# *MuSIC: Mobility-Aware Optimal Service Allocation in Mobile Cloud Computing*

M. Reza Rahimi[1], Nalini Venkatasubramanian[1], and Athanasios V. Vasilakos[2]

[1]School of Information and Computer Science, University of California, Irvine, USA.
[2]National Technical University of Athens, Athens, Greece.
[1]{mrrahimi, nalini}@ics.uci.edu, [2]vasilako@ath.forthnet.gr

***Abstract*—This paper exploits the observation that using tiered clouds, i.e. clouds at multiple levels (local and public) can increase the performance and scalability of mobile applications. User Mobility introduces new complexities in enabling an optimal decomposition of tasks that can execute cooperatively on mobile clients and the tiered cloud architecture while considering multiple QoS goals such application delay, device power consumption and user cost/price. In this paper, we propose a novel framework to model mobile applications as a *location-time workflows* (LTW) of tasks; here user mobility patterns are translated to a mobile service usage patterns. We show that an optimal mapping of LTWs to tiered mobile cloud resources is an NP-hard problem. We propose an efficient heuristic algorithm called *MuSIC* that is able to perform well (78% of optimal, 30% better than simple strategies), and scale well to a large number of users while ensuring high application QoS. We evaluate MuSIC and the 2-tier mobile cloud approach via implementation (on real world clouds) and extensive simulations using rich mobile applications like intensive signal processing and video streaming applications. Our experimental and simulation results indicate that MuSIC supports scalable operation (100+ concurrent users executing complex workflows) while improving QoS. We observe about 25% lower delays and power (under fixed price constraints) and about 35% decrease in price (considering fixed delay) in comparison to only using the public cloud. Our studies also show that MuSIC performs quite well under different mobility patterns, e.g. random waypoint, Manhattan models and is resilient to errors/uncertainty in prediction of mobile user location-time workflows.**

## I. Introduction

The rapid explosion in demand for rich mobile applications has created the need for new platforms and architectures that can cope with the scalability and QoS needs of a growing mobile user population. One of the main bottlenecks in ensuring mobile QoS is the level of wireless connectivity offered by last hop access networks such as 3G and Wi-Fi. These networks exhibit varying characteristics. For example, 3G networks offer wide area ubiquitous connectivity; however, 3G connections are known to suffer from *long delay* and *slow data transfers* [8], [1] resulting in increased power consumption and cost at the user side. In contrast, Wi-Fi deployments, e.g. 802.11 hotspots, exhibit low communication latencies/delays; devices connected to or collocated with Wi-Fi access points can be used to form a nearby local cloud [4], [8]. Using local only solutions with Wi-Fi networks creates *scalability* and *access* issues as the number of users increase. The second key issue is that rich mobile applications often require significant storage and processing abilities (e.g. content transcoding, caching, data interpretation) - despite advances in device technology, resources (energy, storage, processing) at the mobile host are limited.

**M**obile **C**loud **C**omputing (MCC) platforms aim to overcome the resource limitations of mobile devices and networks by leveraging resources available in distributed cloud environments. The goal is to offload compute and data intensive tasks from resource-poor mobile devices to cloud nodes. Recent market studies (e.g from Juniper Research [9]) indicate that the market for cloud-based mobile applications will grow 88% from 400 million in 2009 to 9.5 billion in 2014. A similar forecast made by ABI [12], predicts that the number of MCC subscribers worldwide is expected to grow rapidly over the next five years, rising from 42.8 million subscribers in 2008 to over 998 million in 2014. In our prior work, we discuss the role of public and local clouds in enabling scalable MCC. While public clouds provide resources at scale; there are a limited number of public cloud data centers within close proximity of mobile users resulting in large communication latencies. Recent efforts[11], [2], [1] have demonstrated the role of local resources within close proximity of the mobile user in ensuring improved application latencies. In our prior work[2], we have developed and evaluated a *2-Tier architecture* for the mobile cloud that synergistically combines the capabilities of local clouds and public cloud offerings to increase the performance and scalability of applications executing on mobile devices. Mobility of users introduces new complexities in ensuring QoS in MCC applications. As the number and speed of mobile users increase, mobile applications are faced with increased *latencies* and reduced *reliability*. As a user moves, the physical distance between the user and the cloud resources originally provisioned changes causing additional delays. Similarly, the lack of effective handoff mechanisms in WiFi networks as user move rapidly causes an increase in the number of *packet losses*. In other words, user mobility, if not addressed properly, can result in suboptimal resource mapping choices and ultimately in diminished application QoS.

**Key Contributions :** In this paper, we focus on developing efficient techniques for dynamic mapping of resources in the presence of mobility; using a tiered cloud architecture, we aim to meet the multidimensional QoS needs of mobile users. The

main contributions of this paper are as follow:

1) We propose the notion of a **location-time workflow** (LTW) as the modeling framework to model mobile applications and capture user mobility. Within this framework, we formally model mobile service usage patterns as a function of location and time. We also formally define the service allocation problem for mobile cloud computing. (Section II)
2) Given a mobile application execution expressed as a LTW, we optimally partition the execution of the location-time workflow in the 2-tier architecture based on a *utility metric* that combines *service price, power consumption and delay* of the mobile applications. We show that the resulting service allocation problem is NP-Hard and propose an efficient heuristic called **MuSIC** (**M**obility-Aware **S**ervice Allocat**I**on on **C**loud) to achieve a near optimal solution ( Section III).
3) We develop a prototype of the system using *Amazon WS* as the public cloud, a local campus cloud and Android devices. We implement two real-world rich media mobile applications (mobile video streaming, image and speech processing application). We evaluate our system under varying user mobility patterns including the **random waypoint** and **manhattan** [19] models; our simulation and experimental results indicate that MuSIC scales to a large number of users (100+) and performs within 78% of the optimal solution. Our experiments indicate that MuSIC is tolerant to errors and uncertainty in predicting mobile user location-time workflows - we achieve 62% of optimal performance when the uncertainty of location-time workflow prediction is as high as 30% (Section IV).
4) We also evaluate the performance 2-tier cloud architecture under significant mobility in comparison to using the public cloud alone. Our results indicate that the 2-tier cloud architecture *decreases* power consumption and delay by 25% on average when the price is fixed and decreases the average user's price about 35% (fixed value for delay and power consumption) in comparison to using only a public cloud. (Section IV).

