Resilient Collaborative Cloud Management in Mobile Environments

Ahmed A. Khalifa, Mohamed Azab, Bassem Mokhtar, Mohamed Eltoweissy

Abstract— We are witnessing exponential growth in the number of powerful, independendant, multiply-connected, energy-rich stationary and mobile computing devices.We claim that cloud computing can be realized over such potentially scattered infinite pool of resources. However, Given the contemporary resource-allocation, virtilization-management, and task-management mechanisms supporting modern clouds, it is impossible to construct a cloud that can autonomously adapt to such real-time dynamic variation of heterogeneously-composed potentially- mobile resources. To this end, we propose CoCloud, a comprehensive collaborative cloud management platform, for enabling both resource-infinite computing paradigm over stationary and mobile nodes, and a true on-demand cloud computing. A reference model is presented to reflect CoCloud ability to serve different service delivery models. CoCloud's Global Resource Positioning System (GRPS) employs a global mobile and stationary resource discovery and monitoring to globally position active resources based on a dynamic spatiotemporal calendar. GRPS provides dynamic real-time scheduling, forecasting and tracking of idle, mobile and stationary, resources. Resources are provisioned through CoCloud's smart Virtualization Management Layer (VML). VML fractionizes a user's application; employs a vast number of dynamic, resource-aware micro virtual machines to encapsulate such fractions and facilitate capsule hosting on matching cloud hardware resources. Such employment enables seamless execution over heterogeneous resources, lightweight load migration, and low cost of failure. Using analysis and simulations, results show that CoCloud can guarantee reliable resource provisioning, transparently maintaining applications' QoS, and preventing service disruption in highly dynamic environments. Also, CoCloud resource collaboration enhances both application performance and management overhead by reducing the number of inter-host VM migrations as well as the communication delay.

Index Terms— Cloud Computing, Mobile Computing, Collaborative Computing, Distributed Resource Management, Virtualization.

IJSER

1 Introduction

With the emergence of cloud computing and the advances in mobile computing technologies, the mobile cloud computing (MCC) paradigm was introduced. There are two types of MCC architectures [1-5]:1) a MCC offering access and service delivery to users through their mobile devices where all computations, data handling, and resource management are performed in the static cloud to offload the computational workload from the mobile nodes to the cloud [1] [2] [3]; and 2) utilizing the idle resources of mobile devices and enabling them to work collaboratively as cloud resource providers [4] [5].

In this paper, and in a prior series of papers [6] [7] [8] [9] [10], we adopt and extend the latter definition of MCC as cloud computing, through the collaboration and virtualization of heterogeneous, mobile or stationary, scattered, computing resources forming a collaborative ad-hoc cloud platform that provisions computational services to its users. We term such cloud construction, Dynamic Collaborative Cloud, or CoCloud for short.

- *Ahmed A. Khalifa is currently an Assistant Professor in Switching and Traffic Department, National Telecommunication Institute (NTI), Cairo, Egypt, Egypt. E-mail: akhalifa@nti.sci.eg*
- *Mohamed Azab is currently a Researcher in IRI, The City of Scientific Research and Technological Applications, Egypt. E-mail: mazab@vt.edu*
- *Bassem Mokhtar is currently an Assistant Professor in Department of Electrical Engineering in Alexandria University, Egypt. E-mail: bmokhtar@alexu.edu.eg*
- *Mohamed Eltoweissy is currently a Professor in Department of Computer and Information Sciences, Virginia Military Institute, Virginia, USA E-mail: eltoweissymy@vmi.edu*

Current propositions for MCC solutions, [4] [5] [11] [12] [13], are primarily computing-cluster like rather than cloud-like systems. These approaches facilitated the execution of a certain distributed application(s) hosted on a stationary/semi-stationary stable mobile environment. However, no prior research work has realized the essential characteristics of the cloud model as defined by the National Institute of Standards and Technology (NIST) [14].

In order to build a CoCloud, there are multiple challenges that have to be addressed. These challenges are as follows.

Current resource management and virtualization technologies do not readily provide a virtualization layer that can autonomously adapt to the real-time dynamic variation, mobility, and fractionalization of such infrastructure [4] [5]. Consequently, these limitations make it almost impossible to isolate the resource layer concerns from the executing code logic. Such isolation is an enabler for the cloud to operate and provision its basic services such as, seamless task deployment, execution, migration, dynamic/adaptive resource allocation, and automated failure recovery.

Further, given the dynamic nature of the mobile hardware resources, resource allocation is another vital issue that needs to be addressed to construct a resilient collaborative cloud. Such cloud has a dynamic nature as nodes, usually having heterogeneous capabilities, may join or leave the formed cloud varying its computing capabilities. Also, the number of reachable nodes may vary according to the mobility pattern of these nodes. However, most of the existing task scheduling and resource allocation algorithms [15] [16] [17] [18] [19] did not consider the prediction of resource availability or the connectivity among mobile nodes in the future which affects the performance of submitted applications. Therefore, for the cloud to

operate reliably and safely, we need to accurately specify the expected amount of resources that will participate in the cloud as a function of time to probabilistically ensure that we will always have the needed resources at the right time to host the requested tasks.

Unfortunately, the mobile resources are highly isolated and non-collaborative. Even for those resources working in a networked fashion, they suffer from limited self and situation awareness, and collaboration. Additionally, given the high mobile nature of these devices, there is a large possibility of failure such that permanent connectivity may not be always available. This problem is common in dense mobile wireless networks due to traffic congestion and network failures [20]. In addition, mobile nodes cannot collaboratively contribute to form a cloud anymore if they are susceptible to failure for many reasons, e.g., being out of battery or hijacked. Existing explicit failure resolution and fault tolerance techniques were not effifective enough to guarantee safe and stable operation for many of the targeted applications limiting the usability of such mobile resources.

Consequently, there is a need for a solution that effectively and autonomically manages the high resource variations in a dynamic cloud environment while including the different main types of service offerings and satisfying the five essential characteristics of the cloud model defined by NIST [14].This solution should include autonomic collaborative components for service and resource discovery, scheduling, allocation, monitoringand forecasting to provide elastic and resilient CoCloud.

In this paper, we address the aforementioned challenges by a set of interrelated collaborative solutions (Pillars) towards an actual dynamic collaborative cloud management platform, CoCloud as shown in Fig. 1. However, we presented each of these pillars independently in previous works. Additionally, we perform new evaluations, in this paper, to study the effects of connectivity, density of nodes, reliability, and scalability and management overhead associated with the performance of the formed CoCloud.

Our contributions are to (1) provide a more comprehensive approach, that includes all previous works, which shows the overall architecture of our approach; (2) solve the main limitations of the current attempts towards realizing CoCloud against the essential characteristics of the cloud model as defined by NIST; (3) present a reference model to provide the different main types of X service offerings using our CoCloud; and (4) present our vision and descriptions of probabilistic models and artificial intelligence algorithms implemented in the Prediction Service (PS) Module that enhance the prediction accuracy of resource availability.

CoCloud enables both a new resource-infinite computing paradigm using cloud computing over stationary and mobile nodes, and a true ubiquitous on-demand cloud computing. This has the potential to liberate cloud users from being concerned about resource constraints and provides access to cloud anytime and anywhere. CoCloud provides the right resources on-demand, anytime and anywhere, to form an actual collaborative cloud formed over hybrid mobile and stationary computing resources, while providing the main cloud service delivery models (PaaS, IaaS and SaaS) [14].

CoCloud synergistically manages 1) resources to include resource harvesting, forecasting and selection, and 2) cloud services concerned with resilient cloud services to include resource provider collaboration, application execution isolation from resource layer concerns, seamless load migration, fault-tolerance, and task deployment, migration, revocation, etc. Specifically, the main novelty in this work is the developed CoCloud pillars, which are the resource and cloud management platforms discussed as follows.

CoCloud Resource Management

Global Resource Positioning System (GRPS)

Provides global mobile and stationary resource discovery and monitoring. A novel distributed spatiotemporal resource calendaring mechanism with real-time synchronization is proposed to mitigate the effect of failures occurring due to unstable connectivity and availability in the dynamic mobile environment, as well as the poor utilization of resources. This mechanism provides a dynamic real-time scheduling and tracking of idle mobile and stationary resources. This would enhance resource discovery and status tracking to provide access to the right-sized cloud resources anytime and anywhere [10].

Collaborative Autonomic Resource Management System (CARMS)

Efficient use of idle mobile resources.Our platform allows sharing of resources, among stationary and mobile devices, which enables cloud computing systems to offer much higher utilization, resulting in higher efficiency. CARMS provides system-managed cloud services such as configuration, adaptation and resilience through collaborative autonomic management of dynamic cloud resources and membership. This helps in eliminating the limited self and situation awareness and collaboration of the idle mobile resources [21].

CoCloud Cloud Management

Architecture for resilient cloud operation on dynamic mobile resources to provide stable cloud in a continuously changing operational environment. We presented a preliminary version of this architecturein [7] . Such goal is achieved by using trustworthy fine-grained virtualization and task management layer, which isolates the running application from the underlying physical resource enabling seamless execution over heterogeneous stationary and mobile resources. This prevents the service disruption due to variable resource availability. The virtualization and task management layer comprises a set of distributed powerful nodes that collaborate autonomously with resource providers to manage the virtualized application partitions.

The rest of the paper is organized as follows. In Sections 2 and 3, we give an overview of related works and CoCloud, respectively. We then detail the architecture of the proposed approach to provide resource and cloud management in a dynamic environment in Sections 4 and 5, respectively. In Section 6, we present a scenario of operation of our approach discussing an evalautuon study of the presented approach. Finally, we conclude in Section 7 and outline open research issues.

2 Related Work

The state of the art in research shows that some research work discussed the idea of forming MCC platform relaying on the hardware resources provisioned by stationary or semi-stationary devices. However, these solutions presented computational-clusters hosting certain distributed application(s) rather than generic computing-clouds. In the next subsections we will briefly discuss the latest research targeting CoClouds classified by the associated research objective:

Mobile-resource Sharing

In [4], authors proposed a preliminary design for a management framework that exploits the resources of a collection of nearby mobile devices to construct a virtual ad hoc mobile computing cluster. Therefore, it is a limited scenario that did not consider high node mobility cases where connectivity is not stable, leading to disconnections and faults. Similarly, experiments for job sharing were conducted in [11]over a stable ad-hoc network linking a user group of mobile device. Unfortunately, they shared the same limitation as the aforementioned work; they presented a computing cluster management platform with no consideration for resource mobility, heterogeneity, dynamic connectivity of resources.

Hyrax platform [5]was one of those who introduced the concept of using mobile devices as computation resources. The platform used a central server to utilize data and execute computing jobs on networks of homogeneously configured android smart phones. Hyrax did not consider the general scenario where hardware resources are heterogeneously configured and highly mobile. The system offered limited management automation, Hyrax did not consider a real users' mobility where mobile resources should be automatically and dynamically discovered, scheduled, allocated in a distributed manner largely transparent to the users.

In [12] researchers demonstrated the feasibility of exploiting the resources in mobile devices to execute work units as part of a grid and upload results to a server. However, the model of this approach was specifically developed to a single mobile hardware device or operating system. A more generic solution presented in [13], they proposed a collaborative framework that enables mobile devices to participate in executing computation-intensive tasks in a computing cluster to expand the shared resources. They focused only on task partitioning and offloading where a ratio of participation is determined depending on many factors, e.g., the system capability, the mobile device's performance, and the network state. However, the proposed framework did not consider a real mobile network environment that is relatively unstable. The work presented in [22] addresses how the mobility could enhance the performances of distributed computation and the resilience of services in computing clusters formed from mobile ad-hoc networks. Authors show that improvement can be achieved in the performance of distributed computation with even a small percentage of highly mobile nodes in highly localized networks.

Most of the previously mentioned research works [4] [11] [22] [13] did not take the rapid elasticity, the heterogeneity of pooled resources or the broad network access characteristics into considerations.

