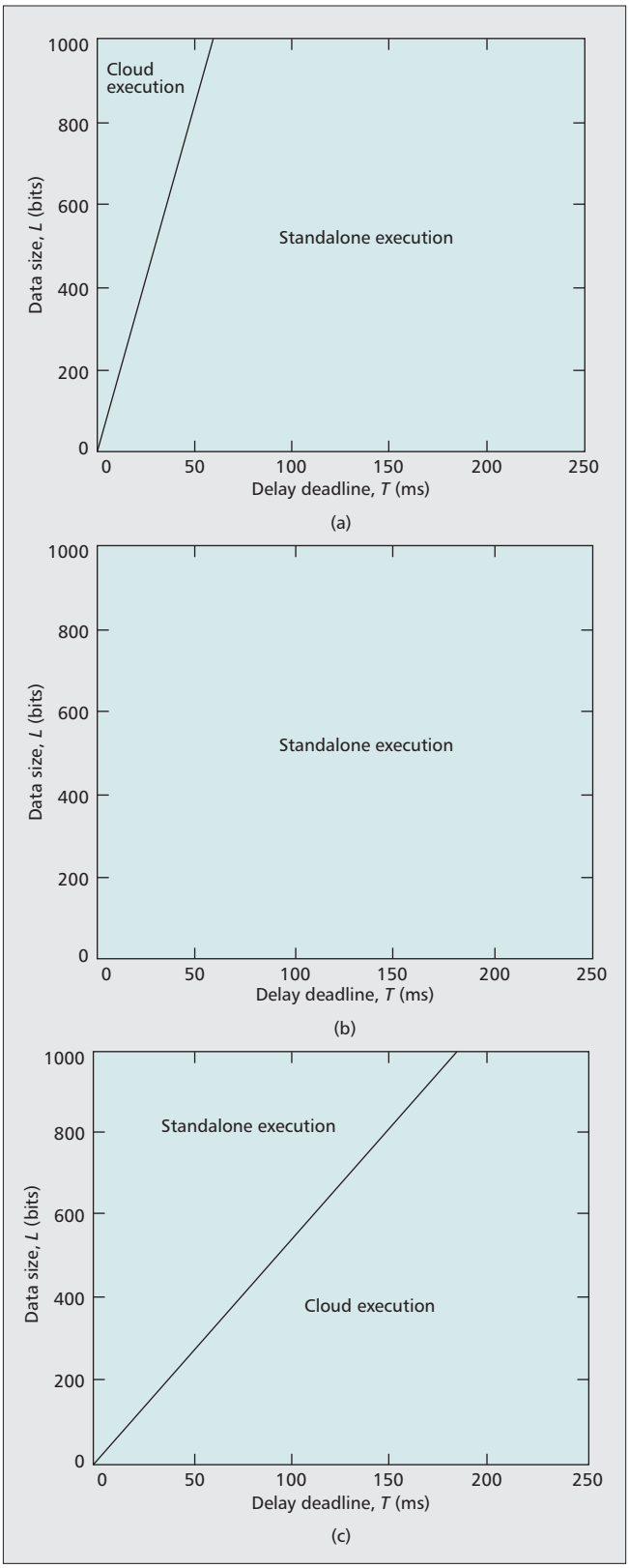

Figure 3. *Optimal application execution for the one node case. For* n = *2 and* n = *4, optimal application execution regions of the standalone execution and cloud execution are separated by a line. The slope of the line is Eq. 1, where* M *is a constant depending on the chip architecture of the smartphone, and* C(n) *is a function of monomial order* n *depending on the energy model of data transmission over a wireless channel. For* n = *3, the decision of optimal execution depends on the comparison between* M *and* C(n).

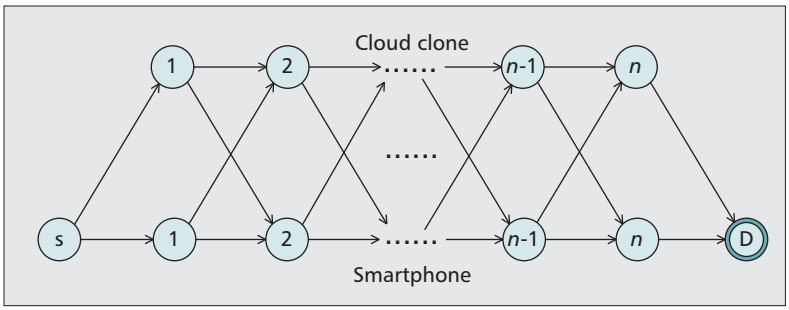

Figure 4. *A graph of task execution flow for linear chain topology. Node* k *represents that task* k *has been completed on the mobile device, while node* $\underline{k}$ *represents that task* k *has been completed on the cloud.*