We conclude with related efforts (Section V) and future directions( Section VI).

## II. Modeling Service Allocation on the Tiered Cloud

Fig. 1 shows the 2-tier cloud architecture[2], [3] for mobile applications . Tier 1 nodes in the system architecture represents public cloud services such as Amazon EC2, Microsoft Azure and Google AppEngine. Services provided by these vendors are highly *scalable* and *available*; what they lack is the ability to provide the *fine grain location granularity* required for high performance mobile applications. This feature is provided by the second tier local cloud, that consists of nodes that are connected to access points. Location information of these services are available at finer levels of granularity (campus and street level). Mobile users are typically connected to these

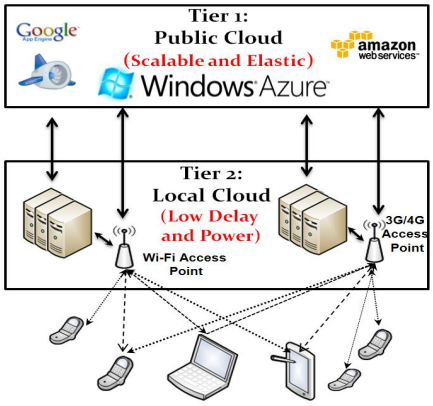


Fig. 1. 2-Tier Mobile Cloud Architecture.

local clouds through WiFi (via access points) or cellular (via 3G cell towers) connectivity - our aim to intelligently select which local and which public cloud resources to utilize for task offloading. In the following subsections, we develop concepts borrowed from service-oriented computing(SOC) literature to formally define the notion of location-time workflows (LTW) for mobile applications and use the LTW concept to define the MCC service allocation problem.

### A. Mobile Application Modeling

In this section we model cloud services, mobile users and mobile applications. Let's start by defining the concept of cloud service set. We use the notation $s_1, ...s_s$ to represent cloud services, $u_1, ..u_k$ to represent the set of mobile users, $l_1, ...l_m$ to represent locations in space and $t_1, ..t_n$ to represent time durations.

*Def. 1*: **Cloud Service Set**: The set of all services ( e.g. compute,storage and software capabilities like multimedia streaming services, content transcoding services, etc ) provided by local and public cloud providers, $C_s$ is formally expressed as:

$$C_s \triangleq \{s_1, s_2, ..., s_{|C_s|}\}$$

*Def. 2*: **Local Cloud Capacity**: Local cloud services can only accept a limited number of mobile client requests. We define a function $Cap(LC)$ which returns the *maximum* number of mobile clients that could be served using local cloud $LC$.

*Def. 3*: **Location Map**: is a partition of the 2-D space/region in which mobile hosts and cloud resources are located. Given a 2D region $R^2$, the location map $L$ is more formally expressed as

$$L \triangleq \{l_1, l_2, ..., l_{|L|}\},\ \forall i, j \in \{1, 2, ..., |L|\},\ l_i, l_j \subseteq \mathbb{R}^2$$
$$l_i \bigcap l_j = \emptyset,\ \bigcup_{i=1}^{i=|L|} l_i = \mathbb{R}^2$$

We assign a vector to the center of location, depicted as $(l_i \Leftrightarrow \overrightarrow{l_i})$.

*Def. 4*: **User Service Set**: The set of all services that a user $u_k$ has on his own device (e.g. decoders, image editors etc.) is represented as:

$$U_k^s \triangleq \{u_k^{s_1}, u_k^{s_2}, ..., u_k^{s_{|U_k^s|}}\}$$

*Def. 5*: **Mobile User Trajectory**: The trajectory of a mobile user, $u_k$, is represented as a list of tuples of the form $\{(\tau_1, l_k), ..., (\tau_n, l_m)\}$ where $(\tau_i, l_j)$ implies that the mobile user is in location $l_j$ for time duration $\tau_i$.

*Def. 6*: **Center of Mobility**: $l_{cm}^{u_k}$ is the location where (or near where) a mobile user $u_k$ spends most of its time. It is calculated as follows:

$$l_{cm}^{u_k} \in \{l_1, l_2, ..., l_{|L|}\}$$

$$l_{cm}^{u_k} \triangleq \min_{\overrightarrow{l_j} \in \{\overrightarrow{l_1}, \overrightarrow{l_2}, ..., \overrightarrow{l_{|L|}}\}} \| \frac{\sum_i \overrightarrow{l_i} \tau_i}{\sum_i \tau_i} - \overrightarrow{l_j} \|$$

*Def. 7*: **Mobile Application Workflow**: A generic mobile application is modeled as a *workflow* $w$ [15], [2], [3] consisting of a sequence of logical and precise steps, each of which is known as a *Function*. A workflow begins at the start function and finishes in the final function. Functions in a workflow can be composed together in different patterns. The SEQ pattern indicates a sequential execution of functions while the AND pattern models the parallel execution of functions. XOR is a conditional execution of the functions and the LOOP pattern indicates an iterative repetition of the functions. Each function is associated with a set of *services* that are capable of realizing and implementing the function.

For each Function $F_i$ in workflow $w$ we define $\chi_{F_i}$ as:

$$\chi_{F_i} \triangleq \{s_k \mid s_k \in U^s \cup C_s,\ s_k\ implements\ F_i\}$$

Intuitively $\chi_{F_i}$ is the set of all services that could realize function $F_i$ (*from cloud service set and all user service set*). For the workflow $w$ consisting of of $n$ tasks, the set $\Gamma$ describes all the feasible solutions or execution plans [15]. It is defined as the cartesian product:

$$\Gamma \triangleq \chi_{F_1} \times \chi_{F_2} \times ..... \times \chi_{F_n}$$

*Def. 8*: **Location-Time Workflow**: We next combine the mobile application workflow concept above and with a user trajectory to model the mobile users and the requested services in their trajectory.

