RMCC: A RESTful Mobile Cloud Computing Framework for Exploiting Adjacent Service-based Mobile Cloudlets

Saeid Abolfazli*, Zohreh Sanaei*, Abdullah Gani*, Feng Xia†, and Wei-Ming Lin‡
*Center for Mobile Cloud Computing Research,University of Malaya, Malaysia
†School of Software, Dalian University of Technology, China
‡Dept. Electrical and Computer Engineering, University of Texas at San Antonio, USA
Email: abolfazli, sanaei, abdullahgani, f.xia@ieee.org, weiming.lin@utsa.edu

***Abstract*—Mobile devices, especially smartphones are increasingly gaining ground in several domains, particularly healthcare, tele-monitoring, and education to perform Resource-intensive Mobile Applications (RiMA). However, constrained resources, especially CPU and battery hinder their successful adoption. Mobile Cloud Computing (MCC) aims to augment computational capabilities of resource-constraint mobile devices and conserve their native resources by remotely performing intensive tasks. In typical MCC solutions, intensive tasks are offloaded to distant VM-based cloud datacenters or cloudlets whose exploitation originates long WAN latency and/or virtualization overhead degrading RiMA execution efficiency. In this paper, a lightweight Resource-oriented MCC (RMCC) architecture is proposed that exploits resources of plethora of Adjacent Service-based Mobile Cloudlets (ASMobiC) as fine-grained mobile service providers. In RMCC, ASMobiCs host prefabricated RESTful services to be asynchronously called by mobile service consumers at runtime. RMCC is a RESTful cross-platform architecture functional on major mobile OSs (e.g., Android and iOS) and realizes utilization of the computing resources of off-the-shelve outdated or damaged-yet-functioning mobile devices towards green MCC. Results of benchmarking advocate significant mean time- and energy-saving of 87% and 71.45%, respectively when intensive tasks are executed in ASMobiCs.**

***Index Terms*—Mobile Cloud Computing, Cloud-based Mobile Augmentation, Computing Offloading, Distributed and Parallel Mobile Computing**

I. Introduction

Mobile devices are increasingly gaining ground in several domains specially healthcare, education, and gaming [1], [2]. Mobile users are insatiably demanding to perform RiMAs on the go with identical experience of desktop computers. However, technological and commercial restraints and intrinsic limitations of mobile devices encumber deployment of powerful long-lasting resources, especially battery. Hence, execution of RiMAs is either impossible or quickly drains the native resources leading to user experience degradation [3]. As a remedy, MCC is rapidly gaining ground aiming to augment mobile devices by increasing, enhancing, and optimizing their computing capabilities while performing compute-intensive components in the cloud-based resources [3].

In typical augmentation approaches [4], [5], VM-based distant immobile clouds are leveraged as remote resources. However, virtualization overhead, long WAN latency due to manifold intermediate hops, and data transmission overhead to distant resources originate noticeable time and energy costs leading to RiMA execution deficiency [6], [7]. Cloudlet [8] leverages VM-based proximate desktops to reduce latency by performing intensive computations in vicinity. it creates a partial VM of mobile run-time environment (VM of all mobile OSs assumably exist in cloudlet), compress it, and transfer it to the cloudlet, decompress it and run it in the cloudlet to perform intensive computations. However, creation, de/compression, and transmission of VM beside virtualization overhead inside cloudlet degrade augmentation yields. Hyrax [9] as another augmentation solution utilizes nearby smartphones to reduce network latency for big data analysis. Hyrax is different from our work since it is designed to perform distributed big data processing rather than compute-intensive processing. However, issues such as platform-dependency (works only for Android), mobility dismissal, and Hadoop overhead for scheduling tightly coupled RiMAs encumber Hyrax adoption. VMCC [10] augments mobile devices using an ad-hoc cluster of smartphones. Augmentation overhead in VMCC is high due to partitioning and transmission costs beside continuous tracing of mobile devices to establish a peer-to-peer network beside necessity to modify applications' source codes.

