A Converged Service Plane for Virtual Infrastructure Containers

Ibrahim Kabiru Musa[1] and Stuart Walker[2]

School of Computer Science and Electronic Engineering
University of Essex, Colchester, UK, CO4 3SQ

Abstract

The Infrastructure deployed in data centers to create cloud services can be extremely large. As such, the magnitude of these resources and the technologies required to enable the cloud services raises various challenges. Composing cloud services from a subset of the data centre as a virtual infrastructure container is a viable option to meet these challenges. We present an efficient automation layer for virtual infrastructure container. Biologically inspired models and algorithms for virtual resource provisioning, service isolation, and task allocation are presented and tested experimentally. We formulate relational model for components interaction in a virtual container. Large biological datasets are used to generate initial inputs to our model. We then implement the model in a simulation experiment to investigate the accuracy of our model and measure the improvement on performance of cloud services creation and delivery. The results from the experiments shows significant significant reduction in provisioning time and more effective resource utilization over similar approaches.

Keywords: *Virtual Infrastructure container, Datacenter, Cloud Computing, virtualization, IAAS*

1. Introduction

Infrastructure As A Service (IAAS) is one way of offering cloud computing. IAAS offers low-level components as Virtual Machines (VMs), which can be booted with a user-defined hard-disk image, and network as services. The IAAS deployment model allows network and IT resources to be offered on a Pay Per Use basis [1] [2]. Cloud computing provides a cost effective and flexible model for performing computationally intensive applications, data-intensive scientific applications such as large-scale numerical analysis [3], and multimedia applications such as video streaming. This approach is currently being pioneered by Amazon and Microsoft [4]. In the model, Virtual Machines (VMs) and storage are offered as services in arrangement that allow flexible scaling.

To meet the requirements of scalability, resources deployed in data centre to offer IAAS model can be extremely large. A typical cloud data centre may comprise thousands of network and IT resources. The magnitude of these resources and technologies to enable the cloud services, such as virtualization, raise numerous challenges including efficient resource utilization, flexibility, and scalability [5]. Numerous proposals to meet these challenges are available in the literature. Inspired by the concept of biological cell [6], a new approach to meet these challenges proposed in [7] envisions the acceptance of request for a converged (network and IT) virtual infrastructure container similar to a biological cell. We refer to this container as service cell offered in Virtual Cells As A Service (vCAAS). vCAAS disambiguates Communication As A Service (CAAS). Components in biological cells (proteins, organelles) interact, based on the organic characteristics of the interacting component and the task to perform, to realise cell functions such as regulation and protection. This type of interaction is similar in context to the kind of interaction in cloud environments. The proteins and their interaction here are virtual machines and network links respectively.

vCAAS is similar to virtual private cloud (VPC). However, while VPC focuses mainly on extending an existing VPN and public cloud concepts to provide secured private cloud services, vCAAS focuses on allowing more control and flexibility over a set of configurable virtual resources. In vCAAS, one or more virtual machines interact to complete a user defined task. To achieve this, VMs and storage resources are abstracted and offered to customers as flexible containers. We refer to each of these containers as service cell (vCell). The vCell owner can control the entire network and IT components. Virtual resource (network and IT) are viewed as a single unit thereby simplifying management and other automation tasks. Furthermore, container based approach to IAAS offers the capability to reuse resources on each virtual machine that is placed in the same container.

As a new approach to next generation IAAS, the functional components of a typical IAAS architecture may not be suitable for vCAAS. There is a need to investigate new models, algorithms, and techniques for container based cloud delivery. This paper proposes a service automation plane suitable for vCAAS. We begin with a description of architectural model for vCAAS and then proceed with the description of our proposed converged automation layer. We apply the undirected graph exponential model (p2-model) to a virtual machines Interaction Network (VIN). We assess the node-specific parameters of the model and use them to identify nodes that dominate the evolutions of the vCell. This type of inference is well established [6] in biological networks and used to predict protein-protein interaction and in identifying virus mutation as well as predicting its evolution [8]. Our major contributions are the formulation of specifications and algorithms required in effective service automation layer suitable for vCAAS delivery. The layer is proposed with the fundamental requirement to provide IAAS users with the illusion of a unique self-healing and flexible infrastructure. All components are contained as single entity and designed to meet specific requirements. The rest of the paper is structured as follows. Section 2 review existing related works, section 3 describes our chosen cloud architecture suitable for vCAAS, Section 4 describes the proposed service layer for our architecture, section 5 presents results of experiment, and finally Section 6 concludes the chapter with a summary of findings.

2. Related Works

Data centres host wide variety of applications each with different requirements. To overcome the challenges of cloud applications, a service based orchestration of virtual resources to accommodate multitude of users each with isolated view is proposed in [9] by abstracting resource management and control functions. The idea is to enable virtual private cloud (VPC) through various distributed Network Resources Managers (NRM) and Network Resource Brokers (NRB) on a single service interface. The NRM performs all virtualization of underlying physical infrastructure. The architecture is designed based on the Open Services Gateway initiative (OSGi) framework. The VPC provides a set of libraries, installed on demand, to enable multiple resource management views. Key modules in the proposed architecture are the control and mediation layers. The mediation layer handles request for virtual resources and performs bandwidth reservation and allocation. Recently, Cells As A Service (CAAS) proposed in [7], combined the visions of these disparate ideas into a next generation cloud delivery model. CAAS envisions the provision of compute, storage, and network services to a large number of multi-tenants each with specific performance criteria such as delay, security, and flexibility all defined in an Extensible Mark-up Language (XML) template. Each service user is assign a view isolated from other services.

Similar to the research initiatives in [7], this work views virtual cloud service as a converged entity comprising network and IT resources. Unlike [7] which demonstrated a simple realization of CAAS on a layer three VPN with security as the main focus, we focus on formulating converged flexible automation layer for virtual infrastructure container. The layer is build on well established cloud architectures. We also explore biological techniques to realise the various functionalities of our proposed automation plane. The next section describes an architecture for virtual infrastructure container with the service automation layer proposed in this work.

