Virtual Networking with Azure for hybrid Cloud Computing in Aneka

Adel Nadjaran Toosi and Rajkumar Buyya

Abstract Hybrid cloud environments are a highly scalable and cost-effective option for enterprises that need to expand their on-premises infrastructure. In every hybrid cloud solutions, the issue of inter-cloud network connectivity has to be overcome to allow communications, possibly secure, between resources scattered over multiple networks. Network visualization provides the right method for addressing this issue. We present how Azure Virtual Private Network (VPN) services are used to establish an overlay network for hybrid clouds in our Aneka platform. First, we explain how Aneka resource provisioning module is extended to support Azure Resource Manger (ARM) application programming interfaces (APIs). Then, we walk through the process of establishment of an Azure Point-to-Site VPN to provide connectivity between Aneka nodes in the hybrid cloud environment. Finally, we present a case study hybrid cloud in Aneka and we experiment with it to demonstrate the functionality of the system.

1 Introduction

Cloud computing is the mainstream paradigm for delivering on-demand and easy-to-use computing services in a pay-as-you-go model. In this paradigm, consumers and organizations adopt cloud-based computational resources and services to deploy their applications and store data. The increasing dependence on information technology (IT) and global explosion of data over the last decade has fostered this adoption more than ever. Moreover, this trend is expected to continue in the upcoming decades as cloud computing becomes the integral and essential part of many emerging IT technologies such as Internet of things [6] and big data applications [1].

Adel Nadjaran Toosi and Rajkumar Buyya
Cloud Computing and Distributed Systems (CLOUDS) lab., School of Computing and Inforamtion Systems, The University of Melbourne, Australia. e-mail: {anadjaran,rbuyya}@unimelb.edu.au

Among the many different forms of cloud computing, hybrid clouds, in which organizations' on-premises infrastructure are expanded by adding third-party public cloud resources, provides one of the best blends for hosting applications. A hybrid cloud allows a seamless integration of an existing on-premises infrastructure (usually a private cloud) and a public cloud, enabling the cloud bursting deployment model. In cloud bursting model, applications run in a private infrastructure and bursts onto a public cloud when the demand for computing capacity spikes. Hybrid cloud delivers the benefits of both the public cloud and in-house cloud computing infrastructures due to its native characteristics such as cost reduction and compliance with the location of sensitive data [18].

In hybrid cloud environments, computational resources are scattered throughout disparate sets of networks (i.e., private and public cloud networks). One of the main issues arises in such a scenario is how nodes (e.g., virtual machines) from multiple sites and clouds are connected together. In other word, there is the issue of managing two separate sets of IP ranges that would have to be combined to enable automated resource provisioning and migration across clouds. Allocation of public IP addresses to nodes requiring communicating with each other provides a viable solution for this issue. However, providing public IP addresses, in particular public IP addresses for private cloud resources in the organizational infrastructure, is not always feasible. Another issue is that in most, if not all, hybrid cloud scenarios, a secure communication channel needs to be built between the private on-premises infrastructure and cloud resources as the public Internet is used to transmit data.

The network virtualization techniques are key enablers to address these issues by constructing of an overlay network over the existing networks such as the Internet. A virtual private network (VPN) is an overlay network that creates a secure network connection to a private network across a public network. As many more cloud providers offering VPN services, enterprises and organizations looking into hybrid cloud solutions can utilize these VPN services to manage their hybrid cloud platforms. In this book chapter, we go through the process of building a hybrid cloud solution for our Aneka platform using virtual network services of Microsoft Azure.

Aneka [7, 17] is a platform-as-a-service (PaaS) solution providing a middleware for the development and deployment of applications in hybrid and multi-clouds. Aneka provides application developers with Application Programming Interfaces (APIs) for transparently harnessing and exploiting the physical and virtual computing resources in heterogeneous networks of workstations, clusters, servers, and data centers. Scheduling and resource provisioning services provided by Aneka allow for dynamic growth and shrinkage of the Aneka cloud to meet Quality of Service (QoS) requirements of deployed applications. This chapter also discusses our extension to Aneka resource provisioning based on Microsoft Azure Resource Manager (ARM) deployment model.

The rest of the chapter is organized as follows: Section 2 describes hybrid cloud and its benefits. Section 3 discusses the connectivity issue between clouds in a hybrid cloud environment. Section 4 proposes virtual private networks (VPNs) as a solution for the connectivity issue in hybrid clouds and explores various Azure VPN connections. In Section 5, our Aneka Platform-as-a-Service (PaaS) is introduced as

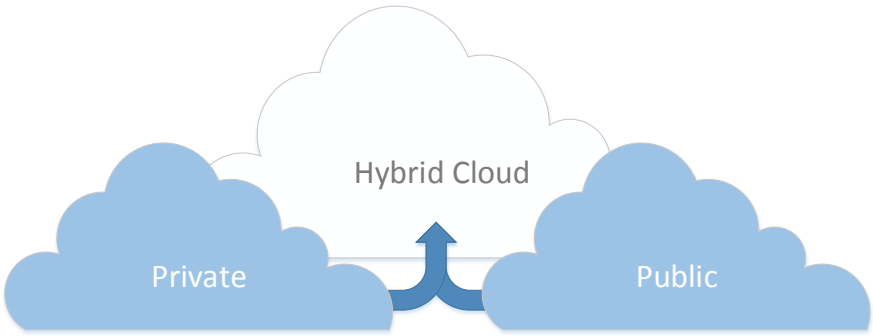


Fig. 1 Schematic view of a hybrid cloud.

a tool for building a hybrid cloud environment. We explain how Azure VPNs can be used to provide inter-cloud connectivity for the establishment of Aneka hybrid cloud in Section 6. To demonstrate the effectiveness VPN for hybrid cloud solutions, in Section 7, we represent details of a case study on creation of hybrid cloud combining private cloud resource (networked desktop computers) and Azure public cloud resources and the deployment of application on the hybrid cloud infrastructure along with experimental results. Section 8 defines some open research problems and pathways for future work. This chapter is summarized in Section 9.

