An Experimental Analysis on Cloud-based Mobile Augmentation in Mobile Cloud Computing

Saeid Abolfazli, *Student Member*, IEEE, Zohreh Sanaei, *Student Member*, IEEE, Mojtaba Alizadeh, *Student Member*, IEEE, Abdullah Gani, *Senior Member*, IEEE, and Feng Xia, *Senior Member*, IEEE

Abstract — *Recent consumer electronic technologies have created momentous ground for smartphones in various domains, particularly healthcare and education. However, smartphones' miniature nature imposes intrinsic limitations on computational capabilities and battery lifetime that encumber performing Resource-intensive Mobile Applications (RMAs). Cloud-based Mobile Augmentation (CMA) is the-state-of-the-art augmentation model that leverages proximate and distant clouds to increase, enhance, and optimize computing capabilities of mobile devices aiming at execution of RMAs, which breeds Mobile Cloud Computing (MCC). This study analyzes impacts of mobile-cloud distance and number of intermediate hops — as influential factors — on CMA performance which are not yet investigated. The results indicate the correlation between distance and intermediate hops on overall execution costs (time and energy) of RMAs. The mathematical modeling and benchmarking unveil that distance has negligible impact on latency, whereas intermediate hops increment and communication overhead significantly degrade application performance and complicate energy and time estimation in CMA system*[1].

Index Terms — Cloud-based Mobile Augmentation, Mobile Cloud Computing, Proximity Analysis, Computation Outsourcing, Offloading

I. Introduction

Latest advances in consumer electronics technologies have provided a momentous foundation for mobile devices, especially smartphones. Human dependency to mobile devices, particularly smartphones is rapidly raising in varied domains such as mobile health [1] towards rich mobile applications [2]. Smartphones are substituting multitude of consumer electronics, particularly digital cameras, Internet browser, and multimedia player [3]. However, smartphones' computational capabilities are constraint by miniature nature and current technological limitations that obligate computational augmentation [4], [5].

Computational augmentation efforts aim to offload the resource-intensive computational code(s) of the mobile applications to the remote resources outside the mobile device to save execution time and energy — as the scarcest resource of mobile devices [6]. Recently, the state-of-the-art Mobile Cloud Computing (MCC) technology has gained momentous ground to alleviate resource deficiencies of mobile devices (i.e., computing, battery power) by leveraging cloud resources towards Cloud-based Mobile Augmentation (CMA). "CMA is the state-of-the-art system leveraging cloud computing technologies and principles to increase, enhance, and optimize processing power of mobile devices" [7]. CMA envisions outsourcing resource-intensive mobile applications (RMAs) (e.g., speech recognition and some mathematical operations like power and prime operations) with least application execution time and optimized power consumption.

In majority of current CMA efforts [8]–[10], researchers utilize distant cloud resources to augment mobile devices. Distant giant clouds feature rich resources and high scalability located far away from mobile nodes that originates long WAN latency and degrades crispness of application response time. In CMA efforts, the ratio of computational energy saved via augmentation against the energy wasted by communication overhead is imperative and requires thorough analysis prior performing augmentation. Although in some recent efforts [8],[11] researchers study various aspects of mobile augmentation in MCC, the impact of distance between mobile clients and cloud servers, which is very crucial in performance and success of CMA is not yet fully studied. Kumar et al [10] study three augmentation decision factors of wireless bandwidth, computation intensity, and data transmission volume, but investigating the impacts of mobile-cloud distance and number of intermediate hops on mobile computational augmentation is not undertaken.

This article employs mathematical modeling and benchmarking on real devices to investigate the impacts of mobile-cloud distance and intermediate hops on performance of mobile applications when they are augmented leveraging distant cloud resources. It also demonstrates feasibility of utilizing RESTful architectural style [12] and Service-

[1] This work is funded by the Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant - UM.C/HIR/MOHE/FCSIT/03.

Saeid Abolfazli is with Mobile Cloud Computing Lab, department of CSIT, University Malaya, Malaysia (e-mail: abolfazli@ieee.org).

Zohreh Sanaei is with Mobile Cloud Computing Lab, department of CSIT, University Malaya, Malaysia (e-mail: sanaei@ieee.org).

Mojtaba Alizade is with Malaysian-Japan International Institute of Technology, University Technology Malaysia (amojtaba2@live.utm.my).

Abdullah Gani is with Mobile Cloud Computing Lab, department of CSIT, University Malaya, Malaysia (e-mail: abdullah@um.edu.my).

Feng Xia is with School of Software, Dalian University of Technology, Dalian 116620, China (e-mail: f.xia@ieee.org).

Oriented Computing (SOC) [13] for augmenting resource-constraint mobile devices. REST (Representational State Transfer) is an architectural style of web services that improves performance of service-based applications compared to traditional styles. SOC is a computing model to build complex systems using independent prefabricated processing building blocks called services that have standard interface. Benefits of using RESTful SOC model in the cloud-based augmentation systems are investigated and impacts of intermediate hops on CMA performance are evaluated. Exploiting SOC contributes to omission of code offloading overhead [14]. Time and energy consumption of two RMAs in three execution strategies of native, proximate cloud, and distant cloud are analyzed and synthesized. The results are validated via benchmarking. The results indicate that augmenting mobile devices via distant clouds originates high anomalous time and energy overheads that degrade application performance and encumber accurate time and energy estimation of applications. Such impacts are increased when data communication rises. To the best of authors' knowledge, this is the first effort to investigate the impact of mobile-cloud distance on CMA using REST and SOC.

The remainder of this article is organized as follows: Related works are discussed in Section II. Section III describes mathematical analysis followed by benchmarking analysis in Section IV. Conclusions and future works appear in Section V.

II. Related Works

Limited efforts investigate varied implications of exploiting cloud-based computation resources for mobile devices augmentation. Miettinen et al [9] investigate energy efficiency of mobile devices when using distant cloud resources with focus on communication technology and computational efficiency of native device and conclude that mobile augmentation efficiency depends on the workload intensity, data communication overhead, and employed technology. Perrucci et al [15] investigate energy consumption of mobile devices and unveil that heterogeneous communication technologies of mobile devices are characterized with different usability and performance and recommend to employ energy-aware communication technology selection for enhanced performance of mobile applications.

