Indian J.Sci.Res. 17(2): 536 - 540, 2018

ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

TOWARDS MOBILE OPPORTUNISTIC IN CLOUD COMPUTING

[1]P.Pratheepkumar, [2]J. Josephine Sharmila, [3]Dr. D. Arokiaraj
[1]Department of Computer Science, NIFT-TEA College of Knitwear Fashion, Bharathiar University, Tirupur, Tamilnadu
[2]Department of Electronics and Communication Engineering, DMI Engineering College, Kanyakumari, Tamilnadu
[3]Department of Management Studies, Aurora's PG College, Ramanthapur, Hyderabad, Telangana

Abstract—With the advent of wearable computing and the resulting growth in mobile application market, we investigate mobile opportunistic cloud computing where mobile devices leverage nearby computational resources in order to save execution time and consumed energy. We propose enabling generic computa-tional offloading to heterogeneous devices that include Cloud, mobile devices, and cloudlets. We propose a generic and flexible architecture that maximizes the computational gain with respect to variant metrics such as minimizing the response time, reducing the overall energy consumption and increasing the network life-time. This novel architecture is designed to automate computation offloading to various compute resources over disrupted network connections due to the nature of mobile devices.

Keywords: cloud lets,computational gain**.**

I. Introduction

Mobile devices, such as smartphones and tablets, are increasingly capable devices with processing and storage capabilities that make significant step improvements with every generation. Yet, various application tasks, such as face recognition, body language interpretation, speech and object recognition, and natural language processing, exceed the lim-its of stand-alone mobile devices. Such applications resort to exploiting data and computational resources such as the Cloud [1], [2]. By 2018, mobile cloud applications will account for 90% of total mobile data traffic which represents an annual growth rate of 64% [3].

Various solutions for computation offloading to more powerful surrogate machines [4], known as cyber-foraging [5] [6], have been proposed such as CloneCloud [7] and MAUI [8]. In addition, and due to the significant impact of large RTT's on energy consumed by mobile devices while offloading to distant clouds, researchers proposed bringing computational resources, known as cloudlets [9] closer to mobile devices. Others, such as Cirrus [10], and Serendipity [11] take some steps towards computational offloading to neighboring mobile devices which we refer to as Mobile Device Clouds (MDC). Also, with processing capacity of mobile devices being mostly unused [12], mobile devices, individually and in clusters, form a potentially significant computational resource that can be tapped at low cost and with low latency. Mtibaa et al. [13] discuss offloading algorithms to neighboring mobile devices to ensure a fair consumption of energy across a group of mobile devices belonging to the same individual or household.

In this paper, we aim at rethinking the perspective of compu-tational offloading from default mobile to cloud techniques tai-lored for certain applications, to a more dynamic and adaptive peer-to-peer offloading architecture. This novel architecture conducts offloading decisions based on a set of parameters thatdefine the nature of the offloading opportunities available and make the best decision based on potentially conflicting user objectives. We propose a generic computational offloading architecture over heterogeneous devices that include mobile MDCs, cloudlets, and clouds. We address specific challenges remaining in each of these options, along with their integration into a single offloading architecture. We also design and im-plement an offloading manager proof-of-concept prototype and present basic performance analysis that outperform offloading to only one computing candidate.

II. Opportunistic Computing Spectrum

The spectrum of computational opportunities will be governed by two dimensions: computation and communication opportunities.

A. Computation Opportunities

With typical node mobility, we envision a spectrum of computational contexts, some of which are shown in Fig. 1. At one extreme, we have the traditional cloud-computing context where devices are intermittently connected to distant cloud resources as shown in Fig. 1(a). We also exploit cloudlets[9] on which we dynamically instantiate the service software to enable mobile device computation offloading as shown in Fig. 1(b). The original cloudlet concept implicitly constrains the offloading of computation to a single cloudlet. In our pro-posed work, we also consider the case when the mobile device is intermittently connected to a set of cloudlet-like resources which we call mobile Cloudlets as shown in Fig. 1(c). At

[1]Corresponding Author

the other extreme of the spectrum, is the domain least explored in literature where mobile devices directly contact other mobile devices (MDC), as shown in Fig. 1(d).