A **Location-Time Workflow** (LTW), shown in Fig. 2, consists of sequences of workflows which are indexed by a mobile user's *location* and *duration/time*. It is represented more formally as follows:

$$W(u_k)_T^L \triangleq (w(u_k)_{t_1}^{l_1}, w(u_k)_{t_2}^{l_2}, w(u_k)_{t_3}^{l_3}, ...,\ w(u_k)_{t_n}^{l_n})$$

where $u_k$ is the $k^{th}$ mobile user and $w(u_k)_{t_n}^{l_n}$ is the user request workflow in location $l_n$ for time $t_n$.

So far we have modeled mobile users and their applications. We next model quality of service parameters for mobile applications.

### B. Modeling Quality of Service for Mobile Applications

Rich MCC applications present several Quality of Service (QoS) needs that capture different dimensions of user satisfaction - they are expressed using parameters such as *delay*, *power* and *price*. Table. I shows the quality of service parameters that we use to model mobile application needs in our tiered mobile cloud computing environment. Mobility introduces a new dimension to these QoS factors since they depend on *user location and requested time*. This is primarily due to the fact that communication link characteristics (Wi-Fi, 3G) vary based on user location and the time of the service. This in turn has an effect on the delay, power and price of the services and hence impacts the QoS. The delay of the service is considered as the difference between the time when a service is called (on the mobile device or cloud) and when the service is terminated. If the service on the cloud is being used we also consider network delay (Wi-Fi or 3G). Power consumption of the service refers to the power consumed on mobile device to execute the service. If the service executes on the cloud, power consumed includes the power overheads of the network connection and data transfer related to that service. Finally, the *price* of the service is the actual price/cost to the end user of executing the service on the public cloud. Table II defines the QoS for the *application workflow* based on the execution plan $\overrightarrow{x} \in \Gamma$. The QoS of a workflow is evaluated based on the QoS of its atomic services while taking into account the composition patterns [15]. For example, the QoS of a SEQ pattern is the sum of the QoSes of the constituent tasks for all QoS parameters (price, power, delay).

| Criteria | Definition |
|---|---|
| $q_{price}(s_i, u_k^{l_i,t_j})$ | The price of using service $s_i$ when user $u_k$ is in location $l_i \in L$ and time $t_j$. |
| $q_{power}(s_i, u_k^{l_i,t_j})$ | The power consumed on user mobile device using $s_i$ when user $u_k$ is in location $l_i \in L$ and time $t_j$. |
| $q_{delay}(s_i, u_k^{l_i,t_j})$ | The delay of executing service $s_i$ when user $u_k$ is in location $l_i \in L$ and time $t_j$. |

TABLE I
QOS PARAMETERS THAT WILL BE USED IN MOBILE CLOUD COMPUTING ENVIRONMENT

We extend the notion of workflow QoS to LTW QoS to

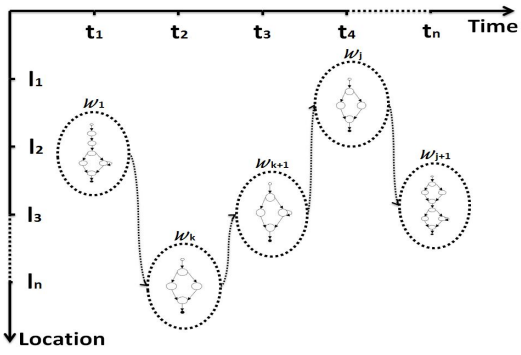


Fig. 2. Location-Time Workflow

| QoS | SEQ | AND | XOR | LOOP |
|---|---|---|---|---|
| $w(u_k)_{price}$ | $\sum_{i=1}^{n} q_{price}^i$ | $\sum_{i=1}^{n} q_{price}^i$ | $\max_i\ q_{price}$ | $q_{price} \times k$ |
| $w(u_k)_{power}$ | $\sum_{i=1}^{n} q_{power}^i$ | $\sum_{i=1}^{n} q_{power}^i$ | $\max_i\ q_{power}$ | $q_{power} \times k$ |
| $w(u_k)_{delay}$ | $\sum_{i=1}^{n} q_{delay}^i$ | $\max_i\ q_{delay}$ | $\max_i\ q_{delay}$ | $q_{delay} \times k$ |

TABLE II
WORKFLOW QOS MODEL

incorporate user mobility as follows:

$$[W(u_k)_T^L]_{price} \triangleq \sum_{i=l_1,j=t_1}^{i=l_n,j=t_n} [w(u_k)_j^i]_{price}$$

$$[W(u_k)_T^L]_{power} \triangleq \sum_{i=l_1,j=t_1}^{i=l_n,j=t_n} [w(u_k)_j^i]_{power}$$

$$[W(u_k)_T^L]_{delay} \triangleq \sum_{i=l_1,j=t_1}^{i=l_n,j=t_n} [w(u_k)_j^i]_{delay}$$

LTW-QoS is a summation of user-experienced QoS in different locations and times.

Calculating the utility of mobile users with LTWs ( LTW-utility) requires us to normalize the multiQoS metrics of price, power, delay that have different units, e.g. dollar, joule and second. First, we apply a normalization process [15] for *atomic services*. We illustrate this in the context of price (it is easily generalized to power and delay).