2 Hybrid Clouds

The hybrid cloud, as shown in Fig. 1, is an integration of a public cloud provider such as Microsoft Azure or Amazon Web Services (AWS) with a private cloud platform which is designed to be used within the boundary of a single organization. In a hybrid cloud environment, the organization that owns the private cloud moves part of its operations to the external public cloud provider. In this scenario, the public and private clouds are independent and distinct entities. This allows the organization to perform protected operations and store sensitive or privileged data on its own private infrastructure while retaining the ability to leverage public cloud computational resources when demand for computation exceeds available capacity. This provides the following benefits to the organizations:

- **Cost savings**: Hybrid cloud solutions can increase cost savings. Public clouds' *pay-as-you-go* model give the organization the flexibility of using public cloud resources as much as they require and removes the need for building an internal infrastructure that endures occasional bursts in demand.
- **Security**: Improved security is another main benefit of hybrid clouds. In hybrid cloud model, the organization utilizing hybrid model can run sensitive operations

and store sensitive data in the private cloud platform. This helps to protect privacy of data and comply with location and regulatory requirements where it is applicable.
- **Scalability**: A hybrid cloud environment provides the opportunity to expand capacity by adding external resources to the pool of available resources. This allows for scaling resources up and down as demands change in order to optimize performance and efficiency.
- **Availability and Reliability**: While private clouds do offer a certain level of availability, public cloud services will offer a higher level of availability. By replicating data across hybrid clouds and moving as many non-sensitive tasks as possible to the public cloud, the organization can benefit from higher availability and fewer outages. The hybrid cloud model is also an appealing choice for disaster recovery plans and more reliable systems.

The benefits of hybrid cloud solutions are not limited to the above list. Other benefits such as business agility, more flexibility, and better accessibility can also be resulted from successful application of a hybrid cloud solution.

3 Connectivity Issue in Adoption of Hybrid Clouds

Similar to many other IT examples, even though building hybrid clouds bring many benefits, they still face some challenges that must be addressed before they can be effectively used. Apart from challenges such as portability, compatibility, and needs for middleware supporting hybrid clouds, the connectivity issue is an integral part of every hybrid cloud deployments. Since computational resources in hybrid cloud environments are scattered across multiple administrative domains, needs for reliable, responsive and secure connections between machines (either virtual or physical) residing in separate networks arise.

Even though one might think that allocation of public IP addresses to these machines resolves the connectivity issue, there are several technical barriers which obstruct such application. They include:

1. Assigning public IP addresses to all machines in the hybrid cloud is not feasible in many cases and is a waste of resources. Because IP addresses are limited and many organizations have access to limited range of public IP addresses.
2. Even if organizations can afford to allocate public IP addresses to private cloud machines, this involves security risks outweighing its benefits. Moreover, machines residing in the private infrastructure often located behind organizational firewalls or network address translation (NAT) that protect them from being directly accessed by devices from outside networks.
3. Lastly, Public IP addresses do not address the issues related to secure communications between private and public cloud resources.

Issues related to the connectivity issue in the hybrid cloud can be addressed by *network virtualization*. Network virtualization is concerned with the construction of

virtual networks over an existing network. In the next section, we discuss how VPN services can resolve the issues related to connectivity and segregated networks in hybrid cloud platforms.

4 Virtual Private Networks

A VPN is a cost effective and secure way of extending an enterprises private network across a public network such as the Internet. It constructs an overlay network on top of an existing network such as an IP network without changing characteristics of the underlying network. VPNs allow remote resources located outside the local infrastructure for example a public cloud to securely access local network resources and vice versa. This creates a secure communication channel for resources scattered over public and private cloud networks and facilitates automated resource provisioning and migration across them.

Thanks to ongoing advances in *Network Virtualization* solutions and technologies employed by public cloud providers such as Amazon Virtual Private Cloud (Amazon VPC),[1] Google Cloud Virtual Network,[2] and Microsoft Azure Virtual Networks (VNet),[3] building overlay VPNs connecting on-premises networks to public cloud virtual networks becomes more convenient.

4.1 Microsoft Azure VPNs

In this book chapter, we present our experience with using Microsoft Azure *Point-to-Site* VPN connections for building a hybrid cloud platform for our Aneka tool (Discussed in Section 5). To send network traffic between the on-premises site and Azure Virtual Network (VNet), Azure provides various VPN connectivity options as follows:

1. **Site-to-Site** A Site-to-Site (S2S) VPN connection is a connection over IPsec (Internet Protocol Security) VPN tunnel. This type of connection requires a VPN device located in on-premises infrastructure with an assigned public IP address not behind a NAT. A virtual private network gateway (VPN gateway) must also be created for the Azure VNet. In Site-to-Site connections, multiple VPN connections from on-premises sites to the VPN gateway can be made which is often called a "multi-site" connection. Fig. 2 shows a sample Site-to-Site VPN connections.
2. **Point-to-Site** A Point-to-Site (P2S) connection allows the creation of secure connections from an individual machine in the on-premises network to the Azure

[1] `https://aws.amazon.com/vpc/`

[2] `https://cloud.google.com/virtual-network/`

[3] `https://azure.microsoft.com/en-us/services/virtual-network/`

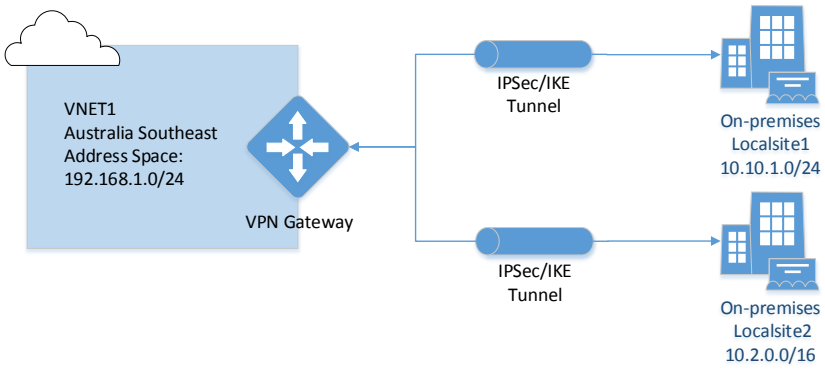


Fig. 2 Azure Site-to-Site VPN connection.