3. Architecture for virtual service container

This section presents the cloud architecture for our vCAAS. The architecture is depicted in figure 2 and is based on existing cloud architectures such as NIST [10] and Open Group Reference Architectures (OGRA) [11]. Similar to OGRA, we proposed a decoupling of NIST services layer into delivery and service automation sub layers, leaving lower layers as described by NIST amd OGRA. This way our architecture complies with a tested and standardized cloud architecture. The various layers and modules interact to create a cloud based virtual infrastructure container - vCell. vCAAS envisages large number of virtual resource. Each vCell is characterized with a service level agreement (SLA) configured by users using a service console (SC). The fundamental assumptions made in our model are:

Rt 1 : A vCell may request multiple virtual resources in form of VMs and virtual storage.

Rt 2 : Virtual Machines allocation is bounded by available capacity of host. vCAAS resources (VMs) may spread across many physical hosts.

Rt 3 : Virtual Machines interact with each other to execute a task. Intra vCell communication increases the total overheads. These overheads are affected by nodes and link characteristics connecting physical hosts.

Rt 4 : A vCAAS provider (Figure 3) may allocate VMs from multiple host locations to achieve business and functional objectives.

Rt 5 : Initial vCell allocations must minimize subsequent services impairments and utilization of resources.

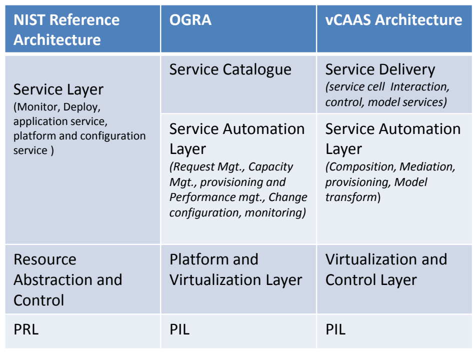
Rt 6 : vCell performance is measured as aggregate rather than separate for each component.

3.1. Various actors in vCAAS

In our proposed model, clear separation of actors ensures flexibility of vCell creation and isolation. Various actors and proposed roles are shown in Figure 3. Physical resources owned by vCAAS physical Infrastructure Providers (vCAAS-PIP) are accessed and virtualized by vCAAS Virtual Infrastructure Provider (vCAAS-VIP) - also acting as a broker. vCell service starts with submission of requests from potential vCell owner to vCAAS-VIP. vCELL owners submit request for specific composition of network and IT to the vCAAS-VIP. vCAAS-VIP then query all vCAAS-PIP and select one vCAAS-PIP. To meet the requirements of the vCELL owner, vCAAS-VIP evaluate existing resource pool and consults vCAAS-PIP to appraise available resources suitable for submitted request. vCAAS-VIP then virtualize and offer the vCell to satisfy initial request (Fig. 3). Our proposed architecture combined the functionalities of the actors and is realised as an IAAS model comprising vCAAS Delivery Layer (CDL), Service Automation Layer (SAL), Control and Virtualization Layer (CVL), and Physical Resource Layer (PRL). Next we describe CDL, CVL briefly. Sections 4 will describe SAL in detail, which is the main focus of this paper.

Cloud Delivery layer (CDL): A distinguishing feature of this layer from the common cloud delivery layer is the presence of a vCell Control Module (COM) and vCell components interaction Module (CIM). COM modules enable vCell users to have direct access to control functionalities available for a vCell. This way, configuration commands can be passed to the underlying SAL. CIM monitors all vCell components interactions and vCell-vCell interactions. The combined functionalities of COM and CIM provide a vCell with adaptation and isolation capabilities.

Control and Virtualization Layer: In our model, a virtual interface (VIF) is created for each VM. The VIF connects a VM to a virtual switch (vSwitch) forming a virtual link. Combinations of these virtual links and connected virtual nodes constitute our definition of a virtual network topology (VNT). We propose an adaptive vSwitch-based traffic shaping strategy for vCAAS (Fig. 6). Each VM in a vCell is assigned a virtual switch (vSwitch) port with initial bandwidth, based



NIST Reference Architecture	OGRA	vCAAS Architecture
Service Layer (Monitor, Deploy, application service, platform and configuration service)	Service Catalogue	Service Delivery *(service cell Interaction, control, model services)*
	Service Automation Layer *(Request Mgt., Capacity Mgt., provisioning and Performance mgt., Change configuration, monitoring)*	Service Automation Layer *(Composition, Mediation, provisioning, Model transform)*
Resource Abstraction and Control	Platform and Virtualization Layer	Virtualization and Control Layer
PRL	PIL	PIL

Fig. 1. Relating our proposed dynamic network architecture for Virtual Infrastructure container, vCAAS, with other widely used cloud architectures.

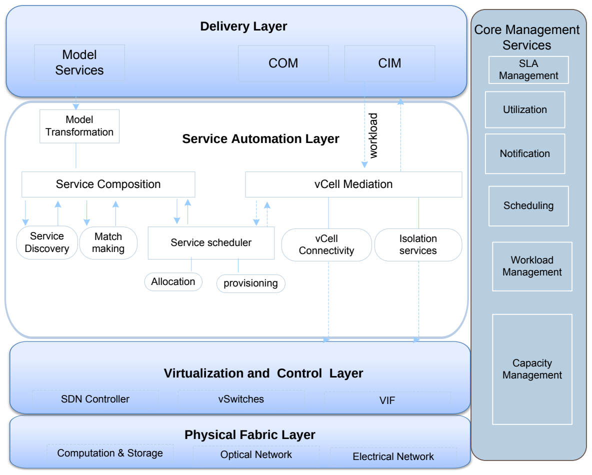


Fig. 2. Proposed dynamic network architecture for Virtual Infrastructure container(vCAAS.)