In this paper, we aim to achieve energy-time efficient execution of RiMA using a lightweight RESTful Service-oriented MCC (RMCC) framework that builds a network of ASMobiCs under supervision of the Mobile Network Operators (MNOs) such as AT&T. Using RMCC, a resource-poor mobile device asynchronously initiates remote execution of intensive services in ASMobiC. We use ASMobiC as a particular type of low overhead Mobile Cloudlet (MobiCloudlet in brief) located in one-hop proximity of mobile users featuring virtualization-free (runs web services) architecture. MobiCloudlet is a wirelessly-connected battery-operating mobile device, such as smartphone, tablet, and car mounted computer, performing computation on behalf of resource-poor Mobile Service Consumer (MSC). MobiCloudlets belong to individual/corporate owners sharing computing resources for a credit (e.g., money or reputation). RMCC advances the state-of-the-art augmentation efforts and compliments related works as a VM-free framework based on service-oriented architecture that enables

remote execution of prefabricated RESTful web services in ASMobiCs without need to transfer full/partial VM of mobile device to the ASMobiCs (unlike cloudlet). RMCC is also an extrapolation of Hyrax and VMCC to compute-intensive mobile applications that omits augmentation management overheads by employing loosely-coupled web services that can be called for remote execution without partitioning and code transmission overheads. We surrender as much managerial tasks as possible to a centralized entity named Trusted Service Governor (TSG) to mitigate client-side management overhead. TSG also opens opportunities to enhance security and privacy of users and ASMobiC owners. To the best of our knowledge, RMCC is the first RESTful service-oriented solution that offers platform-independent RiMA execution in MCC with time-energy saving. MNOs play key role in RMCC security and mobility management considering their reputation-trust among mobile users and also ability to bridge nomadic mobile users with ASMobiCs through cellular communication.

The remainder of the paper is as follows. Section II describes proposed framework followed by its performance evaluation and validation in Section III. Results are discussed in Section IV and paper is concluded in Section V.

II. RMCC Framework

RMCC's aim is to leverage computing capabilities of ASMobiCs for augmenting resource-constraint mobile devices.

A. Key Features

• **Service-Oriented Architecture (SOA):** RMCC inherits cross-platform and lightweight characteristics from SOA featuring loosely coupling of services that significantly mitigates complexity and overhead, enhances elasticity of mobile applications, and realizes platform-independence. Moreover, Representational State Transfer (REST) [11] mitigates communication cost compared to SOAP (Simple Object Access Protocol) used in majority of augmentation approaches.

• **Separation of Responsibilities:** In traditional service-based systems, service providers have both roles of service development and provisioning. However, performing these tasks demands skills that majority of ASMobiC owners lack. To mitigate the problem and enhance RMCC adaptability, feasibility, and complexity, we separate the responsibilities and introduce 'service developer' and 'mobile service provider' roles. The former is responsible to build, describe, and maintain the service, while the latter hosts and executes the service. Thus, RMCC can easily be used by individual owners without IT skills prerequisite. Developers build services and negotiate with TSG to describe and publish them.

• **Asynchronous Communications:** Employing asynchronous communications using Ajax enables light background mobile-cloud communications so that users can continue interacting with their devices when remote execution is taking place on ASMobiCs enhancing user experience and improving RMCC adoption.

• **ASMobiCs:** *ASMobiC* aims to alleviate the latency and heterogeneity overhead in mobile augmentation. The key differences between ASMobiC and other cloud-based resources are summarized in Table I. One key difference is heterogeneity; heterogeneity between mobile devices and cloud/cloudlets is horizontal due to fundamental differences in architectures (e.g., ARM vs x86) and OSs (e.g., Android vs Windows). However, heterogeneity between mobile devices and MobiCloudlets/ASMobiCs is vertical due to architectural similarity (mobiles are dominantly ARM-based) [12]. Handling and addressing vertical heterogeneity is relatively less complex than horizontal. Utilizing ASMobiCs as less heterogeneous remote resources that feature one-hop communication latency with high multiplicity enhances augmentation performance.

TABLE I
CLOUD, CLOUDLET, MOBICLOUDLET, ASMOBIC: KEY DIFFERENCES

Characteristic	Cloud	Cloudlet	MobiCloudlet	ASMobiC
Mobility	NA		Yes	
Elastic Scalability	High	Medium	Low	
Locality	Distant	Proximate		Adjacent
Multiplicity	Low	Medium	High	
LAN Latency	High	Medium	Low	
Operation Time	24/7	Limited to business hours	24/7	
Connectivity	Wired		Wireless	
Execution Platform	VM (Virtual Machine)		VM/HTTP over Physical	HTTP over Physical
Operating Platform	Server, Desktop	Desktop	Mobile	
Context-awareness	Low		High	
Heterogeneity	Horizontal (High)		Vertical (Low)	

B. Building Blocks

Main blocks presented in Figure 1 and explained as follows.