Vehicular Cloud

Exploiting the virtually unlimited power supply in vehicles , researchers in [23] [24] [25] discussed the possibility of having a MCC using a powerful on-board computers augmented with huge storage devices hosted on stationary vehicles acting as networked computing centers on wheels. Given the limited mobility of such solutions, we do not consider them as a realistic implementation of a generic computing cloud. Additionally, the proposed approaches only focused on one delivery service model, the IaaS and provided a virtual environment to satisfy specific client application. In [26], authors presented an overview about a datacenter architecture for the management of physical resources of vehicular nodes. However, this scenario is considered as a stable environment, such that the long-term parking lot of an international airport guarantees that there are at least a specific number of vehicles parked in the airport at any time and ready for utilization. Therefore, this solution does not provide the rapid elasticity characteristic. In addition, no solutions were provided for dynamic environments with neither heterogeneous resources nor task scheduling and resource allocation.

Limitations in Current Approaches

The main limitations of the current attempts towards realizing CoClouds against the essential characteristics defined by the NIST are summarized as shown in Table 1.

TABLE 1
THE MAIN LIMITATIONS OF THE CURRENT COCLOUDS AGAINST THE ESSENTIAL CHARACTERISTICS DEFINED BY THE NIST

NIST Essential Characteristics	Limitations of the current attempts towards realizing CoClouds
On-demand self-service	• Limited provisioning of computing capabilities where no global resource discovery or monitoring is available.
Broad network access	• Limited capabilities are only available over a local network. However, computing capabilities are not globally available and cannot used by heterogeneous platforms (e.g., mobile phones, tablets, laptops, and workstations).
Resource pooling	• Execution is limited to distributed applications built to execute on the targeted static platform. • Resource sharing profile is limited, where resources were shared among tasks built to execute on it. • No virtualization layer and no isolation between the physical resource, the data, and the task logic. • Coarse grain sharing and task execution. • Static task assignment, where no tailoring to the task size to match the resources.
Rapid elasticity	• Provisioning of limited resource pool while giving the illusion of infinite resource availability. • Limited failure resilience leads to unreliable execution.
Measured service	• Limited task mobility leads to limited load balancing. • Poor resource utilization.

3 CoCloud Overview

Figures 1 through6 depict the concept and overall design of CoCloud. CoCloud achieves the five essential characteristics listed by NIST and provides the main service models (PaaS, IaaS, and SaaS).

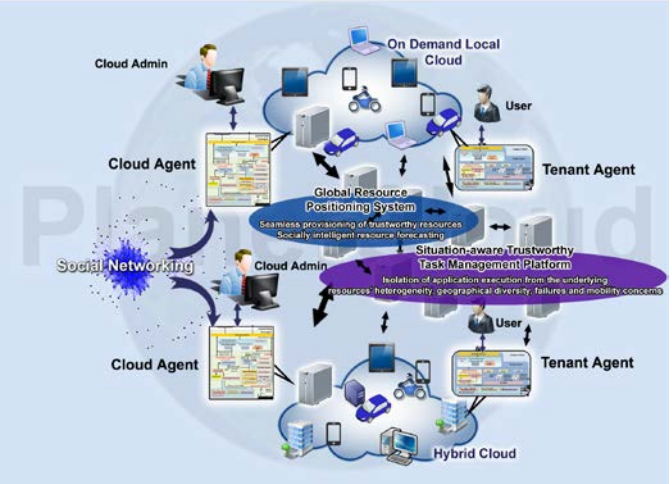

Fig. 1. CoCloud Concept.

The GRPS provides both the broad network access and the measured service characteristics of a cloud model. To achieve these characteristics, GRPS adopts a novel spatiotemporal calendaring mechanism with real-time synchronization to provide a dynamic real-time scheduling and tracking of idle, mobile and stationary, resources. In addition, the GRPS provides a hierarchical zone architecture with a synchronization protocol between different levels of zones to provide the broad network access characteristic and to enable resource-infinite computing.

Both the on-demand self-service and the resource pooling characteristics are provided by CARMS, which maps applications' requirements to the currently or potentially available mobile resources. This would support formed cloud stability in a dynamic resource environment.

On the other hand, the rapid elasticity characteristic is provided by our trustworthy dynamic virtualization and task management layer, which isolates the hardware concern from the task management. Such isolation empowered CoCloud to support autonomous task deployment/execution, dynamic adaptive resource allocation, seamless task migration and automated failure recovery for services running in a continuously changing unstable operational environment. CoCloud platform enhances service resilience against failures via multi-mode recovery and real-time, context and situation-aware adjustment of shuffling and recovery policies.

Our Virtualization and task Management Layer (VML) is an adapted version of CyberX proposed in [27] [28], which is a biologically inspired Cell Oriented Architecture (COA) [27] based platform with active components, termed Cells. Cells support development, deployment, execution, maintenance, and evolution of software. Also, Cells separate logic, state and physical resource management. Cells are realized in the form of intelligent capsules or micro virtual machines that encapsulates executable application-partitions defined as code variants. Applications can be defined in one or more Cell-encapsulated variants. Generic Cells are generated by the host middleware termed COA-Cell-DNA (CCDNA). The virtualization and task management layer dynamically composes such Cells into larger forms representing the full application. Such construction facilitates hiding the heterogeneity of the underlying hardware resources from the application concern enabling seamless deployment, distribution, and migration of application on the cloud mobile nodes.

CoCloud has a portable access application namely, iCloud App interface, which achieves the broad network access characteristic to provide seamless access to and provisioning of resources.

CoCloud comprises two primary types of nodes, as shown in Fig. 2: a fixed control node, and a mobile compute node. Each type of node has an agent running on it, where we propose a hierarchical model based on the concept of an agent as the fundamental building block of our management platform. There are two types of agents: a Cloud Agent (CA), which runs on a fixed control node, and a Tenant Agent (TA), which runs on a mobile compute node. The TA manages the participant's local spatiotemporal resource calendar. It connects with all other agents involved in the cloud formations, and synchronizes the calendar's content with the global spatiotemporal resource calendar on a CA. A CA, as a requester to form a cloud, manages the formed cloud by keeping track of all the resources joining its cloud. The CA is deployed on a high capability node to manage and store the data related to spatiotemporal calendars for all participants within a cloud.

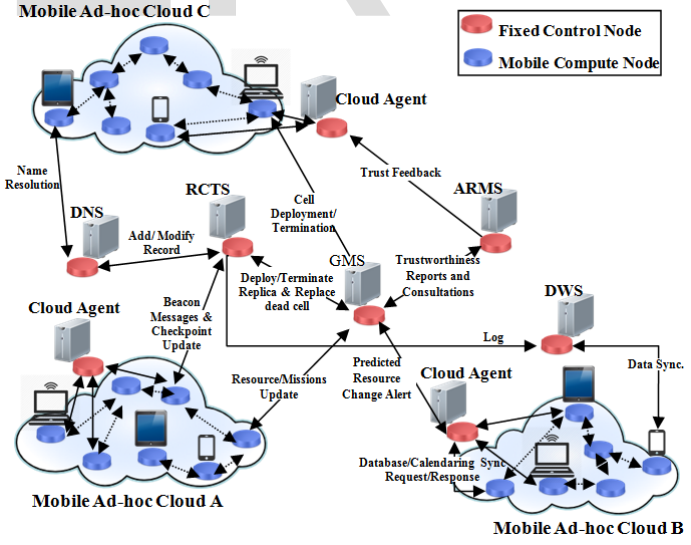

Figure 2. CoCloud Abstract Overview.

Our CoCloud management platform handles all the tasks related to both the Resource Domain concerned with the otemporal resource allocation, and the Task Domain concerned with the task deployment, migration, revocation, etc. as shown in Fig. 3. The next sections provide more details about the two domains.

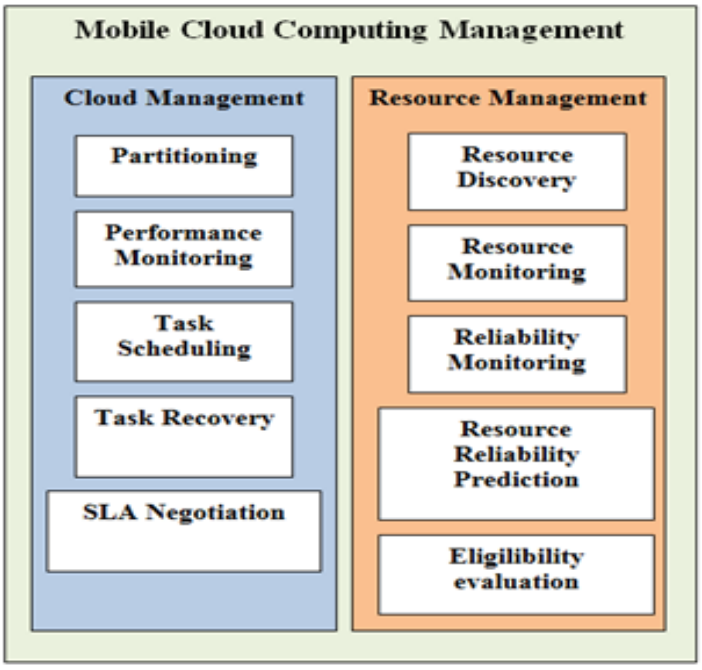

Figure 3. CoCloud Management of Mobile Cloud Computing.

Cloud Reference Model

The left part of Fig. 4 shows the architecture framework of a fixed computing cloud [29], which is proposed by the Cloud Security Alliance. At the two bottom layers, the physical facility and the computing hardware form the most basic computing unit. Since a cloud service provider pools together a vast amount of computation resources that may use different hardware, the computation ability of a set of hardware should be able to be abstracted and each set of hardware must be able to communicate with other's hardware. The facility, hardware, abstraction, and connectivity together form a computing grid that supports any additional service provided by a computing cloud. To enable a client or another cloud to manage and interact with a set of hardware, an API is implemented above the connectivity and abstraction layers. The computing grid together with the API can provide IaaS. A cloud computing service provider can also implement middleware capabilities on which clients can develop software. The infrastructure together with the middleware resembles a platform on which common programming languages and tools can be supported; that is, a cloud provider provides PaaS by overtaking the management task of the infrastructure in the middleware. The cloud provider can then further provide tailored software, content, and their presentation based on the provided platform. This delivery of "the entire user experience" is known as providing SaaS.

Ontology

It is important to note that commercial cloud providers may not neatly fit into the layered service models. Nevertheless, the reference model is important for relating real-world services to an architectural framework and understanding the resources and services requiring security analysis. Our CoCloud platform includes the different main types of X service offerings, as shown in Fig. 4, SaaS, PaaS, andIaaS. In this subsection, we present a simple analogy between CoCloud model and the conventional cloud reference model as follows.

- Realizing IaaS, which includes the entire infrastructure resource stack, on CoCloud is a complicated task as the entire management platform and hypervisor layer should be working through the virtualization and task management layer. The only piece of software that will be statically available on the host is the CCDNA. The hypervisor layer will be operating above the CCDNA where its entire set of services fractionalized and encapsulated in Cells. This model suits the mobile and resource constrained and fractionalized nature of CoCloud resources. We can represent it from a different perspective if we used a complicated version of the Cell built to provision all the features of the regular hypervisor. However, such Cell will not have many of the useful features that the fine-grained Cell had such as, low cost of failure, fast recovery, and resource efficient execution. Additionally, this representation will limit the number of hosts that can cooperate with CoCloud to those hosts with massive computational power. Ultimately, IaaS should provide a set of APIs, which allow management and other forms of interaction with the infrastructure by consumers with our model these API will be encapsulated as Cells too.
- CoCloud provides the PaaS model by virtualizing application development frameworks, middleware capabilities, and functions such as database, messaging, and queuing, and encapsulating it in Cells of the virtualization and task management layer. The CCDNA will be hosting such services or part of these services, while the virtualization and task management layer will seamlessly consolidate these parts emulating the natural behavior of these services similar to the conventional model.
- We presented the SaaS model in details, as this is the simplest model to build in our case given that the services will be built to suit service-execution model of our virtualization and task management layer. The CCDNA represents a static middleware installed on all the hosts facilitating Cell execution, and the software services are encapsulated as a number of Cells operating above it. All the operation management, Cell deployment, revocation, failure recovery, and other management tasks will be autonomously handled by the CoCloud cloud management platform, the virtualization and task management layer, with no involvement from the user or the cloud operator.