We believe that the offloader's (i.e., the task initiator) envi-ronment for remote computation will be, in the most general case, a hybrid of such computational contexts and systems. Therefore, mobile devices need to discover the capabilities of their environment and adapt their remote computation decisions accordingly.

B. Communication Opportunities

Computation opportunities listed above exhibit different communication challenges that range from stable wired com-munication with Cloud, to heterogeneous intermittent multi-hop connectivity in the case of MDCs. Computation tasks need to be potentially routed to specific devices depending on the application, with the premise that results need to be returned back to the initiator. Such requirement will impact decision making regarding which routes and devices to choose when offloading a task. We list three main challenges of our opportunistic computing solutions:

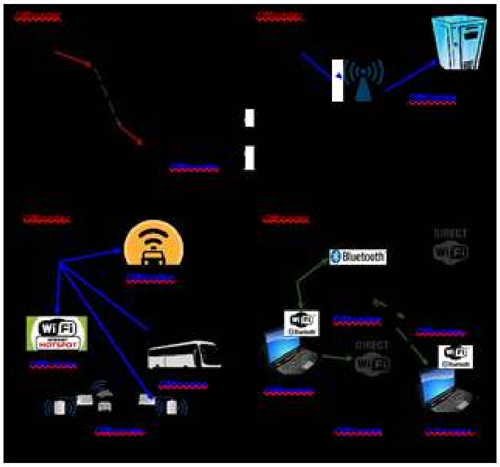


Fig. 1. Opportunistic Computing Spectrum

1) Intermittent connectivity: First, since the underlying connectivity is often variable and unknown, it is difficult to map computations to various nodes with a guarantee that the necessary code and data can be delivered and the results are received in a timely fashion. This leads to a conservative ap-proach for distributing computation geared towards providing protection against future network disruptions. Second, because the network bandwidth is intermittent, it is more likely to be a bottleneck for the successful completion of the offloaded task execution. This suggests avoiding data needed for the next computation traversing the network while scheduling sequen-tial computations on the same chosen node. Third, without reliable control channels, the network cannot be relied upon to provide reachability to all nodes which would be needed for coordination and control. This suggests maintaining local control and developing mechanisms for loose coordination.

2) Partitioning and remote execution: Mobile applications' execution across multiple devices introduces challenges on partitioning the task execution between the devices efficiently, automatically, and optimally. We aim at providing significantly more flexibility in application partitioning. For instance, a given partitioning strategy that can be optimal in one scenario might perform poorly in another. Connectivity characteristics as well as device capabilities are generally unknown to appli-cations' programmers, or the possible configurations render it too difficult to customize.

3) A Common Architecture for computational offloading:

Addressing the application partitioning challenge above re-quires a framework that would enable adaptive and potentially real time decisions of sub-task allocation. Allocation decisions depend heavily on the sub-tasks profiling in terms of commu-nication requirements, running time, and power consumption.

III. Opportunistic Computing Architecture

Computational resources are heterogeneous and should co-operate in an opportunistic mobile computing environment where a mobile device can choose between all computational offloading opportunities based on given metrics. By abstracting all compute resources, our proposed architecture achieves such flexibility and maximizes the computational gain with respect to different metrics such as minimizing the response time, reducing the overall energy consumption, and increasing the network lifetime.

Our system architecture, illustrated in Fig. 2, is designed to (i) automate computation offloading to various compute resources, and (ii) accommodate computation offloading over disrupted network connections due to the nature of mobile devices. It consists of different components that work together to answer two main questions: for a given task T consisting of D data input and C computation requirements, (iii) which compute resource is more suitable to run this task?, and if of-floading is required, given available connectivity opportunities and resources, when and how do a we offload the task?

Our mobile opportunistic computing architecture consists of a task profiler, offloading manager, routing manager, task scheduler, set of computing resources, and a set of databases.

