CLOUD ATLAS: A SOFTWARE DEFINED NETWORKING ABSTRACTION FOR CLOUD TO WAN VIRTUAL NETWORKING

Stephan Baucke, Racha Ben Ali, James Kempf
Ericsson Research
San Jose, CA, USA
first dot last at Ericsson dot com

Abstract
One of the key principles of Software Defined Networking (SDN) is to represent networks as a collection of usable abstractions implemented as an API. Application of this principle has advanced furthest in cloud computing. The OpenStack Quantum network service provides tenants in a data center with an abstraction of an enterprise LAN. The Quantum API implements a virtual network through a plug-in that adapts the API to the physical network. However, existing OpenStack support for wide area connectivity is generally restricted, and does not support Quality of Service (QoS). In this paper, we present Cloud Atlas, a SDN abstraction and API extending the Quantum virtual network into the WAN. Cloud Atlas is built on top of existing WAN network services (L1-, L2, and L3VPNs) that do support QoS. Cloud Atlas makes these services available to OpenStack through a tight integration with Quantum. We discuss two prototypes we have built of Cloud Atlas, one based on command line scripts and one based on a network management system, and a prototype VM snapshotting service. We conclude the paper with some observations on applying the cloud service model to networking and the value of SDN abstractions in supporting such a service model.

Keywords: elastic networking; networking as a service; OpenStack; software defined networking; SDN

1. INTRODUCTION

In most areas of computer science, abstraction has been used to simplify the interfaces between system components, thereby simplifying the construction of large scale, complex systems. Networking has long resisted this trend. Shenker (Shenker, 2012) has pointed out that most research work in networking focuses on mastering the complexity of protocols and interfaces and performance measurement rather than on simplifying complexity to enable the construction of larger systems from components with well-defined interfaces. Rexford (Rexford, 2011) has described the networking literature as being concerned primarily with a proliferation of protocols, a large collection of header formats, and tools to manage them, rather than with abstracting out the essential technical features to make networking easier to use and understand.

In response to these concerns, the networking research community has moved over the last few years toward Software Defined Networking (SDN). SDN is about defining abstractions that expose the appropriate level of detail for complex network functions and implementing those abstractions in an API for programmability. Programmability facilitates automation of provisioning, configuration, and management.

The area of networking where SDN is furthest along is in cloud management/operating systems. For example, OpenStack (OpenStack Foundation, 2013a) defines a network virtualization service for data center networks called Quantum\(^1\) (OpenStack Foundation, 2013b) that provides isolation between tenants, connectivity between VMs for the tenant owning the virtual network, and an API for tenants to directly manage their virtual networks within the data center. The automation of cloud network management removes human intervention; speeding up the deployment and provisioning of tenant networks in the data center in the same way that compute and storage virtualization have for computation and storage. The Quantum API presents an abstraction of an enterprise local area network (LAN) that is easy for tenants to understand and use. Other cloud operating systems provide similar network virtualization APIs.

A complementary service for automatically setting up end-to-end virtual networks to destinations outside the cloud would be desirable. OpenStack supports an API for IPsec or SSL-based tunnels (over-the-top VPNs) between VM gateways and clients outside the cloud (OpenStack Foundation, 2013c) but over-the-top VPNs do not support quality of service (QoS). Most enterprise networks typically require QoS on their VPNs so that they are assured of not running out of bandwidth at critical times. Some public cloud providers offer provider-provisioned L3VPN services into the data center with QoS properties, but they are typically more restricted than over-the-top VPNs and require considerable human intervention when provisioned.

\(^1\)In the Havana release of OpenStack, Quantum was renamed Neutron due to a trademark conflict, but we will continue to refer to it as Quantum in this paper because the work was done exclusively with releases prior to Havana.
What would be desirable is a service offering wide area virtual networks with QoS properties integrated with Quantum and having the same dynamic elasticity and dispatchability as compute and storage resources in the cloud.

To alleviate these issues, we propose Cloud Accessible Transport Layer Service (Cloud Atlas), an SDN abstraction for cloud to WAN virtual networking. Cloud Atlas provides an abstraction of inter-site WAN connectivity, similar to the virtual switch abstraction of local-area connectivity, making wide-area connectivity accessible through the cloud operating system. An easy-to-use API implementing the abstraction allows wide area connectivity to be provisioned by the cloud tenant, thus allowing the tenant to construct an end-to-end virtual network from the data center to destinations in the WAN. The abstraction allows network providers to make their wide-area network assets tenant-manageable and enables tenants to manage their end-to-end virtual networks in coordination with compute/storage resources. Cloud Atlas can additionally be used to manage over-the-top VPNs if desired. The abstraction is largely independent of the underlying physical implementation, which is provided by a plug-in.

In Section 2, we discuss the technical contribution of Cloud Atlas with respect to related work. Section 3 briefly reviews the OpenStack cloud operating system, with focus on the Quantum intra-data center virtual networking component, upon which Cloud Atlas is based. While we believe the general approach is applicable to other cloud operating system, we used OpenStack for our prototype because of its modular architecture and the Quantum virtual networking system. Section 4 presents the abstractions and algorithms in Cloud Atlas and describes how it is integrated with OpenStack and the underlying WAN. In Section 5, we briefly present a couple of prototypes we have built with Cloud Atlas and Virtual Private LAN Service (VPLS) (Augustyn, 2006), a L2VPN service built on top of Multiprotocol Label Switching (MPLS) (Rosen, 2001). VPLS is a widely deployed L2VPN service for connecting enterprise sites and data centers. In Section 6, we describe a few use cases of how Cloud Atlas could implement connectivity services that are difficult to achieve with existing over-the-top, data center to WAN solutions and describe a particular use case, VM snapshotting, which we have implemented. Finally, in Section 7, we summarize the paper and present a few conclusions.

Our contribution is a set of abstractions for virtualized, on demand, QoS-provisioned wide area network access that bring the same kind of elastic “as a service” flexibility to wide area networking as OpenStack and other cloud operating systems bring to networking in data centers. Note that, unlike much other work in networking, our focus is not on performance measurement or analysis but rather on ease of programmability. Cloud Atlas uses whatever underlying wide area VPN technology is available, and therefore the performance will be driven by the performance of that technology.

2. PREVIOUS WORK

Many research projects and most commercial offerings utilize IPsec or SSL VPNs running inside VM gateways to provide cloud to cloud or cloud to user connectivity. Virtuoso, VIOLIN, and QCG (Sundararaj, 2004), (Ruth, 2006), (Bosak, 2012) are three academic projects that provide VPN connectivity between nodes in a grid for grid computing using over-the-top VPNs. Most commercial cloud providers such as Amazon offer IPsec or SSL VPNs for connectivity between enterprise networks and the cloud data center (Amazon, n.d.a). Over-the-top VPNs don’t provide guaranteed QoS Service Level Agreements (SLAs) which most enterprises demand and which are necessary for virtualizing telecommunication services.

Some commercial cloud providers with international data networks offer provider-provisioned L2VPNs with guaranteed SLAs. Savvis offers a L2VPN service for connecting enterprises and data centers based on MPLS VPNs with 6 QoS levels (Savvis, n.d.) called Application Transport Services. While this addresses the need for QoS, the service does not provide end-to-end virtual networking with the same level of elastic dispatchability as virtualized compute and storage. Such features, together with an API integrated into the cloud operating system, are necessary if the service is to be tenant provisioned. Amazon also offers a service called Direct Connect (Amazon, n.d.b) integrated with their Virtual Private Cloud service which consists of BGP peering through a dedicated 1Gb or 10Gb fiber Ethernet link. L2 bridging, which is required for L2VPN, is not supported however.

A few research projects have also explored using L2VPNs with elastic dispatchability. Baldine, et al (Baldine, 2010) describe an architecture and testbed for a distributed network of data centers with emphasis on integrating networking. The control framework is implemented within the ORCA extensible control platform (RENCI, 2013) for leasing resources in a shared infrastructure. The testbed was developed as part of the NSF GENI project (GENI, 2013). In their architecture, the user interacts with brokers to obtain tickets for resources at sites. The user then redeems the tickets for utilizing the resources at the sites. In order to facilitate having the framework understand and reason about heterogeneous resources owned by different owners, the authors developed an ontology using the OWL language. The language is applied to the problem of stitching together VLANs end-to-end to provide a slice of the network. The authors developed three models of stitching – hop-by-hop, centralized, and co-ordinated. The models are distinguished by what entity is responsible for the stitching.

Wood, et al in two papers (Wood, 2009) (Wood, 2011) describe how WAN connectivity resources can be more tightly integrated into the cloud, so that L2VPNs can be
elastic and dispatchable like other cloud resources. In the first paper, the authors describe an abstraction of a virtual data center and a VPLS service connecting the data center to an enterprise or another data center that they call a virtual private cloud. In the second, they describe an implementation of a particular use case, involving moving a VM from one data center to another. For this, they modified the Xen hypervisor to accommodate the lower bandwidth available on wide area networks. Their focus is on traditional networking topics, like improving performance over the wide area link under the load of VM movement, rather than on developing an abstraction that programmers would find easy to use.

In an approach similar to that of Baldine, et al, Schaffrath, et al describe a resource description language for configuring virtual networks called FleRD (Schaffrath, 2012). The language is designed to be used in a particular business ecosystem with virtual network providers offering resources from physical network providers (Schaffrath, 2009). In contrast to similar efforts, the language allows the specification of vague requirements, e.g. that a network connection not run thorough Afghanistan. The language covers virtual servers and storage as well as network connections, and is designed to be a general solution for describing cloud deployments. A generic NetworkElement type describes network objects and a NetworkInterface type allows network objects to be interconnected. NetworkElement objects are associated with two sets of attribute-value pairs: Resource and Features. Resource models any characteristic that can be measured (e.g. bandwidth) while Features indicate characteristics that are qualitative (e.g. geographical location). The virtual service instance description is then handed off to a virtual network provider who brokers the resources from a physical network provider. With FleRD, it should be possible to compose simple qualitative features and build measurable resources with, for example, a lower bound on latency between two geographical locations, or other composed qualitative features (LAN, MAN or WAN).

In contrast to the work of Baldine, et al, Cloud Atlas leaves the details of constructing the end-to-end path to existing underlying WAN mechanisms, and concentrates on the case where the end-to-end path through the WAN is controlled by a single network operator. Stitching is only required between the WAN VPN and the Quantum VPN, and that is handled by the Cloud Atlas elastic networking subsystem. In comparison with the work of Wood, et al, the underlying L2VPN technology used in Cloud Atlas is the same (VPLS), but Cloud Atlas is more focused on integrating WAN networking into the cloud operating system and providing a simple API to allow easy tenant self-provisioning. The Cloud Atlas API could in principle be used with any type of underlying VPN technology even L1VPNs, since it hides the complexity of WAN VPN provisioning. Cloud Atlas also focusses specifically on the WAN network API unlike the work of Schaffrath et al. The abstractions Cloud Atlas presents to the cloud OS tenant are targeted at WAN networking rather than being generic, because there are differences in offered latency and bandwidth and in topology between WAN and internal cloud LAN services that tenants may need to be aware of. Finally, unlike the work of both Baldine et al and Schaffrath et al, Cloud Atlas is an API rather than a language. An API seems a better match for cloud operating systems like OpenStack, and the goal in Cloud Atlas was not to provide a way of describing virtualized systems across providers, as it was in the two earlier works, but rather to provide end-to-end, cloud to WAN virtual networking with QoS that tenants can self-provision across a single provider.

3. THE OPENSTACK CLOUD OPERATING SYSTEM

OpenStack is an open source IaaS cloud operating system, developed under the Apache 2.0 license. Development is managed by the OpenStack Foundation (OpenStack Foundation, 2013a). OpenStack consists of services that manage compute, storage, and networking resources. Additional functions such as an internal message bus and GUI components are also supported. OpenStack operates in the layer above the virtualization layer. It strives to be hardware and hypervisor agnostic and currently supports several major hypervisors as well as a variety of hardware configurations.

OpenStack employs a modular structure consisting of services for specific tasks. In our work, there are three specific services we utilized:

- OpenStack Compute Service ("Nova"), which manages the activation of VM images,
- OpenStack Image Service ("Glance"), which manages and delivers virtual disk images in a variety of formats,
- OpenStack Identity Service ("Keystone"), which provides token-based identity and policy services for use by other OpenStack components,
- The OpenStack dashboard ("Horizon") which provides a graphical user interface for managing an OpenStack data center,
- OpenStack Networking service ("Quantum") (OpenStack Foundation, 2013b) which provides a multi-tenant, intra-data center Layer 2 service modeled on an enterprise LAN.

We built the cloud-managed and network-managed prototypes in Section 5 using Quantum and Keystone and the cloud bursting VM snapshot service described in Section 6 using Nova, Glance, and Horizon.

The Quantum service offers exactly the kind of flexible, elastic support for “Networking as a Service” within the data center that we seek to achieve with Cloud Atlas. Quantum provides simple abstractions and an extensible API that allows cloud tenants to configure and manage

http://hipore.com/ijcc
virtual network topologies and to plug advanced network services such as firewalls into tenant networks.

The base abstractions used by Quantum are Virtual Networks and Virtual Ports. Virtual Networks provide an isolated virtual Layer 2 service upon which tenants can build. Virtual Ports are connection objects where Virtual Interfaces, for example the virtual NICs of virtual machines, can be attached to a Virtual Network. Together, these objects provide a generalized abstraction conceptually similar to the virtual distributed switches that are utilized in many cloud deployments today. However, Quantum is not restricted to any particular underlying network technology (such as VLANs) to implement the virtual tenant networks on the physical hardware. Rather, it employs a plug-in architecture that allows the implementation of the Quantum services using a variety of different open or proprietary network technologies. In this way, Quantum opens up the cloud operating system for new and advanced SDN technologies.

4. CLOUD ATLAS

4.1 OVERVIEW

Cloud Atlas is designed to work with a variety of underlying WAN VPN technologies via a plug-in architecture, and to be interoperable with the OpenStack Quantum service, so tenants can use Cloud Atlas to interconnect Quantum networks with remote sites. To this end, Cloud Atlas performs three main tasks:

- Expose an abstract API that allows the cloud tenant or a higher-level orchestration layer to request and manage WAN connectivity to remote sites, and to interconnect remote sites with the local tenant networks in a dynamic and elastic fashion, thereby achieving end-to-end connectivity for cloud tenants.
- Interact with the local cloud OS used at the sites to interconnect (stitch) the WAN connections with the local tenant networks.
- Interact with an underlying network infrastructure to provision virtual inter-site connections across the WAN. Existing WAN virtualization techniques, for example L2VPN, are used to support multi-tenancy over the WAN.

By providing an abstraction layer and generic cloud OS API for WAN connectivity (rather than requiring cloud tenants to deal with the details of specific WAN technologies themselves), the system simplifies the framework that cloud and network operators use to integrate existing WAN services transparently into the cloud. The API can either be used directly by a cloud tenant, or programmatically by a higher-level distributed cloud orchestration system.

4.2 CLOUD ATLAS COMPONENTS

The Cloud Atlas system consists of three types of components, shown in Figure 1:

- The Elastic Networking (EN) Manager coordinates cross-site connections and provides the Cloud Atlas API for use by tenants or orchestration systems. The EN Manager hosts the Cloud Atlas public API which is exposed to tenants. In addition, it can host plug-ins that configure networks or network elements which do not belong to either of the participating sites, for example by accessing a wide-area network management system (NMS).
- Each participating site runs an EN Agent. The Agents carry out functions that require local access at the site. They interact with the local OpenStack instance to stitch the WAN connection and the local Quantum networks together. They allow the EN Manager to discover the gateways that are available at the site which can be used for WAN connections. They also can host plug-ins to configure internal network elements if required for a specific wide-area network. For example, this can include the configuration of gateways or customer edge routers. In addition to these functions, the agents enable the site's operator to enforce policies for accessing the local resources. The EN Manager interacts with the EN Agents through the Management API.
Network-specific plug-ins, indicated by EN Plug-in in the figure, can be hosted by both the EN Manager and the EN Agents. EN Plug-ins implement functions that are specific to the underlying WAN network deployment and hardware, for example, a CLI script manager for configuring a VPLS service on a particular kind of router.

For each site that wants to participate in cross-site connections managed by Cloud Atlas, an EN Agent needs to be registered with the EN Manager. The EN Manager then interacts with the EN Agents through a well-defined management API not available to tenants. While the EN Agents need to be hosted at their sites, the EN Manager can run anywhere, as long as control plane connectivity with the EN Agents is provided. Typically, it will either run at one of the sites, or together with a higher-level orchestration system at a separate location.

### 4.3 ABSTRACTIONS AND API

Cloud Atlas uses a set of abstractions, similar to the abstractions used by Quantum, which each site exposes to enable the tenant or an orchestration system to create end-to-end connectivity between virtual networks at multiple sites. The main object types are illustrated in Figure 2.

A Site is a collection of cloud resources managed by an instance of a cloud operating system. Each Site is associated with an EN Agent. Sites host Gateways that provide connectivity between local and wide-area networks. Each Gateway has a type representing a particular wide-area network virtualization technology (e.g. IPSec VPN, VPLS, etc.), and is associated with a wide-area network domain. A Virtual Link is a virtual point-to-point connection between two Gateways of the same type at different Sites. A Virtual Port is a logical endpoint of a Virtual Link and represents the local gateway configuration state associated with that Virtual Link. Finally, a VPN is a virtual multipoint-to-multipoint network that connects two or more sites using Virtual Links of the same type. The VPN object itself has a type as well, which corresponds to the common Virtual Link type. Only Sites with a Gateway supporting this type can be added to the VPN.

**Figure 1. Cloud Atlas Architecture**

**Figure 2: Cloud Atlas Abstractions**

**Public API.** The EN Manager exposes a number of operations to tenants or an orchestration layer, based on the abstractions described above. These include the creation of VPN objects, addition of Sites to VPNs, Gateway discovery, and the creation and management of Virtual Links. A Virtual Link re-configuration method can be used to dynamically change the bandwidth. Finally, the API supports a method for retrieving and collecting Quantum-compatible Virtual Interface Identifiers corresponding to Virtual Ports from the EN Agents. These identifiers are forwarded to the tenant who uses them to plug a Virtual Link endpoint into local Quantum networks at the participating sites. The EN Agents stitch EN Virtual Ports to...

http://hipore.com/ijcc
Quantum virtual ports (e.g. using VLANs between EN GWs and Quantum GWs.) creating end-to-end connectivity.

**Management API.** In addition to the public API, a well-defined management API is used by the EN Manager to communicate with the EN Agents. This API provides access to local operations at the participating sites. The management API includes methods for the discovery of local Gateways on behalf of the EN Manager, the management and configuration of Virtual Ports in the local Gateways, and the reporting of local status information to the EN Manager. The Agent also offers a method for the construction of Quantum-compatible **Virtual Interface Identifiers** in cooperation with the local Quantum service as already mentioned.

In our prototype, the APIs are implemented as RESTful interfaces, similar in style to the existing OpenStack APIs. Since the APIs need to support a variety of network technologies with different capabilities, some of the parameters are optional. This particularly affects QoS parameters, since different network types have different QoS capabilities. Over-the-top VPNs don’t support QoS, for example. The methods for the creation and configuration of **Virtual Links** and **Virtual Ports** take this into account by supporting optional parameters in the form of optional JSON or XML attributes.

The management API also supports the transport of opaque objects between the EN Agents and the EN Manager. This accommodates the needs of the network technology-specific EN plug-ins. For example, in our prototype implementation using VPLS as the underlying VPN technology and CLI scripts to implement the router configuration, the MPLS labels used to implement the VPLS tunnels are negotiated between sites. As another example, the mechanism could be used to exchange keys for an IPSec VPN.

**Interaction with OpenStack.** After the creation of a new virtual link over the WAN, the EN Agents interact with the local OpenStack instances at their respective sites to create a binding between the **Virtual Port** representing the local endpoint and a **Virtual Interface Identifier (VIF-ID)** that is compatible with Quantum. This identifier can then be retrieved by the tenant or orchestration layer, and can later be used to “plug” the **Virtual Link** endpoint into a local Quantum network, creating the interconnection between the virtual wide-area link and the local virtual network.

Cloud operating systems typically use specific abstractions like virtual distributed switches (VMware, n.d.) or, for OpenStack Quantum, more generalized abstractions like virtual networks to manage local-area virtual networks. These abstractions include the notions of virtual ports and virtual interface objects. One of the main tasks of the virtual network management system in the cloud OS is to maintain a mapping between virtual and physical ports and interfaces. The EN Agents need to perform a stitching operation in cooperation with the cloud OS to create such a mapping for a specific virtual interface that represents the endpoint of a **Virtual Link**. The result of the stitching operation is a **VIF-ID**, which can, for example, represent a specific VLAN on a specific internal port of the gateway which bridges between the site’s local network and the WAN.

The stitching operation depends on the cloud OS and the underlying WAN technology. In the case of OpenStack using Quantum networks, the binding operation depends on the active Quantum plug-in. For example, in our prototype implementation of an cloud-managed virtualization deployment, we used the Nicira NVP (Nicira, 2012) system with the corresponding Quantum plug-in. In this case, the **VIF-ID** is constructed using an NVP identifier referring to the NVP Gateway and a VLAN tag that is chosen by the Elastic Networking Manager for each virtual link. The NVP **Gateway** identifier is looked up in a pre-configuration table that maps the EN **Gateway** objects to their corresponding NVP identifiers. This table is maintained by the EN Agents. The **Gateway** identifier along with the VLAN tag uniquely identifies a **Virtual Port** on the gateway. The **VIF-ID** is then constructed by putting the **Gateway** identifier and VLAN tag into a special text format that is understood by the Quantum service.