In this paper, we only focus on the SaaS model as tasks are uniform where it is much easier to represent. We build tasks as partitions that are deployed on ready cells, where users interact with the application not the infrastructure.

4 Cloud Resource Management

4.1 Resource Management at Compute Node

Fig. 5 depicts the building blocks of a Compute Node. Resource management components of the compute node are detailed as follows.

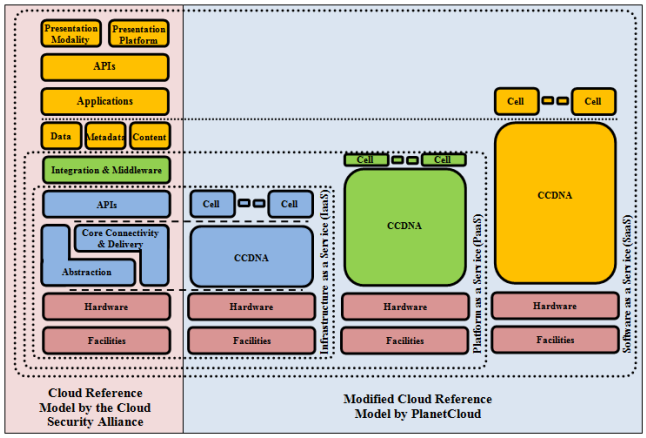

Figure 4. Providing service models by using PlanetCloud

The iCloud interface: It is an interface between the agent and a user/ administrator, or other systems, e.g., social networks and other database systems. A user/ administrator uses the iCloud interface to manage all data in the spatiotemporal resource calendar. In addition, the interface enables defining the settings required for a formed cloud.

The knowledge unit: It consists of two subunits, a local spatiotemporal resource calendar, which includes spatial and temporal information about the available resources, and information bases, that contains predefined or on the fly policies created by a cloud admin. Also, information bases contain information about the formed cloud, e.g., Service Level Agreement (SLA), types of resources needed, amount of each resource type needed, and billing plan for the service, etc.

Participant Resource Calendaring Service (PRCS): PRCS includes a Participant Calendar Manager (PCM) which acts as a service controller for managing the records of the local spatiotemporal resource calendar. PCM interacts with the synchronizer to synchronize the spatiotemporal resource calendar with other GRCSs. Also, PCM automatically monitors the internal state of the participant's resources. On the other hand, PRCS provides the trust management services with the required data to perform trust and security operations.

The Input/Output (I/O) unit: It provides the required communications for different activities such as cloud formation requests and responses.

The lowest layer, of the TA's building blocks, consists of the application, networking, and computing resources, which are involved in the delivery of the service.

4.2 Resource Management at Control Node

The main building blocks of a Control Node are shown in Fig. 6. The functionalities of their resource management are described below.

The knowledge unit: A CA has a global spatiotemporal resource calendar which includes spatial and temporal information, resource profiles, and event calendars of the all available resources of a cloud's participants. Therefore, the CA maintains the overall picture of the resource capability within the cloud. The CA uses a global task repository to store the all tasks within a cloud.

Group Resource Calendaring Service (GRCS): Distributed GRCSs operate on the updated data from participants' calendars. These updated data are stored in a group spatiotemporal resource calendar. GRCS and PRCS are the two primary types of services forming a global resource positioning system (GRPS) [10], for dynamic real-time resource harvesting, scheduling, tracking and forecasting. GRCS comprises four types of modules:The Group Calendar Manager (GCM) module, the Synchronizer, the Prediction Service (PS), and the Trust Management Services.

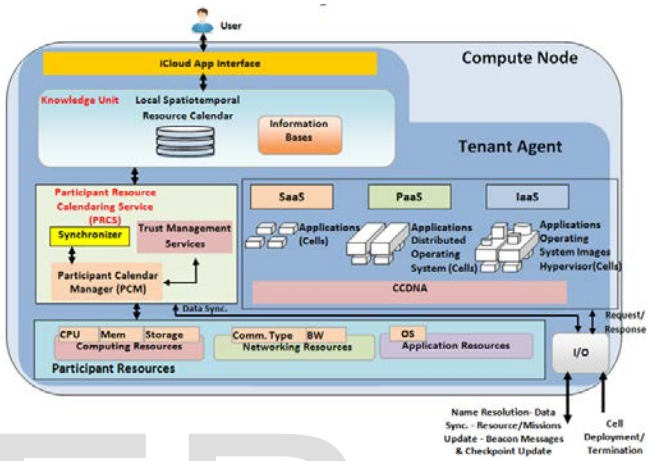

Figure 5. Compute Node Building Blocks.

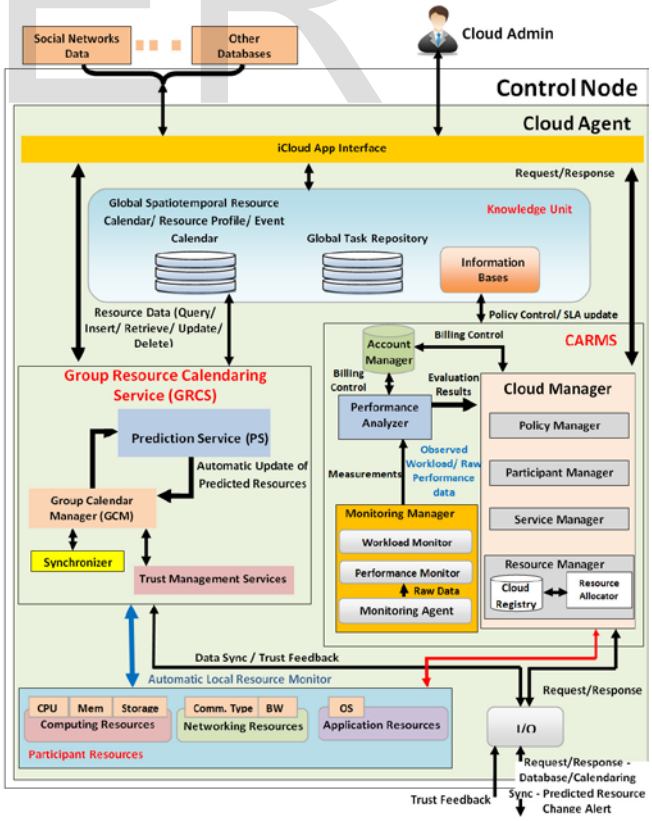

Figure 6. Control Node Building Blocks.

GCM acts as a service controller for managing records of

group spatiotemporal resource calendars. In addition, a calendar manager feeds the PS with the required data to perform resource forecasting. The following subsection will discuss the construction and operation of PS.

The Prediction Service (PS) Module: The PS module employs probabilistic models and artificial intelligence algorithms for forecasting accurately resources clarifying the time they will be available and their expected locations. In addition, the PS learns and maintains rules that aid it in classifying the behavior of resources at the various cloud participants. Accordingly, this will help in having the capability of right resource assignment and achieving reliable collaborative cloud computing

As discussed before, each cloud participant has characteristic data or data attributes that are stored locally and globally at CAs to define a set of features associated to participants. These features can be learnt according to the values of participants' attributes, such as available resources, speed, and location. Those features might be time-based ones and can be expressed as a function of time. The mobility feature of a participant which depends on its speed is an example of these time-based features. Also, some features are time independent, such as the participant's RAM and number of processing cores.

Learning participants' features by PS will aid in knowing the available resources at participant and estimating the behavior of each cloud participant with respect to those resources. For example, the PS can learn that high speed participants cannot offer long-term data storage. Consequently, the availability of heterogeneous participants' resources can be determined effectively at each participant.

For efficient resource forecasting, the PS operation is executed every reasoning window size, in time units, which is set by the GCM. The window size value is dynamic and altered by the GCM according to the frequency of received resources' data updates and the size of modified participants' attributes at the CA. The more recent changes in participants' attributes are received at the CA, the more likely to have changes in the resource forecasting plan. Through each window size, the PS performs the following phases:

The PS learns the more relevant set of attributes of each participant via employing association rule learning (ARL) algorithm [30] [29]. In this phase, the PS investigates the more frequent attributes per each participant where these attributes will refer to the offered resources, characteristics of this participant, such as mobility and so on. As an example, the ARL algorithm might learn the following attributes for a participant: processor, cores, speed, storage, RAM, and operating system.

Then, the PS can classify the learnt attributes of all participants using probabilistic decision models that are built using Fuzzy-based weighted binary decision trees [31]. The probabilistic decision models adopt Fuzzy classifiers, which names the possible output classes using defined Fuzzy membership functions (FMFs). Those FMFs describe the classification thresholds on which each attribute will be classified to a specific class out of a set of output classes. For each registered participant's resource, we will able to recognize the main classified attributes, called features, related to that resource. Then, Entropy-based binary decision trees are used by PS to select the more likely detected feature class at each tree level, or resource attribute, and to form a typical sequence of relevant weighted features related to the studied cloud participant's resource. For each feature, we have two possible output classes. The binary information entropy H_2 (F)about a feature F class is defined in (1).

$$H_2(F) \quad p_{c0}\log_2 p_{c0} \quad p_{c1}\log_2 p_{c1} \qquad (1)$$

Where p_{c0} is the probability of classifying the feature in the first class; and $p_{c1}$is for the other class.

The constructed decision models are dynamic that the used classifiers of embedded weighted binary decision trees are trained based on the stored data at the knowledge units of CAs through a reasoning window. Also, the decision models depend on the set of discovered features per participant's resource in the cloud. Also, the models have categories which are formed according to the resource type and can be applied to every participant which possesses the same resource type. Each feature is considered as a binary random variable (RV) that it can take two values (e.g., numerical, string, etc.). The feature is considered as an independent RV with respect to other features, or RVs. The decision tree per resource consists of parent features and child features at various levels.

For instance, if we have a mobile resource at a cloud participant with ID equals 10 that this resource has the speed attribute which is registered in the knowledge unit through a reasoning window of length 24 hours. According to the defined FMF, the speed attribute might be classified to high speed or low speed features. The registered data show that resource has 87% of its records with high speed and 13% of records with low speed. This means that 87% of the time all over the day, the participant's resource is with high speed. So, the uncertainty about this information (i.e., classifying the resource feature as high speed) can be calculated by the entropy $H_2(F)$ for is approximately 0.56. Since $0 \leq H_2(F) \leq 1$, the more $H_2(F)$ value, the more uncertainty we will have about the discovered information. So, as $H_2(F)$ nears zero, the learnt feature is of high certainty. So, if $H_2(F) < 1$ and $p_c \geq 0.5$, then the feature will classified to class c.