The Task Profiler is responsible for profiling each task and its allocated resources. It receives a task T from a mobiledevice application, and decomposes it (if applicable) into k subtasks Ti; i= 1::k. Each subtask is then profiled asa combination of D_{Ti} a data input, and C_{Ti} computation requirements, then it will be forwarded to the Offloading Manager.

The Offloading Manager is the heart of this architecture. It implements a resource discovery and estimation manager that (1) discovers available resources, and (2) estimates the availability of the resources in the near future. It scans and detects all available resources across all available interfaces and models each as a virtual interface. Once the virtual interface is discovered, the resource discovery and estimation manager estimates the computation resources of each virtual interface (i.e., the available energy, actual cpu and memory utilization, the interface utilization cost, and the available bandwidth). The offloading manager will implement a utility maximization algorithm that compares user-customized gains for each of the 4 main compute resources:

1) Compute locally: Tasks run locally on the mobile device. The device stores (in the Device Capabilities and the Energy Profiling databases) a history of previous task performances on the device, and estimates the perfor-mance of a given task based on similar history tasks.

2) Cloudlet/Mobile Cloudlet: The device offloads the task to a nearby cloudlet. Cloudlets are generally resources characterized by their short RTT connections and high throughput. The connection with such devices is subject to high disruption. The architecture provides a con-nection stability estimation algorithm that predicts the disruption factor between the device and cloudlets.

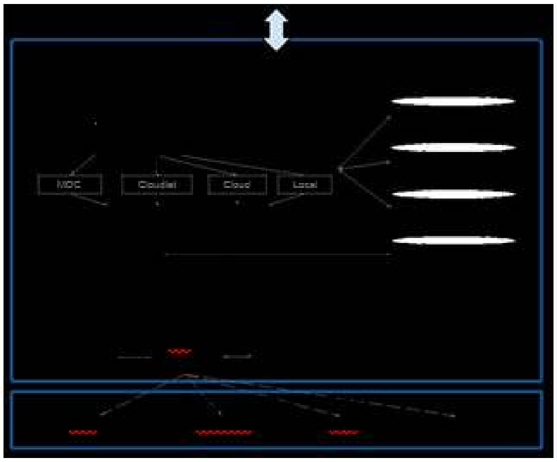


Fig. 2. Mobile opportunistic computing high level architecture.

3) MDC: Offloading to other mobile devices. MDC's main goal is to leverage nearby computational resources in order to reducing energy costs or execution time by bringing task executors closer to mobile task initiators. MDC is motivated by the fact that processing capacity of mobile devices being mostly unused and idle most of the time [12].

4) Cloud: Offload to data centers characterized by their "unlimited" power, storage, and computational re-sources. These resources are usually costly since they adopt the pay-as-you-go paradigm. The connection to a distant cloud is characterized by higher RTT, leading to high energy consumption, and high disruption rates.

The Offloading Manager decides whether or not a given task needs be offloaded or locally executed depending on the characteristics of this task and the capabilities of the local peer. First, it interacts with the privacy and security engine in order to check if the task T is eligible for offloading or not. It then runs a set of objective functions that compare running the task T locally or migrating it to other neighboring resources with minimum resource capabilities $R_{min} = fE_{min}; C_{min}; S_{min}g$ within t, where E_{min}; C_{min}, and S_{min} represent the mini-mum expected energy, computation, and storage capabilities respectively on the neighboring compute resource peers. Once the offloading decision is made, subtasks will be forwarded to the Resource Allocation manager. The resource manager requests, therefore, the allocation of resources for guaranteed tasks (i.e., high priority tasks) upon negotiating the resources that will be used to execute a given task. It also sends a request to free the resources allocated for a particular task if the task is completed or when it received an error signal from the forwarding manager via its fault tolerance engine.

The Routing & Replication Manager is responsible for routing computation tasks to a particular device, with the premise that results need to be returned back to the initiator. Such requirement will impact decision making regarding which routes and devices to choose when offloading a task.