VNet. This VPN connection is built over SSTP (Secure Socket Tunneling Protocol) and does not require a VPN device or a public-facing IP address to work. This solution is used for our case study scenario in Section 7 since our hybrid cloud testbed is built on top of multiple desktop machines behind our organizational NAT. Fig. 3 shows a sample Point-to-Point VPN in Azure.

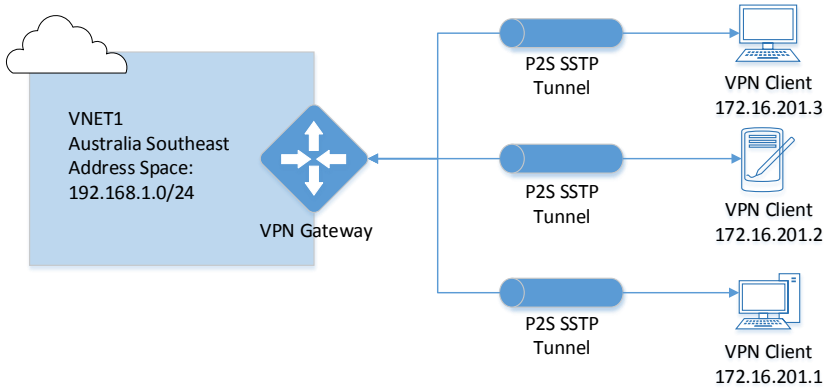


Fig. 3 Azure Point-to-Site VPN connection.

3. **VNet-to-VNet** Connecting an Azure virtual network to another Azure virtual network (VNet-to-VNet) is similar to connecting a VNet to an on-premises site. Both VNets use a VPN gateway to provide a secure tunnel using IPsec/IKE. VNet-to-VNet communication can be combined with multi-site connection con-

figurations. Fig. 4 illustrates the schematic view of a sample VNet-to-VNet in Azure.

In addition to above VPN connections, Azure also provides `ExpressRoute` for those customers in a co-location with Azure cloud exchange to create private connections that directly connects their on-premises infrastructure to Azure data centers.

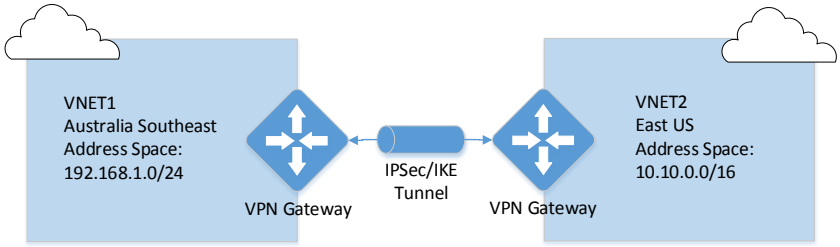


Fig. 4 Azure VNet-to-VNet VPN connection.

5 Aneka Cloud Application Platform

Aneka [17] is a *Platform-as-a-Service* framework to facilitate the development and deployment of cloud applications. It offers a collection of tools to build, control, and monitor an Aneka cloud environment. The Aneka cloud can be composed of a collection of heterogeneous resources on the public cloud or the premises of an enterprise, or a combination of both. Aneka provides application developers with *Application Programming Interfaces* (APIs) for transparently exploiting physical and virtual resources in the Aneka cloud. Developers express the logic of applications using programming models and define runtime environments on top of which applications are deployed and executed. Currently the following four different programming models are supported by the Aneka platform [17]:

1. *Bag of tasks model*: expressing bag-of-tasks and workflow applications;
2. *Distributed threads model*: allowing for execution of applications composed of independent threads (i.e, threads that do not share data);
3. *MapReduce model*: leveraged for processing of large data sets based on the implementation of Google's MapReduce [10]; and
4. *Parameter sweep model*: designed for execution of the same task over different ranges of values and datasets from a given parameter set.

The Aneka framework has been designed and implemented in a service-oriented fashion. Services are the extension point of the platform allowing for the integration of new functionalities and the replacement of existing ones with different implementations. Brief descriptions of some of these services that are central to hybrid cloud deployment are provided below:

Scheduling: The main role of scheduling service is to assign tasks to available resources. Aneka can be configured to use static resources that are available from the beginning of the scheduling process or dynamic resources that can be provisioned based on the requirements. In a dynamic configuration, the scheduling service communicates with the provisioning service to provision or release resources based on the scheduling algorithm decisions. Scheduling algorithms in Aneka can be developed to fulfill specific Service Level Agreements (SLA) required by users such as the satisfaction of deadline or budget constraints for certain applications.

Provisioning: The provisioning service is in charge of acquiring and releasing resources from different cloud providers. To satisfy this requirement, Aneka offers a specialized resource pool connection for each cloud provider that invokes the interface of that specific cloud provider to provision and release resources. These connections are managed by resource pool manager to create a pool of connections that can be invoked any time for the specific cloud provider. Resource pool connections can be created through the *Aneka Management Studio* for adding static resources or as part of dynamic resource provisioning configuration for scheduling algorithms. Fig. 5 shows an overview of dynamic resource provisioning and the interaction between scheduling and provisioning services in Aneka. In Section 5.2, we present our extension to Aneka to support Microsoft *Azure Resource Manager* (ARM) deployment model for the management of Azure provisioned resources.

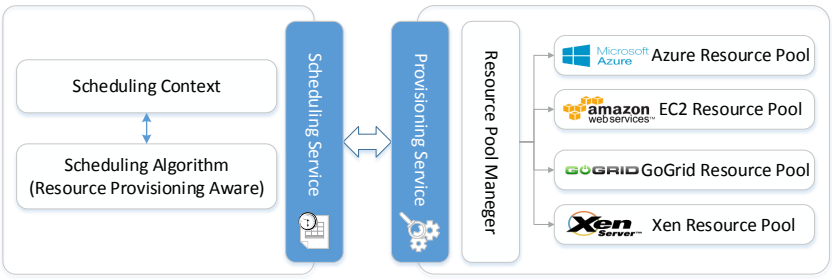


Fig. 5 The interaction between scheduling and provisioning services for dynamic resource provisioning in Aneka.