At the final decision tree level, we will reach the more likely features' sequence that can exist for the related studied resource. The total information entropy concerning sequence of independent features of a resource will be the summation of all binary entropy of each individual feature. Equation (2) shows the entropy of resource's independent feature sequence of length n.

$$H(F_{sequence}^{\ n}) = H(F_1, F_2, \dots, F_n) = H_2(F_1) + H_2(F_2) + \dots + H_2(F_n) \qquad (2)$$

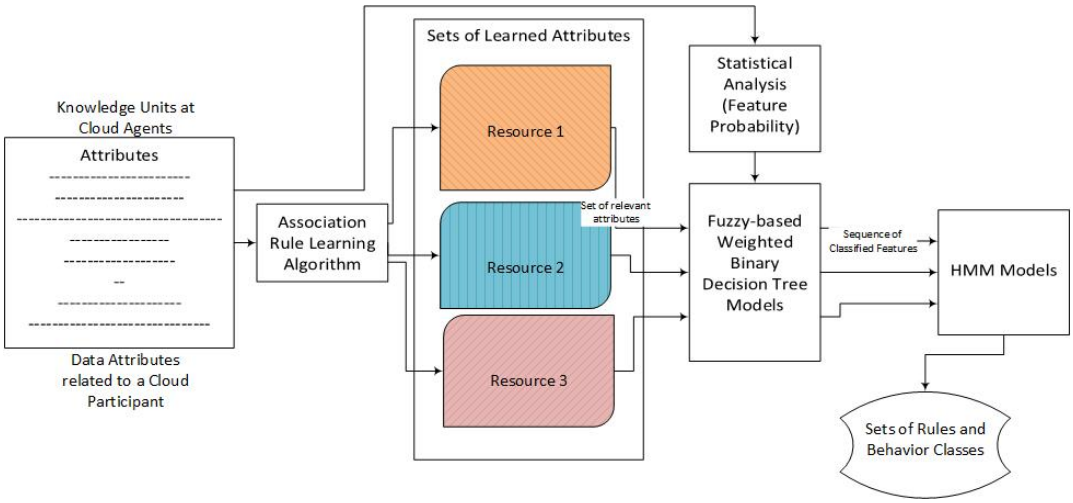

Figure 7. The main operations of PS.

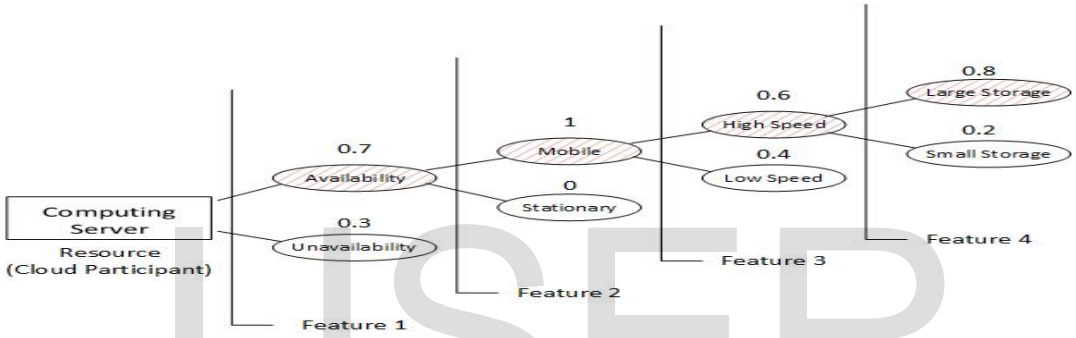

Figure 8. Fuzzy-based weighted binary decision tree for classifying computing resource.

From (2), we can estimate the upper bound for $H(F_{sequence}{}^{n})$ which will be $nH_{2,max}(F)$ which equals n where $H_{2,max}(F) = 1$. We set a limit for learning and storing rules related to studied resources and their related features that the PS will keep the learnt feature sequence as a rule if $H(F_{sequence}{}^{n}) \leq n/2$. Actually, we will have undoubtedly true classification and defined rule if $H(F_{sequence}{}^{n}) = 0$.

- The PS adopts trained hidden Markov models [**32**] to estimate the behavior of participants' resources according to the input typical sequence of features, or input states, learnt from the previous phase. The estimation process for outputs relies on continuous input sequence with different Gaussian distributions. HMM is trained using unsupervised learning algorithm called Baum-Welch algorithm. Then, HMM performs distribution mixture for obtaining the most likelihood output sequence. From the most likelihood output sequence, the PS can generate a set of rules that define the behavior of resources at all participants taking into consideration the condition that $H(F_{sequence}{}^{n}) \leq n/2$}.

The HMM is defined with the following parameters (π , A, B) which are:

- Initial state probability π vector: this defines the initial probability of each classified feature to be the first input feature in the input sequence to the HMM.
- State transition probabilities **A** matrix: the matrix maintains A_{ij} elements where each element defines the transition probability form classified feature *i* to another feature *j*.
- The observation probability ***B*** matrix: this matrix relates each input feature in an input sequence with a possible output through a defined probability.

Fig. 7 illustrates the design and operation of the PS showing its main building blocks, which employ the algorithms discussed before. The figure shows that PS learns behavior and rules related to three different resource types located at a cloud participant based on its maintained data at the knowledge units at the cloud.

The following subsection will discuss an example which clarifies the operation of PS.

Example

IJSER

Fig. 8 shows the classified features for a computing server at a cloud participant based on using the Fuzzy-based weighted binary decision tree model.

Calculating the total entropy of feature sequence of length 4, $H(F_{sequence}^{4})$, according to (2) results in a value of 2.573 where
$H_2(F_1) = 0.881, H_2(F_2) = 0, H_2(F_3) = 0.9709, H_2(F_4) = 0.72188$

Then, this feature sequence is input to a trained HMM that it has two main output resource behaviors which are "reliable resource" and "unreliable resource". According to the defined B matrix, all the four classified features have higher probability related to the first behavior than the one related to second behavior. Here is an example of the B matrix that B: {{0.8, 0.2}, {0.7, 0.3}, {0.65, 0.35}, {0.9, 0.1}} where it consists of four rows r (i.e., number of features) and two columns c (i.e., number of behavior classes)

Since, the total entropy $H(F_{sequence}^{4})$< n/2 (i.e., 2), then the PS will not set a rule for this participant that his server is a reliable one. We can remark here that although the HMM output a classified behavior for the feature sequence, the PS does not accept this output as a rule because it finds to some extent high uncertainty about this information.

Collaborative Autonomic Resource Management System (CARMS): We design our CARMS architecture using the key features, concepts and principles of autonomic computing systems to automatically manage resource allocation and task scheduling to affect cloud computing in a dynamic mobile environment.

Cloud Manager (CM): It provides a self-controlled operation to automatically take appropriate actions according to the results of the evaluation received from the Performance Analyzer, described below. The CM manages interactions to form, maintain and disassemble a cloud. A CM comprises four components, a Service Manager (SM), a Resource Manager (RM), a Policy Manager (PoM), and a Participant Manager (PrM). A SM stores the request and its identifier. The SM maps the responses received from the participants with the service requests from users, and the result is sent back directly to the user. The CM decomposes the requested service, upon receiving a cloud formation request, to a set of tasks. Tasks of a requested service need to be allocated to mobile resources. The RM handles the resource allocation on mobile nodes using its Resource Allocator component. Also, the Resource Allocator obtains the required information about the available resources by interacting with a GRCS. The Resource Allocator interacts with the registry of CA to store and retrieve the periodically updated data related to all participants within a cloud. The CM interacts with CyberX servers to assign a set of virtual resources in cells to these tasks according to the received SLA information from the CM. The PoM prevents conflicts and inconsistency when policies are updated due to changes in the demands of a cloud. The PrM manages the interaction between a cloud requester and resource providers, the cloud participants, to perform a SLA negotiation.

Monitoring Manager: It includes a Performance Monitor unit which monitors the performance measured by monitoring agents. Then, it provides the results of these measurements to the Performance Analyzer component. The workload information about the incoming request is periodically collected by the Workload Monitor component.

Performance Analyzer: It continually analyzes the measurements received from the Monitoring Manager to detect the status of tasks and operations, and evaluate both the performance and SLA. The results are then sent to both the Cloud Manager and the Account Manager.

Account Manager: In case of violation of SLA, adjustments are needed to the bill of a particular participant. These adjustments are performed by the Account Manager component depending on the billing policies negotiated by the requester of cloud formation.

5 CloudVirtualization and task Management

CoCloud uses the trustworthy dynamic virtualization and task management layer to manage the cloud tasks and the running applications on the cloud. This layer virtualizes the cloud resources creating a suitable execution environment for the applications.

The Virtualization and task Management Layer (VML) uses the COA [27] features to enable applications to dynamically adapt to runtime changes in their execution environment. Such feature enables the virtualization and task management layer to tolerate high frequency task preemption and migration that might be induced by failures as a consequence of unexpected resource mobility or power failure. Due to the nature of our resources the high level of heterogeneity is a major concern for task deployment and migration. Using such vitalization architecture adequately resolves this issue.

Figure 9 shows the virtualization and task encapsulation process in CyberX based virtualization management.

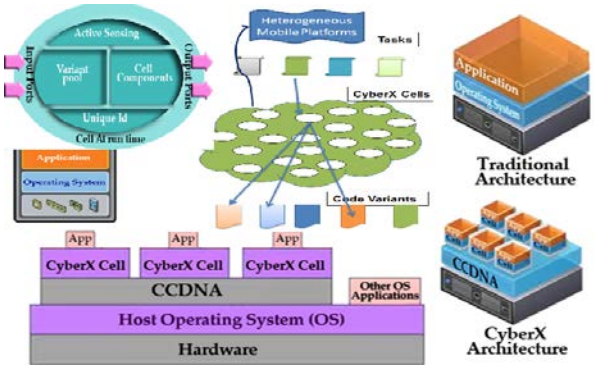

Figure 9. The CyberX based virtulization manament

VML enables the application to exchange real-time status and recommendation messages with the host Cell for administrative purposes to enhance the Cell local application awareness and to enable application driven adaptation. The virtualization and task management layer uses these messages to guide the Cell runtime quality-attribute manipulation towards accu-

rate and prompt adaptation. Further, the virtualization and task management layer collects, analyzes and trustworthy-share these messages and status reports, constructing a real-time sharable global view of the Cell network.

VML enhances the system resilience by multiple recovery modes to cover different application-requirements and host-configurations. The virtualization and task management layer offers a prompt and accurate fine-grained recovery, hot-recovery, for resourceful hosts executing critical applications, and a more resource efficient course-grained recovery, cold-recovery, for less critical applications. In hot-recovery, the Cell can have one or more fully-alive replicas on different mobile nodes which can do achieve virtually no task failure down-time but on the account of increasing resource usage. The cold-recovery might save some of the resources used by replicas, by deploying a replacement of the failed Cell, while compromising some of the execution states, and increasing the failure downtime. The virtualization and task management layer uses the COA loosely coupled features to allow applications to seamlessly change their current active recovery modes based on context, environment, or application-objective change.

VML layer is composed of a set of central powerful nodes or servers. These servers collaborate autonomously to manage the whole network of Cells. This platform is responsible for the organism creation "composition and deployment of Cells", management, the host side API(s) "CCDNA", real-time monitoring and evaluation of the executing Cells, and recovery management. Further, it provides the necessary management tools for system administrators to manage, analyze, and evaluate the working Cells/organisms. Figure 10 shows the VML architecture. The next subsections will briefly describe the main components. More detailed description can be found at [26]

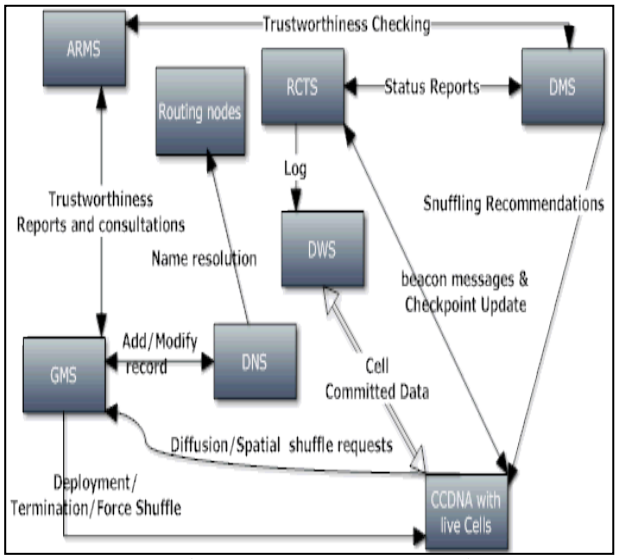

Figure 10. Virtualization management architecture

Cloud Management at Control Node

All related cloud management tasksare performed on fixed powerful control nodes. The following describes the main functionality of the main components running on such nodes:

Auditing and Reputation Management Server (ARMS):Its main task is to monitor outgoing or incoming Cell administrative messages for the lifetime of the Cell. This information is used to assist evaluating the trustworthiness of the Cell. This server cooperates with the recovery tracking servers and routing nodes to frequently evaluate the Cell behavior for any malicious activities. This server will hold comprehensive reports about each Cell for the lifetime of the Cell. A trust feedback will be generated from ARMS and send to the Trust Management Services which helps in the evaluation of the trustworthiness of a participant.