1) Mobile Service Provider (MSP): is predominately the ASMobiC though giant clouds are also provisioned for highly resource-intensive services whose executions need rich computing resources. The terms MSP and ASMobiCs are used interchangeably with the same meaning. ASMobiCs need to register with the central supervisory entity (i.e., TSG) and upon approval install natively the MSP components. The ASMobiCs and TSG negotiate to identify the suitable services for execution on ASMobiCs. Upon conclusion, the service code is installed in ASMobiCs for future execution requests.

'Asynchronous service execution handler' listens to incoming requests and transmits the response to the destination via push mechanism. The MSC's request along with context information are sent to 'execution manager' to initiate and monitor service execution and once the results are ready, the response is forwarded to MSC. QoS metrics like reliability and availability of services are monitored by 'service profiler' for ranking purpose (detailed description is omitted for brevity).

Context information, including location collected by ASMobiCs are used by 'context management' to tailor services according location and environment changes to enhance QoS and QoE. Using the location, the MNO identifies mobile owner's geographical location. This component performance is vital in identifying the closest ASMobiC to the MSC to tackle communication latency.

2) Mobile Service Consumer (MSC): are mobile devices which consume ASMobiCs services at runtime. Prior execution, MSC user decides to whether execute locally or remotely via RMCC. In finer level of control, users can enable remote execution of individual intensive services. 'Asynchronous aug-

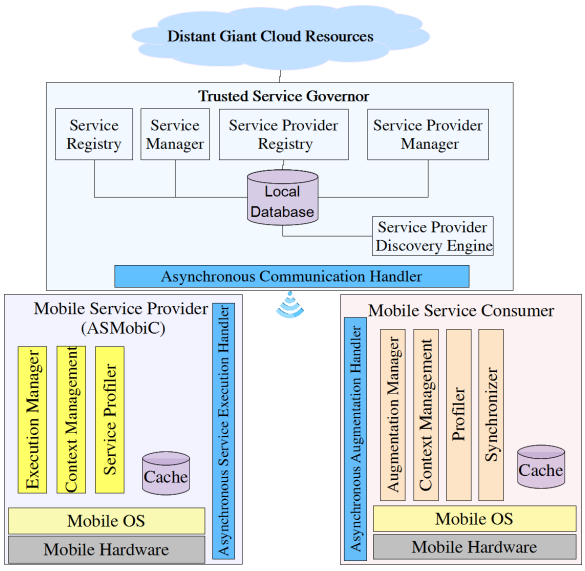


Fig. 1. The Schematic Representation of The RMCC Framework

mentation handler' in MSC asynchronously initiates remote connection with TSG or ASMobiCs.

'Augmentation manager' ensures seamless RiMA execution over MSPs via managing and monitoring the entire augmentation procedure. The RiMA execution starts from MSC and augmentation is initiated when the local execution reaches a resource-intensive service(s) that is/are not executable on the mobile device (or its local execution is not preferred). So, 'augmentation manager' fetches context information from 'context management' and initiates an asynchronous call to the TSG looking for an appropriate ASMobiCs. Considering the user preferences, input data for service execution can either be sent at discovery call or later when calling the MSP. In the former, the TSG forwards the request along with the input data to the nominated ASMobiCs for execution. Upon successful execution, the results can be either returned to the TSG for security verification, if the security is preferred, or directly forwarded to the MSC (required IP is sent by the TSG). In the latter case, upon successful MSP discovery, the TSG pushes back the binding information of all the corresponding nodes (when more than one service are searched) to the MSC. The retrieved information is used by 'augmentation manager' to contact the MSP(s). So, the services are called for execution and results are re-integrated to the local RiMA. The latter case is deployed for our evaluation purpose.