Storage: This service is in charge of the management of data and provides an internal storage for applications. It provides applications with basic file transfer facilities and performs data transfers among Aneka nodes. The current release of

Aneka provides a storage implementation based on the File Transfer Protocol (FTP) service.

Apart from the above services, Ankea provides other fundamental services such as reservation, licensing, accounting, membership and execution services as part of its platform services.

5.1 Aneka Architecture

Fig. 6 provides the architecture and fundamental services that compose the Aneka platform. The figure shows a layered view of the Aneka components. Aneka provides a runtime environment for executing applications by leveraging heterogeneous resources on the underlying *infrastructure* built on the top of computing nodes employed from network of desktop machines, clusters, and data centers. In other words, the infrastructure layer is a collection of nodes hosting components of Aneka middleware.

The *middleware* provides a collection of services for interactions with the Aneka cloud. The container represents the unit of deployment for Aneka clouds and the runtime environment for services. The core functionalities residing in the *Platform Abstraction Layer (PAL)* constitute the basic services that are used to control the infrastructure of Aneka clouds. It provides a uniform interface for management and configuration of nodes and the container instances deployed on them in the infrastructure layer. Middleware is composed of two major components representing the building blocks of Aneka clouds: the *Aneka Daemon* and *Aneka Container*. Each node hosts the Aneka daemon and one or more Aneka container instances. The daemon is a management component controlling the container instances installed on the particular node. A node running the Aneka master container plays the role of resource manager and application scheduler. Nodes running Aneka worker containers are responsible for processing and executing work units of the applications sent from the master node. In addition, each container provides a messaging channel for accessing features of different services provided by the container. There are three classes of services characterizing the container:

1. *Execution services*: are responsible for scheduling and executing applications. Specialized implementations of these services are defined for the execution of work units of each programming model supported by Aneka.
2. *Foundation services*: are in-charge of metering applications, allocating resources, managing the collection of available nodes, and keeping the registry of services updated.
3. *Fabric services*: provide access to the physical and virtualized resources managed by the Aneka cloud. The *Resource Provisioning Service (RPS)* enables horizontal scaling out and allows for elastic and dynamic growth and shrinkage of the Aneka cloud to meet Quality of Service (QoS) requirements of applications.

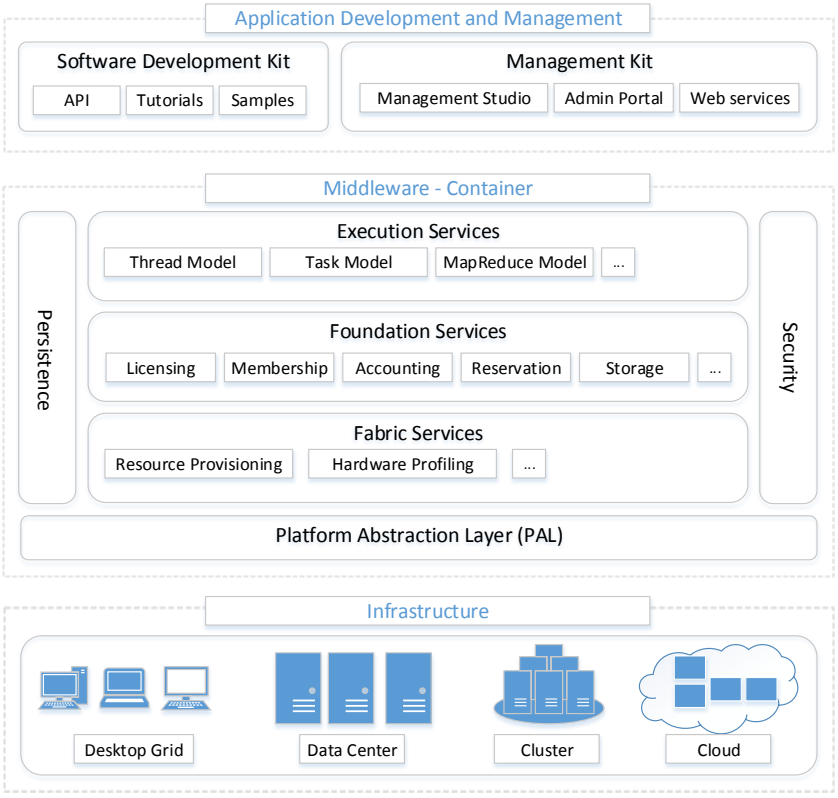


Fig. 6 Aneka Framework Overview.

The services of the middleware are accessible through a set of interfaces and tools in the *development and management* layer. The *Software Development Kit (SDK)* embodies a collection of abstractions and APIs for definition of applications and leveraging existing programming models. The *Management Kit* contains a collection of tools for management, monitoring, and administration of Aneka clouds. All the management functions of the Aneka cloud are made accessible through the *Management Studio*, a comprehensive graphical environment providing a global view of the cloud for administrators.

5.2 Extending Aneka Resource Provisioning with Azure Resource Manager

Aneka resource provisioning service currently supports provisioning requests for cloud providers such as Amazon EC2, GoGrid, and Microsoft Azure. In recent years, Microsoft Azure [4] has undergone a significant transformation and, as a result of that, two different sets of *Azure Resource Manager* (ARM) and *Classic* APIs exist for resource management and deployment in Azure. The ARM and Classic deployment models represent two different ways of managing and deploying Microsoft Azure solutions. Aneka originally supported the Classic deployment model [5] in which each resource (e.g., storage disk, VM, Public IP address, etc.) existed independently and there was no way to group them together. In 2014, Azure offered ARM to simplify the deployment and management of resources by introducing the resource group concept as a container for resources that share a common lifecycle. In order to enable Aneka to use the new Azure APIs for resource provisioning, we extended Aneka by adding an ARM-based resource pool connections.