Recovery and Checkpoint Tracking Server (RCTS):This monitors, and stores checkpoints changes for all running Cells. Checkpoint updates are always enclosed as a part of the Cell frequent beacon message updates. RCTS is also responsible for reporting failure events by comparing the duration between consecutive beacon messages to a certain threshold matching the reporting frequency settings of each Cell. Failure events are validated by comparing the recently noticed reporting-delay for a particular Cell to the average reporting-delay within its neighbors and other Cells hosted on the same host. A Cell failure notice is reported to the global management servers with the last known failure recovery settings, checkpoint, and variant settings to start deploying replacement Cells.

Global Management Server (GMS):The main task of this server is to manage the underlying COA infrastructure. GMS is responsible for Cell deployment, coordinating between servers, facilitating and providing a platform for administrative control. GMS is the only server authorized of issuing Cell termination signals. It can also force Cell migration or change the current active recovery policy when needed. GMS is responsible for assigning the infrastructure global policy, routing protocol, auditing granularity, registering/revoking new hosts, and keeping/adjusting the host-platform configuration file.

The Data-Warehouse Server (DWS):It is the main components of the infrastructure that participates in the separation between the Data, Logic, and Physical-resources. DWSs are distributed through the Cell network, they are responsible for holding and maintaining all the data being processed, and any other sensitive data that the management units want to store. All running Cells are not permitted to store sensitive data on their local memory. All sensitive data has to be remotely stored in a specific DWS through the dedicated data channel. DWSs synchronizes their data independently.

Distributed Naming Server (DNS):It is responsible for resolving the real host IP/Port mapping to the virtual Cell Id and organism names. The working Cells use this mapping at runtime to direct incoming and outgoing communications. DNS is a major player in the COA's separation of concerns that enables virtually seamless, Cell relocation, and workload transition in case of failure recovery. In case of Cell movement, the DNS will be instructed by the GMS to maintain communication redirection.

Cloud Management at Compute Node

GMS uses the resource-forecasting database to allocate resources for the Cells, of the virtualization and task manage-

ment layer, to be deployed on the Compute Node. The SM updates the task repository by the tasks that should be executed, and the code variants associated with it. The GMS encapsulates these variants into one of its Cells forming a suitable container that matches one of the available resources. The selected resource will be the target of the Cell deployment where the CCDNA is installed. That resource shall accept the deployment package from the GMS, instantiate and execute the Cell.
In case of failure, or unavailability, the GMS will relocate the Cells into new active resource seamlessly. All the concerns that might be involved with the task relocation will be autonomously and seamlessly handled by the virtualization and task management layer. The details of task relocation, recovery in case of failure or performance tuning using diversity employment, which has been addressed in [27], is omitted from this paper due to space limitation.

6 Evaluation

6.1 Working Scenario

For evaluation purposes, we present a scenario of dynamic resources in different-sized hospital models with 25, 50, 75, and 100 beds, respectively. The model involves different types of mobile devices such as Smartphones and Laptop Computers and semi-stationary devices such as on-board computing resources of vehicles in a long-term parking lot at a hospital. Such rather large pool of idle computing resources can serve as the basis of a CoCloud as a networked computing center. We start our evaluation by predicting the average number of participants in this scenario at hospitals of different sizes, which reflects the amount of computing resources that might participate in a CoCloud. Then, we perform evaluations, using the obtained average number of participants, to study the effects associated with the performance of the formed CoCloud.

6.2 Expected Number of Participants in a Resource Pool

Patients arrive at a time dependent rate $\lambda_T(t)$, independent of the number of participants already participating in the resource pool at the hospital. The departure rate of participants is $\mu_T(t)$. Further, we assume that for all $t \geq 0$, $\lambda_T(t)$ and $\mu_T(t)$ are bounded by the constants M_1, m_1 , M_2, m_2, where ($0 < m_1$; $0 < m_2$) such that

$$m_1 \leq \lambda_T(t) \leq M_1; \quad m_2 \leq \mu_T(t) \leq M_2 \tag{3}$$

Consider the event {N(t) = k} occurs if the resource pool at the hospital contains k patients at time t, where ($1 \leq k \leq N$). The probability that the event {N(t) = k} occurs is $P_k(t)$.

$$P_k(t) = Pr\,[\{N(t) = k\}] \tag{4}$$

We consider the general case where $\lambda_T(t)$ and $\mu_T(t)$ are integrable functions as in [*26*]. So that if the expected number, $E[N_T(t)]$, of patients in the hospital at time t converges, the limiting behavior of $E[N_T(t)]$ as $t \to \infty$ can be written as

$$\lim_{t\to\infty} E[N_T(t)] = \lim_{t\to\infty} (\lambda_T(t)/\mu_T(t)) \tag{5}$$

Where,

$$E[N_T(t)] = p(t)\,[n_0 + \int_0^t \lambda_T(u)\, e^{\int_0^u \mu_T(s)ds}\, du] \tag{6}$$

Where n_0 is the number of patients in the hospital at t=0. The success probability, p(t) , is given by

$$p(t) = e^{-\int_0^t \mu_T(u)du} \tag{7}$$

Patients arrival, $\lambda_T(t)$, and departure, $\mu_T(t)$, rates into/from the hospital are periodic functions of time, and can be obtained as following:

$$\lambda_T(t) = a + b \sin \theta\,(t) \tag{8}$$

$$\mu_T(t) = c + d \sin \theta\,(t) \tag{9}$$

Where a, b, c, and d are constants.

We can use the previous equations to get the expected number of cars, $E[N_c(t)]$, in the parking lot at time t, where a relationship do exist between traffic and the number of arriving/departing patients. Therefore, we can model the expected number of cars as a percentage factor, v, using the following cars arrival, $\lambda_C(t)$, and departure, $\mu_C(t)$, rates

$$\lambda_C(t) = v^*\lambda_T(t) \tag{10}$$

$$\mu_C(t) = x + y \sin \theta\,(t) \tag{11}$$

Similarly, we can calculate $E[N_c(t)]$ and we set the number of patients' cars in the hospital at t=0 to be equal v*n0. The limiting behavior of $E[N_c(t)]$ as $t \to \infty$ can be written as

$$\lim_{t\to\infty} E[N_c(t)] = \lim_{t\to\infty} (\lambda_C(t)/\mu_C(t)) \tag{12}$$

Let $E[N_m(t)]$ be the expected number of patients' mobile nodes, in the hospital at time t, where each patient holds a mobile node. This allows us to write

$$E[N_m(t)] = E[N_T(t)] \tag{13}$$

In addition, we consider the resources of the hospital's employees as valuable participants to the formed cloud. Such resources may include computational power of the employees' mobile devices as well as on-board computing resources of employees' cars in the employee parking lots at the hospital. We set the expected number of employees, $E[N_e(t)]$, to be

$$E[N_e(t)] = E_{min} \tag{14}$$

Where, E_{min} is the minimum number of employees that should be located in the hospital in their regularly scheduled shifts.

Similarly, we set the expected number of employee cars, $E[N_{ec}(t)]$, as a percentage factor, f, of the number of employees. We can write

$$E[N_{ec}(t)] = f^*E[N_e(t)] \tag{15}$$

The total expected number of participants, $E[N_p(t)]$, in the hospitalcan be obtained by

$$E[N_p(t)] = E[N_c(t)] + E[N_m(t)] + E[N_{ec}(t)] + E[N_e(t)] \tag{16}$$

Using the previously obtained expected number of participants, we can get the total number of available cells hosted by participants in a total resource pool.

Using the previous equations, we set the simulation time to 60 hours. We assumed that at t = 0, n0 equals35, 60, 85, 100 patients, respectively, according to the size of the hospital. Similarly, we set the number of full-time staff employed, $E_{min,}$ equals 35, 61, 94, 116 employees, respectively, according to the size of the hospital [*33*]. We set θ(t) to be πt/12 for a time unit equals one hour. Also, we use a quasi-periodic time-dependent arrival and departure rates as follows.

At hospital size equals 25 beds,

$$\lambda_T(t) = 32+16[1+2\exp(-0.2t)] \sin(\pi t/12) \quad (17)$$

$$\lambda_C(t) = 0.3^* (32+16[1+2\exp(-0.2t)] \sin(\pi t/12)) \quad (18)$$

At hospital size equals 50 beds,

$$\lambda_T(t) = 72+36[1+2\exp(-0.2t)] \sin(\pi t/12) \quad (19)$$

$$\lambda_C(t) = 0.3^* (72+36[1+2\exp(-0.2t)] \sin(\pi t/12)) \quad (20)$$

At hospital size equals 75 beds,

$$\lambda_T(t) = 112+56[1+2\exp(-0.2t)] \sin(\pi t/12) \quad (21)$$

$$\lambda_C(t) = 0.3^* (112+56[1+2\exp(-0.2t)] \sin(\pi t/12)) \quad (22)$$

At hospital size equals 100 beds,

$$\lambda_T(t) = 152+76[1+2\exp(-0.2t)] \sin(\pi t/12) \quad (23)$$

$$\lambda_C(t) = 0.3^* (152+76[1+2\exp(-0.2t)] \sin(\pi t/12)) \quad (24)$$

Where, at each hospital size

$$\mu_T(t) = 2+ [1+ \exp(-0.2t)] \sin(\pi t/12) \quad (25)$$

$$\mu_C(t) = 2+ [1+ \exp(-0.2t)] \sin(\pi t/12) \quad (26)$$

We computed the expected number of mobile nodes at time t as shown in Fig. 11 shows $E[N_P(t)]$. The expected number of mobile nodes, at each hospital size, dropped as illustrated in Fig. 7and settles down to a constant value at 51, 97, 150, and 192, respectively, after t > 20 hours of simulation. The pattern of the unstable fluctuation, before stabilization, depends on the probability of the departure of initially participating nodes and the exponential component of arrival and departure rates.

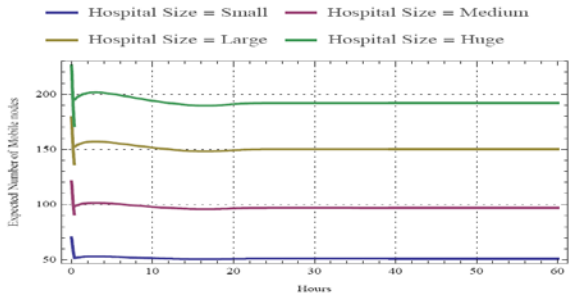

Figure 11. The expected number of mobile nodes versus time.

Similarly, Fig. 12 shows that evaluating the expected number of cars in the parking lot of the hospital stabilizes to a constant number, at each hospital size, e.g. at 19, 36, 55, and 70, respectively, after 20 hours.

Next, we turned our attention to compute the expected number of participants in the hospital versus time. Fig. 13 shows $E[N_P(t)]$ plotted against time. The expected number of participants dropped as illustrated in Fig. 9. $E[N_P(t)]$ stabilizes at 70, 133, 205, and 262, at each size of a hospital, respectively, after t > 20 hours of simulation.

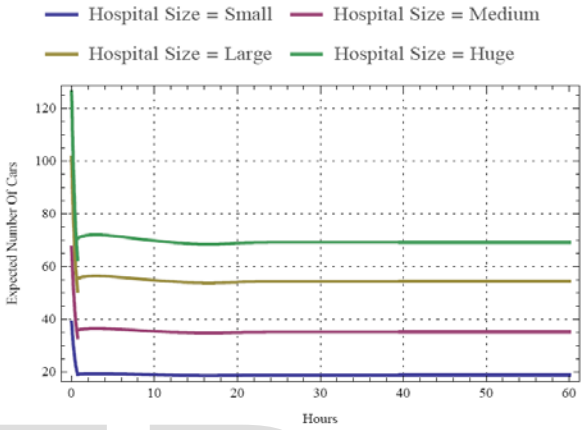

Figure 12. The expected number of cars versus time.