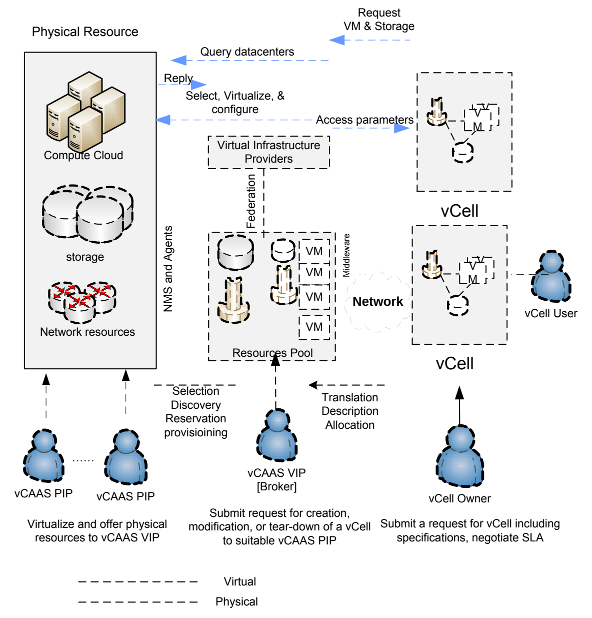

Physical Resource
Request VM & Storage
Query datacenters
Reply
Select, Virtualize, & configure
Access parameters
Compute Cloud
storage
Network resources
NMS and Agents
Virtual Infrastructure Providers
Federation
Middleware
VM
Resources Pool
Network
vCell
vCell User
Selection Discovery Reservation provisioining
Translation Description Allocation
vCAAS PIP
vCAAS VIP [Broker]
vCell Owner
Virtualize and offer physical resources to vCAAS VIP
Submit request for creation, modification, or tear-down of a vCell to suitable vCAAS PIP
Submit a request for vCell including specifications, negotiate SLA
Virtual
Physical

Fig. 3. Main actors in our proposed cloud architecture and the respective functionalities associated to each.

on request, and allowed to expand capacity relative to unused bandwidth in the vCell. This way, VMs adapt dynamically to various traffic conditions. To protect SLA of each container, traffic isolation mechanisms are applied at vSwitch level to police VM activity and ensure activity in one vCell does not have adverse effect on performance of other VMs in different vCell. The module is responsible for dynamic reconfiguration of various virtual components of a vCell. It provides a transparent mechanism for upper layers. Virtual topology is controlled by this layer. Network and computation components are dynamically controlled using openflow controller and virtual resources management application interfaces respectively. Our model adopts the use of a dedicated supervisory channel to facilitate communication between the network components and the controller. This way, control messages are not affected by activities of data flow.

Physical Resource Layer: This consist of heterogeneous physical network and IT resources. Under the control of physical Infrastructure Provider (PIP), access to the physical fabric is through management interface such as Network Management System (NMS), Optical User-to-Network Interface (OUNI), and infrastructure agents. The layer resides in PIP domain and extends to other provider domains using Optical Network-Network Interface (O-NNI).

3.2. Typical vCAAS Scenario

A typical scenario for a cloud application requires the interaction of several virtual components in a workflow to execute a task [5]. Examples of such applications include video transcoding, data mining, and large web indexing. Parallel processing applications such as hadoop - Java version of map reduce [12] requires the coordination of various virtual machines and storage infrastructure each performing sub tasks. Typical map reduce application require up to 256 nodes [2], 400 nodes [12], and other similar variations. By carefully formulating the interaction model for each vCell, efficient use of resources can be achieved. Surplus resources from a vCell can be utilized to accept new request from the same vCell owner. To model these scenarios, we assume that tasks submitted to a vCell are completed in cyclic pattern 4. In figure 5 resource R_1 request resource R_2. At this point R_1 serves as a client to the server R_2. We denote this as $R_1 \longrightarrow R_2$. Similar relationship exists in $R_1 \longrightarrow R_3$, $R_2 \longrightarrow R_4$, $R_1 \longrightarrow R_4$, and $R_4 \longrightarrow R_5$.

The communication and computation overheads of completing a task m, T^m, which requires resources R^m, and submitted initially to resource i is thus given by:

$$T^m = R_i^m + (N_{i,j}^m + R_j^m) * X_{i,j}^m \qquad (1)$$

Where $N_{i,j}$ is the network overhead (e.g delay) between resources indexes i and j, R_i and R_j are computational overheads from resource with indexes i and j respectively, and finally $X_{i,j}$ is defined as:

$$X_{i,j}^m = \begin{cases} 1 & \text{if } R_i, R_j \in R^m andi \neq j \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

From equation 1 the interaction coefficient is derived as:

$$I_{i,j}^m = (R_j + N_{i,j}) * X_{i,j} \qquad (3)$$

Resource R_i receive request for subtask after finish time F_{i-1}. F_{i-1} is the finish time of resource index $i-1$ which immediately precede i in the sequence. If the order at which T^m request resource R_n, $\forall n \neq i$, can be altered then request can be sent to next swappable resources. We say two resources, with indexes k and $k+1$, are swappable if the order of submitting task to the resources does not matter. This way a swappable resources with the smallest availability time can be chosen and the idle time of virtual resources in

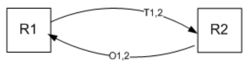


Fig. 4. Circular task representation between virtual components in a cloud environment

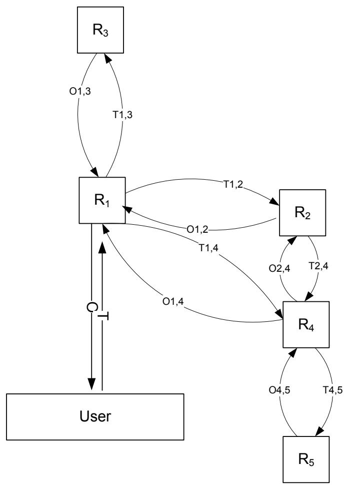


Fig. 5. Example of a circular task in a virtual cloud environment. R is a virtual resource which can be computation or storage,R_i for i=1,2,... is the computational overheads at resource with index i

the vCell can be reduced. To achieve effective utilization in this scenario requires a suitable coordination mechanism for request handling, resource provisioning, and communication processes. We thus propose an automation layer capable of achieving this level of coordination. We describe the design of this layer in the next section.