Several efforts [11], [16]–[20] aim to reduce the impacts of WAN latency on application performance. However, comprehensive investigation on the elements of WAN latency in mobile augmentation is necessary to achieve optimal resource allocation to the resource-intensive tasks. Satyanarayana [21] introduces cyber foraging as a feasible solution to enhance mobile application performance by which mobile users perform computation and data processing on nearby fixed computer(s) in coffee shops or cinema halls. Cloudlet [16] highlight the impacts of long WAN latency on response time and argue that long WAN latency will unlikely be solved. Cloudlet as an alternative of using giant clouds assumes that public computers can be accessed by mobile users if computer owners are incentivized and their security is ensured. However, Cloudlet is likely suitable for low- and medium-intensity tasks which need moderate computational power. MOMCC [19] establishes a cloud of nearby mobile devices to mitigate the WAN latency while augmenting computational capabilities of mobile devices. SAMI [20] leverages multi-tier cloud infrastructures comprises of nearby and distant computational resources to better meet the computational resources of intensive tasks. Tolia et al [22] investigate the application interactivity when mobile device is augmented using remote servers and conclude that long WAN latency decrease application responsiveness. However, because of the assumption that long client-server distance increases latency, scientific investigation on the rationale behind WAN latency remains open research area, which is undertaken in this work.

III. Mathematical Model

In this section, the execution time and energy consumption of RMAs that employ computational resources of clouds are formulated to investigate the impacts of mobile-cloud distance and hops on execution performance of RMAs. In this model, two math utility applications, namely prime and matrix multiplication are analyzed, assuming that their computations are performed using server-side web services without native code execution (except lightweight user interface) inside the mobile device. Data is migrated from the client to the server and results are sent back and displayed on the screen; there will not be any code migration since codes are already available in the cloud.

A. Execution Time Analysis

The overall execution time in mobile-cloud applications is the total values of mobile computation time, round-trip delay, and cloud computation time represented in (1). The mobile computation is the time to perform native computations. The round-trip latency is the delay of sending request to the cloud and receiving the response. The cloud computation delay is the time that request is processed in server side and results are produced. It is feasible to shrink the volume of native computations and demand rich computing resources in the cloud, but decreasing the round-trip delay sounds non-trivial due to very large number of influential factors which are briefly discusses as follows. The overall execution time is represented by T_{total} as

$$T_{total} = T_{mobile} + T_{RT} + T_{cloud} \tag{1}$$

where T_{mobile} is the time to perform native computation that is negligible in this model, T_{RT} is the round-trip delay, and T_{cloud} is the computing latency of the cloud. Computation latency depends on the computational capabilities (CPU clock speed, RAM volume, I/O performance, and environmental context like temperature) of the computing server, whereas round-trip delay comprises of four delays, including transmission, propagations, processing, and queuing delay [17]. Thus, round-trip latency is:

$$T_{RT} = 2 \times \left(T_{prog} + T_{tran} + T_{proc} + T_Q \right) \quad (2)$$

where T_{prog} or propagation delay is the time to transmit the packet through the medium from the client to the server, T_{tran} or transmission delay is the time to send a packet into the communication medium (wire/wireless). T_{proc} or processing delay is the time the intermediate nodes take to handle the packet on the network, and T_Q or queuing delay is the time packet queued before transmission. Due to raising complexity of processing data in propagation stage, processing delay breeds significant impacts on latency, though the processing power of intermediate nodes is dramatically high.

- *Propagation delay* is directly related to the distance between two network nodes and is the time taken for the packet to travel from mobile to the cloud and vice versa. Hence, the propagation delay is the sum of delays between each nodes pair throughout the path between mobile and cloud. Assuming identical delay between each pair of nodes, it is essential to multiply the propagation delay into the number of hops. Moreover, HTTP protocol of TCP/IP needs establishment of connection prior the request and response transmission. It needs to consider delay to establish and terminate the connection, and receive the acknowledgment. If *H* represents the number of hops, propagation delay is:

$$T_{prog} = 4 \times H \times \left(D / S \right) \quad (3)$$

where *D* is the distance between each nodes pair and *S* is the velocity factor or propagation speed of the signal in the medium which is light speed in vacuum (i.e., 3×10^8).

- *Transmission delay:* T_{tran} depends on the amount of data transfer over the network bandwidth *β*. Also, congestion avoidance mechanisms like congestion window, maximum transmission unit, and slow start used in TCP impose transmission delay by limiting packet number and size. Such delays manifest in all TCP transmission with relative impacts. So, they are disregarded in this work to focus on studying the impacts of hops' number and avoid complexity. Thus, for each *PS* bits packet, the total transmission delay of *P* packets is:

$$T_{tran} = P \times \left(PS / \beta \right) \quad (4)$$

- *Queuing delay:* is the amount of time taken in *H-1* hops for transmission of packet over the network with transmission delay of T. So, queuing delay is:

$$T_Q = \left(H - 1 \right) \times \left(PS / \beta \right) \quad (5)$$

- *Processing delay:* When the client-server distance increases, there is high chance of increase in the number of intermediate networking hops. In every hop there is a processing delay of $T_{proc(i)}$ that are assumed to be identical. Thus, for a client-server distance consists of *H* hops T_{proc} is:.

$$T_{proc} = 4 \times \left(H - 1 \right) \times T_{proc} \quad (6)$$

By substituting (3), (4), (5), and (6) in (2), T_{RT} is:

$$T_{RT} = 2 \times \begin{pmatrix} \left(4 \times H \times D / S \right) + \left(P \times PS / \beta \right) + \\ \left(\left(H - 1 \right) \times \left(PS / \beta \right) \right) + \left(4 \times \left(H - 1 \right) \times T_{proc} \right) \end{pmatrix} \quad (7)$$

Equation (7) advocates that the round-trip latency highly depends on the number of hops between mobile and cloud, rather than the physical mobile-cloud distance. Direct impact of distance on round-trip latency is negligible considering the high transmission speed of current networks.

With the help of the following examples, impacts of distance and number of hops on communication latency are described. Consider the common characteristics for these scenarios:

- Transmission speed of each link is 56kbit/s.
- Processing delay for all hops is 5µs.
- Packets size is 8000 bits.
- Propagation speed is 3×10^8.