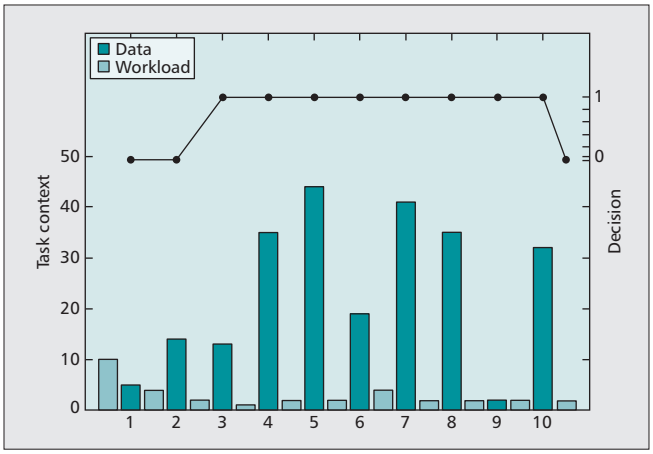

Figure 5. *Offloading policy under the IID channel model for linear chain topology. There are 10 tasks. The horizontal axis denotes the tasks to be executed, the left vertical axis represents task context, including normalized workload (M cycles) and input/output data size (kb), and the right vertical axis represents the optimal task execution location (0 for smartphone and 1 for cloud clone). For a particular task, the darker bar represents the workload of the task, while its left and right lighter bars represent the input and output data sizes of the task, respectively. The line represents the execution decision. The expected data transmission rate over the wireless channel is 10 kb/s. The application time deadline is 0.7s.*

The Offloading Mechanism of the Infrastructure-Based Cloud

The method-level offloading approach can be applied to implement application offloading in the infrastructure-based cloud, due to its high execution efficiency. In the method-level offloading approach, an application is partitioned, and a part of code is executed by remote procedure call (RPC). This approach provides a set of low-level programming interfaces for remote execution, which achieves high execution efficiency. However, it requires programmers to decide which method or module should be offloaded. For this implementation, we can adopt an architecture similar to OpenMobster, which is an open source mobile cloud platform.

We can also choose the system-level offloading approach to implement application offloading in the infrastructure-based cloud due to its easy programmability. In this approach, an image of a smartphone is cloned in the cloud through virtual machine technology (e.g., Xen). With a system identical to that of the smartphone, the clone provides a set of high-level programming interfaces, which is easy to program. However, completely cloning the system of the smartphone can increase the complexity of security control in the infrastructure-based cloud. An alternative solution is to set up a set of weblets [7] elastically via software-oriented architecture (SOA) to execute the application. For this implementation, we can adopt an architecture based on weblets.

Offloading Mechanism of the Ad Hoc Virtual Cloud

In the ad hoc virtual cloud, the method-level offloading approach can be adopted to implement application offloading. As storage of smartphones is limited, the system-level offloading approach via VM clone is impractical in the ad hoc virtual cloud. As a result, we choose the method-level offloading approach via RPC for the mechanism.

Opportunities and Challenges of Unified Elastic Computing Platform

In this section, we discuss opportunities and challenges of the unified elastic computing platform.

Opportunities

The proposed elastic computing platform provides opportunities to enhance the capability of smartphones, which include task delegation, and cloud clone P2P network, data backup, and data staging. These are the advantages of our proposed platform compared to previous work.

Task Delegation — The execution of computation-intensive tasks can be delegated to remote execution engines in the cloud infrastructure, which is referred to as task delegation. Some applications (e.g., media transcoding) can consume more computing resources than the cloud clone can afford. In this case, we can leverage back-end execution engines with sufficient computing resources to execute these tasks.

Cloud Clone P2P Network — A cloud clone P2P network is formed by a set of cloud clones in the infrastructure-based cloud, which can mitigate sporadic network connectivity among smartphones. Cloud clones are logically connected with more stable connectivity and higher bandwidth than their associated smartphones. Communication among smartphones can be performed via cloud clones. In this way, there are new opportunities for energy-efficient collaborative mobile applications among smartphones.

Data Backup — If the smartphone is lost accidentally, the data can be recovered in another secure smartphone by data backup from the cloud clone, which is similar to iCloud.