ARM provides Azure customers with a set of *representational state transfer* (REST) APIs to access Azure IaaS services. These RESTful APIs provide service endpoints that support sets of HTTP operations to access, create, retrieve, update, and delete the Azure cloud resources. This way, resources in Azure can be accessed programmatically. Additionally, ARM supports JSON (JavaScript Object Notation)-based declarative templates to deploy multiple services along with their dependencies. Templates can be used repeatedly to provision resources and deploy applications. In the simplest structure a template contains the following elements:

```
JSON
{
      "$schema": "",
      "contentVersion": "",
      "parameters": { },
      "variables": { },
      "resources": [ ],
      "outputs": { }
}
```

`$schema` describes the version of the template language. Any value can be provided for `contentVersion` and it is used to make sure that the right template is being used for deployments. To customize resource deployment `parameters` values are set. `variables` are used to simplify the template language expressions. `resources` represents resource types that are deployed or updated in a resource group, e.g., VM, VNet or NIC. `outputs` are returned after deployments.

We added `AzureRMResourcePool` class in Aneka as the entry point to access Microsoft Azure resources based on ARM interfaces. This class implements the `IResourcePool` interface of Aneka and in this way, it is transparently included in the design. `AzureRMResourcePool` implements `Provision()` and `Release()` methods of the `IResourcePool` interface using our Azure template stored in

`azuretemplate.json`. The template is responsible for the creation of a Virtual Machine (VM) and its dependencies such as network interfaces and OS disk based on a given URI of the VM image. The VM image is a VHD (Virtual Hard Disk) file containing a Windows Server 2012 operating system on which Aneka Worker container is configured and installed. In the `AzureRMResourcePool` class, we pass to Azure APIs the aforementioned template together with a JSON object containing parameters for VM required configurations (e.g, Admin username and password, type of VM and etc.) and references to other already created resources on which creation of a VM is dependent (e.g., Virtual network, network Security Group for the VM, storage account). These parameters are provided to `AzureRMResourcePoolConfiguration` class and are set by the administrators when they are customizing the Aneka cloud platform. Fig. 7 shows a list of these parameters in a screenshot of Resource Pool Manager Editor window in Aneka Management Studio. Whenever, `AzureRMResourcePool` receives a resource provisioning request, it creates a Resource Group containing the VM and all its dependencies and returns a reference for this deployment to the Aneka Resource pool Manger. Release requests are simply satisfied with deleting the resource group.

6 Configuration of Azure Point-to-Site VPN Connection for Aneka Hybrid Cloud

Our aim is to utilize Aneka to expand the computational capacity of a small desktop cloud build on top several desktop machines by adding extra resources from Azure. Since we do not have privileged access to networking devices in our organization, we choose to employ Azure Point-to-Site connection to create our hybrid overlay VPN. The first step for the establishment of a hybrid cloud network spanning over our on-premises network and the Azure cloud is to create a *VNet* and a *VPN gateway* in Azure.

We created a VNet called `Aneka-VNET-SITE` for our target Azure region which is `Australia Southeast`. The Aneka-VNET-SITE VNet has two subnet address ranges of 192.168.1.0/24 called `FrontEnd` and 192.168.200.0/26 as `GatewaySubnet`. In order to add a VPN gateway to this VPN, a public IP address was created. This IP is used by desktop machines to join the virtual private network. The IP address created this way was configured and connected to the VPN gateway.

We required certificates as VPN clients need to authenticate with Point-to-Site VPNs. For this purpose, a root certificate generated by an enterprise certificate authority (CA), or a self-signed root certificate can be used. We opted to generate a self-signed root certificate. We exported the public certificate data without a private key as a Base-64 encoded X.509. This Root certificate is imported to the Point-to-Site configuration of the Azure Gateway we created earlier. We also used our self-signed root certificate to generate client certificates for our VPN clients. Even though, the same certificate can be used for multiple clients; we exported a unique certificate including the private key for each client. The advantage is that, later on,

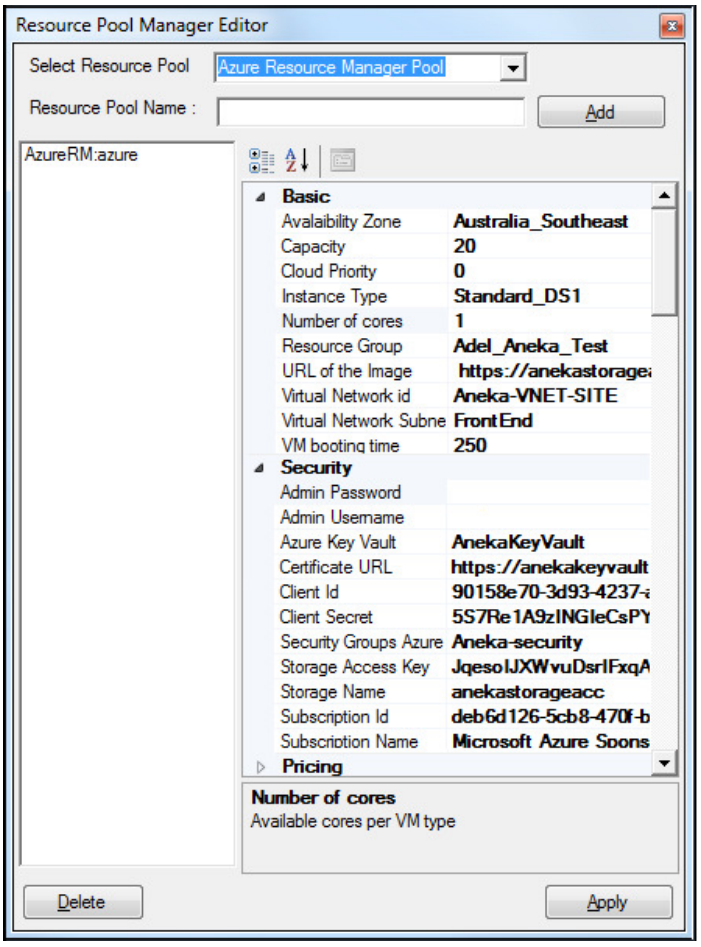


Fig. 7 Azure Resource Manager (ARM) resource pool configuration in Aneka.

we can revoke the certificate for each client individually. The last step is to download and install the VPN client configuration package for all clients. For each and every client connecting to the virtual network, we installed both the client certificate and a VPN client configuration package which can be directly downloaded from Azure. At this point, all clients desktop machines were able to connect to our VPN. Fig 8 shows the Azure Point-to-Site virtual network and IP address ranges created for our platform.