In these scenarios, hops are equal to the number of hops in Section IV. The mobile-cloud distances are 320 km with 14 hops and 6500 km with 23 hops. The number of packets for both scenarios is 20. Hence, T_{RT} can be written as:

$$T_{RT_{(1)}} = 2 \times \begin{pmatrix} \left(4 \times 14 \times \frac{320 \times 1000}{3 \times 10^8} \right) + \left(20 \times \left(\frac{8000}{56000} \right) \right) + \\ \left(13 \times \left(\frac{8000}{56000} \right) \right) + \left(4 \times 13 \times \left(5 \times 10^{-6} \right) \right) \end{pmatrix} = 9.54 \quad (8)$$

For the second scenario,

$$T_{RT_{(2)}} = 2 \times \begin{pmatrix} \left(4 \times 23 \times \frac{320 \times 1000}{3 \times 10^8} \right) + \left(20 \times \left(\frac{8000}{56000} \right) \right) + \\ \left(22 \times \left(\frac{8000}{56000} \right) \right) + \left(4 \times 22 \times \left(5 \times 10^{-6} \right) \right) \end{pmatrix} = 12.2 \quad (9)$$

The difference between the round-trip latency of these two scenarios is as much as 27% which is noticeable difference. Equation (7) depicts that the main influential factor is the number of hops between the mobile and cloud computer (which is very closely related to the mobile-cloud distance) and ration of packet size to bandwidth. However, if the number of hops is fixed, distance increase does not change the results.

To further infer results from the time model, further analysis are performed that are plotted in Fig.1. Fig.1(a) depicts round-trip latency when mobile-cloud distance is increasing from

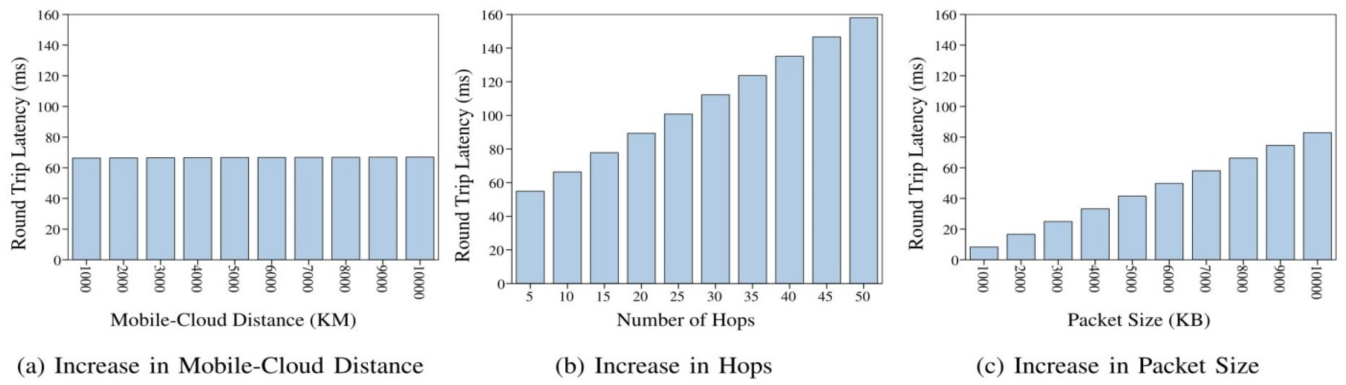

Fig. 1. Round-trip Latency Analysis: Distance has negligible impact whereas number of hops and packet size are significantly influential.

1000 to 10000 km with step of 1000 km. The number of hops is ten and the packet size is 5000 bytes. As illustrated, there is negligible difference when distance increases without raise in number of hops, because the transmission speed of todays' network is remarkably high and increase in distance does not affect much on the latency. Fig.1(b) depicts round-trip latency when the number of mobile-cloud hops is raising from five to 50. The mobile-cloud distance is 1000 km and the packet size is 5000 bytes. As results indicate, the number of hops significantly impacts on round-trip latency. This delay is due to the overheads of forwarding data packets at any hop.

Fig.1(c) depicts round-trip latency when data packet size changes from 1000 to 100000 bytes. The number of hops is ten and the mobile-cloud distance is 1000 km. Fig.1(c) testifies the impacts of packet size on the latency. Although the packet size itself has direct impacts on the latency, increase in packet size originates excess overhead at each hop to perform pre- and post-forwarding processes. Indeed, when the packet size increases, the overhead of congestion avoidance, excess header size, CRC validation, compression, and encryption increase the processing time at each hop. Further analysis is undertaken to better understand the impacts of hop number and packet size (see Fig. 2). For sure, number of hops and packet size directly impact on round-trip latency for cloud-based RMAs. 3-D view of the analysis unveils that round-trip latency proportionately grows when number of hops and packet size rise. Hence, the number of mobile-cloud hops and data packet size are the most influential factors on round-trip latency.

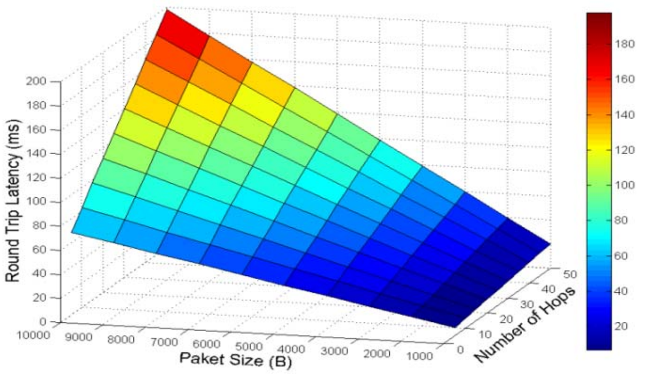

Fig. 2. Joint impacts of number of hop and packet size on round-trip latency for cloud-based RMAs. Number of hops and packet size directly impact on round-trip latency.