4. Service layer for Virtual Infrastructure Container

This section describe our proposed layer for automating the delivery of a virtual infrastructure container. Figure 2 depicts our proposed SAL layer. We regard a cloud service as an application functionality that requires physical or virtual network and IT resources over its execution life cycle. It provides components for translation of submitted request, resource composition, mediation,and provisioning. Design goals are the efficient resource utilization and performance enhancement. Requests are translated to service specific or canonical parameters. The SAL then search out the qualified resource for each parameter in the request, generate candidate resource service set, evaluate each quality of service (QoS) criterion, and select one suitable resource. Here, the Service Layer resides in broker domain. Resource registry maintains information about identity of providers, network access parameters, and vCells. Various modules interact to achieve the automation. These are mediation modules and resource provisioning.

4.1. vCell Mediator

This is a functional unit which ensures vCell-vCell and vCell-infrastructure interaction. Workload submitted by a vCell is appraised and allocated required resources. To realise these functionalities effectively, this module implements interaction prediction model as a two-stage strategy. We refer to these stages as learning and prediction. During the learning stage, practical experiment is conducted using various application execution models (parallel processing, analytic, distributed, and data intensive applications). The output of a learning stage is the interaction matrix M with rows and columns showing component-component interaction evidence.

In the second stage, M is used to predict the future interaction in conjunction with a simple Relational Markov Random Field (RMRF). To formulate our model for RMRF, we begin with equation 4 and taking the logarithm of both sides and the exponential, e, of the Right Hand Side (RHS) we obtain:

$$I^m_{i,j} = e^{log_e(R_j+N_{i,j})*X_{i,j}} \tag{4}$$

Notice that this model satisfies the features of Log-normal distribution reported in literature [13] as suitable for interaction between components within a data center. Next, we define a function $\Psi_{i,j}(X_{i,j}) = log_e((R_j + N_{i,j}) * X_{i,j})$. And thus from equation 4 we have:

$$I^m_{i,j} = e^{\Psi_{i,j}(X_{i,j})} \tag{5}$$

The prediction stage is formulated as inference mechanism based on the posterior distribution of a parameter for any given statistical data, X. Using Bayesian rule for a combination of the prior information and the data we have:

$$P(\Psi/X) = \frac{P(\Psi)P(X/\Psi)}{P(X)} \tag{6}$$

Where the term p(Ψ) represents the prior distribution, obtained from previous experiment, on the parameter.

We model the vCAAS components as projection of potentials Ψ using statistical modelling. Each potential $\Psi_c \in \Psi$ defines interaction (or non-interaction) measure over a set of variables, X. From 5 we define a joint Markov distribution function for independent components i,j ∈ V as:

$$P(X = X_{i,j}) = \frac{1}{Z} \prod_{i,j \in V} e^{\Psi_{i,j}(X^m_{i,j})} \tag{7}$$

$X^m_{i,j}$ is a binary variable define in equation 2, Z is the normalization constant, X represent the virtual component (node) in the network, and V is the set of virtual components in a vCell. Equation 7 assumes non-existence of dependence between the feature of a virtual component and task to be completed. However, input/output (IO) rate for a virtual component depend on the functional requirements of tasks. For instance, storage base virtual components are likely to show higher IO rate in data intensive tasks. For dependent components, where interaction between virtual components depend on the functional feature characteristics (e.g storage, application, control services) of VM, we define a joint Markov distribution function P(X) for dependent components i,j ∈ V as:

$$P(X = x_{i,j}) = P_1(X) \prod_{i,j \in V\, f \in \Gamma} e^{\Psi_{f,i,j}(X^m_{f,i,j})} \tag{8}$$

where

$$P_1(X) = \frac{1}{Z} \prod_{i,j \in V} e^{\Psi_{i,j}(X^m_{i,j})} \prod_{j \in V\, f \in \Gamma} e^{\Psi_{f,j}(X^m_{f,j})} \tag{9}$$

Z is the normalizing factor and Γ is the set of known functional characteristics which determines the level of interaction required by a task. Z is defined as the distribution over the entire components graph, and is given by:

$$Z = \sum_{x} \prod_{i,j \in V\, f \in \Gamma} e^{\Psi_{f,i,j}(X^m_{f,i,j})} \tag{10}$$

Γ is obtained from the learning stage. The projection function Ψ in equation 10 defines two sets of potential values- components to component and task to component. This way both components state (represented by the task feature, f) and components interactions are taken into consideration. It is known that a joint distribution over the interaction variables in equation 10 is equivalent to a distribution over possible interaction networks [6]. Two functionalities provided by this module are isolation and connectivity.

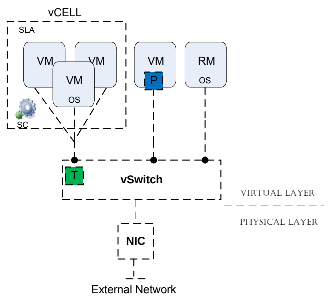


Fig. 6. Architecture of a Traffic Shaper suitable for virtual infrastructure container with a Service Console (SC). P is per VM and T is per vCell rate policy

4.1.1. Virtual components rate isolation. In this work, we formulate an adaptive vSwitch-based traffic shaping strategy for vCAAS. Each VM in the vCAAS is assigned an initial bandwidth by the Resource Manager(RM), based on request, and allowed to expand capacity relative to unused bandwidth in the container. This way, VMs adapt dynamically to various traffic conditions. To protect SLA of each container, traffic isolation mechanisms are applied to police VM activity and ensure activity in one VM does not have adverse effect on performance of other VMs.