Moreover, 'augmentation manager' monitors the communication link of the MSC to maintain/reestablish the connection if the MSC loses the communication. In case of disconnection, results are cached to be used when connection restored. Thus, energy efficiency and responsiveness is improved and system robustness is enhanced. Throughout augmentation, data integrity is maintained via 'synchronizer'. During ASMobiC execution, synchronization runs in background without distracting user. Throughout augmentation, context data including nodes' location is continuously monitored by 'context management' to be used by 'augmentation manager' to more accurately identify the ASMobiC. 'Profiler' logs QoS metrics, like execution time and MSP's reliability upon every successful/failed execution. To mitigate WAN latency, dual caching is used in MSC and MSP to cache data on disconnection.

3) Trusted Service Governor (TSG): is a trusted supervisory entity replicated on multiple servers to supervise and monitor augmentation entities, including MSP, MSC, and service developers. TSG is the main governing entity with several crucial responsibilities. To avoid overcrowded servers and mitigate the WAN latency, MNOs are identified as TSG and can be replicated on MNOs in varied geographical areas to effectively balance load and seamlessly serve mobile subscribers. MNOs have been serving mobile end-users from the beginning and could establish reputation and historical trust [13]. MNOs scattered in urban/rural areas near to mobile nodes to significantly reduce WAN latency. MNOs have started providing cloud services to their clients and, hence, will be able to scale and adapt to the highly oscillating computation and storage of end-users. Most importantly, MNO provides a private Intranet between MSC, TSG, and MSP to perform augmentation without entering the risky channel of Internet.

'Service Registry' acts as a public repository to maintain a local database and store description of registered services. At service registry, developer negotiates with TSG, provides service information, and uploads the service package including core and dependent libraries to the TSG. 'Service manager' analyzes historical QoS data for periodic efficiency assessment to rank services based on functionality and performance. Potential ASMobiCs communicate with the 'service registry' to select the most appropriate services whose required resources are available in the mobile device. This component validates the hosting demands and refuses inappropriate allocation requests (e.g., RAM-intensive service on RAM-poor ASMobiC). Upon successful registry, a unique URI is assigned to services for each ASMobiC. 'Service provider manager' analyzes and synthesizes historical QoS data of ASMobiCs and rank them based on observed performance, availability, and reliability. These ranks are considered when 'service provider discovery engine' searches for MSPs. At selection time, the TSG chooses the MSP with highest rank. Communications are asynchronously performed via 'asynchronous communication handler'. 'Service Provider Discovery Engine (SPDE)' determines the most appropriate ASMobiC capable of performing MSC's requested service(s) with desired preferences. Initially, MSC forwards service name(s) and preferences along with the request to SPDE to acquire address of the best ASMobiC. Considering the user location and current location of MSPs, SPDE identifies the most appropriate MSP that can efficiently meet MSC requirements and preferences. Service discovery overhead is imposed on TSG to achieve higher efficiency.

C. Significance

The novelty of RMCC is manifold. Firstly, it is a cross-platform usable in varied computing devices capable of performing prefabricated utility services. Secondly, lack of run-

time code offloading beside low-hop communication remarkably reduce WLAN latency. Thirdly, no Internet is required considering the MNOs' role to utilize the augmentation benefits. Fourthly, it defines a new role in SOA-based systems called MSP; ASMobiCs can turn into a MSP using a low footprint application. This property of RMCC remarkably contributes to reusing the off-the-shelves and damaged-yet-functioning (damaged speaker, touch-pad, or screen) or old mobile devices. Lastly, it is widely elastic and scalable. Though RMCC is designed and evaluated using mobile devices as MSC and MSP, any wirelessly accessible computing device, such as cloud data centers, desktop PCs, and car mounted computers can be used as MSPs. Considering RMCC features, it can be used in several domains, particularly education, healthcare, entertainment and gaming, vehicular networks, crowdsourcing, and remote monitoring. It also provisions node mobility under coverage of the MNO.

III. Evaluation

RMCC is implemented using jQuery and evaluated via series of benchmarking experiments using three heterogeneous android-based smartphones and a desktop. We devise and validate mathematical models to validate the evaluation results.

A. System-Level Metrics

Application execution time in millisecond (ms) and consumed energy in millijoule (mJ) collected using PowerTutor 1.4 are selected as performance evaluation metrics. Energy consumption of components like LCD are disregarded.