The method described in the previous paragraph is just an example implementation of the stitching operation. For the virtual network management components in other Quantum plug-ins, and potentially in other cloud operating systems, EN Agents that are specific to the cloud OS implement the specific binding operation for each cloud OS.

**Interaction with the Underlying Network.** Cloud Atlas has two axes of variation in interaction with the underlying network. One axis is what particular underlying WAN VPN technologies are supported. The other is how the existing physical network deployment and hardware is programmed to deploy the WAN virtual network. These factors determine how the EN Manager, EN Agents, and network specific plug-ins interact with the network.

Regarding the first point, the capabilities available to clients of the API depend on the capabilities of the supported VPN technologies. In particular, if the virtual network is to support SLAs, it requires virtualization capabilities that enable the partitioning of the existing physical bandwidth to support multi-tenancy. Virtualization can be performed at the edges with over-the-top VPNs at Layer 3, in which case no SLAs are supported, with logical overlays at Layer 3 or Layer 2, e.g. in the case of a managed MPLS or a carrier Ethernet (Metro Ethernet Forum, 2010) network in which case SLAs in the megabit/second range can be supported, or with dedicated physical bandwidth at Layer 1, e.g. in the case of Layer 1 VPNs running over a dedicated wavelength in a DWDM network in which case SLAs in the gigabit/second range can be supported. If a particular VPN technology does not support some of the parameters in an API call, they are ignored if provided.

Regarding the second point, to support over-the-top VPNs, the EN Manager only needs to support VPN gateways that terminate the IPsec or SSL session in virtual

http://hipore.com/ijcc
servers in the data centers or virtual or dedicated physical servers in the enterprise networks. These can typically be handled by the agent with a plug-in running in one of the participating sites. If the supported technologies include L3VPNs (based on MPLS), L2VPNs, or L1VPNs, then the EN Manager either needs code specific to the gateway networking equipment in the data centers and in the intervening network or must use a network management system (NMS). With a small network, a collection of command line interface (CLI) scripts is sufficient, but for larger, multivendor networks, an NMS is essential, particularly for topology management and path computation. An NMS will typically work through element managers that handle a specific vendor’s equipment, in which case, the EN Agents play no role in configuring the hardware. If all the intervening networking equipment is OpenFlow enabled, such as for example in Google’s Gscale network (Open Network Foundation, 2012), an alternative implementation could utilize Cloud Atlas as a northbound API to an OpenFlow controller (McKeown, 2008).

5. PROTOTYPE IMPLEMENTATIONS

We have deployed a testbed consisting of two simulated mini-data centers connected by an emulated wide area networking link within a lab. The “data centers” are two racks of HP servers, one containing 19 HP DL165G5s (HP, 2011) with 32Gb of memory and 160Gb of disk, and the other containing 10 HP DL380G7s (HP, 2013) with 36 Gb of memory and 480Gb of disk. One of the DL380s was upgraded with 2 TB of storage to serve as a NAS for both data centers, so network attached storage is available. The two racks are on separate subnets connected through two 48x1G Ethernet top of rack switches to a wide area network emulation. For the cloud-managed virtualization deployment, the data centers ran OpenStack Essex (OpenStack Foundation, 2012) with Nicira NVP (Nicira, 2012) for Quantum virtual networking. For the network-managed virtualization deployment, the data centers ran OpenStack Grizzly (OpenStack Foundation, 2013d) with the Open Virtual Switch (OVS) (Open vSwitch, 2013) plug-in for Quantum virtual networking.

5.1 CLOUD-MANAGED VIRTUALIZATION

The prototype of a cloud-managed deployment is shown in Figure 3. We connected the top of rack switches to two Ericsson SmartEdge (SE) 800 routers (Ericsson, n.d.a) and connected the routers together though a 1G link to simulate the WAN. In each data center, the SE 800 acts as the Provider Edge (PE) device. NVP uses IP overlay networks to implement per-tenant virtualized network slices within the data center. The NVP Gateway acts as the Customer Edge (CE) device. The Gateway terminates the per tenant overlays for connections outgoing from the data center and trunks them into a VLAN trunk which the SE800 then inserts into a per tenant VPLS VLAN. Note that a deployment based on over-the-top, user deployed IPSec or SSL VPNs would have the same structure, with the routers replaced by VMs running the VLAN termination gateway.

The EN Agent configures the SE 800s in response to API calls using CLI scripts to set up on-demand, per-Virtual Link VPLS pseudowires on fixed MPLS label switched paths. Elastic partitioning of the available aggregate bandwidth is achieved by performing traffic shaping on the gateway routers. Since there are no network devices between the data centers, the EN Manager is only responsible for ensuring the two data centers agree on the per tenant MPLS label. Cloud-managed deployments would typically only be useful in simple networks when the routes between edge routers are fixed, since no topology discovery or path computation services are provided, or for over-the-top VPNs.
A typical call sequence to create an end-to-end Layer-2 network spanning two OpenStack sites A and B using the public EN API and the local Quantum APIs in an cloud-managed deployment is shown in Figure 4. The steps are as follows:

1. Create Quantum networks QA and QB, and ports, PA and PB in QA and QB, at both sites.
2. Discover Gateways of a common VPN type at sites A and B. The EN Manager passes the calls along to the EN Agents in sites A and B, which return the local Gateways.
3. Create a new VPN of the type found in step 2,
4. Add Sites A and B to the new VPN,
5. Create a Virtual Link between A and B. The EN Manager executes a plug-in to configure the gateway routers with CLI scripts. The EN Manager calls the Create Virtual Port method in the EN Agents for sites A and B, passing to them URLs to use as a callback for reporting on link connection status,
6. The Tenant polls the EN Manager for link status. The EN Manager starts threads to respond to periodic reports on status from the EN Agents at sites A and B. The EN Agents call their plug-ins, which send CLI scripts to the PE routers to set up the Virtual Ports. Call back threads running in EN Agents at sites A and B update the link status at the EN Manager after they finish configuring the gateways. When both EN Agents report that the connection is established, the next poll by the Tenant reports “Connected”.
7. Retrieve Virtual Interface IDs (VIF-IDs) VA and VB for the Virtual Link endpoints at sites A and B.

http://hipore.com/ijcc
8. Call the Quantum instances at sites A and B to plug VA into PA and VB in PB.

The Virtual Link re-configuration method can be called at any time after creation of the VPN link to change the bandwidth allocation.

5.2 NETWORK-MANAGED VIRTUALIZATION

The prototype of a network-managed virtualization deployment is shown in Figure 6. We removed the routers and connected the two Ethernet switches to Ericsson SPO 1410 optical/Ethernet switches (Ericsson, n.d.b) acting as the PE device which are connected together to simulate the WAN. There is no CE device in this deployment, the VLANs are trunked on the uplink from the top of rack switch to the SPO. Inside the data centers, we used the OVS plug-in for implementing the Quantum virtual networks. The OVS plug-in isolates tenants based on VLAN id. The per-Virtual Link VPLS pseudowires are configured through Ericsson’s IP Transport Network Management System (IPT-NMS). IPT-NMS (Ericsson, n.d.c) manages L3- and L2VPNs across a connected network of routers and switches. The IPT-NMS exports a CORBA northbound API which is used by the EN Manager to set up the networking. An EN Agent is also not needed in the individual data centers, so the EN Agent functionality was folded into the EN Manager. Bandwidth partitioning is achieved directly by the SPO 1410s. Note that the NMS could be running anywhere, and simulates the situation where an operator OSS is managing the WAN.

Using a network management system simplifies topology discovery and path computation in complex networks, since these functions are typically built-in. Though we only had two optical switches in our testbed, a network-managed deployment could in principle manage a WAN consisting of routers and switches. MPLS-TP (Bocci, 2010) establishes an underlying infrastructure VPN mesh connecting all the switches and routers between the data centers (in our case, only two). In an actual deployment, the infrastructure VPN is set up by the network operator and data center operators, not by the tenant. The VPLS service is then established on demand by the tenant when bandwidth is needed.

In Figure 7, the message sequence diagram for the network based deployment is shown. The numbers correspond to the following steps:

1. As in Figure 5, the tenant first creates Quantum networks QA and QB in sites A and B. In this figure, we assume sites and VPNs have already been associated so we leave out the steps of creating a VPN and adding sites to it to simplify.
2. The tenant asks the EN Manager for a list of sites.
3. In a loop, for each site, the tenant queries the EN Manager for a list of Gateways.
4. The EN Manager obtains authorization tokens from the OpenStack Keystone authentication and authorization managers running in sites A and B. It uses the authentication tokens to obtain a list of Quantum networks for the tenant. Note that in order to obtain the right parameters for stitching the Quantum network to the VPN, the EN Manager must have administrator privileges. Step 4 can proceed independently of the other steps.
5. The tenant requests creation of a virtual link between the two sites. It passes in the Quantum uuids for the virtual networks in the two sites and the Gateway ids. The EN Manager returns immediately with an indication that the virtual link
has been created but is in the state “Disconnected” (indicated by the red VIF id).

6. The EN Manager sends queries to Quantum in both sites to obtain the segmentation ids for the virtual networks to be stitched to the external VPN. Quantum has no specific query for obtaining the segmentation id, it is returned as a side effect of a query for the networks associate with a particular tenant if the query is accompanied by a token with administrator privileges. Because the query is accompanied by a token with administrator privileges, the VLAN ids for the two networks are returned. The EN Manager then makes a CORBA call (indicated by the dashed line) to the NMS system with the two VLAN ids and requests the NMS system to establish VPLS service. This is done in a separate thread to prevent the EN Manager from blocking.

7. The tenant polls the EN Manager for the status of the link. The EN Manager returns the Status as “Connecting” (indicated by orange).

8. When the NMS is finished creating the service, it returns a CORBA call to the EN Manager (indicated by the dashed line) with the status indicating that the link is up. The next time the tenant polls, the EN Manager returns the status as “Connected” (indicated by green). The link is now ready for the tenant to use.

5.3 COMPARISON AND FUTURE WORK

In both the cloud-managed and network-managed implementations, the actual configuration of the VPLS service is asynchronous. In the cloud-managed implementation, the EN Manager hands the orchestration
layer a callback URL that returns when the configuration is complete. In the network-managed implementation, the client must periodically poll the EN Manager. Depending on the size of the WAN and the network equipment spanning it, the configuration of a VPN service may require more time to complete.

Going forward, we plan to support at least one additional VPN type in order to ensure that the API has sufficient flexibility. We also plan to tie into the OpenStack role-based authorization framework and link that with WAN security, so that changes to networking are authorized. Finally, we are implementing a generalized stitching API in OpenStack to allow external provider networks to be stitched to Quantum networks.

6. APPLICATION USE CASES AND PROTOTYPE SERVICE

6.1 USE CASES

Existing over-the-top IPsec and SSL VPNs provide virtual, isolated, elastic connectivity to applications running in the cloud. What Cloud Atlas provides in addition is the support for SLAs to set network bandwidth and possibly maximum latency using enterprise VPNs. In addition, many WAN VPN technologies also feature the ability to extend a corporate LAN into the cloud, rather than just provide a simple point-to-point connection. Applications that require these properties are most likely to benefit from Cloud Atlas.

Bandwidth calendaring. Bandwidth calendaring is an emerging application in which bandwidth is scheduled between an application in the cloud and either another cloud or the enterprise for particular times when backups or bulk data transfer are required. The cloud tenant only pays for the connectivity when it is provided, just as with compute and storage. Typically the network operator can provide larger bursts of bandwidth when the link doesn’t need to be shared by multiple customers, so the tenant’s job completes more rapidly. This can lead to higher reliability, since if the link goes down, only one tenant’s job will be affected.

Disaster recovery. Disaster recovery requires periodic snapshots of running VMs into another data center. These snapshots can be made by bringing up the virtual WAN link, snapshotting the VM, then bringing it down. Disaster recovery is distinguished from bandwidth calendaring in that the bandwidth for the VM copy is not scheduled, it is brought up on-demand at a snapshotting trigger. The snapshotting trigger is a minimal atomic transactional change to a VM memory/virtual disk that can be applied without losing its consistent state. If the data center in which the original VM was running crashes, the archived copy can be quickly brought up, restoring the service.

Hybrid cloud. Hybrid cloud is a service model in which a private data center utilizes service in a public cloud when resources in the private data center are short. With hybrid cloud, the private data center owner can provision the data center for average load and burst traffic to a public cloud when peaks occur, thus saving the physical infrastructure cost of supporting peak load. What Cloud Atlas brings to hybrid cloud service is the ability to rapidly and elastically move an entire corporate LAN into the cloud on demand, rather than just a single application. In addition, because Cloud Atlas supports SLAs, telecommunication applications like real time voice or video over IP can be provided with guaranteed QoS.

6.2 SNAPSHOTTING PROTOTYPE

We implemented a VM snapshotting service using Cloud Atlas. The tenant snapshots a VM through an extended version of the OpenStack Horizon dashboard into the Glance image service in the other data center. Figure 8 shows a screen shot of Horizon during the snapshot process. The process starts with the user setting up an inter-data center VPN between the two tenant networks. When the snapshot is complete, the new VM is brought up. Unlike VM movement, though, snapshotting does not keep the old VM running, but rather shuts it down until the copy has completed, then starts the new VM when the snapshot is complete. This allows the process to proceed much more quickly than VM movement, since data pages touched by users need not be recopied.

Establishing a VPN through the Cloud Atlas API takes around 20 seconds on our simple two node lab network. In a real network, this should be dependent on the load currently being handled by the network management system, the number of network elements, the status of the network elements, and on other factors. Snapshotting a 1.9 Gb image from the Glance service in one data center to Glance in the other takes around 30 sec, but this time is configurable depending on VPN bandwidth. The snapshot application allows the user to set a time constraint for the transfer and then scales the bandwidth on the VPN to accommodate the time constraint. Currently snapshotting is not automated, but automation could allow periodic snapshots to be taken and have them start if the originating data center goes offline. If the two data centers are in the same metro area, the connectivity could be sufficient to ensure uninterrupted service should the originating data center go down.
Our main intention in introducing the Cloud Atlas elastic networking abstractions is to simplify the amount of programming effort needed to implement applications. To give an idea of how much effort was required to implement the checkpointing application; we estimate how long it took for a single programmer to implement the various components:

- Script-based SmartEdge agent: 1 day
- NMS-based agent: 3 months
- Checkpointing application on Horizon: 2 days.

The NMS-based agent took substantially longer because it required co-ordination with the NMS development group and enhancement of the NMS API. But once the agent was complete, developing an application on top of it only required a few days. Development time for another application would also be reduced. This illustrates the leverage programmers obtain from having abstractions rather than having to deal with each piece of networking gear and protocol separately.

7. SUMMARY AND CONCLUSIONS

In this paper, we have described Cloud Atlas, a collection of abstractions implemented by an API for elastic networking between a cloud OS and the WAN. The Cloud Atlas API allows the tenant to dynamically create and configure VPN connections and to plug these connections into their intra-data center virtualized slices, thereby creating end-to-end virtual network slices with QoS between a tenant’s VMs in a data center and clients and servers outside the data center. With Cloud Atlas, networking becomes elastic and dispatchable, exactly as for computation and storage.

Cloud Atlas supports a broad variety of WAN VPN technologies, but is limited in the services that it can support by the underlying VPN technology plug-in. For example, if the VPN technology does not support bandwidth SLAs, then neither will Cloud Atlas. In addition, if a physical connection does not exist, human intervention will be necessary to make it. The analogy with a data center is when a new rack of servers or a new storage array must be installed.

Despite this limitation, we believe that Cloud Atlas has the potential to revolutionize access to the WAN in exactly the same fashion as elastic computation and storage have for data centers. Cloud Atlas reduces the complexity of deploying end-to-end virtual networks by not modeling the details of the particular VPN technologies. Rather, it provides a convenient tenant-friendly abstraction with the details tenants care about that should be adequate for most tenants. By reducing the time and complexity of provisioning VPN service, the deployment of new applications that require provisioned service with SLAs can be automated on cloud platforms just as the current three tiered Web applications have been. Existing provider-provisioned WAN services become self-provisioned for tenants due to the simplicity of the SDN abstractions, reducing operations costs for the network operator. With Cloud Atlas, network operators can aggregate multiple data centers into a true distributed cloud computing facility.

http://hipore.com/ijcc
8. ACKNOWLEDGMENTS

The authors would like to thank Martin Casado and the other folks at Nicira (now VMware) for providing early access to their network virtualization product NVP. Ramesh Mishra at Ericsson Silicon Valley implemented an early version of the cloud-managed virtualization deployment, and Franco Ferioli and Angelo Carossino at Ericsson Italy were responsible for the implementation of the IPT-NMS CORBA adaptor.

9. REFERENCES


http://hipore.com/ijcc


Authors

Dr. Racha Ben Ali earned a PhD. at École Polytechnique Montreal University in 2008 and began working for University of Montreal as a postdoc for a year then at University of Quebec at Montreal for few months then at Ericsson Research as a postdoc. In 2012, he moved to Ericsson Research Silicon Valley where he works on OpenStack and cloud computing. Between university and Ericsson, he was working as a VoIP consultant in a Montreal startup and a network architect at Tele-university of Quebec.

Stephan Baucke is a Senior Researcher at Ericsson Silicon Valley, where he is active primarily in the fields of network virtualization, software-defined networking, and cloud network management. Prior to his current position, he was leading the network virtualization work package in a major European Union research project, and before that he was working in a variety of projects in Europe and the United States involving wireless IP networking, protocol optimization, and signaling transport. Early projects included the design and implementation of the optimized transport protocol for Ericsson's first public EDGE demonstrator.

Dr. James Kempf graduated from University of Arizona with a Ph.D. in Systems Engineering in 1984 and immediately went to work in Silicon Valley. Prior to his current position, Dr. Kempf spent 3 years at HP, 13 years at Sun Microsystems, primarily in research, and 8 years at Docomo Labs USA as a Research Fellow. Dr. Kempf worked for 10 years in IETF, was chair of 3 working groups involved in developing standards for the mobile and wireless Internet, and was a member of the Internet Architecture Board for two years. He is the author of many technical papers and 3 books, the latest of which, Wireless Internet Security: Architecture and Protocols was published by Cambridge University Press in 2008. Since 2008, Dr. Kempf has worked at Ericsson Research in Silicon Valley on Software Defined Networking(SDN)/OpenFlow and cloud computing.

http://hipore.com/ijcc
MAXIMIZING PERFORMANCE OF CO-LOCATED APPLICATIONS: EXPERIENCES AND STRATEGIES WHEN CO-LOCATING MULTIPLE JAVA APPLICATIONS ON A SINGLE PLATFORM

Zhenyun Zhuang, Cuong Tran, Haricharan Ramachandra, Badri Sridharan
LinkedIn Corporation, 2029 Stierlin Court Mountain View, CA 94043, USA
{zzhuang, ctran, hramachandra, bsridharan}@linkedin.com

Abstract
Cloud (e.g., PaaS) Computing promises a cost-effective and administration-effective solution to the needs of sharing computing resources. A widely practiced cloud deployment model is to co-locate multiple applications on a single platform. While bringing efficiency to the users thanks to the shared hardware and software, the multi-tenancy characteristics also bring unique challenges to the backend platforms. In particular, the JVM mechanisms used by Java applications, coupled with OS-level features (e.g., THP), give rise to a set of problems that are not present in other deployment scenarios. In this work, we consider the problem of maximizing performance of co-located mission-critical Java applications when deploying multiple Java applications on a single platform. Based on our experiences with LinkedIn’s platforms, we identify and solve a set of problems caused by such multi-tenancy deployment. We share the lessons and knowledge we learned during the course.

// An earlier version of this paper was accepted in the conference of IEEE Cloud 2014 (Alaska, AL, USA)
// Conference paper title: "Optimizing JMS Performance for Cloud-based Application Servers"
// The differences between the conference paper and this draft are listed at the end of the draft.

Keywords: Linux performance; Multi-tenancy; THP (Transparent Huge Pages); Java performance; Cloud Computing; NUMA

1. INTRODUCTION
Cloud Computing promises a cost-effective and administration-effective solution to the traditional needs of sharing computing resources. A widely practiced cloud deployment model is to co-locate multiple applications on a single platform. While bringing efficiency to the users thanks to the shared hardware and software, the multi-tenancy characteristics also bring unique challenges to the backend cloud platforms. Cloud platforms need to deliver high performance to all the co-located applications/users, subject to SLA (Service Level Agreement). However, since multiple applications share the same computing resources, it is important to prevent different applications from adversely affecting each other. Note that depending on specific cloud solutions, users may use a mix of dedicated resources and shared resources.

Though the general problem of isolating applications and ensuring high performance belongs to the big topic of QoS (Quality of Service), as of today, there are little satisfying solutions to isolating applications while maintaining the cost-effectiveness of multi-tenancy. Note that Some solutions provide hard isolation of resources by dedicating a specific set of hardware and OS to each application (e.g., through Virtual Machines), however it incurs additional administration cost and compromises the benefits gained by resource sharing. In many of today’s multi-tenant backend platforms such as many PaaS platforms, multiple users share the same OS (and in turn the memory and cpu resources). Reasons for adopting such setup includes ease of administration and deployment. In such multi-tenancy deployments, it is critical to protect users at OS level to ensure high performance.