- $Price^{max}(\chi_{Fi})$ : The maximum price of the services that can realize function $Fi$.
- $Price^{min}(\chi_{Fi})$ : The minimum price of the services that can realize function $Fi$.
- For each services $s \in \chi_{Fi}$ the normalized price could be defined as:

$$\|s_{price}\| \triangleq \begin{cases} \frac{Price^{max}(\chi_{Fi}) - s_{price}}{Price^{max}(\chi_{Fi}) - Price^{min}(\chi_{Fi})} & Price^{max}(\chi_{Fi}) \\ & \neq Price^{min}(\chi_{Fi}) \\ 1 & \text{else} \end{cases}$$

For each services $s \in \chi_{Fi}$ the **total normalized QoS** is defined as: $\|s\| \triangleq [\|s_{pow}^2\| + \|s_{price}^2\| + \|s_{delay}^2\|]^{\frac{1}{2}}$ . In general the *higher* the $\|s\|$ is, the better the QoS/performance (small delay, power consumption and price) of the service.

The next step in normalization process is to extend it to the *workflow* $w$. Again, wlog, we illustrate this step using the price (trivially extended to power and delay).

- $C_{price}^{max}$ : The total price of the services in workflow when the most expensive services are selected.
- $C_{price}^{min}$ : The total price of the services in workflow when the cheapest services are selected.
- $\|w(u_k)_{price}\|$ : *Normalized price* of the workflow with specific service plan $\overrightarrow{x} \in \Gamma$ is defined as:

$$\|w(u_k)_{price}\| \triangleq \begin{cases} \frac{C_{price}^{max} - w(u_k)_{price}}{C_{price}^{max} - C_{price}^{min}} & C_{price}^{max} \\ & \neq C_{price}^{min} \\ 1 & \text{else} \end{cases}$$

Finally, we repeat the same procedure for the LTW as follows:

- $[C_T^L]_{price}^{max}$ : The total price of the services in LTW when the most expensive services are selected.
- $[C_T^L]_{price}^{min}$ : The total price of the services in LTW when the cheapest services are selected.
- $\|[W(u_k)_T^L]_{price}\|$ : *Normalized price* of the space-time workflow with specific service plan $\overrightarrow{x} \in \Gamma$ is defined as:

$$\|[W(u_k)_T^L]_{price}\| \triangleq \begin{cases} \frac{[C_T^L]_{price}^{max} - [W(u_k)_T^L]_{price}}{[C_T^L]_{price}^{max} - [C_T^L]_{price}^{min}} & [C_T^L]_{price}^{max} \\ & \neq [C_T^L]_{price}^{min} \\ 1 & \text{else} \end{cases}$$

LTW and QoS give us a formal and solid framework using which we can study the performance and further analyze mobile applications in MCC. We next formalize a *Utility Function* which models the general performance of the system. Note that Different utility functions could be defined - for example, those that consider the service provider benefits, mobile user benefits or both. In this paper, our main concern is mobile user benefits. We define our utility metric, **fair mobile user utility** [13], below:

$$F_{mobile} \triangleq \frac{1}{n} \sum_{u_k} \min\{\|[W(u_k)_T^L]_{price}\|, \|[W(u_k)_T^L]_{power}\|, \|[W(u_k)_T^L]_{delay}\|\}$$

Intuitively this function returns the average of minimum saving of price, power and delay of mobile users as the utility. By combining the utility function and *system constraints* we formalize the following optimization problem for service allocation in MCC:

$$\begin{aligned} &\max F_{mobile} \\ &st: \\ &\frac{1}{n}\sum_{u_k}[W(u_k)_T^L]_{price} \leq B_{price} \\ &\frac{1}{n}\sum_{u_k}[W(u_k)_T^L]_{power} \leq B_{power} \\ &\frac{1}{n}\sum_{u_k}[W(u_k)_T^L]_{delay} \leq B_{delay} \\ &\kappa \leq Cap(LC),\ \kappa \leq n \\ &\kappa \triangleq \text{Number of Mobile users using services on local cloud.} \\ &\forall u_k \in \{u_1, u_2, ..., u_n\} \end{aligned} \tag{1}$$

The first, second and third constraints indicate that the user spent price, consumed power and experienced delay should be less than a threshold/limit. The later constraints capture the local cloud resource limitations which could only accept a limited number of mobile user requests. This problem is shown to be NP-Hard (the Knapsack problem being a special case). In the next section we will propose an efficient heuristic to solve the MCC Service Allocation Problem.

## III. MuSIC: A Heuristic for Optimal Service Allocation in the 2-Tiered Cloud Environment

We extend our previous work in MAPCloud[2] and develop **MuSIC** (**M**obility-Aware **S**ervice Allocat**I**on on **C**loud), an efficient heuristic for tiered-cloud resource allocation which takes into consideration user mobility information. The MuSIC algorithm is a greedy heuristic that generates a near-optimal solution to the tiered cloud resource allocation problem using

$MuSIC\ (W(u_k)_T^L, S, F_{mobile}, \overrightarrow{C}, max_{iter})$

$W(u_k)_T^L$: $u_k$ LTW, $S$: Service Set DB, $F_{mobile}$: Utility Function, $\overrightarrow{C}$: Constraint Vector, $max_{iter}$: Number of Iteration in Simulated Annealing.

Begin
(1) Compute $l_{cm}^{u_k}$.
(2) $Candidate_{Service} = Find_{Service}(W(u_k)_T^L, \overrightarrow{C}, l_{cm}^{u_k})$
(3) $Util_0 = Compute_{F_{mobile}}(Candidate_{Services})$
(4) For i=1 to $max_{iter}$ do
(5) $Candidate_{Services} = Find_{Service}(W(u_k)_T^L, \overrightarrow{C}, l_{cm}^{u_k})$
(6) $Util_1 = Compute_{F_{mobile}}(Candidate_{Services})$
(7) $\Delta = Util_1 - Util_0$
(8) If $\Delta > 0$
(9) $Util_0 = Util_1$
(10) Else
(11) Replace $Util_0 = Util_1$ when $exp(max_{iter}) \geq U[0,1]$
/* U[0,1] means the uniform distribution function */
(12) End if
(13) End for
(14) Return $Candidate_{Service}, Util_0$
End

TABLE III
MUSIC ALGORITHM PSEUDO CODE

a simulated annealing-based approach, which has been shown to be an efficient heuristic for knapsack problems. A simulated annealing approach typically starts out with an initial solution in the potential solution space and iteratively refines this to generate increasingly improved solutions. It uses a randomized approach to increase the diversity of service selection.