B. Benchmarking Model

A RiMA including three different resource-intensive services (prime, matrix multiply, and matrix covariance) are executed in two execution modes (local and ASMobiC). To prepare the experimentation environment, we use a resource-rich desktop computer with 3.3 GHz Intel CPU, 4GB RAM, and 32-bit Windows 7 accessible via wireless connection to run TSG components. The MSC is an HTC Nexus One that hosts and runs the RiMA. Our MSC features a 32-bit Qualcomm Snapdragon S1 QSD8250 chipset with 1 GHz CPU. A Samsung Galaxy S2, HTC One X smartphone with Android OSs, and a DELL Laptop XPS14z with Intel 2.5 GHz CPU, 4GB RAM, and 64-bit Windows 7 are utilized as ASMobiCs. We turn off the MSC's cellular radio, unplug its USB cable, set display brightness to 50%, keep battery level between 60 to 70%, and closed other applications to avoid interruption at data collection stage. To avoid heat impacts in device, we stop data collection on regular intervals. The wireless network was isolated to avoid inference and fluctuations caused by other wireless network consumers. In local execution mode, the Wi-Fi is also off. We evaluate QoS of each RiMA for 30 workloads selected in three intensity levels of low, medium, and high (see Table II). To enhance data collection reliability and mitigate the wireless intermittency impact, each reported data is mean value of 30 executions with 99% confidence interval.

TABLE II
Evaluation Workloads

Workload	Intensity	Workload Selection
Prime	Low	$220063 \leq x \leq 330017, step \approx 10000$
	Medium	$600011 \leq x \leq 690037, step \approx 10000$
	High	$900007 \leq x \leq 990001, step \approx 10000$
Matrix Product	Low	$[30\times m_i].[m_i\times 60], m_i=m_{i-1}+x$, x=10,$m_0$=50,$1 \leq i \leq 10$
	Medium	$[65\times m_j].[m_j\times 120], m_j=m_{j-1}+x$,x=10,$m_0$=85,$1 \leq j \leq 10$
	High	$[130\times m_t].[m_t\times 180], m_t=m_{t-1}+x$, x=10,$m_0$=50,$1 \leq t \leq 10$
Matrix Covariance	Low	$[45 \times 45]$ to $[65 \times 65], step \approx 2$
	Medium	$[72 \times 72]$ to $[90 \times 90], step \approx 2$
	High	$[100 \times 100]$ to $[145 \times 145], step \approx 5$

C. Mathematical Model

Several mathematical and statistical analyses are carried out to validate evaluation results. To devise mathematical models of time and energy, linear regression is used as the predominant observation-based prediction approach [14]. Independent replication technique is used to produce a dataset of workload, time, and energy related to independently selected workloads (no correlation between mathematical and benchmarking workloads). Partial dataset is used to train the regression models to derive time and energy models by identifying the correlations between the workloads and execution time as well as execution time and consumed energy. Split-sample approach and calibration-validation exercise are performed to validate the devised models. For models validation, dataset is randomly split into two different size datasets and correlations between the dependent and independent variables is identified using statistical tests. Moreover, for all models we performed graphical residual analysis to ensure the validity of the models. However, illustrations are omitted due to space limit.

1) Execution Time (ms): Complexity of RiMA calculated using Big-oh $O(f(n))$ notation for building linear regression model of time reported below.

a) Local Execution: The complexities of prime, multiply and covariance are $O(p), O(m^3), O(c^3)$, respectively that are used as independent variables for the regression model. For the sake of clarity and accuracy and to avoid complex calculations, we consider $m_i = T_i \times S_i \times K_i$ and $C_i = N_i \times N_i \times N_i$. Thus, regression model of $TL_{RMA(i)}$ as the total time for executing the i^{th} workloads of p, m, and c is found to be: $TL_{RMA(i)} = (0.007 \times p_i + 34.512) + (0.016 \times m_i - 43.560) + (0.017 \times c_i - 1.193)$. The regression model summary is presented in Table III. Insignificantly different adjusted R^2 in varied samples shown in Table IV advocates that data are well fitted into our model as an evidence of model validation. Degree of Freedom (DF) in figures indicates number of training sets used in building regression models and is dissimilar in different models due to random size of training set.