Contact unpredictability adds two main challenges which are (i) the unpredictable duration of a contact makes the decision of whether such contact is long enough or not to complete the process of handing off the computation subtask, waiting for it to complete on the remote device and receiving the result, and (ii) the unpredictable waiting time for an eventual contact. The routing engine implements different techniques that help the initiator mobile device classify each contact and estimate its duration and the waiting time to meet the next computing entity. It also decides whether or not a task can be replicated in order to ensure success delivery within a given delay.

The Task Scheduler schedules the subtasks belonging to each task to multiple compute resources. It maintains the status of each task and subtask and dispatches them to either local compute resources or to the forwarding engine to be executed remotely. It also implements a failure recovery engine to re-assign a delayed task to other compute resources based on their initial rank given by the offload manager. The manager also receives tasks from neighboring mobile opportunistic computing peers for local execution, it therefore forwards the task to the local computing resources and forwards the results back to the forwarding engine.

The Forwarding Manager is responsible for exchanging tasks and resource history statistics with neighboring compute resources. It updates the databases with up-to-date connectivity stats with new and existing neighbors. Upon receiving an order to migrate the task to distant devices, the forwarder selects the most suitable devices to run the task T within the given time and resource constraints. It uses stored information about historical contact summaries and social information to infer the expected connection and inter-connection duration between neighboring devices. The forwarder is also receiving tasks from neighboring forwarder engines. In this case, it forwards the task to the task scheduler then to the local Computing Resources.

Quality of Service (QoS) monitors error rates and providing different levels of guarantees on the completion time expected for particular tasks. These tasks or applications have gen-erally different priority levels. Therefore, our scheme aims at managing tasks so that top priority applications will not be compromised. It manages the use of resources using a priority scheme that classifies tasks (high, moderate, and low priority) and replicates the most critical tasks to multiple offloadee nodes in order to reduce the error rate and reduce the latency. The fault tolerance module implements a proactive and a reactive technique to deal with task execution errors. The proactive approach involves predicting connectivity failure with a set of offloadee devices and replicates some tasks in order to avoid unnecessary waiting delays. The reactive approach, however, involves initiating a fast recovery upon detecting (or predicting) a task execution failure.

The Computing Resources represent a basic abstraction of the processing capabilities that exist on an opportunistic computing peer device. If the task is scheduled for local execution, it arrives to the computing resources for immediate execution. However, in case of remote execution, the task will be forwarded to a selected remote device then to its task scheduler which will forward it to its computing resources for remote execution on the remote device. Once the task executed, results will be sent back to the task scheduler and then to the offloader's task manager and then to the application.

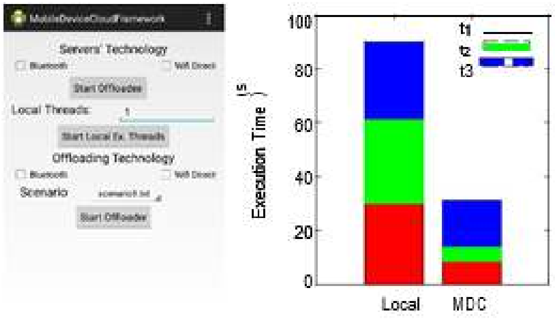
IV. Implementation Prototype & Results

As a first step towards implementing and testing our peer-to-peer architecture, we design and implement the following modules described earlier: the offloading manager and the task scheduler.

A. Proof-Of-Concept Prototype

Our prototype is implemented as an android application installed on all devices regardless of their nature (i.e., offloader or offloadee). Our application allows users to activate or deactivate the collaboration mode at any time (i.e., users' wiliness to offer their computational resources for others). Collaborating users specify the wireless technologies that can be used by the application. The task scheduler then determines which technology (from those specified) can be used to maximize the utility function. In this prototype, we consider minimizing the total execution time.

We create dummy tasks at the offloader device. Theses tasks can either be executed locally or offloaded to one or multiple remote devices (offloadee). Each task consists of computation requirements (measured in FLOPs) as well as input and output data (measured in Bytes). Fig. 3-(a) shows a snapshot of our prototype user interface where the offloadee node specifies the maximum number of threads allowed to run on its device and the offloader node runs a set of tasks with specific parameters and arrival times as stated in a scenario file.