Data Staging — The cloud clone in the infrastructure-based cloud can serve as a proxy for data staging, which can reduce the latency for the smartphone to fetch data. For example, an original high-definition video, stored in the infrastructure, can be transformed into HTTP streaming format with diverse playback rates in any cloud clone [12]. When a mobile user requests video, the cloud clone replies with a required streaming rate.

Challenges

There are also challenges in the elastic computing platform, including performance, security, and energy issue.

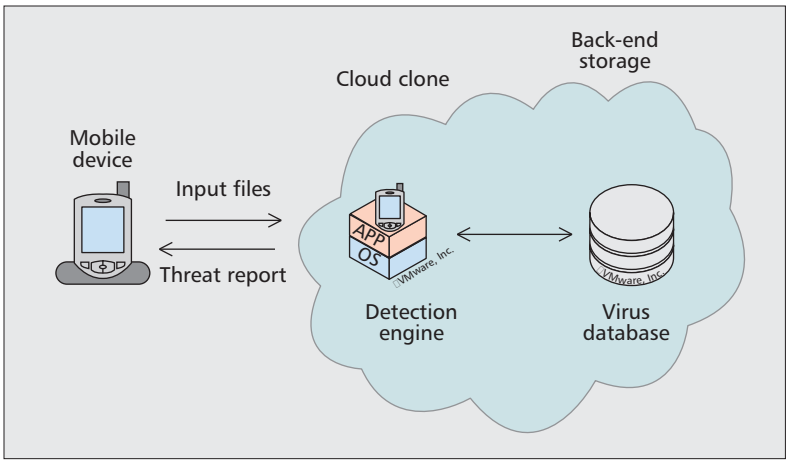

Figure 6. *Architecture of virus scanning by a cloud clone.*

Performance — We refer to performance as the service response time of application offloading for mobile users. To guarantee performance, we need a topology design of cloud clones in the infrastructure-based cloud. In [13], there are peer-based, proxy-based, and clone-based models for the topology design. A suitable topology of cloud clones should be chosen to achieve high performance for different types of mobile applications.

Security — Security issues should be addressed in our proposed elastic computing platform. First, cloud clones should be trusted. The smartphone should identify a trusted cloud clone by itself (i.e., trust establishment [5]) or check the identity of the cloud clone based on trust measurements conducted by a third party (i.e., reputation-based trust [5]). Second, cloud services (i.e., computation and storage) provided by the cloud infrastructure should be trusted. We need to ensure that there are no other hidden programs running behind the service. In addition, security mechanisms should be light-weight without incurring much energy consumption on smartphones.

Energy — Although we have mitigated the battery insufficiency of an individual smartphone by application offloading, energy consumption of the cloud side has not yet been accounted for, especially for ad hoc virtual clouds. Note that the battery systems of smartphones in an ad hoc virtual cloud is also limited. Therefore, we need a strategy to choose smartphones and distribute workload based on their computing capability and residual battery volume for application offloading in order to prolong the lifetime of all the smartphones.

Applications

In this section, we discuss two mobile applications that can benefit from offloading in our proposed elastic computing platform: antivirus and mobile cloud gaming.

Mobile Antivirus Application

In our elastic computing platform, we can exploit the cloud clone to scan for viruses and malicious content, as shown in Fig. 6. Initially, a smartphone sends input files to the cloud clone for scanning. Then the detection engine in the cloud clone scans the content of the input files against a virus database maintained by the infrastructure cloud. Finally, a threat report is sent back to the smartphone.

We can apply the threshold policy to decide whether virus scanning should be executed on the smartphone or on the cloud. The policy depends on the size of files to be scanned, completion time for scanning, and current network channel status. We can evaluate the value of the effective data consumption rate to obtain the energy-optimal execution policy.

Mobile Cloud Gaming

Our proposed elastic computing platform can serve as a mobile cloud gaming system [14], which is a promising paradigm to eliminate the resource constraint on the smartphone for mobile games. First, a cloud clone is delegated to complete the computation-intensive tasks (e.g., 3D graphics rendering), while the smartphone as a thin client receives results displayed on its screen. Second, a cloud clone P2P network can enhance the performance of multi-user mobile games on smartphones. Cloud clones in the P2P network can communicate with each other to obtain the information from mobile users and complete the computation in the game.

We envision that the offloading policy can be applied in mobile cloud gaming. It is quite common for a complex mobile gaming application to be decomposed into multiple tasks in a linear chain, tree, or mesh topology. The offloading policy will optimize the execution of real-time interactive mobile games.