b) ASMobiC Execution: ASMobiC execution mode is when the local execution of the RiMA reaches intensive services to be called for remote execution. The application calls the service governor, forwards the service names, and receives the binding information of designated ASMobiCs. The binding information is used to asynchronously call execution of services in ASMobiC. Upon successful service execution, the results are sent back and integrated with the local code. In

TABLE III
REGRESSION MODEL SUMMARIES OF LOCAL/ASMOBIC TIME AND ENERGY

Model	Regressor	R^2	Adjuster R^2	Sigma
Local Time	Prime	0.99	0.99	0
	Matrix Multiply	1	1	0
	Matrix Covariance	1	1	0
ASMobiC Time	Computing Time	0.99	0.99	0
	Communication Delay	0.99	0.99	0
Local Energy	Execution Time	1	1	0
ASMobiC Energy	CPU	0.92	0.92	0
	Wi-Fi	0.94	0.94	0

this mode, the entire intensive executions take place outside the MSC, and consequently noticeable amount of mobile battery is conserved. $TR_{RMA(i)}$ is the time that i^{th} workload takes for execution and is: $TR_{RMA(i)} = T_{Dis} + TW_{Remote(RMA(i))} + T_{Com(RMA(i))}$ where T_{Dis} is the communication and computation time that the TSG takes to identify ASMobiCs, regardless of the workloads. $TW_{Remote(RMA(i))}$ is the computing time to perform execution of the heaviest service in ASMobiCs and $T_{Com(RMA(i))}$ is the communication delay. Thus, we have $TR_{RMA(i)} = (0.0000425 \times m_i) + (1.3 \times T_{Dis(i)}) + (0.98 \times TC_{Multiply(m_i)}) + (15.785 - 47.28)$. The model summary is reported in Table III.

We validate the model using split-sample method whose results are commonly presented in Table IV. Since the difference between adjusted R^2 is less than 0.05, the model is valid.

2) Consumed Energy: The mathematical model of energy in local and ASMobiC modes are as follows.

a) Local Execution: Energy consumption of MSC comprises of the total energy used by the CPU, LCD, and Wi-Fi (if any). However, LCD energy is not profiled since it has no impact on processing power of MSC. Also, Wi-Fi energy is not profiled in local mode. Hence, the only power consuming component is CPU. If $EL_{RMA(i)}$ is the total energy consumed for the local execution of the i^{th} workload, we have $EL_{RMA(i)} = EL(CPU_i)$ where $EL(CPU_i)$ is the energy consumed to locally execute the entire i^{th} workload. CPU power consumption for each computational component highly depends on the execution time of that particular component, and execution time itself is a direct function of the workloads. So, the more intense is the workload, the higher would be the execution time, and the more will be CPU power consumption. Prediction model of CPU energy is formulated via linear regression which is summarized in Table III. The model is $EL(CPU_i) = (0.281 \times TL_{RMA(i)}) + 119.587$

Validation results of energy model in local mode are summarized in Table IV. Identical adjusted R^2 values in varied samples evidently validate the model.

b) ASMobiC Execution: Similar to the local execution, we disregard energy consumption of the LCD for remote execution. However, in ASMobiC execution CPU and Wi-Fi are two major energy consumers which will be considered for devising the energy model. In remote execution mode, the resource discovery service is consuming some amount of energy. Hence, if $ER_{RMA(i)}$ is the total energy for ASMobiC execution of the i^{th} workload, therefore $ER(RMA_{(i)}) = ER(Dis_{(i)}) + ER(TW_{Remote(i)}) + ER(WiFi_{(i)})$ where $ER(Dis_{(i)})$ is the energy to perform MSP discovery which includes energy consumed while waiting for the MSP information from TSG. $ER(TW_{Remote(i)})$ is the energy consumed while device is waiting for the results to come from the ASMobiC. $ER(WiFi_{(i)})$ is the energy consumed to communicate with external entities, including TSG and MSP(s). Hence, model of consumed energy in ASMobiC mode depends on three components of $TW_{Remote(i)}$ (matrix multiply running time), $Discovery_{(i)}$(discovery delay), and D_i (communication delay).