Table III contains pseudo code for the MuSIC algorithm. While MuSIC uses simulated annealing as the core approach in selecting and refining service selection; custom policies have been designed to make it efficient for the 2-tiered cloud architecture with mobile applications. Given a cloud service set, a set of users with their corresponding LTWs (location-time workflows), MuSIC starts by computing the center of mobility $l_{cm}^{u_k}$ of each user $u_k$. Intuitively it is a location in the mobile user's trajectory where much of the time is spent; the general goal is to select services near that location. MuSIC then uses the service selection function $Find_{Service}(W(u_k)_T^L, \overrightarrow{C}, l_{cm}^{u_k}, F_{mobile})$ that returns the list of services near the user center of mobility $l_{cm}^{u_k}$, which can realize the LTW and satisfy the required constraints. The utility function $F_{mobile}$ of this solution is computed in line 3. Following this, the MuSIC algorithm will enter a loop which is the main core for the simulated annealing based algorithm. The difference between the initial value of $F_{mobile}$ function and current computed value of $F_{mobile}$ function is extracted in line 7. If it is positive, it will be then considered as the new service list; if negative, it may still be retained with some probability and the algorithm will enter the next iteration. The while loop is eventually terminated when the number of iterations exceeds a limit $it$. After the iterations are done the final utility and service set will be returned as the solution.

The main core of MuSIC is the $Find_{Service}$ function which returns the candidate service set for $W(u_k)_T^L$. There are two intuitions behind this function. First of all it is known that services in close proximity to the user usually provide better QoS performance in terms of delay and power consumption.

$Find_{Service}(W(u_k)_T^L, \overrightarrow{C}, l_{cm}^{u_k})$

We assume that the directory service database contains information on the normalized QoS of the service.
$W(u_k)_T^L$: $u_k$ LTW, $l_{cm}^{u_k}$: $u_k$ center of mobility location, const $d_{th}$ : Threshold Distance, const $d_r$ : The increase amount of distance, const $it$ : Maximum number of iteration
Begin
(1) i=0;
(2) while (i $< it$)
begin
(3) $d = d_{th} + i * d_r$
(4) $Candidate_{Services}$=Retrive the related services according to LTW in distance $d$ of $l_{cm}^{u_k}$.
(5) if $Candidate_{Services}$ contains all of the needed services then make 4 different lists sorted according to normalized price, power, delay and total normalized QoS from large to small.
(6) randomly choose from the 3 list services.
(7) check if it satisfies the constraints
(8) if yes return the service set
(9) else
(10) i=i+1;
(11) increase the search radios to $d = d_{th} + i * d_r$
end while
End

TABLE IV
MUSIC $Find_{Service}$ ALGORITHM PSEUDO CODE

Secondly using services with high QoS will increase system utility. In our context, improved QoS can be realized using one of four metrics – normalized delay, power, price and total normalized QoS as defined in previous section.

Table IV illustrates the $Find_{Service}$ routine. It starts with a candidate set of services within a threshold distance $d = d_{th}$ from the $l_{cm}^{u_k}$. Four sorted lists are generated from the candidate set, sorted based on the normalized price, power, delay and total QoS from high to low. MuSIC performs a randomized selection of services from these lists; the random selection is evaluated for satisfaction of the the input constraints. If input constraints are satisfied, the list is returned; else the process is repeated with an increased search distance.

## IV. SYSTEM PROTOTYPING AND EVALUATION

We extend MAPCloud middleware to support LTW and MuSIC [2]. Fig. 3 illustrates the general architecture of MAPCloud platform with the key modules describe below:

*Mobile User Log DB and QoS-Aware Service DB*: The first one contains unprocessed user data log such as mobile service usage, location of the user, user delay experience of getting the service, energy consumed on user mobile device, etc. The second one contains the service lists on local and public cloud and their QoSes in different locations.

*MAPCloud Analytic*: This module processes mobile user Log DB and updates QoS-aware cloud service DB based on user experience and LTW.

*Admission Control and Scheduling*: This module is responsible for optimally allocate services to admitted mobile users based on MuSIC.

The operational flow through this module is simple - a user requested mobile application is forwarded to the MAPCloud. If admitted (based on service availability), the scheduler module will compute and determine the best allocation of services using the MuSIC algorithm. The scheduler modules consult

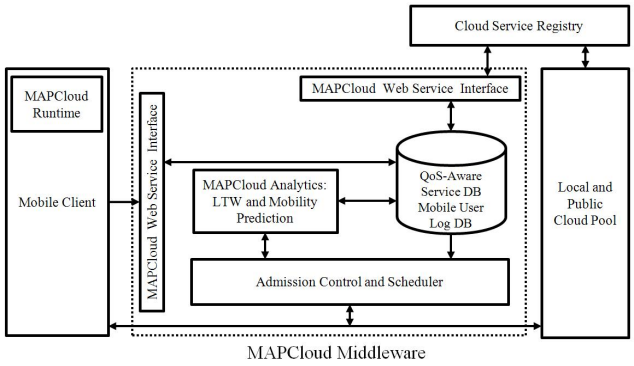


Fig. 3. Middleware Service Architecture

the QoS-Aware Cloud DB and MAPCloud Analytic. The service plan is returned back contains URL of each services in application LTW.