In our isolation strategy for vCAAS, we assume each VM interact with other VMs, in hub and spoke arrangement, to complete work flow tasks. VM experiencing high traffic (such as storage VM) consume high vCAAS bandwidth (Fig. 6). Other VMs require less bandwidth. At the time of vCAAS creation we assign a super port with bandwidth capacity, computed as a function of predicted interaction probability, P(X) in equation 8, and available bandwidth capacity between the two nodes. The allocated bandwidth is thus computed as:

$$B_{i,k} = \sum_{i,j \in V} P(x_{i,j}) C_{i,j}) \tag{11}$$

Where C is the available capacity between nodes i and j, $B_{i,k}$ is the allocated bandwidth for VM_i in virtual container k. This super port can be viewed as aggregation of fixed capacity ports. The bandwidth available for any VM is computed as:

$$A_{i,k} = B_{(i,k)} + \sum_{j} U_{j,k} \tag{12}$$

i,j,k=1,2,...N and i $\neq$ j

Where $U_{j,k}$ is the available bandwidth of VM_j in virtual container k. The summation on the right of equation 12 is unused bandwidth in vCAAS which can be utilized by a transmitting VM. Thus, traffic request R_i at VM_i are accepted as long as $R_i \leq B(i,k) + \sum_j U_{j,k}$

4.1.2. Connectivity Sub Module. In vCAAS we identify 3 networks similar to original idea proposed in [7]. Between user and vCell (internet,private network), between VMs (intra vCell) in a vCell, and between vCells(inter vCell). Our vCell provides an abstracted view of the data centre where compute and storage VMs interact to support a given application task. Interaction between VMs residing in hosts goes through the physical network interface card (PNIC). The interaction is upper- bounded by capacity C of the host. This bound also hold for VMs residing in different hosts. Noticed that since our virtual container (vCell) is isolated, interaction between VMs in a vCell does not affect other VMs and hence there is no need for inter vCell scheduling. In intra vCell communication, a component may require distinct services from all vCell components to complete a task.
vCAAS connectivity further provide functions with ability to forecast future vCells events that require network management operations such as changes in traffic patterns (time of day, location, increased demand, etc.), or possible failure points. vCAAS connectivity interact with management to ensure the ability for communication networks to graciously recover from failures and to maintain a certain level of QoS. Proactive approach before the anomaly occurred in order to keep the network operational and meet vCell requirement in the most effective manner and avoid vCells congestion in any segment over the whole network.

4.2. Resource Provisioning Module

Efficient resource provisioning in cloud computing is critical due to the large number of tenants. For any vCell request represented as a turple vCell(V, D,Q), we

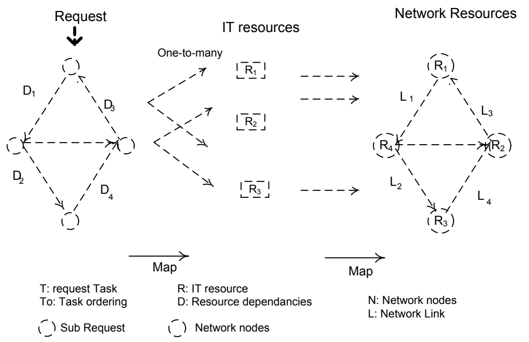


Fig. 7. Non Contiguous provisioning strategy where request is satisfied across multiple physical host based on first fit

let V_i, i=1,2,,N be the set of existing components in a vCell. V = virtual machines set, D = delay requirement, and Q = other quality of service requirements. We denote interaction between vCell components i and j as $N_{i,j}$, the capacity (bandwidth) request between i and j as $\omega_{i,j}$. For simplicity we assume that, for bandwidth request B_i and B_j from components i and j respectively, $\omega_{i,j} = max(B_i, B_j)$.

Next we present one existing algorithm in the literature and one novel provisioning algorithm for instantiating VMs in vCAAS.

4.2.1. Non-contiguous provisioning (NCP). This approach [14] is more efficient as request can be satisfied from many hosts with a mechanism to carry remaining VMs request to next available host, i.e. $vCell(V, D, Q) \in H$. Where H is the set of all available physical hosts. The approach is a variation of first fit with carry on to next available host.

4.2.2. Template-based provisioning (TBP). We propose evolution-based approach to fast provisioning of cloud services. This approach is similar to NCP but with added consideration to specific features of the task. Inspired by biological Cell evolution, the model of vCAAS provisioning is based on template duplication plus rewiring. The process includes the basic ingredients of vCell growth and intends to reproduce the previous set of observation. Such evolutions may involve addition of VM, deletion of VM, addition of interaction between VMs, and deletion of interaction path between VMs. Interaction between VMs depends on feature characteristics of VM. The rules of the provisioning are implemented as follows.

- Choose a virtual component and duplicate;
- **Add** to new vCell
- Links emerging from the new generated node are removed with probability $P(X)$ as defined in equation 8.
- New links (not previously present) are created between the new node and all the rest of the nodes with probability $P(X)$.
- **Repeat** step 1-3 until all requests complete.

Step 1 implements component duplication, in which both the original and the replicated vCell retain the same structural properties and, consequently, the same set of interactions. The rewiring steps 2 and 3 implement the possible mutations of the replicated vCell, which translate into the deletion and addition of interactions, with computed probabilities. Algorithm 1 lists the steps involved in this type of vCell provisioning.

5. Experiment

This section describes the experiments conducted to demonstrate the ideas presented in this paper. The experiments combines both practical and simulated techniques.