Different co-located applications impact each other when sharing limited computing resources including memory and cpu. When users’ consumptions of computing resources are not aligned well due to independent activities and load, the individual user’s performance may be compromised. For example, an user that suddenly aggressively consumes memories (e.g., reading files) and causes another user to swap out its pages, when the latter resumes, its performance will be significantly penalized due to slow swap-in activities. Moreover, with the increasing popularity of multi-core and NUMA (Non-uniform memory access) ("Non-uniform", 2014) systems, additional performance overhead caused by thread migrations and context switches may cause the default application deployment model suboptimal.

http://hipore.com/ijcc
doi: 10.29268/stcc.2014.2.2.2
Given the popularity and powerfulness of Java platforms, a significant portion of today's backend platforms run Java. Though supporting multi-tenancy inside a single JVM has been proposed ("Java Multitenancy", 2014), it is yet to be verified and deployed. As of today, effectively supporting multi-tenancy with Java requires running multiple Java applications on the same platform. Java applications' performance highly depend on the use of JVM and heap spaces. We have observed in our production platforms that the interaction caused by different Java applications can cause severe performance degradations. Briefly, the competitions among multiple Java applications cause inflated application pauses during certain scenarios. Moreover, we have found that some new features of Linux platforms, though intended to improve application performance, could significantly backfire, thanks to some complicated interactions between these features.

In this work, we consider the problem of maximizing the performance of mission-critical Java applications when co-locating multiple Java applications on a single platform. Based on our experiences with LinkedIn's platforms, we identify and solve a set of problems caused by co-locating multiple Java applications. In a nutshell, we consider the performance of Java applications, and identify a set of Linux OS features (e.g., THP or Transparent Huge Pages) that can adversely affect Java application performance in certain scenarios. After root-causing the problems, we provide solutions to mitigate the challenges. Though the solutions have been verified working on Linux OS, on which most of LinkedIn products run, they can easily be applied to similar setups of other platforms to achieve better performance. More importantly, the studies gained could be incorporated into future designs of JVM or OS features.

For the remainder of the paper, after providing some necessary technical background in section 2, we then motivate the problems being addressed in this paper using two scenarios in Section 3. We then present the investigations we conducted in diagnosing the performance problems in Section 4. Based on the findings, we propose the solutions in Section 5. We perform performance evaluation and show the results in Section 6. In Section 7 we highlight the learned lessons during our investigations with LinkedIn's platform. We also present related works in Section 8. And finally in Section 9 we conclude the work.

2. BACKGROUND

We begin by providing background information regarding JVM heap management and GC (Garbage Collection), Linux memory management, and new features of THP (Transparent Huge Pages).

2.1 JVM heap management and GC

Java programs run in JVM (Java Virtual Machine), and the area of memory used by the JVM is referred to as heap. JVM heap is used for dynamic Java objects allocations and is divided into generations (e.g., Young and Old). Java objects are firstly allocated on the Young generation; when unused, they are collected by a mechanism called GC. When GC occurs, objects are checked for reference counters starting from a root object. If the reference counter of an object drops to zero, the object is deleted and the corresponding memory space is reused. Some phases of GC process require applications to stop responding to other requests, a behavior commonly referred to as STW (Stop the world) Pause. One of the important objectives of Java performance is to minimize the durations of GC pauses.

2.1 Linux memory management

Memory management is one of the critical components of Linux OS. The virtual memory space is divided into pages of fixed sizes (e.g. 4KB). Over the years, many features of memory management have been designed to increase the density and improve the performance of running processes.

Page reclaiming Linux OS maintains free page lists to serve applications’ memory request. When the free pages drop to a certain level, OS will begin to reclaim pages and add them to the free lists. When performing page reclaiming, page scanning is needed to check the liveness of allocated pages. Both background scanning (performed by kswapd daemon) and foreground scanning (performed by processes) are used. Oftentimes the foreground page reclaiming is referred to as direct reclaiming or synchronous reclaiming, which represents a more severe scenario where the system is under heavy memory pressure, and the application stops during the process.
Swapping

Swapping is designed to increase process density. When free memory is low, Linux will swap memory pages out to swap spaces to allow for new processes being invoked. When the memory space corresponding to swapped-out pages is active again, these pages will be swapped in.

THP

Transparent huge pages ("Huge pages", 2014) is a relatively new feature that aims to improve the processes’ performance. With larger page sizes (e.g., 2MB), the number of page table entries is reduced for a particular process. More virtual address translations can be covered by TLB (Translation Lookaside Buffer) lookup, hence higher TLB hit ratio (Talluri et al., 1992). Though the benefit of using huge pages has long been understood, the use of huge pages was not easy. For instance, the huge pages need to be reserved when OS starts, and processes have to make explicit calls to allocate huge pages. THP, on the other hand, promises to avoid both problems and thus is enabled by default. THP allows regular pages to be collapsed into transparent huge pages (THPs) through two types of collapsing: background collapsing by khugepaged daemon and direct collapsing by applications requesting memory. Note that we use THP to denote the feature, while use THPs to denote the transparent huge pages.

Non-uniform memory access (NUMA) is a computer memory design which becomes increasingly popular on modern platforms. With NUMA, applications running on a particular CPU can gain better performance when accessing local memory compared to accessing remote memory. OS (e.g., Linux) has advanced mechanisms to optimize the allocation of memory and execution of threads. Linux also exposes tools (e.g., numaclt) ("Control numa", 2014) to allow the control of NUMA policy for scheduling running processes and allocating memory.

3. Motivation

We first provide three motivating scenarios to illustrate the problems that Java applications may face in multi-tenant environments.

3.1 Experiment setup

The machine used in the experiments is an x86 64-based Intel(R) Xeon(R) X5650 SMP (Symmetric multiprocessing) NUMA 2-node machine. Each node has 6 cores, and hyper-threading is enabled. The OS version is Red Hat Linux 2.6.32-220.13.1.el6. The machine has totally 72 GB of physical memory. All default system configurations are used.

Two applications are used, one is written in Java, the other in C++. For easy presentation, we refer to the Java application as JavaApp, and the C++ one BackgroundApp. JavaApp is used to represent real production applications, while BackgroundApp simply consumes computer resources of memory to mimic production environments. JavaApp keeps allocating Java objects, and also periodically discards objects such that they can be reclaimed during JVM GC.

In real production environments, it is common for the system to experience memory pressure. To mimic such situations, BackgroundApp is designed to allocate a fixed amount of memory such that when running JavaApps they can experience certain level of memory pressure.

We consider two types of Java applications: throughput-oriented or response-time-oriented; so we will examine the application throughput as well as the response times experienced by the client requests. For throughput-oriented applications, the throughput is measured as how many Java objects are allocated per second. For response-time-oriented Java applications that serve client requests, ideally the user-experienced response time should be measured. Directly measuring the user response time is not easy, as each type of Java application has different request handling mechanisms. We instead measure the severity of STW GC pauses, since during these GC pauses, Java applications stop responding to the clients.

The JavaApp is started with varying heap sizes and with identical -Xmx and -Xms values. Other important JVM configurations are: UseConcMarkSweepGC, CMSsweepBeforeRemark, CMSParallelRemarkEnabled.

3.2 Scenario 1: During startup state

In this scenario, a single JavaApp is started and running. We first start the BackgroundApp to take 50 GB of memory, which leaves about 20 GB memory unused. We then start the JavaApp with 20 GB heap size.
In Figure 1(a), after JavaApp is started, it has a steady throughput of 12K/sec, which lasts for about 30 seconds. Then the throughput begins to drop sharply. The worst case sees almost zero throughput for about 20 seconds. Interestingly, after a while, the throughput comes back to steady again.

In Figure 1(b) which shows the GC pauses, we also observe similar pattern. The GC pauses initially are well below 50 milliseconds; then they jump to hundreds of milliseconds. We even see 2 pauses are larger than 1 second! After a period of about 1 minute, GC pauses drop again to below 50 milliseconds and become steady.

3.3 Scenario 2: During steady state

In this scenario, a JavaApp is started with 20 GB heap and enters steady state. Then a BackgroundApp starts and begins to allocate 50 GB of memory.

In Figure 2(a), we saw that the JavaApp achieves a steady 12K/sec throughput in the beginning. Then the throughput sharply drops to zero which lasts for about 2 minutes. From then on, the throughput varies wildly. Sometimes it comes back at 12K/sec, other times drops to zero again.

In Figure 2(b), the GC pauses are almost zero in steady state, then it rises to 55 seconds! From then on, the GC pauses vary a lot, but rarely come back to zero. Most of the pauses are of several seconds.

3.4 Scenario 3: Different deployment models

We also consider the third scenario where co-located applications are deployed with different deployment models. On NUMA machines, each node has its local set of CPU and memory. For any CPU, accessing the local memory on the same node gains better performance due to smaller access time. To facilitate the needs of finer control when deploying applications, Linux exposes a set of tools (e.g., numactl) to allow applications to bind to a particular set of CPUs and memory.

We consider two deployment models: (1) default deployment model (without CPU/Memory binding) and (2) improved deployment model (with CPU/Memory binding). We deploy two identical JavaApps on the same machine, and each JavaApp has 5GB of heap. In Figure 3 we show the averaged application throughput (over two applications) of the two models. We see that the default deployment model achieves 3.26 K/sec averaged throughput, while the improved deployment model achieves about 3.72 K/sec, a 14% difference. More importantly, from the curve, we can see that the improved (i.e., with CPU/Memory binding) deployment model has much more stable performance.

We also examined the CPU utilization of these two models and plotted the CPU idle percentage in Figure 4. We observed that the improved deployment model has about 29.8% CPU busy usage, slightly lower than the 31.9% of default deployment model, or a 7% difference. In other words, not only the default deployment has lower and unstable application throughput, but also it uses more CPU resources.
3.5 Summary

We see that in multi-tenant environments, Java applications can experience performance problems both in startup state and steady state. In addition, the default deployment model on NUMA platforms is sub-optimal. The symptoms of these performance problems as seen by users are low application throughput, increased response time and unstable performance. We will investigate the problems and root cause them in Section 4.

4. INVESTIGATIONS

We just saw that Java applications can experience significant performance degradation in multi-tenant setup. We now delve deeper into the two scenarios and identify the causes.

4.1 Investigations into Scenario-1: Startup state

In Section 3.2 we see that during the startup period JavaApp experiences undesired performance. We suspect that the undesired performance of JavaApp has to do with how JVM is started since it occurs when JVM is started. We examine the JavaApp’s RES (Resident size), which is the non-swapped physical memory a process has used. The results are shown in Figure 5(a). We see that though we specify the JVM parameters to be “-Xmx20g and -Xms20g”, JVM does not allocate the heap space from memory all at a time. Instead, the heap is allocated on the go. As more and more objects are instantiated, JVM allocates corresponding memory pages to accommodate. During the allocation process, OS checks the free page list. If the amount of free memory is below certain level, OS will begin reclaiming pages, which takes CPU time. Depending on how severe the shortage of free memory is, the reclaiming process may significantly stall the JavaApp. In Figure 5(b) we see that free memory significantly drops to a very low level. The page reclaiming process incurs CPU overhead, as can be seen in Figure 5(c).

On Linux, when available memory is low, the kswapd daemon wakes up to free pages in the background. If the memory pressure is high, the reclaiming process will free up memory synchronously. The page reclaiming mechanism of Linux is per-zone based (Lameter, 2013), which is governed by certain zone watermarks (i.e., pages min, pages low, pages high). User-land Java applications allocate memory in NUMA nodes’ Normal zones. For our setup, there are two nodes. Node-0’s pages_high is set at 21,498 pages (about 83 MB), while Node-1’s 11,277 pages (about 44 MB). We examined the free pages of Normal zones for the two nodes based on /proc/zoneinfo, and plot the amounts in Figures 6(a,b). We found that the available pages occasionally drop below the watermarks, which could trigger direct-reclaim path. When direct-reclaim occurs, Linux freezes the applications that executing the code path, thus causing high GC pauses. In addition, direct-reclaiming typically scans many pages in order to free unused pages. In Figure 6(c) we plot the number of pages scanned by the direct-reclaiming path as reported by Linux SAR utility. We see that at the peak value, every second about 48K pages (i.e. 200 MB) amount of pages are scanned by direct-reclaiming.

4.2 Investigations into Scenario-2: Steady state
In Section 3.3 we observe that the actions of other applications can severely impact the performance of a Java application. Since the total physical memory is only about 70 GB, while the two applications together request that amount of memory, our first observation is that the system is under memory pressure. We found that there are quite a lot of swapping activities going on, as seen from the Figures 7. In Figure 7(a) we see that many memory pages are swapped out. These swapped-out pages belong to the JavaApp. Later on in Figure 7(b), many pages are then swapped in. We can see from Figure 7(c) that the taken swap space is up to 7 GB. During these time, if JavaApp experiences GC activities, and GC needs to scan objects to collect dead ones. If the scanned objects are allocated on the pages that are swapped out, they need to be brought back to memory from swap space first. Swapping-in takes time, as the swap space is typically on disk drives. Thus, JavaApp see high GC pauses.

Though swapping activities affect GC pauses, we suspect that they alone are unable to explain the excessive pauses we see (i.e. 55 seconds). Our suspicion is justified by our examination of the GC pauses, many of which show high sys time. In Figure 8(a) We observed that the system is also under severe cpu pressure. The high cpu usage cannot be entirely attributed to swapping activities as swapping typically is not cpu intensive. There must be other activities contributing to the cpu usage. We examined various system performance statistics, and identified a particular Linux mechanism of THP that significantly exacerbates the performance degradation.

With THP enabled, when Linux applications allocate memory, preference is given to allocations of transparent huge pages of 2 MB size rather than the regular pages of 4 KB. We can verify the allocation of transparent huge pages in Figure 8(b), which shows the instantaneous number of anonymous transparent huge pages. Since THPs are only allocated to anonymous memory, the figure practically shows the total THPs of the system. At the peak we see about 34K THPs, or about 68 GB.

We also observed that the number of THPs begins to drop after a while. This is because some of the THPs are split in response to low available memory. When system is under memory pressure, it will split the THPs into regular pages to be prepared for swapping. The reason for splitting is because currently Linux only supports swapping regular pages. The number of splitting activities are seen in Figure 8(c). We see that during the five minutes, about 5K THPs are split, which corresponds to 10 GB of memory.

At the same time, Linux attempts to collapse regular pages into THPs, which requires page scanning and consumes cpu. There are two ways to collapse: background collapsing and direct collapsing. Background collapsing is performed by khugepaged daemon. We occasionally observed that the khugepaged daemon tops cpu usage. Direct collapsing by applications trapping into kernel and allocating huge pages has even more severe consequences in terms of performance penalty, which can be seen by direct page scanning count.

An even more troublesome scenario THP can ran into is that the two contradicting activities of compacting and splitting are performed back and forth. When system is under memory pressure, THPs are split into regular pages, while a short while later regular pages are compacted into THPs, and so on and forth. We have observed such behaviors severely hurting application performance in our production systems.

4.3 Investigations into Scenario-3: Different deployment model

In Section 3.4 we observe that the default deployment model does not result in optimal performance of co-located applications on NUMA platforms. The application performance can be improved by enforcing memory and cpu binding. These results highlight the necessity of optimizing the deployment model when co-locating multiple applications.

Facing multiple nodes, OS running on NUMA needs to make two critical decisions: (1) decide which memory needs to be allocated when applications request memory; (2)
decide which CPU should execute a running task (i.e., thread). Though Linux, as well as other modern OS, employ sophisticated algorithms to allocate memory when applications request memory and to assign threads to CPUs when scheduling running tasks, the algorithms oftentimes do not result in optimal performance. OS typically needs to strike the balance among multiple performance factors including CPU load balancing to provide a generic mechanism accommodating all sorts of scenarios. Moreover, enterprise Java applications typically feature more than hundreds of threads, which make the CPU scheduling even harder. Because of this, the memory allocations and threads scheduling may not be optimal for a particular deployment scenario. For instance, a Java thread that is dynamically scheduled to run on a different CPU based on CPU load balancing considerations may have to frequently access remote memory, resulting sub-optimal performance. In addition, migrating among different computing nodes also risk losing the benefits of warm CPU/TLB caches.

On the other hand, for typical business deployment scenarios, the deployment needs are fixed in the sense of: (1) how many applications will be deployed; (2) the memory footprint and CPU usage of these applications; (3) how many computing nodes on the NUMA platform; (4) how much memory and how many CPUs on a computing node, etc. By considering these fixed parameters, a much optimized deployment model can be realized.

4.4 Summary

We saw that JVM mechanisms, coupled with OS-level features, give rise to unique problems that are not present in other deployment scenarios. Though the investigations and findings are Linux specific, given the wide deployment of Linux OS, particularly in server markets, the findings apply to a significant portion of multi-tenant backend platforms. In addition, other platforms are expected to expose similar problems of varying degree of severity. On one hand, JVM mechanisms are largely universal across OS platforms. On the other hand, most OS platforms have mechanisms of swapping and reclaiming. These similarities, along with the identical requirements with regard to multi-tenancy in cloud environments, makes us believe our findings can help on similar problems and solutions in other cloud platforms.

Note that to help illustrate the problem and understand the causes, we consider the particular motivation scenarios where the memory requirement almost exactly matches the physical memory. However, based on our production experience, these scenarios do occur frequently in real productive environment. This is because the fact that deployment often times over-commits applications to allow resource sharing and reduce cost.

5. Solution

We now present solutions to prevent Java applications running with multi-tenancy from performance degradation.

5.1 Overview

Our solution consists of a set of four design elements, each targets a specific aspect of the problems discussed in Section 4. Applying any individual element will help to some extent, however, all the four design elements need to be applied and work together to get the full benefit.

Design Element I: Pre-allocating JVM heap space. We know that JVM heap space is not allocated until they are absolutely used. When new heap space is needed to accommodate new objects allocation requests. Linux OS needs to allocate memory pages for the heap growth, which may trigger heavy page reclaiming and hurt performance. The design element pre-allocates all the heap space such that avoiding on-the-fly page allocation by Linux OS. To enforce heap pre-allocation, Java applications need to be started with "-XX:+AlwaysPreTouch". The side-effect of this design is the increased time taken for the JVM to start, which needs to be taken into account when applying the design element.

Design Element II: Protecting JVM heap spaces from being swapped out. We learned that when GC occurs, the corresponding memory pages need to be scanned. If these pages were swapped out, they need to be swapped in first, which incurs delay and hence GC pauses. The design element protects the pages from being swapped out. Linux OS allows turning off swapping, however it applies to all applications and all memory spaces. An ideal implementation is to allow fine-tuning of swapping in terms of which applications and what memory areas. For instance, the use of cgroup ("Linux cgroups", 2014) to finely control which applications to swap. However, the complexity and administration cost may make such mechanisms a over-killer. On the other hand, we realize that for most multi-tenant platforms as in LinkedIn, these platforms are dedicated to only running homogenous Java applications. Note that Homogenous applications are identical applications running same code base except serving different customers. In these scenarios, it justifies to simply turn off swapping for applications.

At this time, Linux has two approaches to turn off swapping: (1) by issuing "swapoff " command and (2) by setting swappiness=0. The difference between them is that the former completely forbids any swapping activity while the latter only discourages swapping. When the system is unable to handle new process’s memory request, the former approach will kill another process to free memory, while the latter approach begins swapping anyway. So depending on deployment scenarios and requirements, they need to be chosen carefully.

http://hipore.com/ijcc
Design Element III: Dynamically tuning THP. Though we have seen that enabling THP feature could cause critical performance penalty, THP provides performance gains in other scenarios. The bottom line is to enable THP when it can bring benefits, while disable it when it could cause troubles. The first observation we make is that THP exposes performance penalty mostly when the system’s available memory is low. When that happens, existing THPs need to be split into regular pages for swapping out. Thus, it is better to disable THP entirely when the system is under memory pressure. Since Java applications allocate heap when started (particularly when started with "-XX:+AlwaysPreTouch"), it is important to decide on whether to allocate THPs to a Java application when started. Thus, we choose to use the memory footprint size of the Java application as the memory threshold to decide whether to turn on or off THP. When the available memory is significantly larger than the application’s memory footprint size, then THP is enabled, as the system is unlikely under memory pressure after launching the particular application. Otherwise, THP is disabled. Since many dedicated backend platforms like LinkedIn’s are hosting homogenous applications, assigning applications’ footprint size is a simple while effective decision.

Moreover, regular pages need to be collapsed into THPs before the huge pages can be allocated to applications. Thus, part of the element is to decide when to allow THP collapsing. We propose to base the decision on the direct page scanning rate and the cpu usage of khugepaged. In summary, the design element consists of 3 components which control different collapsing types separately: (1) When available memory drops below the threshold, disable THP; (2) When direct page scanning is high, disable direct THP collapsing; and (3) when khugepaged daemon appears to be a cpu hogger, disable background THP collapsing.