B. *Mobile Energy Consumption Analysis*

The overall energy consumption of a typical client/server RESTful application, E_{total}, is the total values of energy consumed to perform client-side computation (E_{client}), to transmit request to the server (E_{tran}), to wait to receive the response (E_{wait}), and to reintegrate and synchronize the results with the client code (E_{sync}) which is formulated in (10).

$$E_{total} = E_{client} + E_{tran} + E_{wait} + E_{sync} \tag{10}$$

The energy consumption of client side can be divided as

$$E_{client} = E_{native} + E_{pre} \tag{11}$$

where E_{native} is the energy consumed to run the native code which is negligible in this evaluation and E_{pre} is the energy consumed to prepare the request for transmission which consists of tasks such as encryption and compression of data. To enhance energy efficiency of the mobile device, native computations considered almost zero, so that the overhead of transmitting request does not exceed the augmentation benefits.

The E_{tran} is the amount of energy used to transmit the data to the cloud and receive the response. If γ represents the energy consumption of transmitting a bit from client to server over the network, then

$$E_{tran} = \gamma \times content \tag{12}$$

E_{wait} is a function of computing latency that has direct correlation with the overall execution time of the application. Hence, for each time unit (ms), there is α energy unit (mJ) consumed. So, E_{wait} can be written as

$$E_{wait} = \alpha \times \left(T_{RT} + T_{cloud}\right) \tag{13}$$

By replacing (2) in (13), E_{wait} can be written as

$$E_{wait} = 2\alpha \times \left(T_{prog} + T_{tran} + T_{proc} + T_{Q}\right) + \alpha \times T_{cloud} \tag{14}$$

If right side of E_{client}, E_{trans}, E_{wait}, *and* E_{sync} are replaced in (10), E_{total} can be written as

$$E_{total} = E_{native} + E_{pre} + (\gamma \times content) + E_{sync} + 2\alpha \times (T_{prog} + T_{tran} + T_{proc} + T_Q) + \alpha \times T_{cloud} \quad (15)$$

In the applications developed in this experiment, the native code is lightweight and transmission overhead is very low since the code stored in the cloud and is not migrating. Hence, energy's need solely depends on the amount of data being sent. Such lightweight design shrinks the value of E_{native} and E_{pre} since encrypting or compressing small amount of data are not resource-intensive tasks. Also, E_{tran} highly depends on the amount of data to and from the mobile device. On the other hand, $\alpha \times T_{cloud}$ is very small considering tremendous power of today's giant clouds. E_{sync} is also negligible due to nature of selected applications. Thus, considering negligible E_{native}, E_{pre}, E_{sync} and $\alpha \times T_{cloud}$ in (15), the energy consumption of the mobile device highly depends on the amount of data traffic and round-trip latency of the mobile application when using the cloud. From the time analysis, it is known that increase in the number of hops significantly raise round-trip latency. It can be concluded that similar delay degrades energy efficiency of mobile devices. Therefore, the less number of intermediary nodes and amount of data transmission, the better would be energy efficiency of mobile-cloud applications. The round-trip latency model and mobile energy results are validated via benchmarking on real device that are presented as follows.

IV. Experimentation

To validate findings of Section III, benchmarking is performed over two cloud servers, because realizing theoretical assumptions (i.e., distance, hops, and packet size) in real environment is impractical. For example, deploying ten servers located in 1000-10000 KM with 5-50 intermediate hops for benchmarking RMAs' performance is practically unfeasible.

A. Model

In this section, benchmarking model is discussed from two aspects of hardware and software.

1) Hardware: The target mobile client is a smartphone featuring dual-core application processor with 1.2 GHz CPU and 1GB RAM. The servers are two cloud VM instances in Singapore and Sydney representing proximate and distant cloud respectively featuring about 2 compute units CPU and 613 MB memory. The wireless access point is a broadband router connecting smartphone to the Internet via 2.4 GHz band of 802.11 wireless technology to access cloud VMs. Proximate cloud in Singapore is about 320km away from the experiment site in University of Malaya, Malaysia. Sydney cloud instance is about 6500 km away from the site.

2) Software: Two computation-intensive mobile math applications (considering noticeable increase in m-math and m-learning applications), namely prime (verifies if the given number is prime) and matrix multiplication (multiplies two matrices) are executed using three different execution strategies of native, proximate cloud, and distant cloud. Prefabricated services are utilized to develop the experiment model and applications by programming varied platform-independent services that perform computation-intensive tasks. These services can be executed in mobile and cloud resources independent from the platform and device (platform-independence is one of the major characteristic of SOC). To reduce complexity and overhead, no code from the mobile device is migrated to the cloud. An HTTP call to an already available service in the cloud is performed and data are transmitted for execution. Upon successful remote execution, the results are returned back to the device. Thus, the excess overhead of identifying, partitioning, and offloading code from mobile device to cloud is mitigated [5].

Using REST architectural style, server-side services are called with the least messaging overhead. REST is opted, because it generates small computation and communication overhead due to its remarkably small header size compared to the SOAP (Simple Object Access Protocol) architectural style [23]. The server component in the cloud is running a web server. The server component on the mobile device is executing on a lightweight open source mobile web service with less than 20 kb footprint. In local execution, client and server codes are stored in the device and executed locally. The mobile web server turns the mobile device to a tiny web server. Thus, the client component can send a POST request to the server component at localhost address and receive the result. To execute the PHP code on mobile device, a PHP plug-in is installed. To avoid connectivity issues and overhead during local execution, the server runs as localhost without IP address. Mobile device and clouds are immobile in this experiment.

Computationally equal VM instances (though VM heterogeneity can make minimal performance differences) in Singapore and Sydney are opted. The *tracert* command is used to identify the number of intermediate hops between test bed platform and the clouds which are 14 and 23 hops to the proximate and distant cloud respectively. The tracing delay is about 34ms and 277ms for proximate and distant VMs respectively. The average data volume transfer in prime application is 189 bytes which is small compared to 615KB of data transferred in matrix application. Amount of data transferred in both experiments are significantly different to demonstrate the impact of communication latency, especially when the distance between mobile and cloud is long.