5.1. Simulation Setup

To implement the interaction prediction model presented in section 4 we run a data intensive application using Map-reduce [15] technique. Our choice of map-reduce is influenced by the success of map-reduce in many research and production environments including Google and Yahoo search engines. The application we considered aim to solve large biological sequencing problems using cloud services. This type of computation is commonly used in bioinformatics and involves the application of informatics and computational methodologies and techniques on analysing biological data [16]. To implement the map-reduce technique, we deploy The Genome Analysis Toolkit (GATK) [17] on a cluster of 10 servers (Fig. 8). GATK provides an open source Java programming framework for writing efficient and robust analysis tools for next-generation resequencing problems. The framework is design in a way that processing can easily be parallelized and distributed. The queue module in GATK allows partition of large data set into any number, analysing each partition, and combining the outputs into a single result. For this experiment we use a publicly available Genome 1000 project dataset to perform the technique of map reduce on the dataset. 5GB of sequence data is uploaded to a Network File Server(NFS). Standard (256 memory, 10 GB storage) virtual machines instances are started to compute the gene expression in parallel. Traversal and analysis walkers calculate the desired expression defined in a binary alignment/map(BAM). BAM files provide reads, quality scores, alignments, and metadata for the gene analysis [17].
We implement experiments in a prototype experimental lab testbed and in scalable simulated environment using Cloudsim [18]. The prototype lab implementation using 10 physical hosts enabled by XEN hypervisor. The prototype implementation of vCAAS is achieved with a Xen Cloud Platform (XCP) in a network of 10 Dell systems, each with similar hardware and virtualization kernel. All the data required for the computations are stored in a Linux based network file server (NFS). We created 5 vCells each with 5 VMs. Virtual machines in a vCell share virtual storage carved from existing storage repository. VMs are created and managed via XEN application programming Interfaces (API). The experiment aims to achieve an effective resource and task allocation by:

- Perform a real practical experiment using a chosen execution model.
- Compute inference values.
- Implement service automation techniques guided by output of inference state.

5.2. Experiment results

To implement our learning algorithm we run a map-reduce application on large sequencing dataset obtained from the Genome 1000 project. Information from the genome dataset contains genetic material on living thing. Such information is the entire set of hereditary instructions for building, running, and maintaining an organism, and passing life on to the next generation. The implemented Genome Dataset experiments consist of up to 10 cores, the algorithm leverage the unique large-scale data processing capabilities of Map-Reduce [15] to reduce the runtime of this important computation from several hours on a desktop workstation to mere minutes on the vCAAS. The result obtained is then used as initial estimates for a Markov distribution likelihood estimation.
First we establish the case for variation in idle times

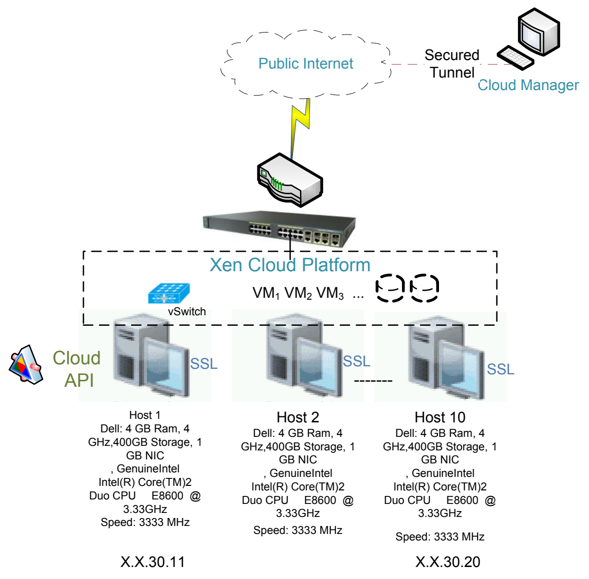

Public Internet
Secured Tunnel
Cloud Manager
Xen Cloud Platform
VM₁ VM₂ VM₃ ...
vSwitch
Cloud API
SSL
SSL
SSL
Host 1
Dell: 4 GB Ram, 4 GHz,400GB Storage, 1 GB NIC , GenuineIntel Intel(R) Core(TM)2 Duo CPU E8600 @ 3.33GHz Speed: 3333 MHz
Host 2
Dell: 4 GB Ram, 4 GHz,400GB Storage, 1 GB NIC , GenuineIntel Intel(R) Core(TM)2 Duo CPU E8600 @ 3.33GHz Speed: 3333 MHz
Host 10
Dell: 4 GB Ram, 4 GHz,400GB Storage, 1 GB NIC , GenuineIntel Intel(R) Core(TM)2 Duo CPU E8600 @ 3.33GHz Speed: 3333 MHz
X.X.30.11
X.X.30.20

Fig. 8. Prototype implementation of Virtual container using Xen Cloud Platform.

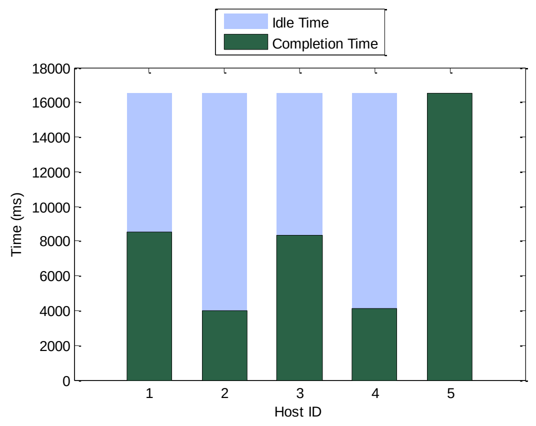


Fig. 9. Graph showing the idle times of various virtual machines in a parallel processing application using map reduce.