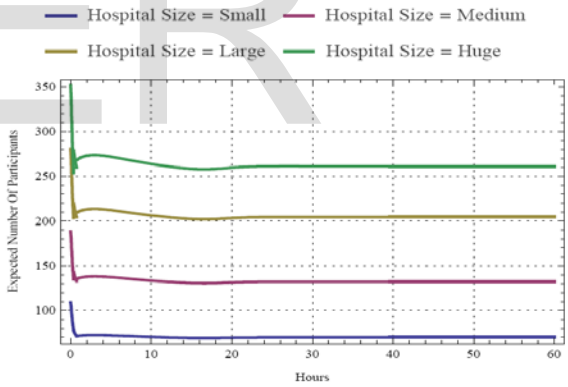

Figure 13. The expected number of participants versus time.

6.3 Simulation Platform

We choose the CloudSim toolkit [*34*] [*35*] to be our simulation platform, as it is a modern simulation framework aimed at Cloud computing environments. To simulate the CoCloud environment in hospital, we have extended CloudSim to support node mobility by incorporating the Random Waypoint (RWP) model, where a mobile node moves along a line from one waypoint W_i} to the next W_{i+1}. These waypoints are uniformly distributed over a unit square area. At the start of each leg, a random velocity is drawn from a uniform velocity distribution.

We designed Java classes for implementing the spatiotemporal data related to resources and their future availability

which is obtained from the calendaring mechanism. In addition, we edited the CloudSim to implement our proposed P-ALSALAM algorithm.

In our evaluation model, an application is a set of tasks with one primary task executed on a primary node. Each task runs in a single VM which is deployed on a participant node. VMs on participating nodes could only communicate with the VM of the primary task node and only when a direct ad-hoc connection is established between them.

Assumptions

The following assumptions are used in all simulation evaluations.

- Communication between nodes is possible within a limited maximum communication range, x (km). Within this range, the communication is assumed to be error free and instantaneous.
- For scheduling any application on a VM, First-Come, First-Served (FCFS) is followed.
- For calculating the collision delay, we consider the worst case scenario, a saturation condition, where each node has a packet to transmit in the transmission range.
- For simplicity, a primary node collects the execution results from the other tasks which are executed on other participating nodes in a cloud.
- A SaaS model is only considered in our model.

6.4 Simulation Evaluation

We start our evaluation by studying the effects associated with execution of applications in a CoCloud, consists of stationary and mobile devices, using different scheduling algorithms, .i.e., Proactive Adaptive List-based Scheduling and Allocation AlgorithM (P-ALSALAM) [*8*], which determines the best participants based on the availability of its resources to participate in a cloud and the random reliability-based algorithm, which does not use this information, where nearby mobile nodes with random availability are selected to execute the submitted application.

Participant nodes are characterized by the number of processing cores, CPU performance in terms of Millions Instructions Per Second (MIPS), amount of RAM, storage and network bandwidth.

In our evaluation model, each task has a pre-assigned instruction length and runs in a Cell. A Cell matches the smallest computational power available in any participants, which is simulated as a single virtual machine (VM) deployed on a participant. A VM can be migrated out from the participating node as the node becomes unreliable to execute a task. Migrations happen when communications are established among participating nodes.

We modify the simulation to include spatiotemporal data, future availability, obtained from the calendaring mechanism. Also, we consider that participating nodes cannot always function well all the time and may fail. In our evaluation, we only consider the cold-recovery mode in case of node failure. We set the number of inactive nodes to be sampled following a Poisson Process during a time t.

Metrics and Parameters

We use several metrics to evaluate the performance of our PlanetCloud and its subsystems as follows.

1. The average application execution time, which is the time elapsed from the application submission to the application completion.

2. The mean number of VM migrations, which is the number of VM migrations during the simulation time.

We set parameters in the simulation according to the maximum and minimum values shown in Table 2.

TABLE 2
PARAMETERS

Parameters	Values	Parameters	Values
Average Mobile Node Speed (Uniform distribution)	1.389 (m/sec)	Number of tasks/Application	10 - 70
Communication range	0.1-1 (km)	Inactive Node rate (Poisson Process)	1/45 -1/15 (Node/Sec)

Simulation Setup

We consider a CoCloud, where a CoCloud at each size of a hospital is composed of previously obtained stabilized number of mobile nodes, in Fig. 7, and stabilized number of semi-stationary cars, in Fig. 8, with heterogeneous characteristics: 512 or 1024 MB RAM, 4 GB Storage, and 54 MB bandwidth. Each node may have one or two cores with processing capabilities of 2000 or 7500 (MIPS), respectively. However, we set all cars to have the highest computing configurations. In our evaluations, we create VMs each has one processing core with processing capability 1256 MIPS and 512 MB RAM.

Results of our evaluations are collected from different simulation runs and the value of sample mean is signified with t-distribution for a 95 % confidence interval for the sample space of 30 values in each run.

In our evaluation, we consider that every car has a fixed location. We consider that every participating car can always function well all the time with high reliability and does not fail. However, we consider that the mobility pattern of mobile nodes follows a Random Waypoint (RWP) model. Also, each node has an average speed equals 1.389 (m/sec). We consider that mobile nodes are different in their reliability, in terms of

IJSER

future availability and reputation, which follow the values of the arrival rate of inactive nodes defined in Table 2.

Results

Connectivity effect at different number of tasks

The average execution time of an application is investigated at different communication ranges of stationary nodes, cars, ranging from 0.1 to 1 (km) when we consider one application is submitted to be executed, with different number of tasks, ranging from 20 to 70 tasks. We consider a small-sized hospital (25 beds) with total number of participant equals 70 (19 cars and 51 mobile nodes). Also, we consider that all nodes are reputable and each mobile node has a transmission range equals 0.4 km, and its average speed equals 1.389 (m/sec). We set the task length to be equal to 500000 MI. We perform this evaluation with an arrival rate of inactive nodes equals 1/45 (nodes/sec). Where, we consider that the effect of reliability of mobile nodes is neglected at this arrival rate of inactive nodes.

Fig. 14 depicts a comparison between the results of applying both P-ALSALAM and random reliability-based node selection algorithms in terms of the average execution time of an application at a small-sized hospital. Results show that P-ALSALAM significantly outperforms the random reliability-based node selection algorithm in terms of the average execution time of an application at all transmission ranges. Similarly, the average number of VM migrations when applying P-ALSALAM significantly is smaller than the case when applying the random reliability-based node selection algorithm as shown in Fig. 15.

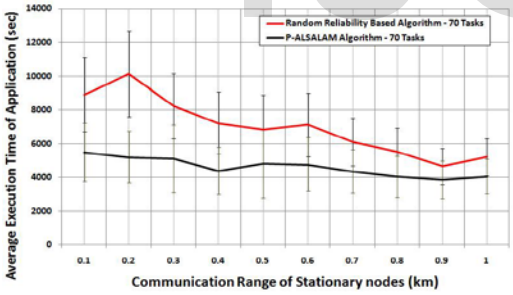

Figure 14. Average execution time of an application when applying different reliability based algorithms at a small-sized hospital (25 beds).

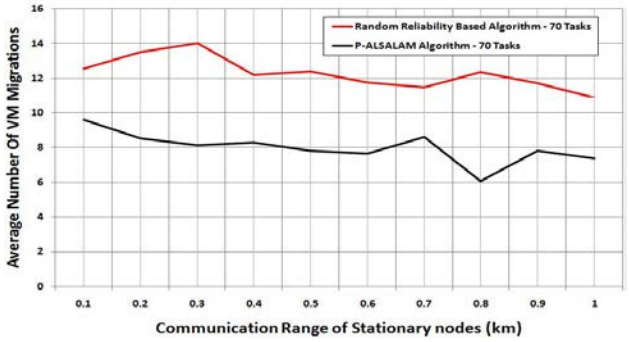

Figure 15. Average number of VM migrations when applying different reliability based algorithms at a small-sized hospital (25 beds).

The results of Fig. 16show that the average execution time of an application has a higher value at a small communication range, e.g. 0.1 (km). This is because the smaller the communication range the larger the probability to depend on mobile nodes, which may fail, as participants in a CoCloud, where the communication delay is dominant. Consequently, the average number of migrations of a VM increases at a smaller communication ranges as shown in Fig. 17. While, a better performance is obtained at higher communication ranges, e.g. 1 (km). Results shows that P-ALSALAM always has a better performance at a small number of submitted tasks.

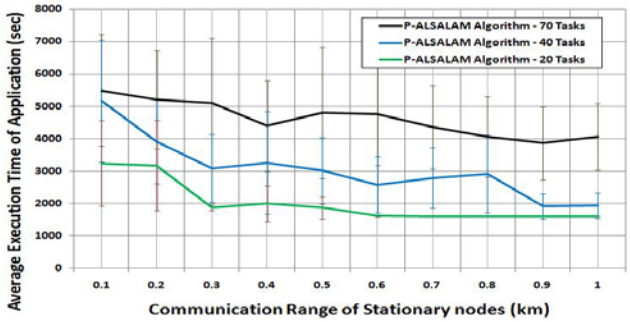

Figure 16. Average execution time of an application vs. communication range (km) when applying P-ALSALAM algorithms at a small-sized hospital at different number of submitted tasks.

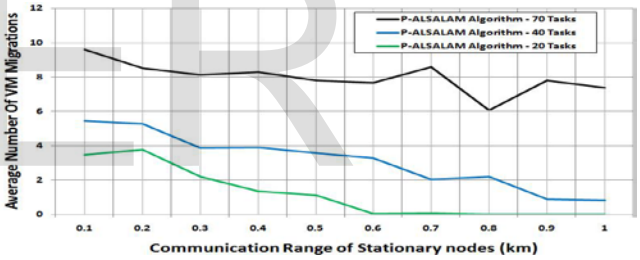

Figure 17. Average number of VM migrations vs. communication range (km) when applying P-ALSALAM algorithms at a small-sized hospital at different number of submitted tasks.

Density effect at different communication ranges of stationary nodes

We repeat the evaluation of P-ALSALAM algorithm at a different size of a hospital, i.e., small, medium, large or huge hospital which represents different node densities when we consider different communication ranges of stationary nodes, e.g. 0.2 and 1 (km), respectively. We set the arrival rate of inactive nodes to be 1/45 (Node/Sec). We consider one application is submitted to be executed, with a number of tasks equals to 70. Fig. 18 shows that a small-sized hospital with low node density, e.g. 70 (Nodes/Km^2), has a high average execution time of an application. This is because of the average number of stationary reliable cars, e.g. 19, is small when compared with a larger number of stationary cars at high density value. Conversely, a better performance is obtained at a huge-sized hospital with high node density, e.g. 262 (Nodes/Km^2), when the average number of stationary reliable cars is large, e.g. 70. This is because the performance is enhanced when our P-

ALSALAM algorithm can assign the requested tasks to a larger number of reliable resources and less depends on variable reliability mobile nodes. Similarly, Fig. 19 shows that the higher the node density the higher dependency on reliable and connected stationary nodes to execute the submitted tasks, and therefore the lower probability of VM migrations is obtained.

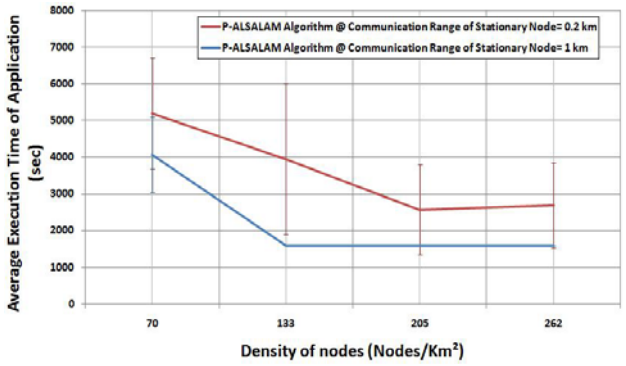

Figure 18. Average execution time of an application vs. node density (nodes/km²) when applying P-ALSALAM algorithms at different-sized hospital models at different stationary nodes' communication ranges.