(a) Protype snapshot (b) Experimental results

Fig. 3. Proof-of-concept implementation and results

B. Experiment & Preliminary Results

In our experiment, we consider a network of three mobile devices: a Galaxy S5 (GS5) running Android 4.4.4, a Nexus 7 (N7), and a Nexus 10 (N10) running Android 5.0.2. GS5 has 3 tasks t_1; t_2; t_3. We first measure the required time to run these tasks locally without offloading to any other device. As shown in Fig. 3-(b), the GS5 requires more than 90s to executestheses tasks sequentially with an average of 30s for each task. However, GS5 achieves a 3 speedup while offloading 2 ofthese tasks to computationally more powerful nearby devices such as the N7 and the N10. It, therefore executes all tasks in only 31.24s using our prototype. Our prototype application distributes the tasks as follow: t_1 is offloaded to N10 taking only 8.7s to run, t_2 is offloaded to N7 which executes it in 14.1s, and the GS5 locally executes t_3 in 31.2s. In this experiment, our prototype uses WiFi Direct and a round robin algorithm that assigns tasks to each encountered device.

V. Conclusion And Future Work

This paper discusses a novel peer-to-peer architecture for mobile opportunistic offloading. It presents a detailed archi-tecture consisting of multiple modules responsible for routing, scheduling, discovering, securing, and incentivizing remote computational offloading. We have also implemented a proof-of-concept prototype running on android devices. Our pre-liminary prototype achieves 3 speedup in execution time using only 3 mobile devices. We plan to develop, test, and evaluate the architecture proposed in this paper. We will propose a multitude of algorithms to investigate different routing strategies, objective functions, incentive systems, and QoS and Guarantee models.

References

[1] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, “Cuckoo: a computation offloading framework for smartphones,” in Mobile Computing, Applica-tions, and Services. Springer, 2012, pp. 59–79.

[2] M. Satyanarayanan, “A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets,” SIGMOBILE Mob. Comput. Commun. Rev., vol. 18, no. 4, pp. 19–23, Jan. 2015. [Online]. Available: http://doi.acm.org/10.1145/2721914.2721921

[3] T. Cisco, “Cisco visual networking index: Global mobile data traffic forecast update, 2012–2017,” Cisco Public Information, 2013.

[4] J. Flinn, “Cyber foraging: Bridging mobile and cloud computing,” Synthesis Lectures on Mobile and Pervasive Computing, vol. 7, no. 2, pp. 1–103, 2012.

[5] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang, “The case for cyber foraging,” in Proceedings of the 10th workshop on ACM SIGOPS European workshop, 2002.

[6] R. K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb, “Simpli-fying cyber foraging for mobile devices,” in ACM MobiSys, 2007.

[7] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “Clonecloud: elastic execution between mobile device and cloud,” in Proceedings of the sixth conference on Computer systems, ser. EuroSys ’11. New York, NY, USA: ACM, 2011, pp. 301–314. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966473

[8] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “Maui: making smartphones last longer with code offload,” in MobiSys’10, 2010, pp. 49–62.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” Pervasive Computing, IEEE, vol. 8, no. 4, pp. 14–23, 2009.

[10] C. Shi, M. H. Ammar, E. W. Zegura, and M. Naik, “Computing in cirrus clouds: the challenge of intermittent connectivity,” in Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ser. MCC ’12. New York, NY, USA: ACM, 2012, pp. 23–28. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342515

[11] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, “Serendipity: enabling remote computing among intermittently connected mobile devices,” in MobiHoc, 2012, pp. 145–154.

[12] L. A. Barroso and U. Holzle,¨ “The case for energy-proportional com-puting,” IEEE computer, vol. 40, no. 12, pp. 33–37, 2007.

[13] A. Mtibaa, A. Fahim, K. Harras, and M. Ammar, “Towards resource sharing in mobile device clouds: Power balancing across mobile de-vices,” in Proceedings of the second edition of the MCC workshop on Mobile cloud computing, ser. MCC ’13. New York, NY, USA: ACM, 2013.