This experiment is performed for 30 sample workloads selected based on three computation intensity levels of low, medium, and high which are summarized in Table I. Each prime workload in three intensity levels is chosen by adding about 10000 to the previous workload. For instance, second low workload is 210011 (first prime number greater than 210000). Matrix workloads are chosen using the series presented in Table I. For each matrix [n×m], elements' values are product of n and m. The execution process for each value is repeated 30 times to ensure consistency and reliability of acquired data and 99% confidence interval is calculated for

TABLE I
BENCHMARK WORKLOADS

Workload	Intensity	Workload Selection
Prime	Low	$200003 \leq x \leq 290011$, step≈10000
	Medium	$600011 \leq x \leq 690037$, step≈10000
	High	$900007 \leq x \leq 990001$, step≈10000
Matrix	Low	$[10\times m_i].[m_i\times 10], m_i=m_{i-1}+x, x=10, m_0=0, 1\leq i\leq 10$
	Medium	$[50\times m_j].[m_j\times 50], m_j=m_{j-1}+x, x=10, m_0=110, 1\leq j\leq 10$
	High	$[100\times m_t].[m_t\times 100], m_t=m_{t-1}+x, x=10, m_0=110, 1\leq t\leq 10$

TABLE II
DESCRIPTIVE STATISTICS OF EXECUTION TIME

	Workload	Execution	Min	Max	Mean
Execution Time	Prime	Local	628.2	1708.5	1168.7
		Prox. Cloud	162.2	386.1	283.3
		Distant Cloud	414	658.6	547.4
	Matrix	Local	128	21037.4	6956
		Prox. Cloud	83.6	14130.7	4124.5
		Distant Cloud	460.9	17958.8	9242

workloads. Three metrics, namely overall application execution time (ms), network latency (ms) –in cloud execution- or I/O latency (ms) -in local execution where both client and server components are running on the same device-, and total energy (mJ) consumed by each application are carefully monitored and measured for 900 iterations. The network or I/O latency in this study excludes computation latency of the servers. Execution times do not include user interaction delay for input and output.

Energy consumption of processor core, main memory, and communication chipset are monitored and other components such as LCD are discarded when collecting energy data. Energy consumed by other software components is not considered in data collection phase. PowerTutor [24] is used to monitor consumed energy on mobile device. To avoid man-made mistakes in collecting time and energy data, auto-logging tasks are performed for the entire experiment.

B. Results:

In this section, benchmarking results of executing both applications on the local device, proximate cloud, and distant cloud are presented and major findings in two parts of time and energy analysis are discussed.

Execution Time: Table II presents descriptive statistics (minimum, maximum, and mean value of 900 executions) of computation time for both applications. Each number in the Table represents the mean value of 900 executions. Descriptive statistics clearly depict the computation differences between prime and matrix applications. From the numbers in this Table, the benefits of CMA on prime and matrix applications are evident. Results of the prime application executions testify that augmenting resource-constraint mobile device leveraging proximate and distant cloud resources can respectively reduce the average application execution time by 75.8% and 53.1% compared to local execution. Whereas, benchmarking results of matrix application executions depict improvement in application execution time only when exploiting proximate outsourcing could reduce the average application execution time of matrix application as much as 40.7% compared to the local execution. In contrast to the proximate cloud, leveraging distant cloud for matrix application prolongs application execution time by 32.8% in average which indicates that using distant cloud not only could not save time but also increases the execution time. Indeed, such difference in results between prime and matrix application is due to the difference in the data transmission volume, which is sharply magnified when the number of hops between mobile and cloud increases.

Networking delays due to constraints of maximum transmission unit, and TCP congestion and receive windows are signified by growth in hop's number. Hence, number of hops between mobile and cloud is a crucial factor that noticeably impacts on the quality and performance of CMA systems, especially when the volume of data transmission is high and packet size increases.

Fig. 3 and Fig. 4 illustrate execution time of different workloads for prime and matrix applications in three execution strategies. Each of 30 workloads is repeated 30 times and each bar in the graphs represents the mean execution time of three workloads (i.e., mean of 90 values). The figures illustrate significant amendments in application execution time compared with the local execution time. In the synthesis of executing matrix application on proximate cloud, it can be observed that the improvement for the light workloads is much smaller than the complex workloads; execution time improvement increases when the computations intensity rises.

Although computing specifications of both cloud VMs are identical for proximate and distant clouds, the average application execution time of distant cloud is 93.3% and 124% more than proximate cloud for prime and matrix applications respectively. Such difference is due to the varied number of hops between mobile and cloud, data volume, and existing heterogeneity between processing and networking infrastructures (e.g., hardware and firmware performance of intermediate nodes, bandwidth, jitter, and VM performance).

Hence, the larger is the number of hops between mobile and cloud, the higher is the applications execution time. In order to better understand the impacts of distance and hop numbers on the execution time, linear interpolation analysis is performed. The results of analysis are illustrated in Fig. 5 and Fig. 6 for prime and matrix applications respectively. Interpolation line of local execution in Fig. 5 indicates sharp rise in prime execution time (low traffic) when the computing overhead of workloads increases, whereas in cloud execution environments, the time growth is lessen due to high performance cloud VMs. Indeed, it reflects the compound impacts of server's computing power, round-trip latency, and data transfer overhead of accessing the remote services on execution time. In contrast, combination of such overhead factors changes the tendency in Fig. 6 for local and cloud execution, which is due to significantly higher complexity and data transfer overhead. Moreover, further analysis of the interpolation lines, highlights that estimation and prediction of application execution time get complicated by the compound impacts on the execution time. Colored, varied-shape, markers depict execution time of low, medium, and high intensities workloads for both applications. Execution times of prime

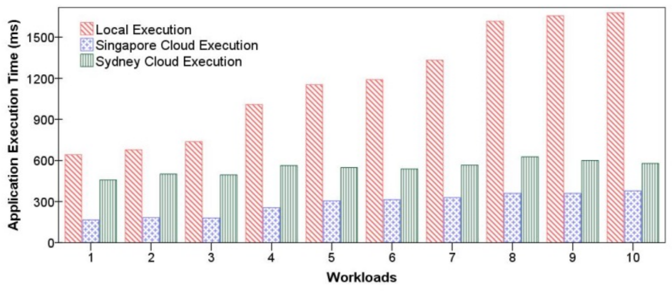

Fig. 3. Prime application execution time for three execution strategies. CMA significantly reduces application execution time. Cloud-based execution is beneficial in all cases.