TABLE IV
REGRESSION MODEL VALIDATION RESULT: TIME AND ENERGY

Model		R2	Adjuster R2	DF
Local/ASMobiC Time/Energy	Split = 1.0	1	1	18
	Split = 0.0	1	1	10
	Full Sample	1	1	29

For any component, except discovery delay, the linear regression can be performed to produce a mathematical model. Discovery delay has no regressor such as workload size or data volume. It only depends on the wireless communication quality between the MSC and TSG. Therefore, the total energy model for the remote execution mode is $ER(RMA_{(i)}) = (3.117 \times Dis_{(i)}) + (0.0938 \times TW_{Remote(i)}) - (0.002 \times WiFi_{(i)}) - 2666.364$. The energy model and its validation results are summarized in Table III and IV, respectively.

IV. RESULTS & DISCUSSIONS

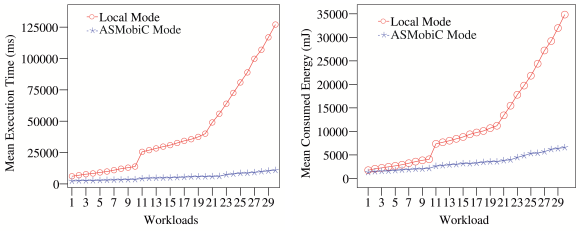
The results are reported in two parts of evaluation and validation for time and energy followed by synthesis of findings.

A. Evaluation Results: Descriptive analysis of benchmarking results are summarized in Table V and plotted in Figure 2. As results unveil, time and energy savings increase as the workloads intensify. The saving in low intensity is least and gradually reaches its peak in high workloads. In average, using RMCC conserves time and energy as high as 87% and 71.45%, respectively. The results illustrated in line charts present the comparison of mean time and energy results for 30 different workloads in local and ASMobiC execution modes. Each value is the mean of 30 iterations of each workload. Red circles represent local execution results and blue asterisks show ASMobiC results. In local mode, increase in workload intensities has significant impacts, whereas the impacts are remarkably lower in ASMobiC mode. Such substantial achievements are due to parallel distribution of intensive tasks on multiple ASMobiCs. Instead of executing all the tasks locally on a single CPU, RMCC leverages extensive computing power of multitudes of ASMobiCs whose access latency do not jeopardize achievements.

B. Validation Results: The validation results via mathematical modeling are depicted in Figure 3. Due to space limit, statistical description of validation results (relatively similar to evaluation results) is not presented. As shown in the Figure, time and energy saving are significant when intensive

TABLE V
DESCRIPTIVE STATISTICS OF EVALUATION RESULTS: TIME & ENERGY

Intensity	Execution Environment	Time (ms) Mean	Time (ms) St. Dev(σ)	Energy (mJ) Mean	Energy (mJ) St. Dev(σ)
Low	Local	9817	829.4	2882.8	248.9
	ASMobiC	2960	136.8	1755.2	87.5
Medium	Local	32174.9	1493.4	9143.2	410.8
	ASMobiC	5116.6	165.3	3143.4	103.8
High	Local	86196.2	8329.6	23587.4	2268.7
	ASMobiC	8503.4	530.2	5267.4	308.9
Mean	Local	42729.4	6552	11871.1	1774.3
	ASMobiC	5526.7	461.9	3388.6	289.3



(a) Execution Time (b) Consumed Energy

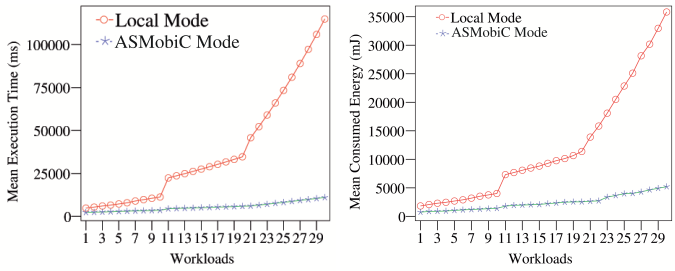
Fig. 2. Evaluation Results: Local vs ASMobiC

computations take place in ASMobiCs. Performance gains increases as the workload intensities hike.

Figure 4 depicts synthesis of evaluation and validation results for energy and time. The differences in evaluation and validation results are shown insignificant which advocates reliability and validity of RMCC.