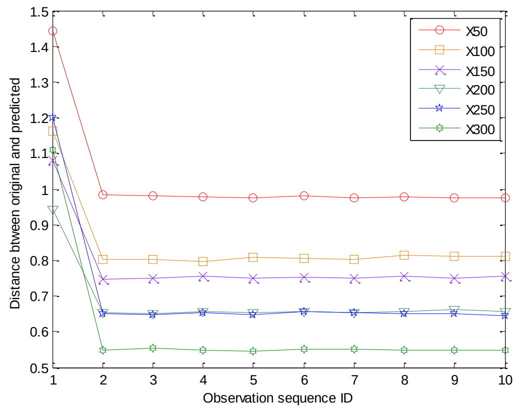


Fig. 10. Measuring the distance between learnt observation and predicted sequence in a 4 States Markov

of various VMs coordinated to perform a map reduce application. Figure 9 shows a 48.6 seconds deviation in idle time. For large data set, upto terabytes in biological data, this deviation impacts on cost and performance of computation resources. Using the feature characteristics obtained from the experiment we then implement a prediction experiment using the model presented in section 4. Each request is modelled as a state in RMRF model. This way task that request 4 resources is assigned 3 states. Figure 10 shows the result of implementing the proposed learning algorithm. As shown in figures 7 and 11, the distance between the observed and predicted states is less than 0.06 when 300 observations are used.

To investigate the ideas proposed in cloud environment we conducted simulation experiment using Cloudsim [18]. First, we investigate performance of the strategy described in relation to communication overheads, ease of service creation, and utilization. We investigate the performance of the two virtual machines and tasks allocation algorithms presented in section 4. The focus is on creation time for various number of requested virtual machines. Thus the results in 12 compares the time to create virtual service cells with varying VMs sizes. Figure 12 shows that our algorithm out performs NCP algorithm in speed of service creation for varios VMs sizes.

We also investigate the impact of prediction model on task distribution. Using RMRF the tasks are allocated to host with less workload as opposed to first fit task allocation. Figure 13 shows that applying RMRF prediction capabilities ensures that jobs are scheduled with higher fan out. In the figure 13, tasks, 1-100, from a virtual service cell are scheduled in a way that ensures under utilization is minimized. The vertical axis in figure 13 gives the total workload on the selected resource at the time of submitting tasks 1,2,...,100. The tasks are shown on the horizontal axis. This result also shows improvement in effective resource utilization.

We further demonstrate our proposed isolation strategy with 100 vCells each with 10 VMs based on request submitted in a template. VMs are assigned bandwidth in the range 2-10MB. Each vCell is assigned a port of capacity equal to total predicted bandwidth requirement by all VMs in vCell. Figure 14 shows the

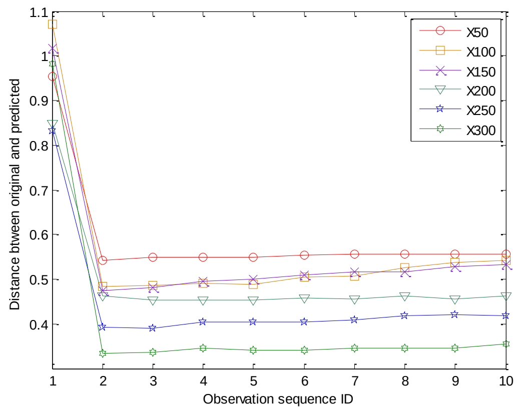


Fig. 11. Measuring the distance between learnt observation and predicted sequence in a 5 States Markov

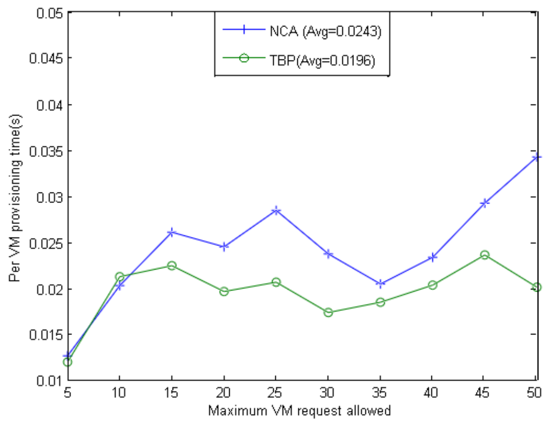


Fig. 12. Comparing our novel vCell creation with first fit technique.

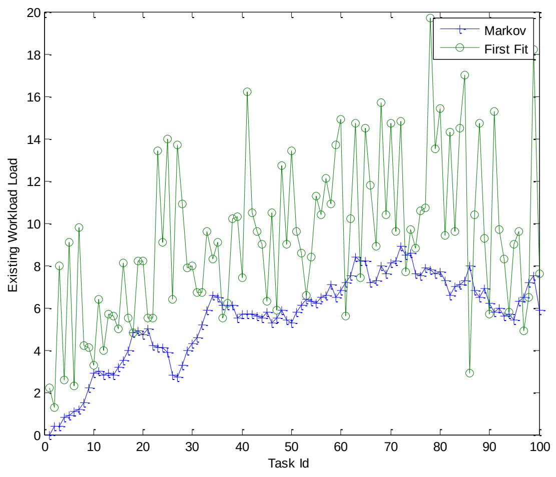


Fig. 13. Comparing task allocation using RMRF prediction and first fit.