Design Element IV: Explicitly hardware binding. When the deployment requirements (e.g., number of applications, memory/CPU footprint, characteristics of NUMA platforms) are known, this design element binds the hardware resources (i.e., CPU and memory) to the applications to encourage local memory access and warm CPU/TLB cache. With the presence of multiple computing nodes (each with local CPUs and memory) on a NUMA platform, this design element will categorize co-located applications into a number of sets, each set is bound to one or more computing nodes. Note that when an application’s memory or CPU footprint exceeds that of a single computing node, multiple computing nodes can be combined to form a group. For any binding between (one or more) applications and (one or more) computing nodes, the following targets are required: (1) the aggregated memory footprint of the bound

Algorithm 1:
1. Initialize $NumNode$ application sets ($Set_k$);
2. Create $List_{app}$ which sort all $app_i$ based on cpu footprint;
3. While $List_{app}$ is not empty, do:
   4. Starting from the first application $app_i$:
   5. If moving $app_m$ into $Set_k$ causes $Mem_{total} > Mem_{Node}$:
      i = i+1;
   7. Else:
      8. Move $app_i$ to $Set_k$;
      9. $k = (k+1)\%NumNode$;
10. Reverse $List_{app}$;

Algorithm 2:
1. Characterise Java applications into three sets:
2. $Set_{lowpause}$: Java apps that require low GC pause;
3. $Set_{slowstart}$: Java apps that can tolerate slow startup;
4. $Set_{others}$: Other Java applications;
5. If need to protect all Java applications:
6. If cannot kill Java applications when starting others:
   7. Set swappiness to 0;
   8. Else:
      9. Issue command of swapoff;
10. Else:
11. Turn off swapping for $Set_{lowpause}$;
12. Apply AlwaysPreTouch to $Set_{lowpause} \cap Set_{slowstart}$ and start Java applications

Algorithm 3:
1. Every time of $T$ (Adjustment period):
2. Perform system resource monitoring (e.g. free mem);
3. If free memory > $Th_{freemem}$:
   4. Enable THP;
   5. Else:
      6. Disable THP;
      7. Continue;
   8. If average direct page scanning rate > $Th_{dld}$:
      9. Disable direct THP collapsing;
   10. Else:
      11. Enable direct THP collapsing;
12. If khugepaged average cpu usage > $Th_{cpu}$;
13. Disable background THP collapsing;
14. Else:
15. Enable background THP collapsing;

Figure 9 Algorithm
applications should not exceed the total memory of the computing nodes; (2) the aggregated CPU footprint of the bound applications should not exceed the total CPU resources of the computing nodes. For example, assuming a 2-node NUMA platform, and each node has 6 CPUs and 48GB of memory. Given 3 applications with memory/CPU footprints of (4-CPU, 20GB), (2-CPU, 20GB) and (5-CPU, 40GB), the first two applications will be bound to one computing node, and the third application will be bound to the other computing node.

5.1 Algorithm

We classify the designs into a set of 3 algorithms as following. All the Java applications that will be run on a particular Linux system is firstly deployed with explicit hardware binding. This problem is similar to the traditional knapsack problem, which is NP-complete. We propose a greedy algorithm to expedite the processing, as shown in Figure 9 Algorithm 1. For easy description, we assume no application has larger CPU/memory footprint than a single computing node. Specifically, the greedy algorithm firstly initialize empty “knapsack” for each computing node (totally NumNode nodes), denoted by Seti. Then it ranks all applications based on CPU footprint, forming a list of Listapp.

The sorted applications then will be moved to the knapsacks one by one, starting from the application with largest CPU footprint. For every movement, the aggregated memory footprint for the current knapsack is compared against the memory resource provided by a computing node. If the former exceeds the latter, then the current application is skipped; otherwise the application will be moved to the knapsack. Every time a movement is completed, the algorithm will goes to the next knapsack. Once all knapsacks are checked, for the purpose of balancing the CPU usage among computing nodes, the ordering of applications based on CPU footprint is reversed. Then the previously described movement process will start until Listapp is empty.

For each computing node (or set of computing nodes), all the Java applications are characterized into three sets based on two types of requirements: (1) strict low pause requirements and (2) short startup delay. Specifically, as shown in Figure 9 Algorithm 2, Setlowpause contains the Java applications that require low pauses, Setslowstart contains those can tolerate slow startup, while Setothers are others. Note that the first two sets may have common elements.

The algorithm then adjusts the swapping configurations of the system. If all the Java applications need to be protected from swapping, then it simply adjusts the system swapping configuration, as described in Figure 9 Algorithm 2. Otherwise, individual Java applications can be separately adjusted with regard to swapping.

Before any Java applications is started, the algorithm applies “-XX:+AlwaysPreTouch” to the Java applications that belong to the intersection of Setlowpause and Setslowstart. In other words, only Java applications that need to guarantee low GC pauses and can tolerate slow startup are started with pre-allocating heap spaces.

After Java applications are started, every T time (adjustment period), the algorithm finely tunes THP. Firstly, the algorithm obtains current system performance statistics including free memory size. It then decides to enable or disable THP, as shown in Figure 9 Algorithm 3. Note that once THP is turned off, the algorithm simply skips the following steps as finer knobs are disabled inside THP.

The algorithm then checks whether it should turn on or off the background and direct THP collapsing independently. For direct THP collapsing, it relies on the past period’s statistics of direct page scanning. If it appears to be having heavy direct page scanning activities, it will turn off the knob. Otherwise, direct THP collapsing is enabled. Similarly, for background THP collapsing, it relies on the
activities of `khugepaged` as shown in cpu monitoring utilities such as `top`. If khugepaged appears to be hogging cpu, that knob is turned off. Otherwise, the knob is turned on. The tuning of THP is shown in Figure 9 Algorithm 3.

6. Evaluation

We now present the evaluation results of all four design elements. We use the same JavaApp as described in Section 3, and also use the BacgroundApp for creating deployment scenarios with different available memory sizes.

**TABLE 1: STARTUP DELAY OF PRE-ALLOCATING JVM HEAP SPACE**

<table>
<thead>
<tr>
<th>Heap size (GB)</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start delay (Sec)</td>
<td>0.5</td>
<td>0.8</td>
<td>1.6</td>
<td>2.5</td>
<td>7.3</td>
<td>11</td>
<td>15</td>
</tr>
</tbody>
</table>

### 6.1 Pre-allocating JVM heap space

We begin by evaluating the design element of pre-allocating JVM heap space. We let the BackgroundApp to take 50GB of memory, and then start the JavaApp with 20GB heap. Recall that the machine has 72 GB of memory, the above setup creates a scenario with memory pressure. For comparison we start the JavaApp in two scenarios: without and with pre-allocating heap. As shown in Figure 10(a,b), without heap pre-allocation, the application throughput sharply drops to almost zero after about 30 seconds, which is correlated with significant GC pauses. In the scenario with pre-allocated heap, as shown in Figure 10(c), the application performance is very stable.

Pre-allocating JVM heap space does increase the startup latencies of Java applications. Applications begin to respond only after all the heap space is allocated. We quantify the startup latencies of a JavaApp of different heap sizes and found that it roughly takes about 4 seconds for every 10GB of heap size, as shown in Table 1.

#### 6.2 Protecting JVM heap spaces from being swapped out

To evaluate the effectiveness of protecting JVM heap space from swapping, we firstly run the JavaApp with pre-allocated heap of 20GB for 150 seconds. We then start the BackgroundApp to take 50GB of memory, which encourages page swapping. In the scenario where JVM heap space is not protected (i.e., by setting swappiness to 100), we see that after about 3 minutes, the JavaApp’s performance drops sharply, as shown in Figure 11(a). In Figure 11(b), we confirmed that some heap memory is swapped out. In the scenario where JVM heap is protected, we observe much better performance in Figure 11(c). Note that the lower performance after 3 minutes are caused by THP activities.

#### 6.3 Dynamically tuning THP

We then evaluate the design element of dynamically tuning THP. We consider the scenario where JavaApps are started with abundant available memory and less-abundant available memory. Specifically, the first JavaApp is started without other applications running. It then runs for 3 minutes and stops. After that, a BackgroundApp takes 45GB of memory, then another JavaApp is started and runs for 3 minutes. Note that for these runs, the first two design elements are both enabled.

For the above scenario, 3 mechanisms are considered: THP is turned off, THP is turned on, and THP is dynamically tuned. The adjustment period is set to be 2 seconds. The results are shown in Table 2. We see that when THP is off, JavaApp-I achieves the lowest throughput of 12 K/s, while the other two mechanisms have THP enabled and hence see higher throughput of 15 K/s.
TABLE 2: JAVAAPP THROUGHPUT (K ALLOC/SEC)

<table>
<thead>
<tr>
<th>Mechanisms</th>
<th>THP OFF</th>
<th>THP ON</th>
<th>Dynamic THP</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaApp I</td>
<td>12</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>JavaApp II</td>
<td>13</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

For JavaApp-II, since it is running under memory pressure, turning THP off gives the highest throughput. Dynamically tuning THP results in less throughput (12K/s) than turning-off THP, but outperforms THP-on since it turns off THP when necessary. Note that dynamically tuning THP brings the benefit of accommodating more scenarios, particularly in scenarios where the system usage is unpredictable thus manual configuration of THP is not desirable.

We also notice that for JavaApp II, the performance benefit brought by Dynamic THP when compared to THP-ON is not significant (i.e. 1K/s), that is because of the relatively simple scenario we considered and hence less performance improvement. However, in other scenarios such as those illustrated in Section 3, as well as in real productive environments, we have observed much more significant improvement by turning off THP appropriately. Thus, we believe the algorithm used by Dynamic THP needs to be finely tuned, which will be our future work.

6.4 Explicit hardware biding

As we described in Section 3, the NUMA system has 2 computing nodes, and each node has about 36GB memory and 12 CPUs. We consider 3 applications with the following memory/CPU footprints. App-A and App-B each uses 5GB memory and about 3 CPU; they also contain a single working thread. App-C contains 2 concurrent working threads, and the CPU footprint is about 6 CPUs using 10GB memory.

TABLE 3: JAVAAPP THROUGHPUT (K ALLOCS/SEC)

<table>
<thead>
<tr>
<th>Application</th>
<th>App-A</th>
<th>App-B</th>
<th>App-C</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>W/O binding</td>
<td>2.57</td>
<td>2.72</td>
<td>4.52</td>
<td>9.81</td>
</tr>
<tr>
<td>W/ binding</td>
<td>2.95</td>
<td>2.93</td>
<td>5.38</td>
<td>11.26</td>
</tr>
<tr>
<td>Improvement</td>
<td>14.8%</td>
<td>7.7%</td>
<td>19.0%</td>
<td>14.8%</td>
</tr>
</tbody>
</table>

Following the greedy algorithm we presented in Section 5, App-C is bound to the first computing node, while App-A and App-B are bound to the other computing node. In the following, we will show the performance results of all applications, as well as the CPU usage of the entire platform.

Figure 12 shows the application throughput without explicit hardware binding, while Figure 13 shows the case with explicit hardware binding. We also display the average values in Table 3. We see that for all applications, explicit hardware binding achieves significant performance gains. Overall, the gain is about 15%. Probably more importantly, the performance of all applications are much more stable with explicit hardware binding, as displayed in the figures.

We also show the cpu usage comparisons in Figure 14 and observe that with-binding actually uses less CPU resources (i.e., 47% vs. 46.2%).

7. LESSONS LEARNED

During the investigations into the performance problems experienced with our production platforms, we have learned several lessons which are summarized in this section.

Multi-tenant Java platforms expose extra performance challenges. Multi-tenant platforms expose unique challenges with regard to the tradeoff between cost and performance. On one hand, the goal of serving multiple tenants on a single system is to save cost by encouraging resource sharing; on the other hand, independent activities of tenants may affect each other. The interactions between tenants oftentimes are further complicated by the OS features, particularly the memory management. When the tenant applications are running in JVM, we also need to consider the JVM and GC mechanisms due to the fact that Java maintains its own heap. We have seen in this work that the Java heap is not entirely allocated in the beginning. Though the design has its own benefits, it could cause performance problems in certain scenarios.

Be cautious about Linux’s new features (optimizations). Linux has been constantly incorporating new features in the hope of optimizing the systems and improve performance. Most of the mechanisms are successful over the years, however performance engineers should never be careful enough about these features. In this paper, we have shown that the new feature of THP proves to cause significant...
performance problem when the system is under memory pressure. We have to admit that the idea of THP is fantastic, as it significantly eases the way huge pages are used. The pitfalls of THP we have seen are mainly due to the less-mature implementations of the idea. We believe as its implementations mature, THP will be able to bring more performance benefits in more circumstances. However as a general guideline, we need to deeply understand the internal mechanisms of any new OS features as well as the scenarios where a feature works or does not work.

Root causes can come from seemingly insignificant information. Linux OS emits significant amount of system statistics (e.g., /proc), and most of us most of the time mostly only examine a small subset of these statistics. Over years of performance investigation lives, we have encountered many scenarios where commonly examined statistics shed little light on the performance problems. For instance, in this work, we have seen heavy direct-reclaim-caused page scanning activities, however free memory is sufficient. After we turn our eyes to examining per-zone statistics emitted by /proc/zoneinfo, we nailed down the zones that are desperate for free memory and hence triggers direct page scanning.

8. RELATED WORKS
Multi-tenant cloud platform Cloud Computing model is being increasingly deployed as a fast and economic solution to serving multiple users. Despite the distinctions among the three commonly accepted service models (i.e., IaaS, PaaS, and SaaS), any large scale cloud computing environment requires the instantiation of multiple tenant applications (Woolen, 2010) or VMs (Virtual machines) (Bugnion et al., 2012). (“Oracle virtualization”, 2014) on a single hardware and/or software. Though certain techniques such as Logical Domains (“Oracle virtualization”, 2014) can isolate resources dedicated to individual tenants, to achieve highest level of resource sharing and hence cost saving, system resources have to be shared by multiple tenant applications. Such resource sharing, particular memory-sharing, gives rise to certain unique challenges.

Java performance for server applications Bearing many advantages including platform-independence, Java is used as one of the top languages to build and run server applications. Ever since its birth, extensive works have been done to finely improve its performance in various deployment scenarios (Hunt and John, 2011), (Taboada et al., 2013). ("Usage share", 2014). However, most of the performance studies focused on the Java/JVM itself. Our work, on the other hand, deals with an undesirable deployment scenario caused by system level resource shortages and multitenant interactions.

Linux memory management and system optimizations Linux has long been the mostly deployed OS in server market ("Usage share", 2014). Its key component of memory management has seen a plethora of advanced features being designed over the past years (Bovet and Cesati, 2005). To accommodate the fact of limited RAM and requirement of supporting multiple processes, paging and swapping are continuously optimized to improve the system/application performance ("Huge pages", 2014), (Cervera et al., 1999), (Oikawa, 2013). Meanwhile, to better fit a particular setup, various system and configuration optimizations are researched extensively (Johnson et al., 2005).

9. CONCLUSION
We studied the challenges faced in a typical multi-tenant cloud platforms with multiple co-located Java applications. We identified that JVM mechanisms used by Java applications, coupled with OS-level features, give rise to a set of problems that are not present in other deployment scenarios. We propose a solution suite that addresses these problems and shared the experiences we learned.

10. REFERENCES
“Control numa policy for processes or shared memory,” http://linux.die.net/man/8/nmactl, 2014

http://hipore.com/ijcc


Authors

Zhenyun Zhuang received his Ph.D. and M.S. degrees in Computer Science from Georgia Institute of Technology. He also received his M.S. degree in Computer Science from the Tsinghua University and B.E. degree from in Information Engineering from Beijing University of Posts and Telecommunications. He has been actively conducting research in various areas including system and Java performance, wireless communications, mobile networks, distributed systems, cloud computing and middleware platforms.

Cuong Tran currently is a staff engineer at LinkedIn. He worked at Yahoo and Symantec as a system architect. He also worked as an architect at Veritas Software and director of technology at Adforce. With 10+ patents/papers/blogs, he is particular interested in the areas of system/application performance and web/mobile performance, system performance and performance monitoring/analysis tools.

Haricharan Ramachandra currently is an engineering manager at LinkedIn in LinkedIn. He received his MS in Computer Science from California State University-Chico and BS in Engineering from University of Mysore. He serves in the board of directors at Santa Clara Tech Academy. He also worked at Yahoo as a technical lead. His interest includes system/application performance.

Badri Sridharan currently is a director of engineering at LinkedIn in LinkedIn. He received his MS in Computer Engineering from University of Wisconsin-Madison, and BE in Electronics and Communication from B.M.S College of Engineering. Leading the performance division, his interest covers all areas of system/application performance.

Differences between the conference paper (IEEE Cloud 2014) and this draft

1. Major extensions:

- **Motivation**
  Added an entirely new subsection of 3.4 (Scenario 3: Different deployment models) and 4 figures

- **Investigations**
  Added an entirely new subsection of 4.3 (Investigations into Scenario-3: Different deployment model)

- **Solution**
  Added an entirely new design element of "Explicitly hardware binding" Added a new algorithm (Algorithm 1)

- **Evaluation**
  Added an entirely new subsection of 6.4 (Explicit hardware binding) and 8 figures

2. Minor extensions/changes

- **Title:**
  Changed the title to more properly cover all the contents.

- **Abstract**
  Modified the abstract correspondingly to accommodate the added contents.

- **Introduction**
  Added the new contents and rephrased several paragraphs.

- **Background**
  Added the knowledge of NUMA and tools to control NUMA policy.

- **References:**
  - Added NUMA reference.
  - Added numactl reference.
CLOUD COMPUTING FRAMEWORK FOR AGILE DEVELOPMENT

Gardner Mwansa, Ernest Mnkandla
University of South Africa
Email: gmmwansa@gmail.com, mnkane@unisa.ac.za

Abstract
The emergence of cloud computing is influencing how businesses develop, re-engineer, and implement critical software applications. The cloud requires developers to elevate the importance of compliance with security policies, regulations and internal engineering standards in their software development life cycles. Cloud computing and agile development methodologies are new technologies that have come with new approaches in the way computing services are provisioned and development of quality software respectively. However, the synergy between the two is bonded with technical and non-technical challenges. In this paper, a conceptual framework is proposed to support the process of migration of South African small, medium and micro enterprises (SMMEs) who are using agile software development methodologies to cloud computing environment. The framework is also analysed based on critical cloud computing adoption factors as recommended from previous studies on SMMEs adoption practices.

Keywords: Cloud Computing, Agile Development Methodologies, SMMEs.

1. INTRODUCTION
Cloud computing is trending within social and corporate realms and experts believe that it will reshape information technology processes in the next few years (Armbrust et al., 2009). Cloud computing affords traditional and ubiquitous smart end user devices such as PCs, tablets and mobile smart phones to access computing services that include software applications, storage facilities, processing and application development by connecting to the Internet through Web 2.0 (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009). These resources are provided and kept by providers who are remotely situated. There are generally four cloud deployment models: private cloud - the company owns and controls its infrastructure and applications running behind a firewall with virtualization, tools and policies including deployments; public cloud - resources and applications are offered as services on a subscription basis by providers; hybrid cloud – a mix of public and private clouds and Community cloud provides an infrastructure shared by more than one organisation. Each of these deployments have advantages and disadvantages associated with them (Marinescu, 2012; Mell, Grance, & Grance, 2011).

The whole cloud computing model is attractive to users of different needs as it provides the following benefits: cost saving in operation, development and fast deliveries; resources such as data, applications and tools can be accessed anywhere and by any Internet ready device with Web 2.0; offer customized computing infrastructure with convenient task-centric, on-demand way of sharing configurable shared pool of resources; facilitates collaboration and provides good conditions for green computing (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009; Marinescu, 2012).

Despite being endowed with benefits, cloud computing has challenges such as security concerns; data ownership concerns; lock-in and interoperability concerns; enterprise Support and Service Maturity; requirement for online connectivity and; anxiety among developers about a new cloud computing platform without appropriate guidance and understanding of how to effectively utilize cloud computing standard architecture (Conway & Curry, 2012; Venkatraman & Wadhwa, 2012).

In spite of significant challenges that the technology platform faces, many users, vendors and industry observers predict an optimistic future for cloud computing (Buyya et al., 2009). Worldwide, some agile developers have migrated to cloud computing environment. For instance, the R&D of Salesforce motivated migration of all software development to the cloud environment (Salesforce, 2008). However, it is evident that this migration to private cloud has mainly involved large scale companies that have the capability to create private cloud infrastructures of their own with easy access to resources and tools. Small, medium and micro enterprises (SMMEs) on the other hand have challenges in adopting private cloud computing for reasons such as lack of capital base for investing in cloud infrastructure that will accommodate all resources needed for their development activities. This leaves them with the option of subscribing to...
public clouds only. This puts them at a disadvantage and subjected to the challenges of cloud computing associated with public clouds.

The South African Government currently considers SMMEs as vital enterprises for the economy (Berry et al., 2002). They contribute 56% of private sector employment and 36% of gross domestic product (Fatoki & Smit, 2011). According to the National Small Business Act (1996), an SMME in South Africa’s finance and business services sector is an organisation of micro-business which employs up to 5 employees, or a very small business employing of up to 10 employees, or a small business employing up to 50 employees, or a medium sized business employing up to 100 employees.

A significant adoption of cloud computing solutions in South Africa especially for business owners who are technologically proficient has been observed. These adoptions have mainly been in web hosting and ecommerce (94%), email hosting/archiving (75%), customer Relationship Systems (58%), configuration and data backup (58%) and application development with 40% (Hinde & Van Belle, 2012). It will be interesting to investigate application methodologies, programming environments and tools used by these organisations who have adopted cloud computing as it was not part of Hinde & Van Belle (2012)’s study. Reason being that, certain development methodologies such as agile emphasize specific practices that may bring about issues in the form of non-technical and technical problems associated to cloud computing environment. User/developer communication limitations and programming environment lock-in are examples of non-technical and technical problems respectively (Khajehhosseini, Greenwood, Smith, & Sommerville, 2012; Sillitti & Succi, 2004).

It is against this background that this research paper aims at proposing a framework based on apparent characteristics, practices and contexts that are critical in agile development processes in order to determine successful migration to cloud computing specifically for SMMEs in the South African context. According to Sahandi, Alkhalil, and Opara-Martins (2013), a global phenomenon of cloud computing adoption in the SMMEs sector is evident; however, this research is specific to the South African context due to its unique standards and regulatory frameworks that guide Internet use.

It is envisaged that the framework will contribute to; 1. Theoretical knowledge and perceptions of technological innovation adoption frameworks as applied to agile development methodologies and cloud computing environment; 2. Determine effective interactions among the factors that contribute to successful migration; 3. Will provide guidelines to SMMEs in South Africa who are using agile development methodologies in effective transition into use of cloud computing without compromise on software quality.

The rest of this paper is organized as follows: Section 2 reveals the available literature on cloud computing environments and software engineering with a focus on agile development methodologies. In Section 3, an analysis of problems arising from developing on the cloud environment are discussed. Section 4 proposes a framework. Finally, the paper ends with a conclusion and recommendation for future work.