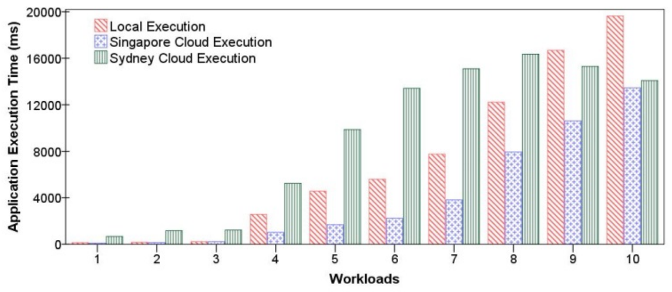

Fig. 4. Matrix application execution time for three execution strategies. CMA using distant cloud deteriorates application execution time in most of the cases. Sydney results illustrated high overhead of long distance that exceed the local execution time.

application are predictably increasing for each workload intensities due to very small round-trip latency (hops and data volume) between client and server components. Such constant rise facilitates estimation and prediction of execution time (using linear interpolation polynomial) for the whole population. Predictability intensifies in cloud execution strategy, especially in distant cloud when computing complexity, data transfer, client-server distance, and number of hops increase.

The observed anomaly in execution time of matrix application (see Fig. 6) in distant cloud execution strategy encouraged repetition of data collection process for several times to ensure efficiency and accuracy of data collection and analysis methods. However, no constant change was observed during repetition which assures that predicting execution time of data-intensive compute-intensive mobile applications on distant cloud is not easily feasible. Large number of factors, particularly the momentary bandwidth fluctuations, variable jitter, climate changes, mobile resource levels, and instantaneous cloud performances are affecting the overall execution time of applications, which magnified when data volume and mobile-cloud hops increase in the presence of congestion control. Convergence of results from mathematical analysis of overall execution time and benchmarking process advocates that in CMA systems, number of mobile-cloud hops beside several other factors such as computation intensity, mobile-cloud data transmission volume, network bandwidth and congestion, slow-start delay, and monitoring overhead of augmentation impact on the execution time of mobile applications and intensify estimation for the entire population.

Error! Reference source not found. depicts execution times of proximate and distant modes where Hop number is distinguishing metric. It helps to

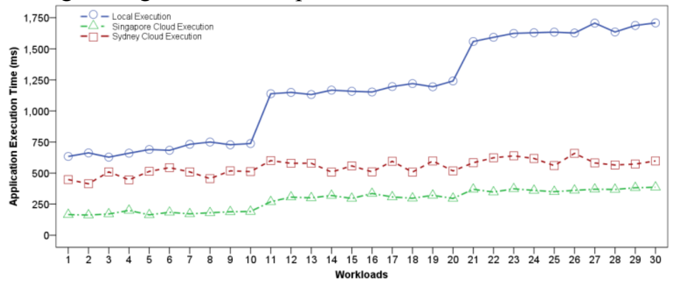

Fig. 5. Prime application scattered plots with interpolation lines for application execution time. Predictability is feasible due to constant changes in application execution time.

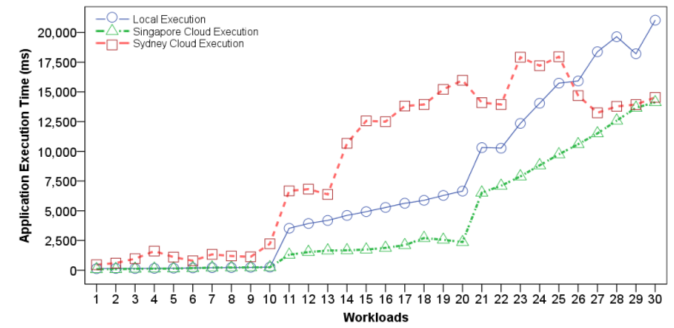

Fig. 6. Matrix application scattered plots with interpolation lines for application execution time. Predictability is less feasible due to bursty changes in application execution time.

conclude the impacts of hops on the application performance. Each bar represents the mean execution time of ten workloads.

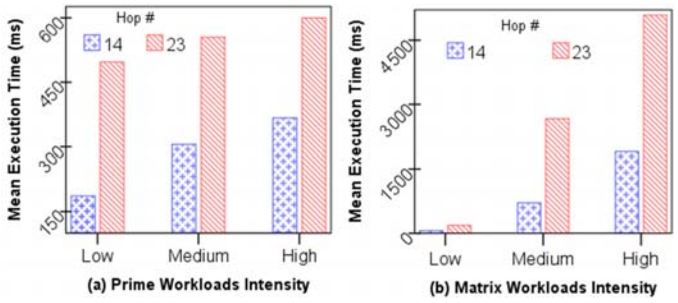

Fig. 7. Impact of hop numbers on execution time of matrix and prime.

Mobile Energy Consumption: Descriptive statistics of mobile energy consumption are summarized in Table III. Results of prime application consumed energy advocate that augmenting resource-constraint mobile device leveraging proximate cloud resources can reduce the energy requirement of the mobile application 76.6% in average, whereas it is about 56.6% when using distant cloud. The difference in energy saving between proximate and distant cloud is significant considering the low data transfer. Descriptive statistics for matrix application depict that exploiting proximate cloud significantly reduces the average energy consumption up to 40.6% for data-intensive mobile applications, whereas using distant cloud increases energy consumption in average by 60%. Hence, exploiting distant clouds for augmenting matrix application increases the energy consumption of the mobile application. Fig. 8 and Fig. 9 illustrate consumed energy of varied workloads for prime and matrix applications in three execution strategies. Each 30 workloads is repeated 30 times and each bar in the graphs represents the mean energy consumption of three workloads (i.e., mean of 90 values). Energy consumption results of prime application indicate significant energy conservation, especially when the workload intensity increases. For matrix application, as Fig. 9 depicts, leveraging proximate cloud resources is beneficial in saving energy, whereas augmenting mobile device via distant cloud results in adverse and leads to more energy consumption compared to the local execution. Energy in utilizing distant cloud is saved when the computation extensively increases and could surpass the communication overhead leading to energy saving. Thus, leveraging distant cloud for augmenting data-intensive mobile applications can originate excess energy overhead.