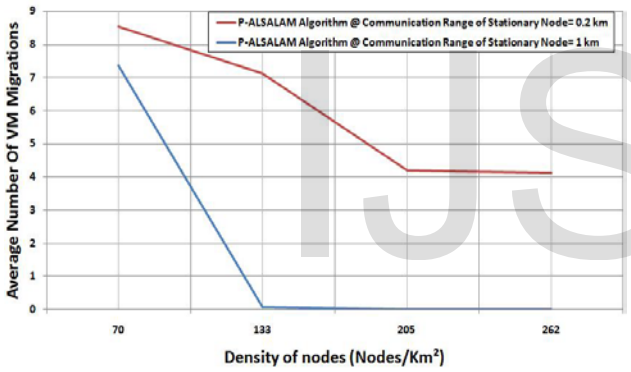

Figure 19. Average number of VM migrations vs. node density (nodes/km²) when applying P-ALSALAM algorithms at different-sized hospital models at different stationary nodes' communication ranges.

Density effect at different arrival rates of inactive nodes

Similarly, we repeat the evaluation of P-ALSALAM algorithm at a different arrival rate of inactive nodes equals 1/45 and 1/15 (Node/Sec), respectively. Fig. 29, showed that the average execution time of an application at a large density of nodes, e.g. 262 (Nodes/Km²) has a better performance than the case of a small density of node, e.g. 70 (Nodes/Km²), at the same number of tasks, e.g., 20 tasks and the same arrival rate of inactive nodes, e.g. 1/45 (Node/Sec). This is because the larger the density of nodes the more dependency on reliable stationary nodes. However, the smaller the density of nodes the more dependency on variable reliability mobile nodes that could fail, and therefore the performance may degraded due to the migration delay. Results depict that the effect of node failure may be neglected at arrival rate of inactive nodes equals 1/45 (Node/Sec).

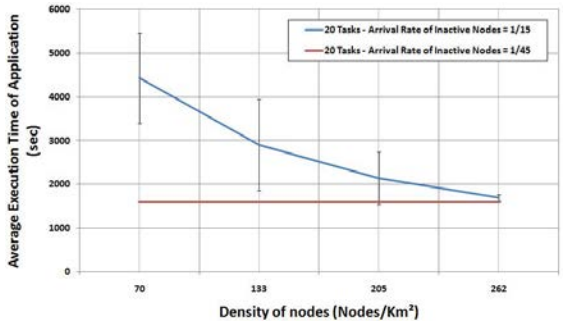

Figure 20. Average execution time of an application vs. node density (nodes/km²) when applying P-ALSALAM algorithms at different-sized hospital models at different arrival rates of inactive nodes.

Reliability Effect

In this evaluation, we evaluated the average execution time of an application and the mean number of VM migrations at a small-sized hospital with node density equals 70 (Nodes/Km²) at a high load of submitted tasks, e.g. 70 tasks. To neglect the effect of connectivity we consider the communication range of a stationary node equals 1 (km). Also, we consider that all nodes are reputable and each mobile node has a transmission range equals 0.4 km, and its average speed equals 1.389 (m/sec). The average execution time of an application is investigated at different values of the arrival rate of inactive nodes, ranging from 1/45 to 1/20 (nodes/sec).

Fig. 21 shows that at a larger value of arrival rate of inactive nodes, e.g. 1/20 (nodes/sec), the worst performance is obtained than in the case of results at a smaller arrival rate of inactive nodes, e.g. 1/45 (nodes/sec). This is because of the probability a node could fail is high when compared with a lower arrival rate of inactive nodes value. Consequently, the average number of migrations of a VM increases when the arrival rate of inactive nodes is increased as shown in Fig. 22.

The node failure forces a VM to migrate to another reliable node. This leads to an extra time overhead of VM migration which is added to the execution time of an application. These results showed that our PlanetCloud performs well in terms of the average execution time of application with a smaller number of VM migrations even in case when a large number of mobile nodes have left the CoCloud.

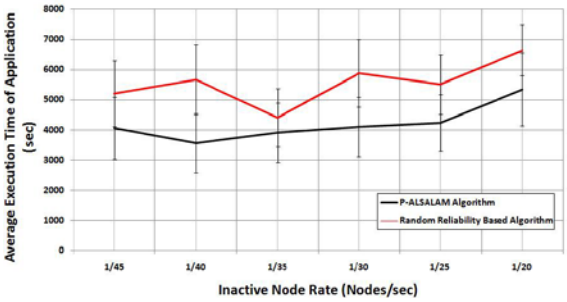

Figure 21. Average execution time of an application at different arrival rates of inactive nodes at a small-sized hospital (25 beds).

IJSER

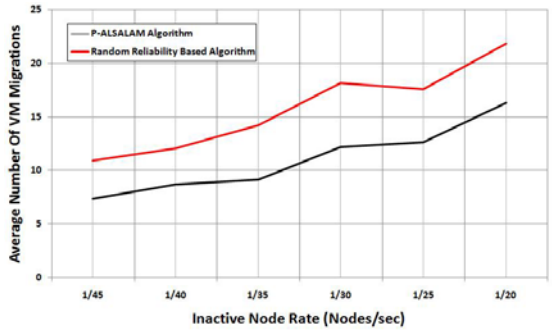

Figure 22. Average number of VM migrations at different arrival rates of inactive nodes at a small-sized hospital (25 beds).

Scalability and Management Overhead effect

In this experiment, we consider a CoCloud at three different configurations, as depicted in Table III. A distinct VM instance is created for each task; therefore there is no overhead of VM scheduling for task processing. We set the number of stationary nodes equals 10 nodes, and all of them have the highest computing configurations. On the other hand, we consider that each mobile node has a transmission range equals 0.2 km, and its average speed equals 1.389 (m/sec). The performance is investigated at arrival rate of inactive nodes equals 1/50 (nodes/sec).

We start this evaluation by investigating the performance of a CoCloud as more nodes, resources, are added to the system, ranging from 40 to 100 nodes, when we consider one application is submitted to be executed, with a fixed number of tasks equals 35.

TABLE 3
COCLOUDS WITH DIFFERENT HOST CONFIGURATIONS.

	Values of Resource Configurations		
Parameters	**low**	**Medium**	**High**
Number of CPUs/node	1, 2	2, 2	2, 4
Processing Capabilities	2000 ,7500 (MIPS)	7500, 14564 (MIPS)	7500, 49160 (MIPS)
RAM	512, 1024 (MB)	1024 , 2048 (MB)	1024 , 4096 (MB)
Storage	4 (GB)	16 (GB)	16 (GB)
Bandwidth	54 (MB)	54 (MB)	54 (MB)

Fig. 23 depicts a comparison between the results of applying both P-ALSALAM and random reliability-based node selection algorithms in terms of the average execution time of an application at a CoCloud with low resource configurations. Results show that P-ALSALAM significantly outperforms the random reliability-based node selection algorithm at all node densities. Analysis of the results indicates that average execution time of an application decreases with the increasing in number of nodes that could participate in a CoCloud, i.e. more resources are added to a CoCloud computing pool. This is because the more resources added to the computing pool of a CoCloud, the larger the probability to select a new VM placement, another node, where a VM could migrate to. This helps the migrated VM to get better resource availability and to scale up its computation. While, noticeable performance degradation in P-ALSALAM results appear at higher node densities, e.g. 100 (Nodes/Km²), due to a great effect of the delay due to collisions in addition to the transmission delay.

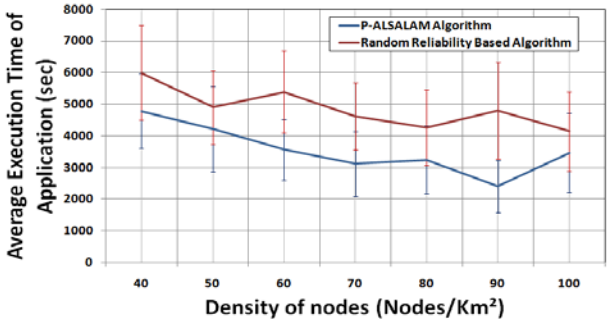

Figure 23. Comparison of application average execution time as more resources are added to a CoCloud at different reliability based algorithms.

Similarly, the average number of VM migrations when applying P-ALSALAM is smaller than the case when applying the random reliability-based node selection algorithm as shown in Fig. 24.Results show that the higher the node density the lower probability of VM migrations is obtained. This is because of the probability a node could fail is high at high node density, e.g. 40 (Nodes/Km²), at the same arrival rate of inactive nodes, when compared with a lower node density, e.g. 100 (Nodes/Km²). Consequently, the average number of migrations of a VM decreases when the density of nodes is increased as shown in Fig. 24.

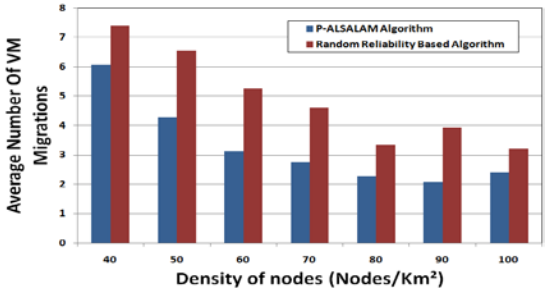

Figure 24. Comparison of Average number of VM migrations as more resources are added to a CoCloud at different reliability based algorithms.

We repeat the evaluation both P-ALSALAM and random reliability-based node selection algorithms when we consider different task load, i.e. the number of submitted tasks, ranging from 10 to 60 tasks. In this part, we set the density of nodes to be 70 (Nodes/Km²). Fig. 25 shows noticeable differences between results of the two cases, with/without using the P-ALSALAM, appear at a higher number of submitted tasks/application, e.g. 60 tasks/application. Also, results show the increasing trend in average execution time of application

with the increasing in number of submitted tasks, and consequently the number of VMs. This is because a distinct VM instance is created for each task. Consequently, the deployment and management of each VM requires additional overhead for application processing which increases the execution time of an application. In a CoCloud, the time required to migrate a VM from one node to another constitutes the time overhead of VM migration. It is clear that more use of VMs makes the execution time of application degrades faster. Also, the figure suggests that relationship between the number of submitted tasks, their VMs, and the execution time of application is linear.

Similarly, the results of the average number of VM migrations when applying P-ALSALAM significantly outperform the results when applying the random reliability-based node selection algorithm as shown in Fig. 26. Results show that more use of VMs the more probability of VMs migrations could occur.

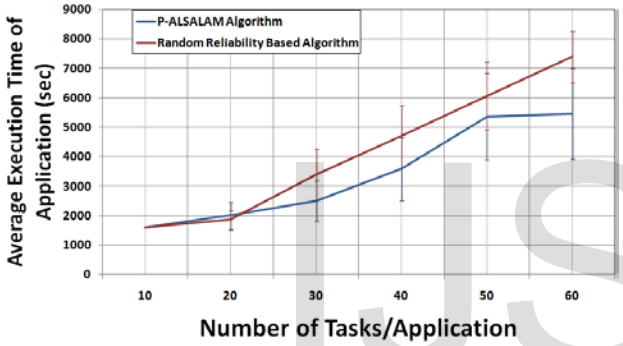

Figure 25. Comparison of application average execution time of a CoCloud with low resource configurations as the number of submitted tasks increases when applying different reliability based algorithms.

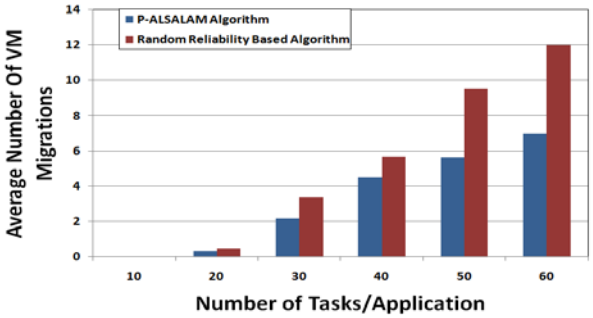

Figure 26. Comparison of Average number of VM migrations of a CoCloud with low resource configurations as the number of submitted tasks increases when applying different reliability based algorithms.