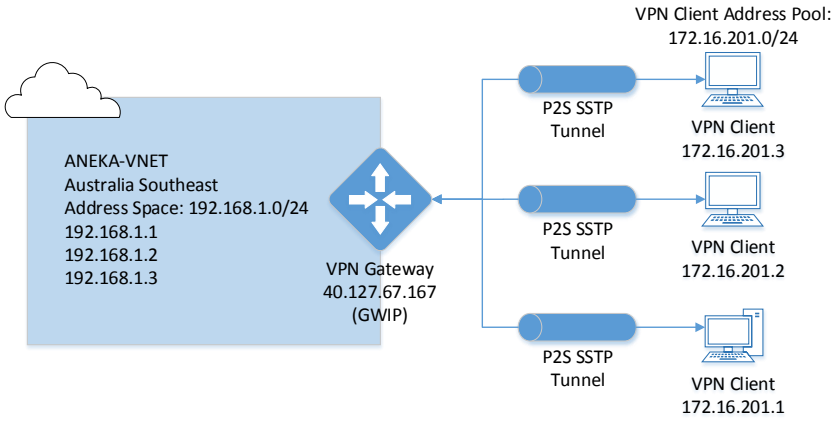


Fig. 8 Azure Point-to-Site VPN for Aneka.

7 Case Study: A hybrid Cloud Using Aneka

In this section, we describe a case study hybrid cloud built on top of Aneka platform and uses Azure virtual networking for connecting computational resources scattered over two separate networks. First, we make a brief review of some related works utilizing hybrid cloud environments for executing applications. Second, we describe the hybric cloud setup and the established VPN network. Then, we present the parameter sweep application used to be executed on the hybrid cloud environment. Finally, some experimental results are provided to demonstrate the functionality of the design and implementation.

7.1 Related works

The idea of using public cloud resources to expand the capacity of local infrastructure has been explored by many studies. Mateescu et al. [14] propose an architecture that provides a platform for the execution of High-Performance Computing (HPC) scientific applications. The cornerstone of the proposed architecture is called *Elastic Cluster* which makes an expandable hybrid cloud environment. Assunção et al. [9] analyze the trade-off between performance and usage cost of different provisioning algorithms for use of resources from the cloud to expand a cluster capacity. Javadi et al. [12] propose failure-aware resource provisioning policies for hybrid cloud environments. Xu and Zhao [18] propose a privacy-aware hybrid cloud framework which supports a tagging mechanism for the lacation of sensitive data. Belgacem and Chopard [2] conduct an experimental study of running a large, tightly coupled,

Table 1 Configuration of machines used in the experiments.

Machine	Type	CPU	Cores	Memory	OS
Master	Intel Core i7-4790	3.60CHz	8	16GB	Windows 7
Worker	Intel Core 2 Duo	3.00CHz	2	4GB	Windows 7
Azure Instances	Standard DS1	2.4 GHz	1	3.5	Windows Server 2012

distributed application over a hybrid cloud consisting of resources from Amazon EC2 clusters and an existing HPC infrastructure. Mattess et al. [15] presents a provisioning algorithm for extending cluster capacity with Amazon EC2 *Spot Instances*. Yuan et al. [19] propose a profit maximization model for a private cloud provider by utilizing the temporal variation of prices in hybrid cloud. Scheduling and resource provisioning in hybrid cloud has been researched for many other types of application as well, for example, big data analytics [8], workflows applications [16], online commerce [13], mobile phone applications [11] and compute intensive applications [3].

7.2 Hybrid Cloud Setup

The hybrid cloud environment we experiment with constitutes two desktop machines (one master and one worker) locating in our CLOUDS laboratory at the University of Melbourne and a number of virtual machines provisioned from Microsoft Azure. Table 1 shows the configurations of machines used to setup the hybrid cloud. A schematic view of the hybrid cloud platform used for running the application is also depicted in Fig. 9.

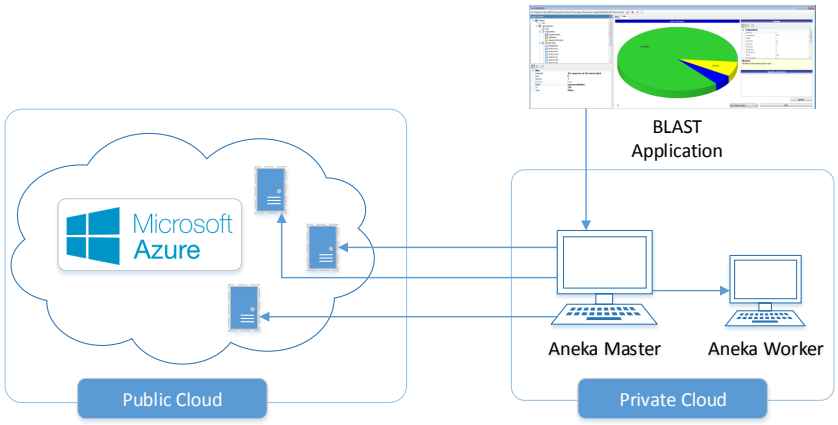


Fig. 9 Hybrid cloud testbed.