2. BACKGROUND

Cloud computing has over the last decade been a catchword in the computing circles and has escalated promises of a new paradigm shift in the manner in which computing services are provisioned to users individually as well as an organisation computing (Buyya et al., 2009; Pallis, 2010). Its use currently involves users using services on different levels of its architecture computing (Buyya et al., 2009; Marinescu, 2012; Mell et al., 2011). Users get access to services that include storage, access to application software, processing and application development by using various devices such as smart phones, laptops, personal computers etc. (Buyya et al., 2009). In addition to this, there are other benefits such as cost savings, increased capacity and capabilities to Information Technology departments.

While there has been apparent significant benefits in the use of cloud computing, adoption of cloud technologies is still faced with doubts by many would-be users due to some challenges such as those of security, privacy, lock-ins and uncertainties in the regulatory frameworks (Conway & Curry, 2012; Venkatraman & Wadhwa, 2012). However, there has also been substantial research in this area especially addressing challenges of the technology offerings as it will be discussed in this literature review.

2.1 HISTORICAL PERSPECTIVE AND DEFINITION OF CLOUD COMPUTING

The dawning age of cloud computing spans long before the advent of the Internet where researchers had a vision of what was termed as computer utility. For instance, in 1961, Professor John McCarthy predicted that computing would in future be structured like any other public utility such as telephone or electricity (Arutyunov, 2012). The cloud computing ideology can also be traced back to Advanced Research Projects Agency Network (ARPA-NET) in 1969 when Joseph Carl Robnett Licklider visualized a network of data and programs interconnected for everyone to use globally (DARPA, 1981). All these ideas had a theoretical concept of commoditizing computing services by providers who would make available services according to user requirements.

http://hipore.com/ijcc
The philosophical ideas of the 1960s were introduced in the mainframes or datacentres managed by computer companies such as IBM from single installations. These were characterized by “dumb terminals” that never had any processing capacity but totally dependent on connectivity with the mainframe or minicomputer (Marshall, 1990). The target users were mainly corporate or Government institutions who also actually set them up internally due to the complexity and huge cost of maintaining them.

However, during the early 1980s, most organisations started acquiring personal computers and workstations which emerged within affordable levels. This technological landscape was perceived as bringing to an end the original utility computing philosophy. The personal computers brought about the second wave of computer revolution that focussed on digitalization where users were increasingly using computers for documents, spreadsheet and databases (Barnatt, 2010). By the 1990s, as digitalization extended to storage of pictures, company documentation, music, video etc., it started to become almost impossible to store these forms of digitized information on stand-alone computers. This led to traditional systems of client-server architectures that accommodated a dedicated storage or application server of which individual PCs would connect to and access required information (Berson, 1996).

Since the beginning of the new millennium, a new wave of computer revolution started to emerge. This new revolution calls for “atomization” and “ubiquitous computing”. Atomization is the opposite of digitalization that entails digital content to be turned back into atoms that can be realised by vision, touching and hearing. Ubiquitous computing involves development of non-traditional computing devices that promote atomization Barnatt, 2010. For instance, smart phones and iPods are ubiquitous computing devices.

This type of requirements has led to the rise of cloud computing which in a way also has evolved through a number of stages that includes grid and utility computing, application service provision (ASP) and Software as a Service (SaaS) (Marinescu, 2012; Desai & Currie, 2003).

Cloud computing creates a situation where a user application accesses computing resources through a type of service and not necessarily directly by talking to the specific CPU for processing or hard drive for storage. A precise definition of cloud computing can be difficult to define due to the fact that different technology specialists would go for different emphasis in their definition rather than most end-users. Gartner defines cloud computing as a style of computing where massively scalable IT related capabilities are provided “as a service” across the Internet to multiple external customers while Forrester defines it as a pool of abstracted, highly scalable, and managed infrastructure capable of hosting end-customer applications and billed by consumption (Gartner, 2013); while IBM states that it is an emerging computing paradigm where data and services reside in massively scalable data centres and can be ubiquitously accessed from any connected devices over the internet (Gartner, 2013; Staten, 2008).

The world’s developer of international standards, The International Organisation for Standardization (ISO) is still crafting cloud computing definition paradigms (ISO/IEC JTC 1/SC. 2011, August). Currently, The National Institute of Standards and Technology (NIST) offers a succinct definition which describes cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell et al., 2011). Building on this, the user’s perspective can in this manner mean cloud computing being dynamically scalable, device-independent and provides task-centric resources that are accessed from the Internet at a charge as per use basis from service provider’s infrastructure (e.g., Google Apps, Amazon EC2, or Salesforce.com.

With the evolution of the web to 2.0, it is prudent to speculate that cloud computing technology is geared to achieve the philosophical objective of making computing services as the 5th utility after water, electricity, gas and telephony computing (Buyya et al., 2009). It entails a radical move from the traditional client-server architecture into web service.

Figure 1 below shows the difference between a traditional computing (client-server) model and the cloud computing model. The first part (1a) in Figure 1 shows traditional client-server settings where local software is installed and data stored on personal computers. Users of these personal computers have access to enterprise applications, data storage including processing power from corporate servers (data centres). In case of software development, all the development tools and necessary databases are either stored on the local server or personal computers. The Internet is not a critical requirement until deployment time or only if there is need to access some websites and communication in terms of emails.

The second part (1b) of Figure 1 shows the cloud computing model. In this scenario, software applications and data are not stored on user’s or corporate computing devices but in the cloud. In this case, Internet connectivity is critical to have access to the required resources. Unlike in the traditional architecture, the cloud computing model requires third parties in order to facilitate access to resources. That is, you need an Internet service provider and a cloud
services provider. Challenges of cloud services provision emanate from around these third parties (Ren, Wang, and Wang, 2012). For example; trust on how secure a connection is and not to allow intrusion.

Figure 1. Comparison of Traditional and Cloud Computing Models

Source: (Barnatt, 2010)

Cloud computing has essentially five characteristics that are supposed to be available in its infrastructure, namely; on-demand self-service, broad network access, resource pooling, rapid elasticity and; measured service (Sitaram & Manjunath, 2012).

On-demand self-service aims to reduce the configuration tasks from the user’s point of view where resources such as the compute, storage or platform are self-provisioned or automatically configured. Other than creating accounts on a service provider, a user may not interact physically with the service provider’s staff to have access to resources. Broad network access is a ubiquitous characteristic that allows access to resources using any device such as phones, PCs etc. as long as it is connected to the Internet and running a web browser. Resource pooling implements virtualization and multi-tenancy by supporting many concurrent users. Rapid elasticity creates a service platform or resource that increases or decreases according to user requirements. It is possible to declare the number of servers that one needs. This significantly aids cost saving in capital investments where organisations would not invest in computing resources that are often idle. Measured service is a “pay as you go” facility that literally removes the element of computing equipment being a fixed cost (Barnatt, 2010; Sitaram & Manjunath, 2012).

2.2 CLOUD DEPLOYMENT MODELS

There are four different ways in which cloud services can be deployed depending on the structure of an organisation and the provisioning location. Mell et al. (2011) defined cloud computing deployment models as private cloud, public cloud, community cloud and hybrid cloud.

Private cloud is the most secure and risk-averse cloud that has the whole cloud infrastructure belonging only to a single organisation (Armbrust et al. 2010). Normally, private clouds are considered a step to the growth of a corporate data centre where the organisation shares in-house infrastructure for cloud services. Mainly targets virtualization solutions for mission critical applications with demands for high security and low latency, and custom service levels. The main advantage is that the organisation has full control over its data, security aspects and performance. Ideally, the cloud user owns this infrastructure. This implies that such organisations should have the necessary capital outlay to host such infrastructures which in some cases result in poor economies of scale. Usually SMMEs do not have such capability, making this option unfeasible for them.

Public cloud are in real sense the early manifestation of cloud computing. The cloud infrastructure owned by a service provider that offers cloud services to the public on commercial basis, available through a public network; the Internet. Cloud services are usually sourced from very large resource pools that are shared by many other clients specializing in elastic workloads such as software development and testing application. They are synonymous to plants or factories that cater for services or utilities to clients on as demand with size of requirement arises. Structurally, they are distributed systems consisting of one or more data centres. They are normally considered an attractive option for SMMEs because they provide an economical plan for organisations to reduce IT costs and capital expenditure. SMMEs are capable of starting up or running a business with a rent an infrastructure option without an upfront capital investment in IT services. However, since public clouds are meant to serve many users on the same infrastructure, a multitenancy characteristic is created. A number of issues such as security, QoS performance management etc. are associated to this multi-tenancy effect (Ren et al. 2012). Other concerns evident in the public cloud are issues of data ownership, lock-ins, interoperability, support maturation and connectivity (Sitaram & Manjunath, 2012). Most popular public cloud providers are proprietary overlooking the challenges mentioned due to lack of cloud computing standardization.
Examples of some well-known public clouds are the Amazon Web Services (AWS) comprising of the Elastic Compute Cloud (EC2) and the Simple Storage Service (S3) which form an IaaS cloud offering and the Google App Engine which offers PaaS to its clients.

Community cloud provides an infrastructure shared by more than one organisation that have similar interests for serving a particular community. Interests can be of an industry or a business sector nature. According to NIST, “the infrastructure is shared by several organisations and supports a specific community that has shared concerns (e.g. mission, security requirements, and policy or compliance considerations). It can be managed by organisations or third parties and may exist on premise or off premise”. It differs from public cloud in the sense that cloud services are provided for a certain need of end users rather a multitude of needs to different users as in the public cloud. It also differs from the private cloud due to the fact that cloud services are not provided and owned by one organisation. Architecturally, community clouds are usually implemented over various administrative domains. An example of a community cloud would be a scientific research community sharing a large distributed infrastructure. Another example would be a community of SMMEs sharing common infrastructure in application development. However, the problem is that community clouds would require hosting standard software which may not be appropriate for organisations with different approaches to software development.

Hybrid clouds are a combination of different deployments (Armbrust et al. 2010). For example, a company may decide to run its software applications on a public cloud but make storage on its private cloud. This arises in cases where private clouds are unable to meet user’s quality of service requirements. They allow organisations to exploit their own IT infrastructure for maintaining sensitive information within locations at the same time be able to grow and shrink by provisioning external resources which they are able to release when not needed. Common workloads are those of regulated data that require elasticity and agility such as Business Intelligence solutions. They are sometimes referred to as heterogeneous cloud due their heterogeneity nature in distributing integrated services or resources from one or more clouds (Buyya, Vecchiola & Selvi, 2013). Being hybrid make them inherit problems of associated deployment models.

In this study we focuses on public clouds as SMMEs do not have capacity to invest in private clouds.

2.3 Cloud Technologies

In order to replace the traditional client-server approach with cloud computing, there are basically three options or service types in which services can be provisioned. These are: 1. Software as a Service (SaaS); 2. Platform as a Service (PaaS) and; 3. Infrastructure as a Service (IaaS) (Marinescu, 2012).

Figure 2 shows the services types and their relationships. SaaS is designed to provide applications as a service to end users. The approach is to provide off-the-shelf and existing web applications. Users can access the applications and still be able to customise it to their conditions and requirements. In case of off-the-shelf application not being present in the cloud infrastructure, then the SaaS becomes unsuitable. Then the user may have to use other service types that allow application development. SaaS is currently the most noticeable and used in the cloud as it mostly deals with end user software packages such as word-processing and spreadsheets. Example SaaS services are those from Google Docs and email services such as Gmail, Hotmail and Yahoo mail (Barnatt, 2010; Sitaram & Manjunath, 2012).

PaaS is designed to provide a platform service mainly for online application deployment for developers. The platform entails the operation system and the hardware associated with it. An environment is created to allow software development including test runs using development tools that are present within that particular service provider’s cloud infrastructure (Barnatt, 2010; Sitaram & Manjunath, 2012). It also facilitates speed of programming by automating some coding tasks and allows programmers to work on their programming languages and associated tools. Hence, technical programming knowledge and skills are necessary for most use of PaaS offerings. Therefore this service type is suitable for companies that choose to cloud compute or development of software although it can be restrictive in terms of resources provided by the cloud provider leading to the problem of vendor lock-in. A vendor lock-in is a situation created when a user of a service or product fails to easily change to another competitor’s service or product due to incompatible proprietary technologies. An example of PaaS is the App Engine offered as service by Google which can allow any user to write new cloud applications and be able to deploy them to the web using the Google’s cloud infrastructure (Barnatt, 2010; Sitaram & Manjunath, 2012).

The IaaS service type is a major cloud computing development meant for IT operators. It has a capability of offering services of processing, storage, networks and many other vital computing resources where a user is able to deploy and run arbitrary software (Mell et al., 2011). It includes services such as operating systems and applications. Without control of the underlying hardware in the cloud infrastructure, the user has control over the operating systems, storage, deployed applications and some limited control over networking components. Cloud providers of this service rent out servers using a process called virtualization. Server virtualization involves masking and
pooling of server resources. For example, one physical server may be configured using a special administrator software into multiple virtual servers (machines) and each acts like a distinctive physical device, capable of running its own operating system (Barnatt, 2010). In cloud computing, these virtual servers are mostly referred to as instances. The IaaS service provider can either offer dedicated physical servers or virtual server instances. Although, these two services can perform the same functions, virtual instances are sometimes regarded as insecure especially by users who do not want to share server hardware with others. For this reason, some customers may choose to use specific deployment models like private cloud only or a combination depending on the security requirements of their services or products. One example of IaaS vendors is the Amazon Web Services (Barnatt, 2010; Sitaram & Manjunath, 2012).

2.4 CLOUD TECHNOLOGIES.

Considering current demand from work and personal needs for online engagements and growth of the web, cloud computing could be a manifestation of a new paradigm of a large-scale distributed computing utility for business and society solutions (Pallis, 2010).

The South African e-government initiative strongly supports IT research in development of solutions that are directed to the future IT trends and offering (Department of Public Service and Administration-Republic of South Africa, 2001). Research studies on SMMEs using cloud computing have been conducted widely in South African and at an international level. We reviewed a few studies and decided to cite two local studies mainly because most of the studies had a common approach and presented the similar results although in different contexts.

The first local study was conducted by Hinde and Van Belle (2012) on cloud adoption by SMMEs in South Africa. The study showed a potential growth in cloud computing and that slightly over 52% of respondents accepting cloud adoptions. In the same study, 65% were aware of cloud computing existence, 25% thought it was for bigger companies and 34% had an adoption model in place.

The second study by Schofield (2013) conducted with the research team of Johannesburg Centre of Software Engineering (JCSE) summed up surveys conducted in the last two years of the study which included SME Survey 2012 (Goldstuck, 2012) involving 2,000 respondents; Microsoft SMB Cloud Adoption Study 2011 (Microsoft, 2011) involving 3,258 respondents and IDG Cloud Computing Survey 2013 (IDG Enterprise, 2013) involving 1,358 respondents. The majority of companies considered in these studies belonged to the SMME category (employing up to 100 employees).

The study results agreed with Hinde and Van Belle (2012)’s findings on cloud computing adoptions. The study concluded that company owners, who were technologically capable, appreciated the value of cloud computing in usage as well as economical use as compared to those who did not understand it. Further, challenges of cloud computing such as security, bandwidth connections were also highlighted as impediments to making adoption decisions.

Going further, a study based in UK showed that SMMEs stand to benefit in reducing, costs, improving flexibility and scalability when they decide cloud computing migration. However, issues relating to security, vendor lock-in, and technical hitches with data privacy and data protection need attention (Sahandi et al. 2013). In another report, The European Commission technical report on ICT – Information and communication Technologies - Work programme 2013, a recommendation was made to strengthen software and services technologies by exploiting Internet-based services such as cloud computing. It also recommended that adoption of cloud computing should be taken with careful consideration of legal, socioeconomics and technical issues. In conclusion, the report indicated that the potential of cloud computing and its models has not yet been fully exploited in terms of development and research to the degree of full utilisation by stakeholders (European Commission, 2013).

Based on industry commentaries, it is interesting to note that although South African organisations have approached adoption of cloud computing with scepticism, South Africa has however taken a critical role in cloud computing adoptions in Africa. This is according to Sudarshan Roongta, vice-president of Oracle’s Industry Strategy and Insight programme for Europe, the Middle East and Africa (EMEA) (itWeb, n.d.).

SMMEs have taken a leading role in adoption followed by large enterprises. Roongta reported that 66% of enterprises in South Africa have shown “very high” confidence in the security aspect of the cloud services and only one in 10 of the decision-makers have no trust in cloud
security. This has made the security concern dropping to the third on the list of other challenges. The recent investment in the Telcos and access to international bandwidth has improved the reliability of cloud computing. He noted that by end of 2014, adoption rate in South Africa will increase from the current 56% to 66% being led by the retail and mining sectors. According to him, the Compound Annual Growth Rate (CAGR) of 35% and an investment worth $215 million would be realized in 2017. Some other useful statistics that Roongta also reported are:

- Globally, cloud computing is taking the mainstream with 82% adoptions in SaaS; 52% cloud storage; 36% IaaS; and 21% adopting hybrid cloud.
- In terms of cloud usage, most organisations (57%) are using it for human resources; 54% for e-mail collaboration; 52% for sales and marketing; 51% for customer care; 42% for supply chain; 41% for finance; 36% for sourcing; and 35% for operations management.
- 70% of the respondents indicated that cloud computing is providing tangible cost savings.

On an international arena of industry analysis, statistics from North Bridge (2013), an active partner for early-stage entrepreneurs providing seed-to-growth financing for innovative companies looking to disrupt big markets in America, indicated that SaaS has taken the main role in cloud computing adoptions although the fastest in terms of growth is the IaaS. This implies providing way to growth in the PaaS. The report indicates that SaaS is the most popular with current (year 2013) 63% from 55% a year before. However, IaaS recorded a 29% annual increase making it the fastest while PaaS is forecasted to grow fastest in the next five years. A growth in IaaS or PaaS indicates application development activities.

[44] predicts IT cloud services will have a CAGR of 23.5%, five times that of the IT industry as a whole over the 2013–2017. Another study on current actual adoption rate from a study by TheInfoPro (2013), a service entity of 451 Research, predicts an average growth rate of 36% from this year until 2016. This study was conducted during the first six months of 2013 and involved IT management and primary decision makers of medium sized to large organisations in Europe and North America. Some notable findings in this study were:

- That sixty percent respondents believe that cloud computing is a natural evolution of IT service delivery and do not need to allocate a budget it. Out of those with a separate budget for cloud computing also believe that their spending will increase in 2013 and 2014 as compared to previous years.
- IaaS and SaaS activity has doubled to levels between 35% and 33% on projects declared, with 35% respondents indicating that private cloud activity are dominating.
- Despite increased cloud computing activity, 83% of the respondent have challenges in deploying their cloud computing initiatives. Mostly the challenges are non-technical but lie with the domain of processes, people, policy and organisational issues.

Jacobs (2013) of ITWeb, indicated that Gartner’s predictions has positioned cloud computing to number four out of the top ten technological trends for 2014 with a bulk of new IT spend in 2016.

Considering Schofield (2013), Hinde and Van Belle (2012) and Roongta’s report, we can confidently conclude that there is a positive trend in the growth of cloud computing in the South African context. However, we still not clear on the levels of cloud adoption of PaaS services specifically for software development purposes.

Some of the reasons or benefits that have led organisations migrating to cloud computing are:

- Cloud computing is being perceived as a new paradigm or next generation platform for future practices and philosophy of computing.
- Cost savings in operation, development and fast deployment of software with less failovers. There is no consideration for hardware or software for cloud services.
- Resources such as data, applications, tools and web services can be accessed from anywhere on the Internet and offers a one-stop facility for software development. It also offers easy integration of these resources with other enterprise solutions.
- Offers highly customized computing infrastructure online using the Web 2.0 strategy. These are provisioned in a convenient, task-centric, on-demand manner to a shared pool of configurable computing resources such as networks, servers, storage and applications.
- Cloud computing is collaborative, facilitating software development practices such as those of agile development methodologies.
- Cloud computing offers legal and good conditions to use less energy and waste fewer resources computing (Buyya et al., 2009; Marinescu, 2012; Mell et al., 2011).

Certain drawbacks that are associated to cloud computing especially in the absence of a cloud computing adoption framework are as follows:

- Security concerns
- Data ownership concerns
- Lock-in and interoperability concerns
• Enterprise Support and Service Maturity
• Requirement for online connectivity
• Anxiety within developers about a new cloud computing platform without appropriate guidance and understanding of how to effectively utilize cloud computing standard architecture (Conway & Curry, 2012; Venkatraman & Wadhwa, 2012).

Generally, technology innovation adoption models and frameworks have been thoroughly invested and applied in information technology projects. For instance, the Technology, Organisation and Environment model (TOE) (Tornatzky, Fleischer & Chakrabarti, 1990; Hage, 1980), Diffusion of Innovation (DOI) (Rogers, 1995) and the Technology Acceptance Model (TAM) (Wixom & Todd, 2005). These models have also been combined in certain adoptions just to meet the requirement of the technology situation (Oliveira & Martins, 2011).

In most of these models, an element of social context has been considered very critical (Tornatzky et al., 1990). More recently, Werfs, Robert, Baxter, Allison and Sommerville (2013) considers that adaptive social-technical issues can inform adoption processes of disruptive technologies such as cloud computing. These involve complex interaction among humans, technology and the environment.

With specific reference to cloud computing, we found a wealth of literature on cloud computing adoptions of which may also be applicable to SMMEs in the South African context. We reviewed the Cloud Computing Toolkit by Khajeh-Hosseini et al., (2012) which has not yet reached maturity, the cloud adoption Goal-Oriented Requirements Engineering Approach (GOR), an interactive process of adoption (Zardari & Bahsoon, 2011), Bidgoli (2011)’s six step process model that does not pay much attention to organisational issues and finally the Alshamaila and Papagiannidis (2013) analysis of the TOE based on the SMEs in the UK.