To study MuSIC performance OCR+Speech (OCRS) and video streaming and transcoding (VS) applications have been developed as the rich mobile applications. In the first application the user takes a picture of the text page and the application will return a file which contains the spoken text. The second application is video streaming and transcoding application in which the video clip is streamed to the mobile users. For both of these two applications different services has been extracted such as image filtering, noise cancelation, etc. We measure the delay and power consumption of services in different situation for both local and public cloud. Fig. 4 shows our average delay and power consumption for mentioned applications considering different data size. (a) and (c) show the average power consumption of transmitting/recieving data from Android G2 to local and public cloud with different data size. For example for typical 2Mb of file size the average Wi-Fi power consumption is 15435 mjole. This power consumption is more when using public cloud as shown in figure Fig. 4 (c). For 2Mb of file size the average Wi-Fi power consumption is 19345 mjole.

### A. Simulation Studies

Simulation platform is used to test the performance and scalability of the MuSIC considering users mobility. In particular, we used MATLAB and CloudSim [5], an open source cloud simulator. We used the experimental result obtained by profiling real applications (OCRS and VS) to tune the simulation environment.

The basic simulation setup models a region with 225 cells ($15 \times 15$). Local clouds have valid Wi-Fi in 6 cells around and there exists 3G connectivity in whole region. A LAN provides a backbone for local cloud connectivity and data transfer. We used two important mobility model in our simulation environment one is **Random Waypoint** (RW) model and the other one is **Manhattan** model [19]. Manhattan mobility model is mainly used for the movement in urban area, where the streets are in an organized manner. we used the $15 \times 15$ grid size in our simulation. In our simulation we used the speed range in [1m/s, 10m/s]. we combine these two models in our simulation environment in a sense that 50% of mobile users have RW model the remaing have manhattan model. In our simulation environment we assumed that half of the time mobile users are using OCRS and half of time they are using VS applications. We set the maximum number of MuSIC iterations to 20. In our experiments, we varied data sizes which were uniformly distributed from [1Mb, 5Mb]. Each simulation results is the average of 15 runs. We test the performance of the system based on different number of users, different number of public and local cloud instances and uncertainty in prediction of mobile users' LTW. For example 10% uncertainty in LTW consists of 10 sub-workflow means that in average one from 10 is not predicted correctly (error in prediction of user's location or requested service). In our simulation we considered the uncertainty in the range of [0%-30%].

***MuSIC Optimality Study under Mobility:***

To measure the MuSIC optimality we compare it with *Random Service Allocation* (RSA) and optimal solution derived by *brute-force search* of Eq.1 . We used the following metrics to measure throughput of the service allocation algorithms:

$$MuSIC_{Throughput} = \frac{MuSIC\ output}{Optimal\ Solution\ of\ Eq.\ 1} \times 100$$

$$RSA_{Throughput} = \frac{RSA\ output}{Optimal\ Solution\ of\ Eq.\ 1} \times 100$$

Fig. 5 shows the throughput of MuSIC and RSA for mobile applications when there are several mobile users in the system with varied uncertainty. As it is shown in Fig. 5 (a), when there is no uncertainty in LTW prediction, MuSIC could achieve to 56% performance when there are 15 mobile users. The performance increases to 78% in average when there are 100 mobile users. By having uncertainty in prediction of LTW in the range of [0%,30%] as it is shown in Fig. 5 (b), MuSIC

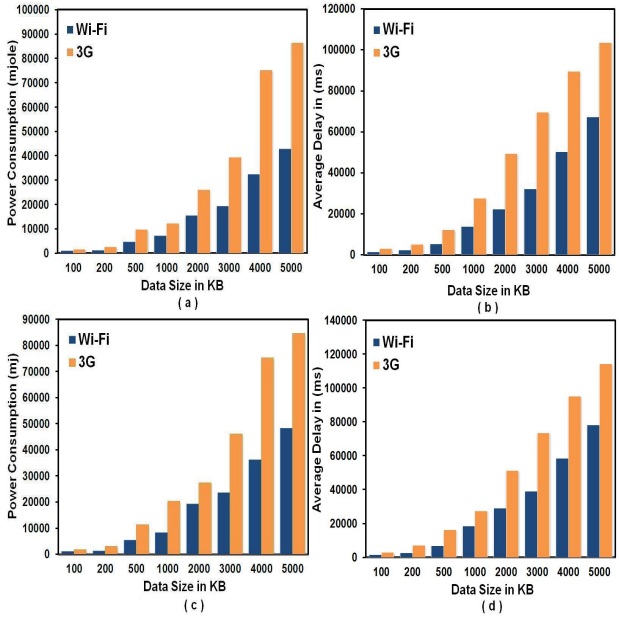


Fig. 4. Delay (in ms) and power consumption (in mjole) of different wireless network types regarding to data size when using local cloud ( Fig. a and b) and Amazon Public Cloud (Fig. c and d). (It has been averaged for different services of OCRS and VS applications.)

| Number of Users | 20 | 40 | 80 | 100 |
|---|---|---|---|---|
| MuSIC (seconds per person) | 3s | 8s | 15s | 30s |
| RSA (seconds per person) | 1s | 3s | 6s | 10s |
| Brute-Force Search (seconds per person) | 8s | 55s | 240s | 590s |

TABLE V
PROCESSING TIME OF MUSIC, RSA AND BRUTE-FORCE SEARCH

could achieve about 62% of optimal solution when there are 100 mobile users. Fig. 5 (c),(d) show the same results for RSA algorithm. With zero uncertainty RSA have 43% performance when there are 100 mobile users in the systems. Adding uncertainty does not change the performance (still 40%). This unchanged performance in RSA throughput makes sense while RSA randomly assigns services to mobile users without using user trajectory information.

Table V shows the running time of MuSIC, RSA and brute-force search method (on a 64bit Windows dual-core Intel with 8GB of memory and 500GB of hard). According to this table when the number of mobile users is small such as 20 MuSIC execution time is about 3 times faster than brute-force search method and 3 times slower than RSA. This ratio increases to about 20 in comparison to brute-force search method when there is a large number of users in the system.