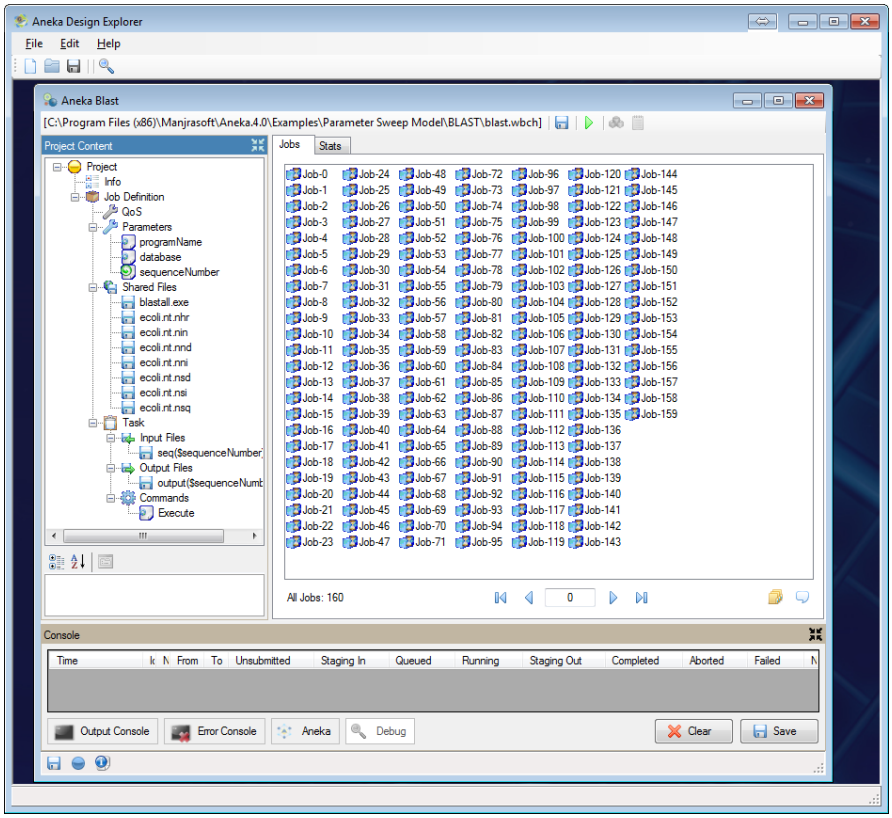


Fig. 10 Aneka Design Explorer and BLAST Project.

7.3 BLAST Application

To demonstrate the functionality of the virtual network established for supporting connectivity among Aneka hybrid cloud resources, we run a parameter sweep application using Aneka programming APIs. The application, we execute is called BLAST (Basic Local Alignment Search Tool) and is a tool for looking similarities among a given sequence of genes and those stored into classified databases. The BLAST application can be downloaded from the National Centre for Biotechnology Information (NBCI) website.[4] The website also provides a classified repository of all the databases that can be used for similarity searches. We used the database called "ecoli.nt" for our demonstration purpose.

There are many ways of parallelizing a BLAST query against a database. Here, we use the Parameter Sweep Model in order to automatically perform several BLAST queries against the same database. A parameter sweep application is a dis-

[4] `https://blast.ncbi.nlm.nih.gov/Blast.cgi`

tributed application that can be defined by a template task characterized by a set of configurable parameters. The template task identifies the set of operations that define the computation. The configurable parameters represent the way in which the template task is specialized.

Aneka provides an integrated tool, called *Design Explorer*, for quickly composing Parameter Sweep applications, controlling and monitoring their execution on Aneka Clouds. Parameter sweep applications in Aneka are expressed by the Parameter Sweep Model (PSM). The Aneka PSM provides the logic for creating the sequence of task instances from a template task and a given set of parameters. The Design Explorer provides a way to serialize its PSM data into an XML.

In our case study, we compose two BLAST-based parameter sweep applications using the Design Explorer tool to run 160 and 320 different queries each has a size of 1kb over the ecoli.nt database of size 4652kb. Each query is a sequence of characters representing the target genes and it maps to an Aneka independent task. Fig. 10 displays a sample screenshot of the Aneka Design Explorer including a BLAST project with 320 tasks and the below XML shows its related PSM:

```
<?xml version="1.0" encoding="Windows-1252"?>
<psm xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <name>Aneka Blast</name>
   <description>BLAST simulation</description>
   <workspace>C:\Projects\Explorer\blast</workspace>
   <parameters>
     <single name="programName" type="String"
      comment="The name of the program" value="blastn" />
     <single name="database" type="String"
      comment="The database file" value="ecoli.nt" />
     <range name="sequenceNumber" type="String"
      comment="The sequence of the input/output"
      from="0" to="319" interval="1" />
   </parameters>
   <sharedFiles>
     <file path="blastall.exe" vpath="blastall.exe" />
     <file path="ecoli.nt.nhr" vpath="ecoli.nt.nhr" />
     <file path="ecoli.nt.nin" vpath="ecoli.nt.nin" />
     <file path="ecoli.nt.nnd" vpath="ecoli.nt.nnd" />
     <file path="ecoli.nt.nni" vpath="ecoli.nt.nni" />
     <file path="ecoli.nt.nsd" vpath="ecoli.nt.nsd" />
     <file path="ecoli.nt.nsi" vpath="ecoli.nt.nsi" />
     <file path="ecoli.nt.nsq" vpath="ecoli.nt.nsq" />
   </sharedFiles>
   <task>
     <inputs>
       <file path="seq($sequenceNumber).txt"
        vpath="seq($sequenceNumber).txt" />
```

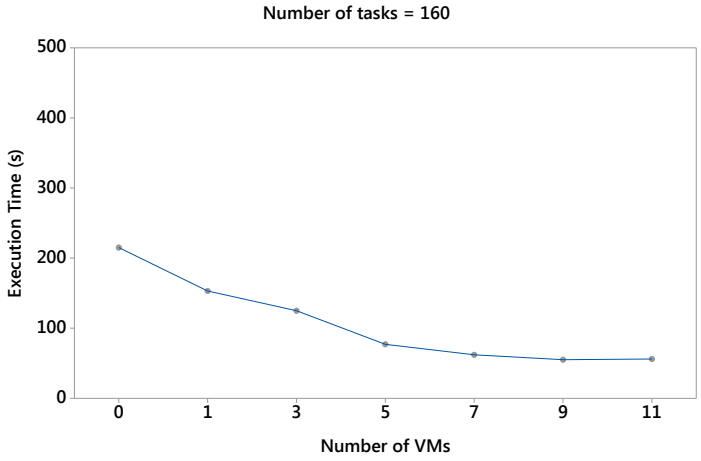
```
      </inputs>
      <outputs>
        <file path="output($sequenceNumber).txt"
         vpath="output($sequenceNumber).txt" />
      </outputs>
        <commands>
          <execute cmd="blastall.exe" args="-p ($programName)
           -d ($database) -i seq($sequenceNumber).txt
           -o output($sequenceNumber).txt" />
        </commands>
    </task>
</psm>
```