TABLE III
DESCRIPTIVE STATISTICS OF MOBILE ENERGY CONSUMPTION

	Workload	Execution	Min	Max	Mean
Energy	Prime	Local	215.1	871.1	559.6
		Prox. Cloud	75.8	189.4	131
		Distant Cloud	169	302.1	244.3
	Matrix	Local	73	13485.1	4355.3
		Prox. Cloud	68.4	9828.9	2587.8
		Distant Cloud	273	12797.1	6980.4

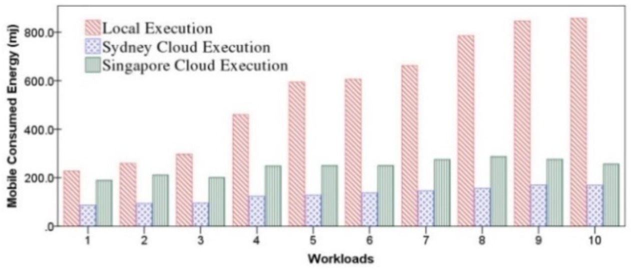

Fig. 8. Prime application energy consumption for three execution strategies. Cloud-based execution is beneficial in all cases.

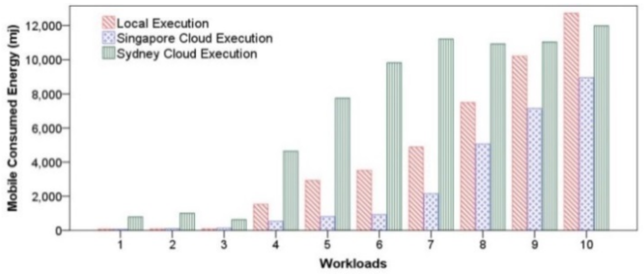

Fig. 9. Matrix application energy consumption for three execution strategies. Cloud-based execution is not beneficial in most cases.

Similar to the analysis performed in application execution time, linear interpolation analysis is undertaken for energy (see Fig. 10 and Fig. 11) to investigate predictability for other workloads. Synthesizing the Fig. 10 and Fig. 11 unveils that energy consumption of application is more predictable in local execution compared to the cloud execution due to lack of round-trip latency. For cloud execution, fluctuations for both applications are more significant when using distant cloud compared with proximate resources, which complicates accurate prediction of energy consumption of other workloads

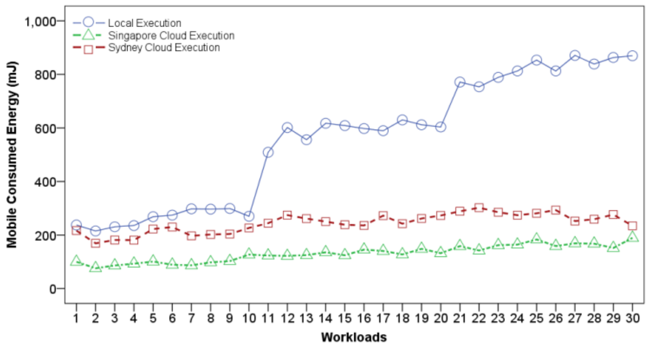

Fig. 10. Prime application scattered plots with interpolation lines for application consumed energy. Predictability is feasible in local and proximate cloud execution due to constant changes in application consumed energy. Fluctuations are high in distant cloud strategy.

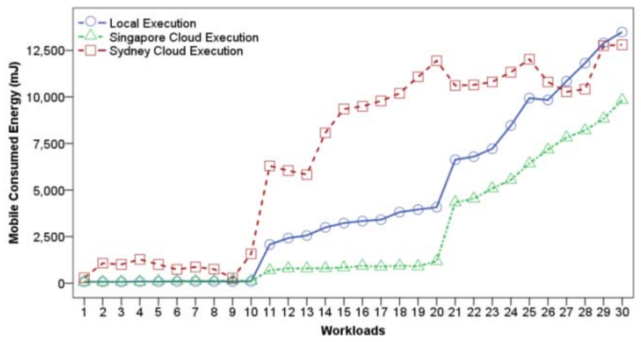

Fig. 11. Matrix scattered plots with interpolation lines for consumed energy. Predictability is less feasible in distant cloud strategy due to bursty changes in application energy consumption.

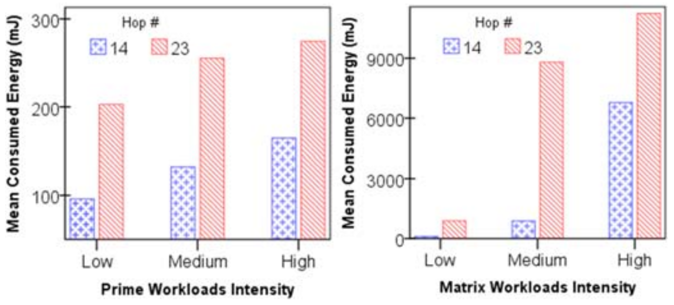

Fig. 12. Impact of hop numbers on Energy usage of matrix and prime

Therefore, considering the identical computing energy of cloud VMs and equal size of data transfer, number of mobile-cloud hops is the major factor that impacts on the energy dissipation of mobile augmentation.

Fig. 12 shows energy consumed in proximate and distant modes and helps to conclude impacts of hops on application performance. Each bar is the mean energy of ten workloads.

V. CONCLUSIONS AND FUTURE WORKS

The ultimate MCC augmentation goal is to realize the vision of unrestricted functionality, storage, and mobility regardless of underlying constraints. In this paper, impacts of exploiting geographically distributed cloud resources on performance of RMAs running on resource-constraint smartphone are mathematically and experimentally analyzed. Mathematical analysis of overall execution time and energy consumption unveils that number of intermediary hops and data transmission volume have significant impacts on RMA's performance. Findings are validated using series of experiments in real environment. Benchmarking results advocate that exploiting distant cloud to augment computational capabilities of resource-constraint mobile devices is a time- and energy-intensive approach that degrades the performance of augmented mobile applications in MCC. Also, increase in distance, significantly degrades predictability of RMAs time and energy. Augmenting mobile devices leveraging distant clouds originates an anomaly and encumbers accurate time and energy estimation of RMAs. Such impact is increased when data communication between mobile and cloud raises. Therefore, reducing mobile-cloud distance, especially for data-intensive RMAs significantly enhances the performance on resource-constraint devices.