Fig. 6 (a) and (b) show the real delay and power consumption according for different number of mobile users with LTW uncertainty in the range of [0%,30%]. For example as shown in Fig. 6 (a), by having 8 local clouds the average power consumption would be 50 jole/person (when there are 100 users). Adding 8 public services could decrease the power consumption about 20%. This shows that increasing computing and storage resources does not necessarily increase the performance linearly while the communication bandwidth is a bottleneck.

***Evaluating the 2-Tiered Cloud under Mobility :***

In this section we study the performance of the 2-tier cloud

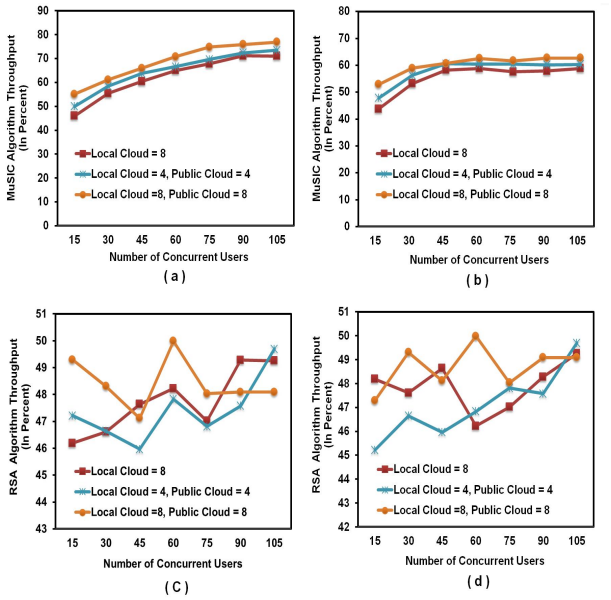


Fig. 5. MuSIC and RSA algorithms average throughput: (a) without any uncertainty, (b) with uncertainty in the range of [0%,30%] (c) without any uncertainty, (d) with uncertainty in the range of [0%,30%].

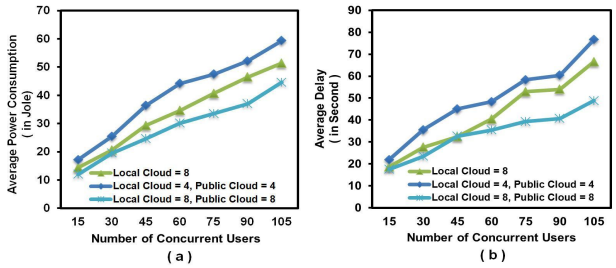


Fig. 6. MuSIC algorithm real averaged values for delay and power consumption

architecture in comparison to only using public cloud services. One way of comparing the 2-tier cloud architecture is using the metrics described in table VI. For example if a mobile application should experience constant low delay such as in some video streaming, then we could measure the gain that we get in power consumption and price by using 2-tier cloud architecture in comparison to only using public cloud. For example by using 2-tier cloud services the average power consumed on user device is 8 joules. If only public cloud is used then it will be 10 joules (due to long delay). Then the mobile user will gain $[1 - 8/10] \times 100 = 20\%$ by using this 2-tier architecture in comparison to only using public cloud. By averaging this metric over all of the mobile users $u_k$ we could gain the average mobile users gain. The same procedure could be extended to power and price according to the table (*Amazon large Instance* and *T-Mobile* prices were used as the data and cloud price model.).

Table. VII shows the values of the mentioned metrics for 100 users using MuSIC and RSA algorithms with varied uncertainty in location-time prediction. With constant delay by using MuSIC one could get 35% gain in price and 7% gain in power consumption when there is no uncertainty in LTW prediction. Having [0%,30%] uncertainty, mentioned values become 27% and 2%. These results are intuitively correct while the lower the delay the lower power consumption is. The same is true for price because of cheaper price of WiFi and local services in comparison to 3G and public clouds virtualized services. With constant power consumption and zero uncertainty in LTW prediction by using MuSIC one could get 30% percent saving in price and 10% saving in application delay. These values are changing to 11% and 2%

| | Metrics |
|---|---|
| Constant Delay | $[1 - \frac{Price(Public + Local\ Cloud)}{Price(Public\ Cloud)}] \times 100$ |
| | $[1 - \frac{Power(Public + Local\ Cloud)}{Power(Public\ Cloud)}] \times 100$ |
| Constant Power | $[1 - \frac{Price(Public + Local\ Cloud)}{Price(Public\ Cloud)}] \times 100$ |
| | $[1 - \frac{Delay(Public + Local\ Cloud)}{Delay(Public\ Cloud)}] \times 100$ |
| Constant Price | $[1 - \frac{Delay(Public + Local\ Cloud)}{Delay(Public\ Cloud)}] \times 100$ |
| | $[1 - \frac{Power(Public + Local\ Cloud)}{Power(Public\ Cloud)}] \times 100$ |

TABLE VI
PERFORMANCE METRICS FOR EVALUATION OF 2-TIER CLOUD ARCHITECTURE

| | | 2-Tier Cloud Architecture Performance | | | |
|---|---|---|---|---|---|
| | | MuSIC | | RSA | |
| | | 0% Uncertainty | [0%-30%] Uncertainty | 0% Uncertainty | [0%-30%] Uncertainty |
| Constant Delay | Price | 35% | 27% | 12% | 10% |
| | Power | 7% | 2% | 2% | 3% |
| Constant Power | Price | 30% | 22% | 11% | 13% |
| | Delay | 10% | 4% | 2% | 2% |
| Constant Price | Power | 26% | 17% | 8% | 7% |
| | Delay | 22% | 15% | 7% | 10% |

TABLE VII
2-TIER CLOUD PERFORMANCE RESULTS

when uncertainty is in LTW prediction. With the constant price one could get 26% decrease in power consumption and 22% decrease in application delay using MuSIC. As it is clear from the figure the performance of RSA doesn't change much with having uncertainty in LTW prediction. This makes sense while RSA does included the uncertainty in itself.