Conclusion

We combine the ad hoc virtual cloud and infrastructure-based cloud as the fabric of the unified elastic computing platform to enhance the scalability of smartphones. Under this platform, we present an offloading policy by a unified optimization framework and an offloading mechanism to implement application offloading. We also discuss two applications (antivirus and mobile cloud gaming) that can benefit from the elastic computing platform. In the future, we will consider more generic graphs for general offloading policy and build real mobile applications.

References

[1] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," *Proc. ACM Symp. Principles of Distrib. Computing*, 1996, pp. 1–7.
[2] R. Balan *et al.*, "The Case for Cyber Foraging," *Proc. 10th ACM Special Interest Group on Op. Sys. European Wksp.*, 2002, pp. 87–92.
[3] K. Kumar and Y. H. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," *IEEE Computer*, vol. 43, no. 4, 2010, pp. 51–56.
[4] M. Chen *et al.*, "Enabling Technologies for Future Data Center Networking: A Primer," *IEEE Network*, 2013.
[5] M. Satyanarayanan, R. C. P. Bahl, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, 2009, pp. 14-23.
[6] B. G. Chun *et al.*, "Clonecloud: Elastic Execution Between Mobile Device and Cloud," *Proc. 6th Euro. Conf. Computer Sys.*, 2011, pp. 301–14.
[7] X. W. Zhang *et al.*, "Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing," *Mobile Networks and Applications*, vol. 16, no. 3, 2011, pp. 270–84.
[8] G. Huerta-Canepa and D. Lee, "A Virtual Cloud Computing Provider for Mobile Devices," *Proc. 1st ACM Wksp. Mobile Cloud Computing and Services: Social Networks and Beyond*, 2010.
[9] Y. Wen *et al.*, "Energy-Optimal Execution Policy for a Cloud-Assisted Mobile Application Platform," tech. rep., 2011.
[10] W. Zhang, Y. Wen, and D. Wu, "Energy-Efficient Scheduling Policy for Collaborative Execution in Mobile Cloud Computing," *32nd Annual IEEE Infocom*, 2013, pp. 190–94.
[11] E. Cuervo *et al.*, "MAUI: Making Smartphones Last Longer With Code Offload," *Int'l. Conf. Mobile Sys., Applications, and Services*, 2010, pp. 49–62.
[12] W. Zhang *et al.*, "QoE-Driven Cache Management for HTTP Adaptive Bit Rate Streaming Over Wireless Networks," to appear, *IEEE Trans. Multimedia*, 2013.
[13] G. Hu, W. P. Tay, and Y. Wen, "Cloud Robotics: Architecture, Challenges and Applications," *IEEE Network* Special Issue on Machine and Robotic Networking, vol. 26, no. 3, 2012, pp. 21–28.

[14] W. Cai, V. C. Leung, and M. Chen, "Next Generation Mobile Cloud Gaming," *Proc. IEEE Int'l. Symp. Mobile Cloud, Computing, and Service Engineering*, 2013.

Biographies

WEIWEN ZHANG (wzhang9@ntu.edu.sg) received his Bachelor's degree in software engineering and Master's degree in computer science from South China University of Technology (SCUT) in 2008 and 2011, respectively. He is currently a Ph.D. candidate in the School of Computer Engineering at Nanyang Technological University (NTU) in Singapore. His research interests include cloud computing and mobile computing.

YONGGANG WEN (ygwen@ntu.edu.sg) received his Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT) in 2008. He is currently an assistant professor with the School of Computer Engineering at NTU. Previously, he worked at Cisco as a senior software engineer and system architect for content networking products. He also worked as a research intern at Bell Laboratories and Sycamore Networks, and served as a technical advisor to the chairman at Linear A Networks, Inc. His research interests include cloud computing, mobile computing, multimedia networks, cyber security, and green ICT.

JUN WU (wujun@tongji.edu.cn) received his B.S. degree in information engineering and M.S. degree in communication and electronic system from Xidian University in 1993 and 1996, respectively. He received his Ph.D. degree in signal and information procesing from Beijing University of Posts and Telecommunications in 1999. He joined Tongji University as a professor in 2010. He has been a principal scientist at Huawei and Broadcom before joining Tongji. His research interests include wireless communication, information theory, and signal processing.

HUI LI (huili0154@hotmail.com) received his M.S degree in computer science from Simon Fraser University in 1997. He received his Ph.D. degree in computer science from Sichuan University in 2008. He is currently a professor in the College of Computer Science at Sichuan University, China. He worked at Northern Telecom as a senior software developer. He also worked as a key system engineer in Wisesoft. His research interests include virtual reality, multimedia systems, computer graphics, cloud computing, and social networks.