In the next evaluation, we evaluated P-ALSALAM algorithm at different resource configurations of a CoCloud as shown in Table III.Fig. 27 depicts a comparison between the application average execution times, at density of nodes equals 70 (Nodes/Km²), when we consider different the number of submitted tasks, ranging from 10 to 60 tasks. Results show that the higher the computing capabilities of a node participating in a cloud, the better performance is obtained. This is because the higher resource configurations of a participating node the higher ability of our P-ALSALAM Algorithm to allocate many VMs to a single physical node and to perform efficient VM consolidation. Consequently, the efficient selectivity of a reliable node, provided by our P-ALSALAM, decreases the inter node VM migrations and their management overheads as depicted in Fig. 28.

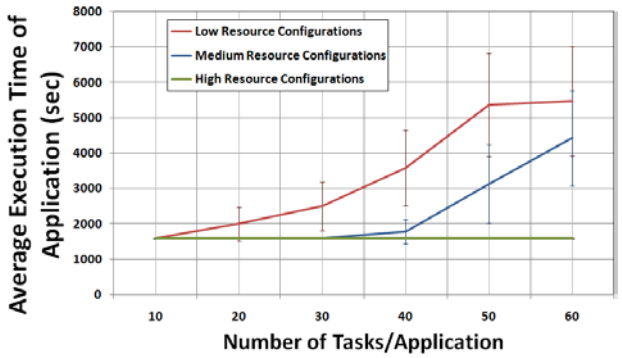

Figure 27. Application average execution time comparison for CoClouds with different resource configurations when applying P-ALSALAM Algorithm.

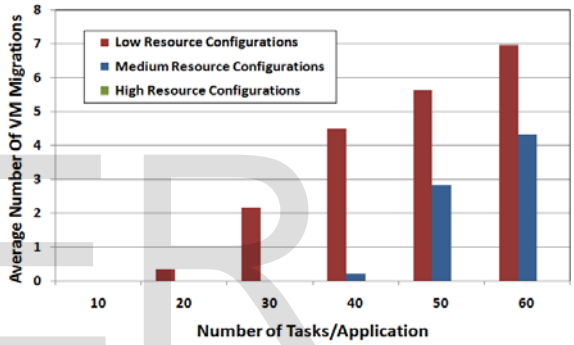

Figure 28. Average number of VM migrations comparison for CoClouds with different resource configurations when applying P-ALSALAM Algorithm.

Findings

Our findings can be summarized as follows:

- The performance is affected by the percentage of the number of stationary nodes within the total density of available nodes. It means the more stationary reliable nodes, participate in a CoCloud, the less dependency on mobile variable reliability nodes. This could enhance the performance of the submitted application.

- A better performance may be obtained, even at a shorter transmission range, if we apply our P-ALSALAM algorithm. This is because our algorithm frequently reschedules the delayed tasks and this minimizes the effect of communication delay.

- Performance enhancement of a CoCloud is provided by applying the efficient dynamic VM consolidation. This could be achieved by selecting reliable and powerful computing nodes to participate in a CoCloud. This enhances both scalability and management overhead by reducing the number of inter host VM migrations.

IJSER

- The performance is affected by the percentage of the reliable resources that could be added to the computing pool of a CoCloud. It means the more reliable nodes, participate in a CoCloud, the larger the probability to select a new VM placement, where a VM could migrate to. This leads to get better resource availability and scalability.

7 Conclusion

In this paper, we proposed PlanetCloud, a collaborative cloud management platform with an intrinsic support for highly-mobile heterogeneously-composed and configured clouds. Our construction fulfills the essential characteristics of the cloud computing model and offers different service models. Our design liberates cloud computing from being concerned about resource constraints. PlanetCloud is powered by a cloud management subsystem for resilient cloud operation on dynamic mobile resources to provide stable cloud in a continuously changing operational environment. Also, PlanetCloud is powered by a resource management subsystem which enables efficient use of idle mobile resources, by using collaborative autonomic resource management and providing a global mobile and stationary resource discovery, forecasting and monitoring. Additionally, this work presents our vision to enhance the prediction accuracy of resource availability including the descriptions of probabilistic models and artificial intelligence algorithms that could be implemented.

Analytical and simulation results showed that PlanetCloud can safely and reliably provide and maintain the needed computational power to form a reliable cloud operating in a dynamic, unstable, and heterogeneous resource environment. Also, results showed the benefits of enabling resource collaboration, provided by PlanetCloud, to achieve better capability to minimize management overhead. Additionally, results showed that PlanetCloud can "softly" guarantee reliable resource provisioning, transparently maintaining applications' QoS and preventing service disruption in highly dynamic environments.

Our ongoing research includes the following.

1) Develop a security mechanism to preserve the privacy and security constraints of a cloud resource provider, while allowing multiple users to share its resources and data for calendaring. There is a need to support distributed data access and computations without compromising the raw data of cloud nodes; and

2) Extend our proposed architecture to enhance the prediction accuracy of resource availability by utilizing complementary data sources such as from social networking.

References

[1] A. Klein, C. Mannweiler, J. Schneider, and H. D. Schotten, "Access schemes for mobile cloud computing," in Eleventh International Conference on Mobile Data Management (MDM), 2010, pp. 387-392.

[2] T. Xing, D. Huang, S. Ata, and D. Medhi, "Mobicloud: A geodistributed mobile cloud computing platform," in 8th International Conference on Network and ServiceManagement (CNSM), 2012, pp. 164–168.

[3] T. Xing, H. Liang, D. Huang, and L. X. Cai, "Geographic-based service request scheduling model for mobile cloud computing," in IEEE 11th International Conference on Trust,Security and Privacy in Computing and Communications (TrustCom 2012), 2012, pp. 1446–1453.

[4] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, California, USA, 2010, pp. 1-5.

[5] E. Marinelli, "Hyrax: cloud computing on mobile devices using MapReduce," Carnegie Mellon University, Master thesis 2009.

[6] A. Khalifa, M. Azab, and M. Eltoweissy, "Resilient Hybrid Mobile Ad-hoc Cloud Over Collaborating Heterogeneous Nodes," in the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2014), Miami, Florida, United States, October, 2014.

[7] A. Khalifa, M. Azab, and M. Eltoweissy, "Towards a Mobile Ad-hoc Cloud Management Platform," in the 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014), London, United Kingdom, December, 2014.

[8] A. Khalifa and M. Eltoweissy, "MobiCloud: A Reliable Collaborative MobileCloud Management System," in the 9th IEEE International Conference on Collaborative Computing:Networking, Applications and Worksharing, United States, 2013, pp. 158 – 167.

[9] A. Khalifa, R. Hassan, and M. Eltoweissy, "Towards Ubiquitous Computing Clouds," in The Third International Conference on Future Computational Technologies and Applications (FUTURE COMPUTING), Rome, Italy, September, 2011, pp. 52-56.

[10] A. Khalifa and M. Eltoweissy, "A Global Resource Positioning System for Ubiquitous Clouds," in the Eighth International Conference on Innovations in Information Technology (Innovations'12), Abu-Dhabi, United Arab Emirates, March, 2012, pp. 145-150.

[11] N. Fernando, S.W. Loke, and W. Rahayu, "Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing," in Fourth IEEE International Conference on Utility and Cloud Computing (UCC), Australia, 5-8 Dec. 2011, pp. 281-286.

[12] M. Black and W. Edgar, "Exploring mobile devices as Grid resources: Using an x86 virtual machine to run BOINC on an iPhone," in 10th IEEE/ACM International Conference on GridComputing, 2009, pp. 9-16.

[13] W. Lee, S. K. Lee, S. Yoo, and H. Kim, "A collaborative framework of enabling device participation in mobile cloud computing," Mobile and Ubiquitous Systems: Computing, Networking, and Services, Springer Berlin Heidelberg, vol. 120, pp. 37-49, 2013.

[14] P. Mell and T. Grance, "The NIST Definition of Cloud Computing (ver. 16th and final definition)," National Institute of Standards and Technology, Information Technology Laboratory, Sept. 2011.

[15] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," Journal of Internet Services and Applications, vol. 2, pp. 207-227, December 2011.

[16] C. Lin and S. Lu, "Scheduling scientific workflows elastically for cloud computing," in IEEE 4th International Conference on Cloud Computing, USA, 2011, pp. 746 - 747.

[17] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Bi-criteria workflow tasks allocation and scheduling in cloud computing environments," in 5th International Conference on Cloud Computing, USA, 2012, pp. 638-645.

[18] B. Yang, X. Xu, F. Tan, and D. H. Park, "An utility-based job scheduling algorithm for cloud computing considering reliability factor," in International Conference on Cloud and Service Computing (CSC), Hong Kong, 2011, pp. 95-102.

[19] L. Wang, G. von Laszewski, J. Dayal, and F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS," in 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID'10), Australia, 2010, pp. 368–377.

[20] M. Bourguiba, K.A. Agha, and K. Haddadou, "Improving networking performance in virtual mobile clouds," in Third International Conference on the Network of the Future (NOF), 21-23 Nov. 2012, pp. 1-6.

[21] A. Khalifa and M. Eltoweissy, "Collaborative Autonomic Resource Management System for Mobile Cloud Computing," in the Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, Spain, 2013.

[22] Anh-Dung Nguyen, P. Senac, and V. Ramiro, "How Mobility Increases Mobile Cloud Computing Processing Capacity," in First International Symposium on Network Cloud Computing and Applications (NCCA), 21-23 Nov. 2011, pp. 50-55.

[23] M. Eltoweissy, S. Olariu, and M. Younis, "Towards autonomous vehicular clouds," in Ad Hoc Networks, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.Springer Berlin Heidelberg, vol. 49, pp. 1-16, 2010.

[24] M. Abuelela and S. Olariu, "Taking VANET to the Clouds," in the 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM), New York, USA, 2010, pp. 6-13.

[25] T. Hristov, and G. Yan S. Olariu, "The Next Paradigm Shift: FromVehicular Networks to Vehicular Clouds," in Mobile Ad Hoc Networking:The Cutting Edge Directions.: Wiley-IEEE Press, 2013, pp. 645 - 700.

[26] S. Arif et al., "Datacenter at the Airport: Reasoning about Time-Dependent Parking Lot Occupancy," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 11, pp. 2067-2080, November 2012.

[27] M. Azab and M. Eltoweissy, "CyberX: A Biologically-inspired Platform for Cyber Trust Management," in 8th International Conference on Collaborative Computing, Oct. 2012, pp. 655- 663.

[28] M. M. M. Azab, "Cooperative Autonomous Resilient Defense Platform for Cyber-Physical Systems," Virginia Polytechnic Institute and State University, Doctor of Philosophy 2013.

[29] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1," 2009.

[30] Rakesh Agrawal, Tomasz Lmielinski, and Arun Swami, "Mining association rules between sets of items in large databases," in ACM SIGMOD Conference, Washington DC , USA, 1993, pp. 207-216.

[31] et al. K. Debray, "Weighted Decision Trees," in JICSLP, 1992, pp. 654-668.

[32] L. Rabiner and B. Juang, "An introduction to hidden Markov models," IEEE ASSP Magazine , vol. 3, pp. 4-16, 1986.

[33] Margaret K. Schafer, "Staffing the General Hospital: 25 to 100 Beds," U.S. Federal Security Agency, Public Health Service, Division of Hospital Facilities, Hospital Services Branch, 1955.

[34] S. K. Garg and R. Buyya, "NetworkCloudSim: modelling parallel applications in cloud simulations," in 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011), Melbourne, Australia, Dec. 2011, pp. 105–113.

[35] R. N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, no. 1, pp. 23-50, January 2011.

IJSER