In this research we, consider the cloud computing adoption using Alshamaila and Papagiannidis (2013) analysis. The reason for this choice is motivated by the fact that it is more recent, involved SMEs and the process framework addresses most of issues associated to information technology companies. The main factors which were found to be playing a critical role in the adoption process were relative advantage, uncertainty, geo-restriction, compatibility, trialability, size, top management support, prior experience, innovativeness, industry, market scope, supplier efforts and external computing support.

Those that fall in the technology context are as follows:

• **Relative advantage:** refers to as: “the degree to which an innovation is perceived as being better than the idea it supersedes” (Rogers, 2003). It is considered as a central indicator to adopting a new technology in information systems innovation. The probability of adoption is enhanced when a business realizes a relative advantage in an innovation (Thong, Yap & Raman, 1994; Lee, 2004).

• **Uncertainty:** referred to as “the extent to which the results of using an innovation are insecure” (Fuchs, 2005). This indicates knowledge deficiencies on an innovation by stakeholders. In the case of cloud computing lack knowledge expertise in areas such as security, privacy and lock-in are evident especially for SMMEs.

• **Geo-restriction** is the uncertainty factor about data location. There is a possibility that consumers may not be able to know the exact location where their data is stored and processed. Depending on need, consumers should be allowed to sign SLAs with an option of knowing about data location.

• **Compatibility:** Refers to: “the degree to which an innovation is perceived as consistent with the existing values, past experiences, and needs of potential adopters” (Rogers, 2003). Compatibility is considered an important factor of an IT innovation (Rogers, 2003).

• **Trialability:** Refers to: “The degree to which an innovation may be experimented with on a limited basis” (Rogers, 2003).

The organisation context has the following factors:

• **Organisation Size:** this refers to the organisational size (Alshamaila & Papagiannidis, 2013). Small businesses are more motivated to adopt cloud services.

• **Top management support:** Refers to: the plan of action that dedicates time for ICT program in relation to cost and potential, plan reviews, results follow-ups and coordinating integration of ICT with management processes of business (Young & Jordan, 2008).

• **Prior technology experience:** Refers to “the extent of a user’s experience with previous similar technologies” (Lippert & Forman, 2005).

• **Innovativeness:** Refers to: “the extent to which a client adopts innovations earlier than other members of the same social context” (Rogers & Shoemaker, 1971).

Finally, the environmental context involves the following factors:

• **Industry:** Refers to: “the sector to which the business belonged” (Goode & Stevens, 2000).

• **Market scope:** Refers to: “the horizontal extent of a company’s operations” (Zhu, Kraemer & Xu, 2003).
• **Supplier efforts and external computing support:** Refers to: “the supplier activities that can significantly influence the probability that an innovation will be adopted “(Frambach, Barkema, Nooteboom, & Wedel, 1998).

2.5 **SOFTWARE ENGINEERING AND AGILE DEVELOPMENT METHODOLOGIES**

Software can be classified as a product of a design process by software engineers. It is a systematic amalgamation of programs that are made to run within a computer system that can be of any size, and architecture. Today’s business can hardly optimally be operational without a presence of software in their systems. The choice of software by companies varies depending on the requirement which has a direct influence in the way software is created. Pressman (2010) gives a textbook definition of software as one that consists of three pillars as follows: 1. instructions consisting of programs that when executed provide function and performance, 2. data structures that enable the programs to adequately manipulate information, and 3. documents that describe the operation and use of the programs. However, it is paramount to understand software when you consider its characteristics. The basic characteristics are that software systems are abstract and intangible. As a product, it is developed, does not wear out and mostly it is custom built. These characteristics are varied from those of hardware engineered products. It is usually developed for a particular customer even when concepts such as re-usability are encouraged.

The early software applications until the 1960s were largely developed devoid of an explicit information system development methodology. These practices brought about a number of challenges in user satisfaction ranging from cost, time and scope perspective. After this era a number of thoughtful efforts such as Systems Development Life Cycle (SDLC) have been made to understand the software development process. These efforts were mainly done in order to improve the quality of software during and after its development by addressing challenges of the previous unconventional era (Avison & Fitzgerald, 2006).

The result of these efforts has been value addition to the final software product and improvement in delivery times. However, these achievements could not preclude technical challenges as well as development process skills that continue to affect the SDLCs (White & Leifer, 1986). In South Africa, it is common to find problems within developing organisations such as software failures, budget over runs and late delivery to satisfy clients who are in need of quality software due to problems that are everywhere within the development environment. Mostly, these are associated with incomplete user requirements.

Newer approaches such as agile methodologies were introduced to software development in order to address issues of software quality although the quality aspect has been and continues to be subject of research in the software engineering domain. In agile development, the quality aspect is inherent in the development process. Agile methodologies are an alternative to traditional waterfall approach of software development. It can be defined theoretically as a group of software development processes that are iterative, incremental, self-organizing, and emergent (Keith, 2002).

With agile methodologies, prescribed values, principles and practices are recommended for successful software project implementation (Keith, 2002). Agile development requires distinctive tools such as feedback, transparency in communications, and time-boxing. Therefore, organisations that adopt agile methodologies need to implement an environment with an integrated toolset comprising tools for measurement, bug tracking, design, analysis, testing, coding, business intelligence and critiquing, just to mention a few. In addition, open source tools and proprietary tools need to be carefully coordinated to deliver successful projects (Sillitti & Succi, 2004). Success in this context means delivering a software product within the agreed time and budget constraints and at the same time meeting the anticipated user requirements from the project sponsor (Mnkandla, 2008).

In principle, cloud computing environment facilitates speedy provision of tools and infrastructural resources to agile development teams who also add value by continuous development of a software product through iterations and incremental approach. However, Werfs et al. (2013) classifies cloud computing among disruptive technologies. They further claim that when making decisions to adopt cloud computing, careful analysis should be made in light of 1. The type of cloud that is intended for use; 2. How the product’s functionality will be offered; 3. The cloud service provider to use and; 4. The pricing structure to be used for the services and products. As much as one would be keen to make decisions of cloud exploitation, a decision making process based on the above could complex depending nature of activities or services required.

Research shows an increase in the adoption of agile methods by developers in South Africa. However, there is little evidence to show which specific agile methodology is being adopted. The development platform has mostly been on stand-alone and traditional client-server architectures. However, as observed from Hinde and Van Belle (2012), application development within the cloud environment by SMMEs within South Africa is evident but it is not clear that these adoptions involve agile methodologies. Worldwide agile development in cloud environment has been successful although these experiences are only for large
companies (Vaquero, Rodero-Merino, Caceres, Lindner, 2009).

Considering the benefits of cloud computing, SMMEs agile software development adopters can enjoy faster, production, improved quality and more flexible and collaborative processes that embrace change. Some benefits include the following:

- Automated build in the cloud. Development organisations would reduce costs by using virtualization in accelerating their work through existing images residing on multiple platforms. This reduces utility pricing on servers as compared to the use of dedicated servers.
- In the cloud environments, access to production environments is quicker and supports automated production deployment. This results in reduction of feedback cycle within the technical team and business owners.
- Development teams are able to use virtualization aspect of cloud computing for unlimited number of servers and be able to do parallel work within the agile philosophy. Successful Agile development projects depends on strong and extensive communications.
- The virtualization aspect of cloud computing will facilitate quicker provisioning and testing of code while at the same time developing and testing a new version. Cloud testing allows substantial advances in speed and agility by using multi-platform testing on virtual images. Unit tests can be done in parallel on cloud machines which also results in cost serving as compared to using dedicated servers.
- Exploration and innovation within a team by trying new ideas on server working environments (Brynjolfsson, Hofmann & Jordan, 2010).

Ramesh, Cao, Mohan and Xu (2006) also identified five specific challenges that apply to agile distributed software development as follows: 1. Communication need vs. communication impedance, 2. Fixed vs. evolving quality requirements, 3. People- vs. process-oriented control, 4. Formal vs. informal agreement, and 5. Lack of team cohesion. Distributed software development is modelled around IT development teams spread out geographical locations but collaborate with each other on applications through mini-projects in order to develop final software. Modern web based techniques and tools such as cloud computing facilitate smooth running of activities in a distributed manner.

**Communication need vs. communication impedance.**

As indicated earlier, agile development methodologies do not depend on formal documentation but informal interactions within the team of developer and users. The distributed software development environment however requires that formal mechanisms such as designs are put in place for geographically separated locations. This raises a question on how you can balance formality of communication in agile distributed software development platforms.

**Fixed vs. evolving quality requirements.**

Distributed software development will normally require fixed and upfront agreements on quality requirements because of limited capability to control activities of distantly located teams. On the other hand agile requires an ongoing negotiations environment between developers and users as in the process of arriving at acceptable levels of quality on different phases of development. The issue of balancing between fixed and evolving quality requirements need to be addressed in these circumstances.

**People- vs. process-oriented control.**

The question of concern here is how you apply a suitable balance between people and process-oriented control in agile distributed development. The reasoning behind this question is based on the nature of distributed environments that are process oriented while agile is more of people oriented through informal processes and practices.

**Formal vs. informal agreement.**

Agile development environments normally involve informal contracts while distributed development requires formal agreements especially on targets, milestones and requirement specifications. This situation requires a balancing act between levels of contract formality appropriate in the agile environment.

**Lack of team cohesion.**

While cloud computing has the capability of facilitating agile development practices in theory, the actual practical aspect has some challenges arising from non-technical and technical assumptions and constraints. Some challenges as depicted from an industry expert include non-technical problems such as inadequate training, poor leadership, and rigid adherence to agile principles that do not fit into the project (Stafford, 2013). Technical problems arise from Internet access and its assumptions about co-locations, latency, and errors cannot be easily made. As a result, problems such as not having required meetings, inadequate documentation and issues related to short iterations are experienced. In addition, due to the fact that computing resources can shrink and grow on demand requires proper planning if the benefit of cost saving are to be realised while keeping good qualities of service, otherwise this may affect development processes (Armbrust et al., 2009).
Team cohesion in distributed development where developers and users are in different locations is not as binding as in co-located environments. This even makes it worse when agile development processes are used because they emphasise on continuous collaboration on all stages and aspects of the development project.

Some guidelines in form of a framework on migration are necessary to aid these SMMEs in making decisions on how to maximize benefits and optimize usage of cloud environment.

3. ANALYSIS

The current rate of emergence of cloud computing poses a big challenge for the need to embrace it. For many reasons outlined by computing (Buyya et al., 2009; Conway & Curry, 2012; Venkatraman & Wadhwa, 2012; Pallis, 2010; Arutyunov, 2012; European Commission, 2013; Brynjolfsson et al. 2010; Armbrust et al. 2010; (Vaquero et al., 2009), it is an indication that we are sitting at a critical stage of the most significant trend in information technology industry. Despite the explosion, there has been no clear contextual definition of cloud computing while at the same time it is crucial to understand the requirements and challenges of cloud applications if one has to fully benefit from its environment (Pallis, 2010). This is a problem, for instance, agile development proponents would like to emphasize certain characteristics of cloud computing to meet their goals. Hence the need to define its own cloud computing framework within their requirements and use.

Without a framework and specific cloud computing description, there are a number of challenges that are likely to be experienced especially by SMMEs as they decide to migrate to the cloud environment such as anxiety within developers about a new cloud computing platform without appropriate guidance and understanding of how to effectively utilize cloud computing standard architecture. These problems are likely to emerge from the perspective of technical and non-technical limitations (Hinde & Van Belle. 2012; Sillitti & Succi, 2004). Literature has shown that there are several frameworks and decision models for cloud migration (Oliveira & Martins, 2011). However, agile migration is the most desired and concerns raised by the researchers are on some critical aspects associated to agile that are lacking in current frameworks.

4. PROPOSED FRAMEWORK

Based on literature surveyed in this research on current trends in cloud computing and agile software development practices, we propose to develop a framework that addresses the following problems:

- Identify factors necessary for successful migration of SMMEs that are using agile development to cloud computing;
- Determine effective interactions among the factors that contribute to successful migration; and
- Provide guidelines to SMMEs agile developers in South Africa for effective transition into use of cloud without compromise on software quality.

In order to achieve the above, an innovative approach is required to leverage all the benefits of cloud when used with agile software development so as to mitigate technical and non-technical challenges. We therefore hypothesize framework building with the following considerations:

- Important factors to consider in migrating agile development methodologies to cloud computing.
- Management of the process of migrating agile development methodologies to cloud.
- Roles of different stakeholders within and outside the organisation in ensuring successful migration.

Table 1 shows a framework exposition that addresses proposed activities and information required for the framework in order to address envisaged challenges during the migration process.

Table 1. Activity Versus Information Requirement for Framework

<table>
<thead>
<tr>
<th>Activity</th>
<th>Information Required for Framework</th>
<th>Variable(s) and/or Relationships measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETERMINE EXISTING ENVIRONMENT</td>
<td>Agile methodology in use</td>
<td>These will be identified through the coding of interviews transcripts, observation schedules, literature and document reviews.</td>
</tr>
<tr>
<td></td>
<td>Cloud computing services in use/required</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type of applications and tools in use/required to develop software in cloud computing environment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Perceptions held by agile software developers with respect to cloud computing</td>
<td></td>
</tr>
</tbody>
</table>
DETERMINE AND EVALUATE CONDITIONS FOR SUCCESSFUL MIGRATION

Factors responsible for success in migrating organisation's agile development to cloud computing

Difficulties or weaknesses encountered during the process of migrating to cloud computing

Factors responsible for the difficulties and weaknesses during the process of migrating to cloud computing

These will be identified through the coding of interviews transcriptions, observation schedules, literature, and document reviews.

Tests: Content Analysis and Correspondence Analysis

Perceptions held by agile software developers with respect to cloud computing

Perceived usefulness and perceived ease of use with respect to cloud migration

Based on literature, we analyze each activity in the proposed framework in order to give further information and clarity as in Tables below:

Table 2.1 shows the approach to determine existing environment.

Table 2.1 Existing Environment

<table>
<thead>
<tr>
<th>Information Required for Framework</th>
<th>Possible factors for analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile methodology in use</td>
<td>Extent to which an agile methodology such as scrum/Extreme Programming is used</td>
</tr>
<tr>
<td>Cloud computing services in use/requred</td>
<td>Extent to which developers use PaaS, IaaS Service Types</td>
</tr>
<tr>
<td>Type of applications and tools in use/requred to develop software in cloud computing environment</td>
<td>Extent of programming experience with programming tools in the cloud</td>
</tr>
</tbody>
</table>

Table 2.2 shows the approach to evaluate conditions for successful migration. It addresses information required for factors responsible for success in migrating organisation's agile development to cloud computing with specific reference to technology factors.

Table 2.2 Technology factors

Information Required for Framework

Possible factors for analysis (Alshamaila & Papagiannis, 2013)

- Relative advantage:
  - The degree to which an innovation is perceived as being better than the idea it supersedes.

- Uncertainty:
  - The extent to which the results of using an innovation are insecure.

- Geo-restriction:
  - The degree of uncertainty about data location.

- Compatibility:
  - The degree to which an innovation is perceived as consistent with the existing values, past experiences, and needs of potential adopters.

- Trialability:
  - The degree to which an innovation may be experimented with on a limited basis.

Table 2.3 shows the approach to evaluate conditions for successful migration. It addresses information required for factors responsible for success in migrating organisation's agile development to cloud computing with specific reference to organisational factors.
Table 2.3 Organisational factors

<table>
<thead>
<tr>
<th>Information Required for Framework</th>
<th>Possible factors for analysis (Alshamaila &amp; Papagiannidis, 2013)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Organisation Size:</td>
<td>o The size of a business in terms of:</td>
</tr>
<tr>
<td></td>
<td>✓ The market share of the business;</td>
</tr>
<tr>
<td></td>
<td>✓ The level of sales turnover</td>
</tr>
<tr>
<td></td>
<td>✓ The number of employees.</td>
</tr>
<tr>
<td></td>
<td>✓ The value of the business</td>
</tr>
<tr>
<td></td>
<td>✓ The value of capital employed</td>
</tr>
<tr>
<td>• Top management support:</td>
<td>o The degree of support from management on:</td>
</tr>
<tr>
<td></td>
<td>✓ The plan of action that dedicates time for ICT program in relation to cost and potential,</td>
</tr>
<tr>
<td></td>
<td>✓ Plan reviews,</td>
</tr>
<tr>
<td></td>
<td>✓ Results follow-ups and coordinating integration of ICT with management processes of business.</td>
</tr>
<tr>
<td>• Prior technology experience:</td>
<td>o The extent of a user’s experience with previous similar technologies:</td>
</tr>
<tr>
<td>• Innovativeness:</td>
<td>o The extent to which a client adopts innovations earlier than other members of the same social context.</td>
</tr>
</tbody>
</table>

Factors responsible for success in migrating organisation’s agile development to cloud computing

Table 2.4 Environmental factors

<table>
<thead>
<tr>
<th>Information Required for Framework</th>
<th>Possible factors for analysis (Alshamaila &amp; Papagiannidis, 2013)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Industry:</td>
<td>o Determine a business sector within software development.</td>
</tr>
<tr>
<td>• Market scope:</td>
<td>o The horizontal extent of a company’s operations.</td>
</tr>
<tr>
<td>• Supplier efforts and external computing support:</td>
<td>o Determine supplier activities that can significantly influence the probability that an innovation will be adopted.</td>
</tr>
</tbody>
</table>

Factors responsible for success in migrating organisation’s agile development to cloud computing

Table 2.5 shows the approach to evaluate conditions for successful migration. It addresses information required for assessing difficulties and weaknesses for the framework.

Table 2.5 Difficulties and weaknesses

<table>
<thead>
<tr>
<th>Information Required for Framework</th>
<th>Possible factors for analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulties or weaknesses encountered during the process of migrating to cloud computing</td>
<td>The extent of system weaknesses and difficulties in the migration process.</td>
</tr>
<tr>
<td>Factors responsible for the difficulties and weaknesses during the process of migrating to cloud computing</td>
<td>Decode from the weaknesses identified earlier.</td>
</tr>
<tr>
<td>Difficulties or weaknesses encountered during the process of migrating to cloud computing</td>
<td>Decode from the difficulties identified earlier.</td>
</tr>
</tbody>
</table>

Table 2.6 shows the approach to evaluate interactions of successful factors for the framework.
Table 2.6 Interaction of Success factors

<table>
<thead>
<tr>
<th>Information Required for Framework</th>
<th>Possible factors for analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction between factors that are responsible for success in migrating to cloud computing</td>
<td>Analysis of interactions of identified factors in “Factors responsible for success in migrating organisation’s agile development to cloud computing”.</td>
</tr>
<tr>
<td>Interaction between factors that are responsible for the difficulties/weaknesses in successful migration</td>
<td>Analysis of interactions of identified factors in “Factors that are responsible for the difficulties/weaknesses in successful migration”.</td>
</tr>
<tr>
<td>Relationship between factors that account for success and those that account for difficulties and weaknesses?</td>
<td>Analysis of relationships of identified factors in “Factors that account for success and those that account for difficulties and weaknesses”.</td>
</tr>
</tbody>
</table>

5. CONCLUSION/FUTURE WORK

Developing software in a cloud computing environment differs from the traditional approach. It makes it even more challenging when methodologies such as agile are used due to the fact that there is great need for interaction both technical and non-technical (such as sharing applications or development tools, communication and coordination) during development and deployment processes.

In this paper, a conceptual framework is proposed yet to be tested empirically through further investigation. The main thesis of this paper is that the migration process to cloud computing by SMMEs should be guided by a framework in order to mitigate all the challenges that are associated to cloud computing environments.

6. ACKNOWLEDGEMENT

This work was partly funded by the masters by dissertation and doctoral study bursary scheme from the University of South Africa.

7. REFERENCES


http://hipore.com/ijcc


Authors

Gardner is a lecturer at Sol Plaatje University in the Northern Cape, South Africa. He is currently instructing Information Technology courses. He completed his Masters of Science in Information Technology at university of Namibia in 2008. He is currently reading towards a PhD in Information Systems. His research interests are in emerging technologies such as cloud computing, agile development methodologies and big data analytics.

Ernest is an associate professor at the University of South Africa’s School of Computing; he lectures Software Engineering, Software Project Management, and Formal Program Verification to BSc honours students. He also supervises a number of master & doctoral students & has lectured in ICT for over 17 years. Ernest’s educational background includes an honours degree in Electrical Engineering, a master degree in Computer Science, and a PhD in Engineering. He is a certified Scrum Master. His research focuses on software engineering, software project management (tools and techniques), the relations between PMI’s PMBOK and Agile Project Management, the adoption issues in agile software development methods, the broader issues of software process improvement (SPI) and predictive analytics and project management intelligence. He is passionate about improving the quality of software development and has a focus on software defects management.
A CLOUD-FEDERATION-ORIENTED MECHANISM OF COMPUTING RESOURCE MANAGEMENT

Haopeng Chen, Wenyun Dai, Wenting Wang, Xi Chen, Yisheng Wang
School of Software, Shanghai Jiao Tong University
chen-hp@sjtu.edu.cn, {wwtvanessa, scorpiodwy}@gmail.com, {april-622, easonyq}@hotmail.com

Abstract
As the cloud of clouds, cloud federation, sometimes called as inter-cloud, provides a feasible and attractive kind of infrastructure to the cloud applications which want to obtain high performance, availability or profit. Meanwhile, the cloud providers also may consider that cloud federation is an ideal form to manage their computing resource since it can improve the utilization of their computing resources and extend the computing power of an individual provider. This paper classifies the cloud federations into different models according to their structure and control modes, including vertical, horizontal and hybrid federation, autonomous and centralized controlled federation. A design of the framework of multi-objective constrained resource management for cloud federation is proposed based on the analysis of the common objectives of such a framework, including dynamic profit-driven provisioning, availability-aware placement and power-saved consolidation. The result of our simulation has demonstrated the feasibility and effectiveness of the proposed framework.