Future works will investigate implications of exploiting multiple heterogeneous clouds in different locations for RMAs execution. The major issue is to develop an optimal resource scheduler that dynamically allocate appropriate resources based on the user preferences (e.g., cost and trust), application requirements (e.g., CPU and RAM), and QoS (e.g., energy).

References

[1] J. Winkley, P. Jiang, and W. Jiang, "Verity: an ambient assisted living platform," IEEE Trans. Consum. Electron., vol. 58, no. 2, pp. 364–373, May 2012.

[2] S. Abolfazli, Z. Sanaei, Gani Abdullah, F. Xia, and L. T. YANG, "Rich Mobile Application: Genesis, Taxonomy, and Open Issues," J. Netw. Comput. Appl., in Press, 2013.

[3] "BCC Research Global Markets for Smartphones and PDAs (IFT068A)" May 2009, p. 7215, 2014.

[4] S. Abolfazli, Z. Sanaei, and A. Gani, "Mobile Cloud Computing: A Review on Smartphone Augmentation Approaches," in Proc. WSEAS International Conference on Computing, Information Systems, and Communications, Singapore, Feb 2012.

[5] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and B. Rajkumar, "Cloud-based Augmentation for Mobile devices: Motivation, Taxonomies, and Open Challenges," IEEE Commun. Surv. Tut, in Press, 2013.

[6] Z. Sanaei, S. Abolfazli, A. Gani, and R. H. Khokhar, "Tripod of Requirements in Horizontal Heterogeneous Mobile Cloud Computing," in Proc. WSEAS International Conference on Computing, Information Systems, and Communications, Singapore, Feb 2012.

[7] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges," IEEE Commun. Surv. Tut, In Press, 2013.

[8] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in Proc. ACM International Conference on Mobile Systems, Applications, and Services, 2010, pp. 49–62.

[9] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 2010, pp. 4–11.

[10] K. Kumar and Y. H. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," IEEE Comput., vol. 43, no. 4, pp. 51–56, 2010.

[11] Z. Sanaei, S. Abolfazli, T. Khodadadi, and F. Xia, "Hybrid Pervasive Mobile Cloud Computing," Information-an Int. Interdiscip. J., 2013.

[12] R. T. Fielding, "Architectural styles and the design of network-based software architectures," University of California, 2000.

[13] M. Bichier and K. Lin, Service-oriented computing. 2006.

[14] M. Shiraz, S. Abolfazli, Z. Sanaei, and A. Gani, "A study on virtual machine deployment for application outsourcing in mobile cloud computing," J. Supercomput., vol. 63, no. 3, pp. 946–964, Dec. 2012.

[15] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer, "Survey on Energy Consumption Entities on the Smartphone Platform," in Proc. IEEE International Conference Vehicular Technology Society - Spring, Budapest, Hungary, 2011, pp. 1–6.

[16] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," IEEE Pervasive Comput., vol. 8, no. 4, pp. 14–23, 2009.

[17] R. and W. N. and T. W. Ramaswamy, "Characterizing network processing delay," in Proc. IEEE International Conference on Global Communications, 2004, pp. 1629–1634.

[18] N. Cordeschi, M. Shojafar, and E. Baccarelli, "Energy-saving self-configuring networked data centers," Comput. Networks, vol. 57, no. 17, pp. 3479–3491, 2013.

[19] S. Abolfazli, Z. Sanaei, M. Shiraz, and A. Gani, "MOMCC: Market-oriented architecture for Mobile Cloud Computing based on Service Oriented Architecture," in Proc. IEEE International Workshop on Mobile Cloud Computing, Beijing, China, 2012, pp. 8–13.

[20] Z. Sanaei, S. Abolfazli, M. Shiraz, and A. Gani, "SAMI: Service-Based Arbitrated Multi-Tier Infrastructure Model for Mobile Cloud Computing," in Proc. IEEE International Workshop on Mobile Cloud Computing, Beijing, China, 2012, pp. 14–19..

[21] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," IEEE Pers. Commun., vol. 8, no. August, pp. 10–17, 2001.

[22] N. Tolia, D. G. Andersen, and M. Satyanarayanan, "Quantifying interactive user experience on thin clients," IEEE Comput., vol. 39, no. 3, pp. 46–52, 2006.

[23] J. H. Christensen, "Using RESTful web-services and cloud computing to create next generation mobile applications," in Proc. ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications, Orlando, Florida, 2009, pp. 627–634

[24] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in Proc. ACM International Conference on Hardware/software codesign and system synthesis, 2010, p. 105.

Biographies

S. Abolfazli (S'12) is a Ph.D. candidate and senior research assistant in faculty of CSIT, University of Malaya, Malaysia. He received M.Sc. (Information Sys) and BE (Software Eng.) from India and Iran respectively. He worked in various IT companies for ten years. His main research interests are mobile cloud computing, cloud-based mobile augmentation, M. big data and SOA.

Z. Sanaei (S'12) is a Ph.D. candidate and senior research assistant in the faculty of CSIT, University of Malaya, Malaysia. She received her M.Sc. (Information Sys.) and BE (Software Eng.) from India and Iran respectively. Her main research interests include mobile cloud computing mobile big data, pervasive computing, and resource scheduling in high performance computing.

M. Alizade (S'12) is a Ph.D. candidate in Malaysia-Japan Int'l Institute of Tech. in the Comm. systems and Net research group, Universiti Teknologi Malaysia (UTM). He received M.Sc. in Info Sec from UTM and BE in Electrical Electronic Engineering in 2007 from Iran. His research interest is in Cryptography, Network Security, and Mobile Cloud Computing

A. Gani (M'00-SM'12) is an Associate Professor at the faculty of CSIT, University of Malaya, Malaysia. He received his PhD from the University of Sheffield. He has published more than 100 papers in conferences and respectable journals. His interest of research includes self-organized system, reinforcement learning, and wireless-related networks.

F. Xia (M'07-SM'12) is an Associate Professor and PhD Supervisor in School of Software, Dalian University of Technology, China. He published one book and over 110 scientific papers. His research interests include mobile and social computing, network science, and cyber-physical systems. He is a member of ACM and ACM SIGMobile.