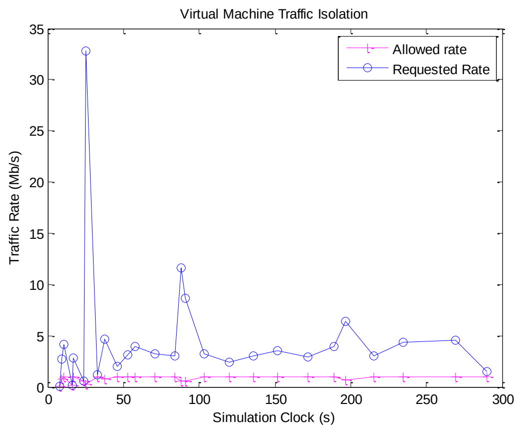


Fig. 14. Effect of container based isolation of virtual machine on bandwidth request and allowed usage.

result of running 100 work flow applications each assigned a uniform random VM. The vertical axis shows the average traffic. The traffic policy ensures that sample virtual machines are not allowed rate above actual allocated rate. VM activities in Figure 14 are shaped to conform to allocated bandwidth. The result shows a tight margin between allowed and requested bandwidth. This result very the effectiveness of effectively isolating vCells by assigning guided probabilistic computed bandwidth at point of vCell creation.

6. Conclusion

In this work, a container based cloud service delivery architecture is presented and a converged (network and IT) service layer for such model is described. Mathematical models, widely used in biological networks to describe cell interaction, are adopted in context and modified to realise the functionalities of the proposed service layer. The outcome of prototype experiment on a research testbed are used as input to the model for inference mechanism. Our interaction prediction model shows accuracy and considerable efficiency on rapid service creation. Based on the prediction model, isolation models are applied at virtual switching layer to avoid adverse activities of virtual machines from affecting other virtual machines. Although in this paper, data intensive application is used to demonstrate the novel ideas, the same learning and prediction model can be applied to many other execution models and techniques.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," *Technical Report No. UCB EECS-2009-28*, pp. Retrieved from http://www.eecs.berkeley.edu /Pubs/TechRpts/2009/EECS–2009–28.html, February 2009.

[2] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, sept. 2008, pp. 5 –13.

[3] Y. Tabaa and A. Medouri, "Towards a next generation of scientific computing in the cloud," *International Journal of Computer Science(IJCSI)*, vol. 9, Issue 6, No 3, 2012.

[4] M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to the theory of np-completeness," in *W. H. Freeman and Co*, 1979, pp. 261–262.

[5] A. S. Zamani, M. M. Akhtar, and A. S, "Emerging cloud computing paradigm," *International Journal of Computer Science(IJCSI)*, vol. 8, Issue 4, No 2, 2011.

[6] A. Jaimovich, G. Elidan, H. Margalit, and N. Friedman, "Towards an integrated protein-protein interaction network: A relational markov network approach." *Journal of Computational Biology*, vol. 13, no. 2, pp. 145–164, 2006.

[7] P. Banerjee, R. Friedrich, C. Bash, P. Goldsack, B. Huberman, J. Manley, C. Patel, P. Ranganathan, and A. Veitch, "Everything as a service: Powering the new information economy," *Computer*, vol. 44, no. 3, pp. 36 –43, march 2011.

[8] E. N. Mohamed, K. Samar, and S. Nabila, "Profile hidden markov model for detection and prediction of hepatitis c virus mutation," *International Journal of Computer Science(IJCSI)*, vol. 9, Issue 5, No 3, 2012.

[9] T. Miyamoto, M. Hayashi, and K. Nishimura, "Sustainable network resource management system for virtual private clouds," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 30 2010-dec. 3 2010, pp. 512 –520.

[10] R. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "Nist cloud computing reference architecture," in *Services (SERVICES), 2011 IEEE World Congress on*, july 2011, pp. 594 –596.

[11] B. Michael, G. Bernard, K. Petra, D. Robert, B. Gerd, P. Stefan, K. Heather, and A. Ali, *Introduction and Architecture Overview IBM Cloud Computing Reference Architecture 2.0*, 2011. [Online]. Available: https://www.opengroup.org/cloudcomputing/uploads/40/ 23840/CCRA.IBMSubmission.02282011.doc

[12] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "An analysis of traces from a production mapreduce cluster," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, may 2010, pp. 94 –103.

[13] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 267–280.

[14] E. Elghoneimy, O. Bouhali, and H. Alnuweiri, "Resource allocation and scheduling in cloud computing," in *Computing, Networking and Communications (ICNC), 2012 International Conference on*, 30 2012-feb. 2 2012, pp. 309 –314.

[15] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[16] M. Zakarya, I. U. Rahman, N. Dilawar, and R. Sadaf, "An integrative study on bioinformatics computing concepts, issues and problems," *International Journal of Computer Science(IJCSI)*, vol. 8, Issue 6, No 1, 2011.

[17] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M. A. DePristo, "The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data," *Genome Research*, vol. 20, no. 9, pp. 1297–1303, Sep. 2010.

[18] R. Buyya, R. Ranjan, and R. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *High Performance Computing Simulation, 2009. HPCS '09. International Conference on*, june 2009, pp. 1 –11.

Ibrahim K. Musa Received his B.Sc. (Hons.) degree in computer science and masters from Federal University of Technology Nigeria in 2006 and 2009 respectively. He was employed as lecturer in the Federal University of Technology Nigeria in 2008 and worked as a Cisco and Microsoft instructor in 2007-2009. In 2010 he started his Ph.D. in Computer Science and Electronic Engineering department of University of Essex, United Kingdom. His current research interest is resource virtualization in cloud Computing.

Stuart D. Walker received the B.Sc. (Hons.) degree in physics from Manchester University, Manchester, U.K., in 1973 and the M.Sc. and Ph.D. degrees in electronic systems engineering from the University of Essex, Essex, U.K., in 1975 and 1981, respectively. After completing a period of contractual work for British Telecom Laboratories between 1979 and 1982, he joined the company as a Staff Member in 1982. He worked on various aspects of optical system design and was promoted to Head of the Regenerator Design Group in 1987. In 1988, he became Senior Lecturer and then Reader in Optical Communication at the University of Essex. He has led eight patents and authored over 140 publications. In 2004, he was promoted to a Professorship and is currently Head of the Optical Systems Laboratory at the university. His current interests focus on modeling and analysis of advanced optical network components and systems.