Keywords: cloud federation; resource management; multi-objective constrained; profit-driven provisioning; availability-aware placement; power-saved consolidation

1. INTRODUCTION
Nowadays, the cloud computing paradigm is regarded as a revolution to the conventional information technology. The features of flexible pricing, rapid provisioning and infinite scaling enable make cloud computing appealing for the applications with massive data or large-scale concurrent clients. With cloud computing, the developers can rent the software, platform and infrastructure as services to facilitate rapid application development and reduce the cost of operation and maintenance of their applications. Consequently, more and more applications have been or will be migrated and deployed into clouds.

The cloud providers also can enhance the utilization of their computing resource and obtain extra profit by leasing their idle resource as service in clouds. As a result, almost all of the IT giants build their own public clouds in forms of Software as a Service, Storage as a Service, Infrastructure as a Service and Platform as a Service. For example, Amazon EC2 (2013) and Amazon S3 (2013) respectively “provide resizable computing capability and storage space in order to make web-scale computing easier for developers”; Google (2013) states that “Google App Engine, in form of Platform as a Service, enables enterprises to build web applications on the same scalable systems that power Google applications”; Microsoft (2013) claims that “Azure enables users to quickly build, deploy and manage applications across a global network of Microsoft-managed data centers”; IBM (2013) describes that “SmartCloud is the IBM vision for cloud computing, and it is used to accelerate business transformation with capabilities from IBM cloud offerings”. Besides these enterprises mentioned above, many companies such as Salesforce, AT&T, GoGrid, NetSuite, Rackspace and RightScale also provide cloud computing service in a variety of different manners.

The diversity of public clouds provides the providers of applications with more choices to deploy their own applications. On the one hand, an application can obtain the independence of cloud providers and improve its availability by deployed into an integration of resources from multiple clouds. For example, two instances of an application can be respectively deployed into Amazon EC2 and Windows Azure to improve its global availability. On the other hand, the multi-tier architecture of web applications allows each of the tiers to be deployed into different clouds in order to ensure all the rented services are with best quality. For example, the web tier, application tier and database tier of an application can be respectively deployed into Amazon EC2, Google App Engine and Amazon S3 to ensure each tier is deployed into the cloud with best quality. Both of the cases involve ‘Cloud Federation’ of public clouds which is the cloud of multiple public clouds. As a consequence of cloud collaboration, the cloud federation of public clouds is an inevitable development of cloud computing.

With the advancement of virtualization, it is feasible for an enterprise to make use of virtualization to effectively integrate its heterogeneous computing resource into a private cloud. Thus, more and more enterprises have built or are building their own private clouds. The computing resource of a private cloud is limited, so it is necessary for a private cloud to cooperate with other private or public clouds in order to scale up its computing power when the utilization of its computing resource is saturated. Consequently, the ‘Cloud Federation’ of private...
and public clouds, sometimes called as ‘hybrid cloud’, becomes very important for cloud owners and providers. Actually, for the customers, performance and availability are concerned, while power saving is an important issue for cloud providers. But existing research on cloud federation or hybrid cloud mainly focuses on how to scale up the computing power of a single cloud by building cloud federation. Meanwhile, the research on the cloud federation built for archiving high availability, independence of cloud provider and high quality of services is not adequate yet. Aware of this situation, we analyze the common objectives of resource management in cloud federation and put forward a design of multi-objective constrained framework for it in order to archive the goal of win-win for both cloud customers and providers.

The remainder of the paper is structured as follows. Section 2 classifies the models of cloud federations according to their structure and control modes; Section 3 analyzes the objectives of resource management in cloud federation; Section 4 describes our proposed design of the framework of multi-objective constrained resource management in details; Section 5 discusses the mechanism of multi-objective constrained resource management; Section 6 describes how we perform the simulation and analyzes its result; Section 7 briefly summaries the related works; and conclusion is in Section 8.

Figure 1. Structures of cloud federations: (a) architecture of typical multi-tier applications, (b) a vertical cloud federation, (c) a horizontal cloud federation, (d) a hybrid cloud federation

2. MODELS OF CLOUD FEDERATIONS

According to the structure of various cloud federations, we can classify them into three types: vertical, horizontal and hybrid cloud federation. If we classify the cloud federations by the control modes,
they can be divided into two types: autonomous federation and centralized controlled federation.

2.1 Structures of Cloud Federations

Most cloud applications inherently have the multi-tier architecture which at least includes web tier, application tier and database tier. Each of the three tiers can be individually deployed in order to meet the customized demand of applications. Each tier can be refined into more fine-grained tiers according to the requirement of applications. For example, the web tier can be divided into presentation tier and controller tier; the application tier can be divided into service tier, domain tier and data access tier. However, the fine-grained tiers of a coarse-grained tier are not deployed separately otherwise the performance will be damaged drastically.

*Figure 1(a)* is an example of multi-tier application in which the clusters of web tier, application tier and database tier are respectively composed of four, four and three VMs (Virtual Machines). There is a load balancer in front of each cluster to dispatch the requests and balance the workload among the VMs in the same cluster. The multi-tier architecture and the flexible deployment mode of cloud applications enrich the diversity of cloud federations by enabling the vertical cooperation between clouds.

The first type of cloud federation is “Vertical Cloud Federation”, shown as *Figure 1(b)*, in which the clouds vertically collaborate with each other to provide the application with all the necessary services. The “vertical collaboration” means that each cloud provides hosting environment for only one tier and any request dispatch involving multiple tiers needs to be accomplished by the vertical collaboration between clouds. For example, in *Figure 1(b)*, the web tier, application tier and database tier are respectively deployed into cloud A, B and C. The resource from cloud A, B and C allocated to the target application forms a vertical cloud federation. Such a deployment solution is totally determined by quality of services.

The second type of cloud federation is “Horizontal Cloud Federation”, shown as *Figure 1(c)*, by which a cloud application can obtain the independence of cloud providers and improve its availability by deployed multiple instances into an integration of resources from multiple clouds. As the saying — ‘Don’t put all your eggs into one basket’, the instances of an application can be horizontally deployed into different clouds to reduce the failure probability. For example, in *Figure 1(c)*, each of cloud A, B and C has a complete instance of the application. Such a deployment solution is also effective to solve the problem of saturation of computing resource.

The third type of cloud federation is “Hybrid Cloud Federation”, shown as *Figure 1(d)*, which is a combination of vertical and horizontal cloud federations. For example, in *Figure 1(d)*, the collaborations between cloud B and cloud D, cloud C and cloud D, and cloud D and cloud E are vertical while the ones between cloud A and cloud B, cloud B and cloud C, and cloud A and cloud D are horizontal. It is obvious that this type of federations is not so common than the other two types due to its high complexity. However, it is possible that some applications still prefer this type due to their special requirements.

2.2 The Architecture of Cloud Federation

Cloud federation is a cloud of clouds, which means the resource of cloud federation is from multiple cloud providers. Since all the resource of cloud providers is encapsulated as services, there should exist a service registry to facilitate the cloud providers to publish their services. Consequently, the service registry is a necessary component of cloud federation. However, the service registry is not enough to provide all the necessary support for constructing the cloud federation. So we consider the architecture of cloud federation should be the one shown in *Figure 2*, in which the core is the Cloud Federation Center which contains the lightweight kernel, the infrastructure and the extensions. The Cloud Federation Center acts as an agent for cloud customers and cloud providers to facilitate the construction and deconstruction of cloud federation coordinate the services provided by clouds and distribute revenue among clouds.

For cloud providers, the CF(Cloud Federation) Manager is used to communicate with other clouds, which is a new component for most existing clouds. The App Manager and the Cloud Manager are the extension of existing components of clouds, which means that new features should be added into them to support cloud federation. All the cloud applications are deployed into the Virtualized Resources.

Each cloud involved in cloud federation can communicate with Cloud Federation Center and other clouds to provide necessary services to cloud applications and coordinate the cooperation among services.

The details of these components will be given in section 4.
2.3 Control Modes of Cloud Federation

The control modes of cloud federation can be classified into the *Autonomous Model* and *Centralized Controlled Model* according to the way of management of cloud federation.

Autonomous Cloud Federation is autonomously managed by cloud providers. The initiators of Autonomous Cloud Federation are cloud providers, especially private cloud providers. They just query available resources through the lightweight core of the Cloud Federation Center and autonomously complete the process of building cloud federation which is transparent to the cloud consumers. This model is used to build horizontal cloud federation in order to extend the computing power of the initiators’ cloud.

Centralized Controlled Cloud Federation is managed by the Cloud Federation Center. The initiators of Centralized Controlled Cloud Federation are usually the cloud consumers. Their request of building cloud federation can be divided into two types. In the first case, the cloud consumers retrieve the information of available resources, lease the resources from cloud providers and deploy their applications. But obviously, it requires the cloud consumers must be very professional. Thus, in the second case, more common than the first one, cloud consumers send their functional requirements of rental resources and SLA (Service-Level Agreement) constraints to the cloud federation center, and then the cloud federation center generate the solution of building cloud federation in form of service composition based on the registered service. This mode is used to build both horizontal and vertical cloud federation.

3. Objectives of Resource Management

No matter in which cloud federation, the common objectives of resource management are to achieve high performance, high availability and power saving.

3.1 Common Objectives of Resource Management

Either the consumers or the providers of cloud federation have their own objectives of resource management.

For the consumers, namely the providers of cloud applications, it is their primary goal to reduce the leasing cost by optimizing the utilization of leased resource under the condition of guarantee reasonable quality of services they provided. The balance between quality of services and cost will be taken into account in order to maximize the profit of application providers. So the profit-driven resource provisioning is the first objective of resource management.

Availability is also a concerned issue for application providers. Since it is the relative but not absolute locations of the VMs that determine the availability of given resource, the availability-aware resource placement will give the feasible topology of VMs which indicates the network distances among VMs but not determines their physical locations.

The cloud providers will map the relative locations of VMs onto physical nodes according to the runtime status of the latters. The cloud providers need to ensure that the utilization of their resource is limited into a reasonable range in which the power consumption is optimized. Furthermore, the cloud providers need periodically consolidate their resource by VM migration due to the fluctuation of utilization of resource. So the power-saved resource consolidation is another important objective of resource management.

In summary, the common objectives of resource management are profit-driven resource provisioning, availability-aware resource placement, and power-saved resource consolidation. Consequently, the resource management in cloud federation is a multi-objective constrained one.

3.2 Profit-driven Resource Provisioning
From the viewpoint of consumers of cloud federation, the computing power of their cloud federation instances had better be able to be dynamically scaled up and down with the change of real-time workload of their cloud applications, since their profit will be maximized by renting the computing power in an economical way.

The profit of a cloud application is from the difference between its revenue and cost. The revenue is determined by the SLA assigned by the application and its consumers. In general, it is in proportion to the performance level achieved. Monotonic non-increasing utility functions are quite realistic to model the relationship between the revenue and the achieved performance, since the better the achieved performance is, the higher the revenues gained per request are. The cost is determined by the amount and price of computing power in the hosting cloud federation. The more the rented computing power is, the more the cost is. Given the unit price of computing power is fixed, there is a linear dependency between the cost and the amount of rented computing power. It is common that more computing power will result in better performance. But it is possible that the cost increased by the more rented computing power is greater than the revenue increased by improved performance. Thus, the better performance would probably result in less profit. So profit-driven resource management needs to find the point in Figure 3 at which the difference between Revenue(R) and Cost(C) is the maximum.

In Figure 3, the horizontal coordinate presents the number of VMs the cloud application rented, while the vertical coordinate presents the sum of profit. The Cost(C) linearly depends on the number of VMs (Virtual Machines). The Revenue(R) also increases with the increase of the number of VMs, but the relationship is not linear. Given the number of VMs is specified, for example B, we can find point A at which MR equals to MC and the profit is maximal. Given the number of VMs is changed, for example, it is changed to B' or B'', the point A will be changed to A' or A'' correspondingly. The Revenue(R) does not only depend on the number of VMs, it also depends on the real-time workload of the cloud application, since the achieved performance is determined by these two factors. The global maximal profit is the maximum of all of the maximal profit under each number of VMs.

Besides of performance, the availability is also a concerned objective for the owner of a cloud application. The profit-driven resource management only focuses on the appropriate amount of resource to be rented, while the availability-aware resource placement aims to determine the places of the rented resource. When some computing resource is rented by a provider to build the infrastructure of hosting environment of its application, the availability of such an infrastructure can be calculated with the availability and the topological structure of the physical hosts.

For example, Figure 4 shows two infrastructures of an application both of which are composed of two physical hosts. Both of the physical hosts of the blue one are in the cloud A while the two physical hosts of the green one are respectively in cloud A and cloud B. It is obvious that the availability of the green one is higher than that of the blue one, since the blue one is not available when both the clusters of Cloud A the hosts belong to are simultaneously not available while the green one is not available when both the cluster of Cloud A and the cluster of B the hosts belong to are simultaneously not available. In many clouds, for example, in Amazon EC2, the hosts are clustered into availability zones, and the zones are grouped into availability regions. In such a structure, the further the hosts of a cluster are away from each other, the more available the cluster is. But we should be aware that the high availability is obtained at the cost of performance.

3.3 Availability-Aware Resource Placement

The green one are respectively in cloud A and cloud B. It is obvious that the availability of the green one is higher than that of the blue one, since the blue one is not available when both the clusters of Cloud A the hosts belong to are simultaneously not available while the green one is not available when both the cluster of Cloud A and the cluster of B the hosts belong to are simultaneously not available. In many clouds, for example, in Amazon EC2, the hosts are clustered into availability zones, and the zones are grouped into availability regions. In such a structure, the further the hosts of a cluster are away from each other, the more available the cluster is. But we should be aware that the high availability is obtained at the cost of performance.
As a result, the geographic distribution of the hosts of a cluster should be limited into an acceptable range.

3.4 Power-Saved Resource Consolidation

From the view of cloud provider, the aim of resource consolidation is to save power and then cut down the operating cost. On the one hand, the cloud providers want to satisfy the resource requirements of cloud applications with minimal number of running physical hosts. On the other hand, they also hope that all the running physical hosts are running at appropriate status which means the utilization of computing resource on each physical host is greater than the lower bound and smaller than the upper bound.

The input of power-saved resource consolidation is the output of availability-aware resource placement in which the places of VMs are logical ones. The physical places of VMs will be located according to the runtime status of physical hosts. Furthermore, they are not fixed since the utilization of computing resource of physical hosts varies with the runtime workload of cloud applications. As a result, the periodical check for overloaded and underloaded nodes is executed and then the dynamical balancing is accomplished by VM migration.

In a cloud federation, it is possible for cloud providers to lease computing resource from each other which will result in a leasing loop. In such situation, the performance of cloud applications will be impacted since it incurs unnecessary remote communication cost. As a result, the resource consolidation needs to eliminate the leasing loop by VM migration too.


This section gives a design of the framework of cloud-federation-oriented resource management, as shown in Figure 5.
4.1 CLOUD FEDERATION CENTER

The Cloud Federation Center is the core of the framework we proposed. As shown in Figure 5, the Cloud Federation Center is composed of three parts.

The Lightweight Kernel is a service registry in which various service descriptions are published by the cloud providers. The cloud customers and providers query the kernel to discover desired services. We have designed and implemented a service registry which can discover the alternative services that meet the demand according to the specified functional and QoS requirements (Xiong & Chen, 2009). Therefore, the lightweight kernel can be implemented by reusing and extending the existing service registry to support the cloud-specific semantic descriptions. This kernel is necessary for either the Autonomous or the Centralized Controlled cloud federations.

The Extensions is used in the Centralized Controlled Cloud Federations but not in the Autonomous ones. The CF(Cloud Federation) Generator generates a solution for deploying Cloud applications according to the SLAs. The generated solution can be a single cloud, or a vertical, a horizontal or a combined cloud federation. We have designed mechanisms for determining the required quantity of computing resource based on the predicted performance (Chen, et al., 2011) and allocating the computing resource dynamically based on the required availability (Wang, et al., 2012). The Provider Manager is used by the administrator of the Cloud Federation Center to ensure that only qualified providers can register their services into the Lightweight Kernel. The Service Composition component is called by CF Generator to obtain a composite service from multiple clouds when the latter fails to find a single cloud as the hosting environment.

The Infrastructure is also used in the Centralized Controlled Cloud Federation but not in the Autonomous ones. The resource involved in cloud federation is monitored by the Resource Monitor at runtime to get their real-time status. The Revenue Distributor can reasonably distribute the revenue of cloud federation to all the involved clouds. The Service Coordinator can do the API and protocol transformation in order to coordinate the service cooperation across clouds.

4.2 DYNAMIC RESOURCE MANAGEMENT

The dynamic resource management in our framework is realized by the collaboration of App Manager, Cloud Manager and Resource Monitor in the Cloud Federation Center. There are three parts of dynamic resource management shown in Figure 5.

The Resource Monitoring Component is located in the Infrastructure part of the Cloud Federation Center. As we mentioned, in this component, the statistics and analysis of real-time monitoring data collected from the clouds in the cloud federation assure the cloud federation center grasping the global real-time state of computing resources.

The App Manager is one module of cloud providers. The Real-time Monitor component monitors the runtime status of Cloud applications deployed in the cloud, including the response time, throughput, failures and so on. It can be implemented in the manners of packet filtering or proxy. The Performance Analyzer will model and predict the performance of cloud applications based on the monitored data. The data obtained from the analysis can be used by the Resource Scheduler to allocate or reclaim computing resource for the Cloud applications. The Resource Arranger periodically rearranges the allocated computing resource by live migration of VMs in order to minimize the resource fragmentation generated at runtime.

The Cloud Manager is an existing module of cloud providers. We need to add some new functions to its existing components. The Resource Manager determines how and when to construct and deconstruct the Autonomous Cloud Federation based on the global utilization of its computing resource. The Payment Manager discriminates the revenue from cloud federation from that completely from its own cloud since the former needs to be distributed among the clouds involved into the cloud federation. The Leasing Manager doesn’t only manage the leasing contracts signed with the Cloud applications, but also manages the ones signed with other clouds in cloud federation. Meanwhile, the User Manager manages all the registered users and trusted cooperative cloud providers.

4.3 SERVICE COOPERATION

As shown in Figure 4, the service cooperation is implemented through the Service Coordinators in Cloud Federation Center and clouds. The Service Coordinator in cloud is a part of CF Manager, and comprises of the following modules.

The Comm(Communication) API should be consistent with the Service Coordinator in the Cloud Federation Center. Meanwhile, various cloud providers should provide adapters for the Comm API, and map it to their proprietary implementation.

The Security Controller realizes the strict control access and encryption of sensitive data which are necessary for all cloud providers.

The Logger realizes the log management. The configured log system is designed to assure the effective log management.

The Semantic Matcher provides a mechanism of semantics extending in which the ontologies and other formal methods are utilized to describe the semantics of collaborative behavior in cloud federation.

http://hipore.com/ijcc/
4.4 Revenue Distribution

As shown in Figure 5, the revenue distribution of the framework we proposed is implemented through the Revenue Distributor in Cloud Federation Center and the Profit Engines in clouds. The Profit Engine in cloud is a part of CF Manager, and comprises of the following modules.

The Pricing Manager is supported by the dynamic pricing mechanism of existing price management module of cloud providers. Meanwhile, the scheme of constructing cloud federation generated by the cloud federation center is used as an additional factor to determine the dynamic prices of computing resource.

The Revenue Distributor should be consistent with the Revenue Distributor in the Cloud Federation Center. The multi-objective optimization algorithm is utilized to design and implement the strategy of revenue distribution for multi-win.

The Budget Manager determines the leasing policies according to the SLAs of the Cloud application, including the quantity, location and VM types.

The Billing Manager is an existing component of cloud providers to compute the charge of cloud consumers which are either the Cloud applications or the other clouds.

5. The Mechanism of Multi-objective Constrained Resource Management

In this section, we will discuss the mechanism of multi-objective constrained resource management in details, including the management flow, the ways to implement the objectives.

5.1 The Flow of Resource Management

As discussed in section 3, the resource management of cloud federation is a multi-objective constrained one of which the objectives are profit-driven provisioning, availability-aware placement and power-saved consolidation.

In general, there are three ways to implement the multi-objective resource management: parallel process, serial process with feedback and serial process without feedback.

In the parallel process, all the objectives are independently processed to obtain the solution sets for each objectives and then the intersection of these solution sets is calculated. If the intersection has several solutions, one of them will be chosen randomly or by some other rule as the final result. If the intersection is an empty set, the objectives will compromise with each other and the intersection will be iteratively calculated until get a nonempty result. This way is apt to get a global optimal solution since all the objectives are processed independently. But it is obvious that this way is time-consuming one which makes it hard to be applied in practice.

In the serial process with feedback, all the objectives are processed one by one in some order and the output of the process of one objective will be the input of the process of next objective and is fed back to the process of previous objective. For example, the output of profit-driven provisioning is the amount of desired computing resource, which is the input of availability-aware placement. The latter will generate a topological structure of the desired computing resource which possibly violates the SLA on performance due to the overlong distance between VMs. So the topological structure needs to be fed back to the profit-driven provisioning in order to verify its feasibility. If the verification is passed, the topological structure will be sent to the power-saved consolidation to locate the VMs. If the verification is failed, the availability-aware placement will repeatedly generate a new output and feed it back to profit-driven provisioning until one output passes the verification. This way also can get a global optimal solution, however, it is still too time-consuming to be accepted in practice.