## V. RELATED WORK

The concept of exploiting external resources to address the resource limitations of mobile platforms is not new. The typical application runs a simple GUI on the mobile device and intensive-processing tasks on a remote server, [8], [6], [17], [1]. Platforms such as *MAPGrid* [14] exploits nearby idle grid computing resources to intelligently process and cache data for rich mobile applications such as video streaming. *Cloudlets* [11] use virtualizing technology for creation of resources near access points (AP) and *CloneCloud* [4] uses workflow concepts to partition applications between a mobile device and local cloud. Techniques for service partitioning have been studied in early systems such as *Spectra* [18] and *Chroma* [16] that use client-server architectures to offload resource intensive tasks; both these systems use prior knowledge of application performance and resource utilization to learn behavior and subsequently optimize performance. In [7], they proposed the architecture based on group of mobile devices to upload the task. They claimed that this architecture could improve the mobile application performance but they did not considered the performance of the application such as power and delay which are critical for mobile applications. In *WhereStore* [10], the authors considered the data sharing application. They showed that the locality of these storage can significantly improve the performance of the application, specially for location-based data search and sharing. In this work they mainly target to reduce the missing rate of replicas in such applications. In contrast to the above approaches, the MAPCloud [2] approach consisting of the tiered cloud architecture, location-time workflows and the MuSIC service allocation algorithm aims to support service partitioning in a tiered cloud architecture, such a integrated approaches allows us to handle simultaneously the multiple dimensions of QoS, scalability and mobility.

## VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we proposed a new framework to model mobile applications as a Location-Time Workflow - the unique aspect here is that this abstraction models the mobile user service usage patterns based on user mobility. Our main goal was to use this concept to optimally decompose the set of tasks to execute on the mobile clients and the 2-tier cloud architecture, considering multiple QoS factors like power, price, and delay. We proposed an efficient algorithm called *MuSIC* that is able to achieve about 78% of optimal solutions when the number of mobile users is high. Our studies also show that MuSIC performs quite well under uncertainty in prediction of mobile user LTW and different mobility patterns like random waypoint and Manhattan models. In our future work, we aim to expand the MAPCloud approach to support group-based mobile applications such as mobile multimedia file sharing and mobile social networks.

## REFERENCES


[1] Marco Valerio, Sokol Kosta,Alessandro Mei,Julinda Stefa: " To Offload or Not to offload? The Bandwidth and Energy Costs for Mobile Cloud Computing", In IEEE INFOCOM 2013.

[2] M. Reza. Rahimi, Nalini Venkatasubramanian, Sharad Mehrotra and Athanasios Vasilakos, "*MAPCloud: Mobile Applications on an Elastic and Scalable 2-Tier Cloud Architecture*", In IEEE/ACM *UCC 2012*.

[3] M. Reza. Rahimi "*Exploiting an Elastic 2-Tiered Cloud Architecture for Rich Mobile Applications*", In IEEE/ACM *WoWMoM 2012*.

[4] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, Ashwin Patti, " *CloneCloud: Elastic Execution between Mobile Device and Cloud*", In ACM *EUROSYS 2011*.

[5] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, " *CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms*", Wiley Press, 2011.

[6] Shivajit Mohapatra, M. Reza. Rahimi, Nalini Venkatasubranian "Power-Aware Middleware for Mobile Applications", Chapter 10 of the Handbook of Energy-Aware and Green Computing, Chapman & Hall/CRC, 2011.

[7] G. H. Canepa, D. Lee " *A Virtual Cloud Computing Provider for Mobile Devices*", In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing; Services: Social Networks and Beyond (*MCS '10*), New York, 2010.

[8] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl " *MAUI: Making Smartphones Last Longer with Code Offload*", In *MobiSys 2010*.

[9] Juniper, "*Mobile Cloud Computing: $9.5 billion by 2014*", Juniper, http://www.readwriteweb.com/archives, Tech. Rep., 2010.

[10] P. Stuedi, I. Mohomed, and D. Terry " *WhereStore: location-based data storage for mobile devices interacting with the cloud*", In the Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services, *MCS '10* New York, NY, USA, 2010.

[11] M. Satyanarayanan, P. Bahl, R. Cceres, N. Davies " *The Case for VM-Based Cloudlets in Mobile Computing*", In IEEE *PerCom 2009*).

[12] ABI, "*Mobile Cloud Computing Subscribers to Total Nearly one Billion by 2014*", ABI http://www.abiresearch.com/press/1484/, Tech. Rep., 2009.

[13] C. Wilbaut, S. Hanafi, S., S. Salhi, "*A Survey of Effective Heuristics and Their Application to a Variety of Knapsack Problems*" IMA Journal of Management Mathematics, 19(3), 227-244, 2008.

[14] Y. Huang, N. Venkatasubramanian " *MAPGrid: A New Architecture for Empowering Mobile Data Placement in Grid Environments*", In IEEE/ACM *CCGrid 2007*.

[15] L. Zeng, B. Benatallah, A. H. NGU, M. Dumas, J. Kalagnanam, and H. Chang " *Qos-Aware Middleware for Web Services Composition* ", In *IEEE Trans. Software. Eng*, 2004.

[16] R. Balan, M. Satyanarayanan, S. Park, T. Okoshi, "*Tactics-based Remote Execution for Mobile Computing*", In *MobiSys 2003*.

[17] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau and N. Venkatasubramanian, "*Integrated Power Management for Video Streaming to Mobile Handheld Devices*", In *ACM Multimedia 2003*.

[18] J. Flinn, S. Park, M. Satyanarayanan, "*Balancing Performance, Energy, and Quality in Pervasive Computing*", In IEEE *ICDCS 2002*.

[19] T. Camp, J. Boleng, and V. Davies, "*A Survey of Mobility Models for Ad Hoc Network Research, in Wireless Communication and Mobile Computing (WCMC)*", Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002.