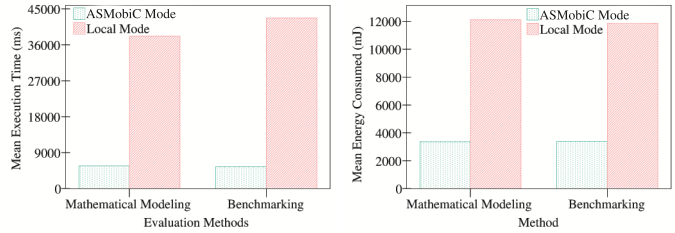
V. CONCLUSIONS

In this research, we proposed a lightweight framework, called RMCC that harnesses computational resources of plethora of ASMobiCs instead of distant clouds to efficiently execute compute-intensive tasks. Lightweight feature of the RMCC is realized by employing SOA and REST architectural style in design and development. The performance of the RMCC is evaluated and validated via benchmarking and mathematical modeling, respectively. The evaluation results advocate mean time and energy saving as significant as 87% and 71.45%, respectively when RMCC administrates execution of intensive services in ASMobiCs. Our findings advocate feasibility and benefit of leveraging computational capabilities of increasingly empowering and dominating smartphones as servers. In future, a lightweight hybrid resource scheduling algorithm like [15] to optimally allocate resources to intensive tasks is essential. We also need to enhance RMCC adaptability to varied wireless communication technologies like Bluetooth, 3G, and 4G to efficiently perform a throughput-energy trade-off in CMA communications. A billing system also is needed to manage financial issues and ASMobiC incentive handling matter in this framework.



(a) Execution Time (b) Consumed Energy

Fig. 3. Validation Results: Local vs ASMobiC



(a) Execution Time Synthesis (b) Consumed Energy Synthesis

Fig. 4. Synthesis of Evaluation and Validation Results

REFERENCES

[1] S. Abolfazli, Z. Sanaei, A. Gani, F. Xia, and L. T. Yang, "Rich Mobile Application: Genesis, Taxonomy, and Open Issues," *Journal of Network and Computer Applications*, vol. 40, pp. 345–362, 2014.

[2] W. Cai, V. C. M. Leung, and M. Chen, "Next Generation Mobile Cloud Gaming," in *IEEE CCNC '05 International Symposium on Service-Oriented System Engineering*, 2013, pp. 551–560.

[3] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 337–368, Jan. 2014.

[4] M. Shiraz and A. Gani, "A lightweight active service migration framework for computational offloading in mobile cloud computing," *The Journal of Supercomputing*, vol. 68, no. 2, pp. 978–995, Jan. 2014.

[5] F. Xia, F. Ding, J. Li, X. Kong, L. T. Yang, and J. Ma, "Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing," *Information Systems Frontiers*, vol. 16, no. 1, pp. 95–111, Oct. 2013.

[6] E. Ahmed, A. Akhunzada, M. Whaiduzzaman, A. Gani, S. H. Ab Hamid, and R. Buyya, "Network-centric performance analysis of runtime application migration in mobile cloud computing," p. In Press, 2014.

[7] S. Abolfazli, Z. Sanaei, M. Alizadeh, A. Gani, and F. Xia, "An Experimental Analysis on Cloud-based Mobile Augmentation in Mobile Cloud Computing," *IEEE Transactions on Consumer Electronics*, vol. 99, no. 1, pp. 1–9, 2014.

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

[9] E. E. Marinelli, "Hyrax: cloud computing on mobile devices using MapReduce," Master Thesis, Computer Science Department, Carnegie Mellon University, 2009.

[10] G. Huerta-Canepa and D. Lee, "A Virtual Cloud Computing Provider for Mobile Devices," in *Proc. ACM MCS'10: Social Networks and Beyond*, San Francisco, USA, 2010, pp. 1–5.

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

[12] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 369–392, 2014.

[13] Z. Sanaei, S. Abolfazli, T. Khodadadi, and F. Xia, "Hybrid Pervasive Mobile Cloud Computing: Toward Enhancing Invisibility," *Information-An International Interdisciplinary Journal*, vol. 16, no. 11, 2013.

[14] R. Jain, *The art of computer systems performance analysis*. Wiley, 2008.

[15] S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu, and A. Abraham, "Hybrid Job Scheduling Algorithm for Cloud Computing Environment," in *Int'l Conf. Innovations in Bio-Inspired Computing and Applications*, 2014, pp. 43–52.