Serial process without feedback is designed to reduce the time consumption by removing feedback and adding constraints onto output. For example, the output of profit-driven provisioning is the amount of desired computing resource and the maximal acceptable distance between VMs. The availability-aware placement will generate a topological structure of VMs under such constraint which makes the feedback unnecessary since the performance is guaranteed by the constraint. This way cannot guarantee to generate a global optimal solution and just can generate an acceptable one which is possible far away from the former. However, the time consumption of this way is dramatically reduced due to the remove of feedback. Consequently, we adopt this way in our proposed mechanism.

In the serial process without feedback, the order of processing of objective embodies the importance of the objectives. The profit-driven provisioning determines the amount of desired computing resource which is the base for other two objectives. As a result, it is the first processed objective. The availability-aware placement determines the topological structure of the desired computing resource which is independent of specific physical nodes. So it is the second processed objective. The power-saved consolidation maps the topological structure onto the physical nodes. It certainly should be the last processed objective.

Summarily, the flow of multi-objective resource management is shown in Figure 6.
In Figure 6, the profit-driven provisioning module reads the monitored runtime status of cloud applications and physical nodes, SLA and resource descriptors as inputs, and calculates the amount of desired computing resource as output for scaling up or down. The availability-aware module receives the output of previous module and reads the static configuration about availability in resource descriptor and requirement in SLA to generate the topological structure of resource. This module needn’t read the runtime status because its output is independent of physical nodes. The power-saved consolidation module reads the monitored data and maps the topological structure onto physical nodes. Furthermore, this module periodically reads its output as input in order to consolidate the resource in background. Meanwhile, the whole flow will also be executed periodically to implement dynamic resource management.

5.2 **PROFIT-DRIVEN RESOURCE PROVISIONING**

Since the runtime workload is varying from time to time, the profit-driven resource provisioning needs to dynamically find the global maximal profit point in order to determine the amount of desired computing resource.

We have proposed a performance model for analyzing and forecasting the runtime workload of cloud applications and a method for determining the amount of computing resource need to be added or removed (Chen, et al., 2011). This model and method can be applied into the scenario of cloud federation.

For a cloud application deployed into a cloud federation, its operation cost at time $t$ is:

$$
\text{Cost}_t = \sum_{i=1}^{k} \left( \text{Price}_{rl_i} \times N_{rl_i} + \min(\text{Price}_{al_i}, \text{Price}_{od_i}) \times N_{d_i} \right)
$$

where $\text{Cost}_t$ is leasing cost, $\text{Price}_{rl_i}$, $\text{Price}_{al_i}$ and $\text{Price}_{od_i}$ denote the price of long-term lease, auction lease and on-demand lease of the computing resource of cloud $i$ respectively, $N_{rl_i}$ and $N_{d_i}$ represent the amount of resource of cloud $i$ for long term and short-term tenancy. This equation means that the cloud application will require a certain amount of long-term leased computing resource to process its regular workload and require the on-demand or on-spot resource to deal with the boom growth of workload. It is obvious that $N_{rl_i}$ and $N_{d_i}$ is determined by the runtime workload since the application needs appropriate amount of resource to guarantee the quality of service in SLA with the fluctuation of workload.

The revenue of a cloud application is:

$$
\text{Inc}_t = \sum_{i \in S} \sum_{j = 1}^{x_{it}} (c_i \times P_{ij} - \eta_i \times (1 - P_{ij}))
$$

where $\text{Inc}_t$ is the revenue, $i$ denotes the $i$th service the cloud application provided, $x_{it}$ denotes the number of accepted requests to the $i$th service while $j$ denotes the $j$th one of the $x_{it}$ requests. $P_{ij}$ is the probability that the measured quality of the $i$th service for the $j$th request is superior to the desired one. If a request is processed by the $i$th service without any violation of SLA, the application provider will obtain the incoming of $c_i$, otherwise, it will suffer the penalty of $r_i$. $P_{ij}$ is dependent on the quality of services, because the heavier the workload is, the poorer the quality of service is, and then the more possible that the provider will suffer penalty.

The difference between the $\text{Cost}_t$ and $\text{Inc}_t$ is the profit of application provider. Both of them are dependent variables of quality of service and runtime http://hipore.com/ijcc/
workload. We can find the global maximal profit point by analyzing the runtime status, including average response time and workload. Then, we can dynamically calculate the most appropriate amount of desired computing resource.

The cloud federation center will determine the construction and separation of cloud federation instances according to the provisioning result. The resource provisioning just determines the amount of computing resource needed but not specifies the source of the computing resource. The cloud federation center will determine how to locate the computing power according to the requirements of consumers. For instance, if the consumer wants to maintain the status that multiple instances of the cloud application should be deployed into multiple cloud providers, when the computing resource needs to be scaled down, the cloud federation center will just scale down the computing resource of some instances but not remove any instance. Either the profit of cloud applications or the one of cloud providers will be guaranteed as high as possible in such a dynamic scaling mechanism.

5.3 Availability-aware Resource Placement

We have proposed an availability-aware approach to place VMs for dynamic scaling of cloud applications (Wang, et al., 2012). In this approach, we used Bayesian formula to evaluate the availability of the infrastructure of a cloud application. For instance, the green solution in Figure 4 will be unavailable under the following situations: both the hosts are failure, one host is failure and the other is normal while the master host of its cluster or cloud is failure, both the hosts are normal while both the master hosts of their clusters or clouds are failure. With Bayesian formula, we can calculate the conditional probability that both the hosts of green solution are unavailable and then derive its availability.

In this approach, the maximal acceptable distance between any two VMs was also taken into account to prevent the violation of SLA on performance. However, we just put a single upper limitation on this distance which means all the VMs are equivalent to each other. In fact, if a multiple layered application is deployed as Figure 1 (b), (c) or (d), such a single upper limitation is not suitable any more. For example, the VMs in web layer can be distributed far away from each other since they needn’t to communicate with each other, while a VM of web layer should be close to a VM of application layer in order to reduce the communication cost between layers. So when this approach is applied into cloud federation, a set of the maximal acceptable distances but not a single distance is specified according the relationship between VMs, such as the distance between two VMs in same layer or the one between two VMs in two adjacent layers.

When applying this approach, we will define the availability for each cloud separately because the cloud federation is constructed with the resource from multiple clouds which are highly likely heterogeneous with each other. Thus, the physical nodes of each cloud, including physical machines, network switches and other network equipment, will be assigned with their own availability. The calculation of conditional probability in cloud federation is more complicate than that in a single cloud, so it is a bit time-consuming.

During the availability-aware resource placement, a VM placement plan is generated in which the places of VMs are logic ones but not physical ones. On the one hand, from the view of cloud application, the logic places, such as the relative distances between VMs are more important than the physical places because the physical hosts in a cluster are equivalent to each other. For example, for the blue solution in Figure 4, the cloud application concerns that its two VMs must be deployed into two clusters of a cloud while doesn’t care the VMs are deployed into which physical hosts of the two clusters. On the other hand, the physical places should be determined by cloud provider according to the runtime load of physical hosts.

5.4 Power-saved Resource Consolidation

We proposed an approach to dynamic workload balancing, which periodically checks the overloaded and underloaded nodes and then dynamically balances the workload by VM migration (Zhang, et al, 2011).

In this approach, we define the upper bounds and lower bounds of utilization of computing resource for VM and physical nodes. If the runtime utilization of a physical node is beyond the upper bound, this node will be marked as an overloaded node and its some VM(s) will be migrated to a paired physical node in order to ensure all the physical nodes are operating in normal mode. If the runtime utilization of a physical node is below the lower bound, this node will be marked as an underloaded node and it will be merged with other underloaded nodes or paired with an overloaded node. The virtual nodes are consolidated in the similar way. Taking the performance into account, we execute such consolidation locally but not globally. For example, a VM can only be migrated to the physical nodes in the same region of its host node.

6. Simulation and Analysis

We performed some simulation to show the effect of multi-objective constrained resource management in cloud federation. This section will give the details about the simulation.

6.1 Toolkit and Dataset of Simulation

http://hipore.com/ijcc/
We extended CloudSim (Calheiros, et al., 2011) with some modules designed by ourselves to support our research. The CloudSim toolkit we used was CloudSim 3.0, which was run at a Mac book with two 2.8GHz CPU and 4GB RAM running OS X 10.8. We used CloudSim to simulate two clouds each of which had tens of VMs.

The dataset of our simulation was NASA Ames iPSC/860 log (Feitelson, 1995). We parsed it and counted the numbers of jobs of each hour in order to simulate the workload. Actually, the iPSC/860 machine located at NASA Ames was a 128-node hypercube. At the time, it was the workhorse of the NAS facility for scientific computations. Up to nine jobs could run on the system at the same time, by using distinct subcubes.

Firstly, we used the autoregressive model to predict the runtime workload at next time node and calculated the amount of VMs needed to meet the performance requirement in SLA. Secondly, we calculated the availability to determine the topological structure of the desire amount of VMs in a cloud federation and then we placed the VMs in such a topological structure. Finally, we periodically consolidated the computing resource by VM migration.

6.2 WORKLOAD FORMATTING AND FORECASTING

The workload file of NASA Ames iPSC/860 log was suffixed with “swf” (Standard Workload Format). We preprocessed the workload and reserved some useful data fields. We calculated the amounts of jobs per hour, shown in Figure 7 (a), as the initial source input of our simulation.

As mentioned in Section 5.2, the real meaningful input is the forecasting workload of next time node but not the workload monitored at present time. So we used autoregressive model, based on the workload of eight previous time nodes, to forecast the dynamic workload of next time node. The result is shown in Figure 7(b).

In the corresponding of the real and forecasting workload, the deviation is about 22%, mostly happened in the peak nodes and some others in the nadir nodes. In most time nodes, the forecasting workloads are acceptable. If more precise forecasting workloads are needed, other complex models can be applied.

6.3 RESOURCE PROVISIONING

Based on the forecasting workload, the number of VMs, that is the amount of computing resource, is calculated according to the specific requirements in SLA. In CloudSim, we created two datacenters to simulate two clouds, each datacenter had two hosts. The first host had four CPUs at 10000 MIPS CPU and the second one had two same CPUs, while they both had 2G of memory. The VMs in each host are no more than twenty, and the number of VMs can be changed with the dynamic workload and requirement. Each VM has one CPU at 2000 MIPS and 512 MB of memory.

We chose the first one-hundred hours workload and set the upper bound of response time required of each job was not longer than 200ms. If the response time is beyond the upper bound, scaling up will be executed. Otherwise, we will check whether the computing resource is overprovisioned in order to make a decision to scale down. Figure 8 shows the correspondence of jobs and number of VMs, and we can see the effect of scaling up and down since the number of VMs is changed with the number of jobs.
6.4 Resource Placement

We had two clouds simulated in CloudSim with their own average availabilities of regions, zones and hosts, shown as Table 1.

Suppose the communication costs between two VMs on a single host, on different hosts in a single zone, in different zones of a single region, in different regions of a single cloud, and in different clouds are respectively 0, 1, 2, 3 and 6. To satisfy the availability requirement of that being greater than 99.99%, we used availability-aware policy with relocation of VMs (Wang, et al., 2012), and the Figure 9 shows the average availabilities of the system under the two-day workload, which corresponds to the first 48 hours of Figure 8. The reason why we chose the first two-day workload but not the whole five-day workload is just for simplification since the figure will be unclear if we rendered the latter.

<table>
<thead>
<tr>
<th>Cloud</th>
<th>Region</th>
<th>Zone</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>99.99%</td>
<td>99.98%</td>
<td>99.8%</td>
</tr>
<tr>
<td>B</td>
<td>99.98%</td>
<td>99.97%</td>
<td>99.7%</td>
</tr>
</tbody>
</table>

Table 1. Availabilities of clouds

Figure 9. The average availabilities of the system under the 48 hours workload. The blue line shows the VM amount needed per hour. The green line shows the average availabilities with neither placement policy nor VM relocation. The red line with squares shows the average availabilities with availability-aware policy and VM relocation.

From the result shown in Figure 9, the availability-aware resource placement is useful and meaningful for the aim of high-availability since it can guarantee the availability is always higher than the required one.

6.5 Resource Consolidation

We consolidated the deployment of applications in the CloudSim. Our aim is improve the utilization of the whole physical resources. To prove our strategy is useful, we expanded our simulation environment from 4 hosts to 40 hosts, and each host still had 20 virtual machines. Every 20 hosts were in the same zone where the old 2 hosts in. We set the lower bound of utilization is 20% and the upper bound is 70%. We deployed the
application into 40 hosts with 800 virtual machines initially, and then we did the consolidation in every 300 seconds.

![Figure 10](image.png)

**Figure 10.** The result of consolidation. The red line shows the number of physical hosts. The black line shows the number of VM migration.

In the first time of deployment without consolidation, almost all the 800 virtual machines were used, and the 40 hosts were of low utilization which is from 2% to 16%. Under our consolidation strategy, the virtual machines in the minimum-utilization host were migrated to another host in the same zone. After doing consolidation for one day (about 245 times), as shown in Figure 10, the amount of physical hosts reduced from 40 to 3. The final three hosts were of the utilization of 57.07%, 55.63% and 62.72% respectively.

7. Related Works

The concept of cloud federation was first mentioned as Intercloud by Kevin Kelly in 2007, and he said “eventually we’ll have the Intercloud, the cloud of clouds.” (Kelly, 2007). Sam Johnston further expatiated that “the Intercloud is a global cloud of clouds as the Internet is a global network of networks” (Johnston 2009). However, the concept of Intercloud didn’t receive enough concerns, because there was little consensus on how to define the Cloud and many people considered that cloud computing was just a redefinition of the commercial by existing technology.

With the continuous development of cloud computing, more and more people have profoundly understood the essence of cloud computing and realized the importance of cloud federation. During 2009, some researchers used cloud federation to describe the future data center. One of the most important papers was the “Blueprint for the Intercloud” (Bernstein, D., et al., 2009). This blueprint concerned protocols and formats for cloud computing interoperability but didn’t put forth the scheme of many other problems such as when and how to build intercloud and how to distribute profits among all the providers.

In 2009, Global Inter-Cloud Technology Forum (GICTF, 2009) was established in Japan and attempted to promote development of intercloud. In 2010, this forum published a White Book about use cases and functional requirements for intercloud computing (GICTF, 2010).

Research on the architecture of cloud federation is the most fundamental among all researches about cloud federation. One of the two main architectures is using an independent third-party heavyweight cloud federation center as the core which takes charge of the dynamic combination and resolution of cloud federation, such as an architecture proposed by Buyya, et al. in 2010. This architecture is convenient to use and don’t have drastic changes to the existing cloud architectures, but cloud consumers must change the mode of using cloud resources and the center is the single point of failure of this architecture.

Considering of the autonomy of cloud federation, more researcher prefer the other architecture — the lightweight cloud federation center. In this architecture, the center takes charge of the registration and query of resource information from every cloud; meanwhile, all the clouds combine into and split from the cloud federation dynamically autonomously. Two typical representatives of this architecture are a cloud federation mode proposed by Antonio Celesti, et al. (Celesti, et al., 2010) and RESERVIOR mode proposed by IBM (Rochwerger, et al., 2009). In this architecture, the pressure of cloud federation center is largely reduced. This architecture is suitable for active collaboration between cloud providers, but not so helpful for Cloud applications of which all the tiers are not deployed into a single cloud. The two modes mentioned above both are based on performance, not considering other factors, such as availability and power saving, so they can’t fully meet the actual multi-objective demand.

Rodrigo N. Calheiros et al designed Aneka, a platform for developing scalable applications on the Cloud, supports such a vision by provisioning resources.

http://hipore.com/ijcc/
from different sources and supporting different application models (Calheiros, et al., 2012). They mentioned that the key concepts and features of Aneka support the integration between Desktop Grids and Clouds. Like almost all existing research on cloud federation, Aneka aims at how to scale up the computing power by integrating the computing resource from multiple providers. They ignored the cloud federation which is built for improving the availability and obtaining independence of providers and best quality of services.

The key of cloud federation is that the clouds can communicate with each other by a unified API and specific adaptors. Apache Deltacloud is right such a project that “gives customers an opportunity to manage cloud instances in the way they want” (Apache, 2013). This project facilitates the construction of cloud federation and makes it feasible. But it doesn’t provide customers any functions to automatically request computing resource according to their constraints.

In conclusion, the most existing research on cloud federation focuses on how to scale up the computing power by cloud federation but not how to improve the quality of services by cloud federation. Both of the two aspects are important for cloud applications. So this paper tried to give a more comprehensive analysis and design of resource management of cloud federation.

8. Conclusions

This paper classify the cloud federations into different models according to their structure and control modes, including vertical, horizontal and hybrid federation, autonomous and centralized controlled federation. A design of the framework of multi-objective constrained resource management for cloud federation is proposed based on the analysis of the common objectives of such a framework, including dynamic profit-driven provisioning, availability-aware placement and power-saved consolidation. The result of our simulation has demonstrated the feasibility and effectiveness of the proposed framework.

With an implementation of the framework proposed in this paper, the independent cloud federation center would be able to schedule computing power for the providers of cloud applications in a transparent way, which would greatly lower the technical threshold of application of clouds.

9. References


**Authors**

Haopeng Chen, Associate Professor of School of Software at Shanghai Jiao Tong University. He received his Ph.D degree from Department of Computer Science and Engineering, Northwestern Polytechnical University in 2001. His research group focuses on Distributed Computing and Software Engineering.

Wenyun Dai, born in Yangzhou, Jiangsu, P.R. China. He graduated from Xiamen University major in Software Engineering in 2010. He is now a third-year master student in School of Software, Shanghai Jiao Tong University. He is interest in resource management in clouds and multi-objective optimization.
Call for Articles
International Journal of Services Computing

Mission
The International Journal of Services Computing (IJSC) aims to be a reputable resource providing leading technologies, development, ideas, and trends to an international readership of researchers and engineers in the field of Services Computing. To ensure quality, IJSC only considers extended versions of papers published at reputable international conferences such as IEEE ICWS.

From the technology foundation perspective, Services Computing covers the science and technology needed for bridging the gap between Business Services and IT Services, theory and development and deployment. All topics regarding Web-based services lifecycle study and management align with the theme of IJSC. Specially, we focus on: 1) Web-based services, featuring Web services modeling, development, publishing, discovery, composition, testing, adaptation, and delivery, and Web services technologies as well as standards; 2) services innovation lifecycle that includes enterprise modeling, business consulting, solution creation, services orchestration, services optimization, services management, services marketing, business process integration and management; 3) cloud services featuring modeling, developing, publishing, monitoring, managing, delivering XaaS (everything as a service) in the context of various types of cloud environments; and 4) mobile services featuring development, publication, discovery, orchestration, invocation, testing, delivery, and certification of mobile applications and services.

Topics
The International Journal of Services Computing (IJSC) covers state-of-the-art technologies and best practices of Services Computing, as well as emerging standards and research topics which would define the future of Services Computing. Topics of interest include, but are not limited to, the following:

- Services Engineering
- XaaS (everything as a service)
- Cloud Computing for Internet-based services
- Big Data services
- Internet of Things (IoT) services
- Pervasive and Mobile services
- Social Networks and Services
- Wearable services
- Web 2.0 and Web X.0 in Web services
- Service-Oriented Architecture (SOA)
- RESTful Web Services
- Service modeling and publishing
- Service discovery, composition, and recommendation
- Service operations, management, and governance
- Services validation and testing
- Service privacy, security, and trust
- Service deployment and evolution
- Semantic Web services
- Scientific workflows
- Business Process Integration and management
- Service applications and implementations
- Business intelligence, analytics and economics for Services
Call for Articles
International Journal of Big Data

Mission
Big Data has become a valuable resource and mechanism for the practitioners and researchers to explore the value of data sets in all kinds of business scenarios and scientific investigations. New computing platforms such as cloud computing, mobile Internet, social network are driving the innovations of big data. From government initiative perspective, Obama Administration in United States launched "Big Data" initiative that announces $200 Million in new R&D investments on March 29, 2012. European Union also announced "Big Data at your service" on July 25, 2012. From industry perspective, IBM, SAP, Oracle, Google, Microsoft, Yahoo, and other leading software and internet service companies have also launched their own innovation initiatives around big data.

The International Journal of Big Data (IJBD) aims to provide the first Open Access publication channel for all authors working in the field of all aspects of Big Data. Big Data is a dynamic discipline. One of the objectives of IJBD is to promote research accomplishments and new directions. Therefore, IJBD welcomes special issues in any emerging areas of big data.

Topics
IJBD includes topics related to the advancements in the state of the art standards and practices of Big Data, as well as emerging research topics which are going to define the future of Big Data. Topics of interest to include, but are not limited to, the following:

**Big Data Models and Algorithms** (Foundational Models for Big Data, Algorithms and Programming Techniques for Big Data Processing, Big Data Analytics and Metrics, Representation Formats for Multimedia Big Data)

**Big Data Architectures** (Cloud Computing Techniques for Big Data, Big Data as a Service, Big Data Open Platforms, Big Data in Mobile and Pervasive Computing)

**Big Data Management** (Big Data Persistence and Preservation, Big Data Quality and Provenance Control, Management Issues of Social Network enabled Big Data)

**Big Data Protection, Integrity and Privacy** (Models and Languages for Big Data Protection, Privacy Preserving Big Data Analytics Big Data Encryption)

**Security Applications of Big Data** (Anomaly Detection in Very Large Scale Systems, Collaborative Threat Detection using Big Data Analytics)

**Big Data Search and Mining** (Algorithms and Systems for Big Data Search, Distributed, and Peer-to-peer Search, Machine learning based on Big Data, Visualization Analytics for Big Data)