7.4 Experimental Results

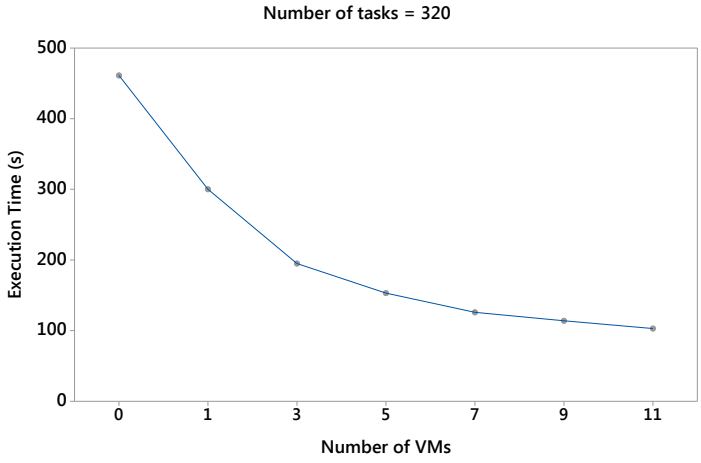
We expanded our hybrid cloud case study by increasing the number of virtual machines provisioned in Azure. In this scenario, the size of private cloud is constant during the experiments where ir remains at two desktop machines and public cloud resources are provisioned as required. Table 2 and Fig. 11 show how the total execution time is reduced when more public cloud resources were added to the Aneka hybrid cloud. When there are no public cloud resources available, the execution of BLAST for 160 and 320 tasks only using private cloud resources takes 215 and 461 seconds, respectively. These values, respectively, are reduced to 55 and 103 seconds when 11 VMs are provisioned from the public cloud. It is worth mentioning that the relative gain in performance decreases with adding more computational resources into the pool of resources, which is consistent with *Amdahl's law*. This can be clearly seen when the number of tasks is 160 and the total number of computational cores is increased from 11 to 13.

Table 2 Configuration of machines used in the experiments.

Tasks (#)	VMs (#)	Private Cloud CPU Cores (#)	Total CPU Cores (#)	Execution Time (mm:ss)
160	0	2	2	03:35
160	1	2	3	02:33
160	3	2	5	02:05
160	5	2	7	01:17
160	7	2	9	01:02
160	9	2	11	00:56
160	11	2	13	00:55
320	0	2	2	07:41
320	1	2	3	05:00
320	3	2	5	03:15
320	5	2	7	02:33
320	7	2	9	02:06
320	9	2	11	01:54
320	11	2	13	01:43



(a)



(b)

Fig. 11 Total execution time of BLAST application with (a) 160 and (b) 320 tasks on different hybrid cloud size.

Fig. 12 shows the distribution of tasks among Aneka nodes scattered on private on-premises infrastructure and public cloud resources. Please note that the Aneka master node is also located in the private infrastructure and communicate with all worker nodes (See Fig. 9). Our experiments demonstrate that the established azure VPN provides an appropriate platform for running application over Aneka nodes scattered in multiple separate networks (in different administrative domains) possibly located behind a NAT or firewalls boundary. Moreover, since the VPN encrypts

the traffic between its connected nodes, all communications are protected against unauthorized disclosure and confidentiality and privacy of information are enforced.

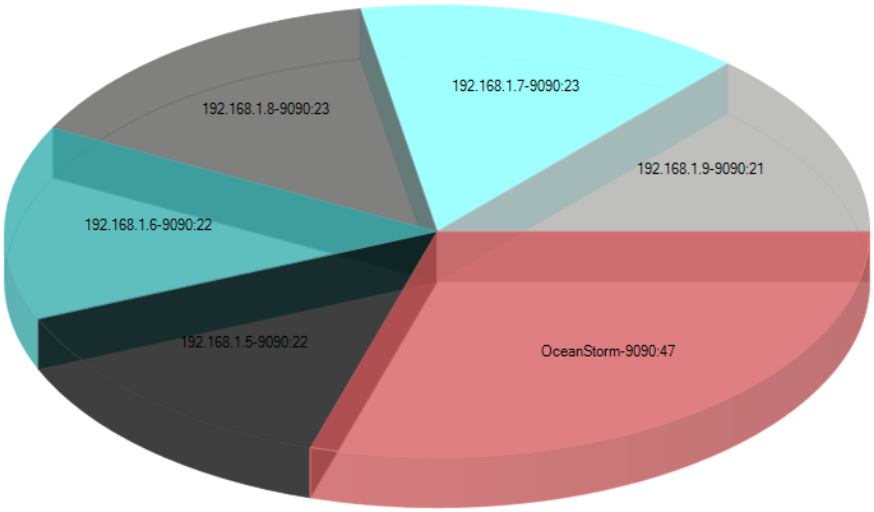


Fig. 12 Distribution of task among Aneka nodes when there are 160 tasks and 7 CPU cores available. Labels in the pie chart shows: address of machine - Aneka worker container port : the number of tasks allocated to that machine. OceanStrom's VNet IP address is 172.16.201.5.

8 Open Research Problems

The fundamental idea of uncoupling resource provisioning from vendor-specific dependencies is of a great interest for many application platforms such as Aneka. In this direction, Aneka can be extended to use features of multi-cloud toolkits and libraries such as JClouds.[5] However, to setup the VPN network for the Aneka platform, a great deal of manual work is needed as discussed in this chapter. A promising avenue to advance technologies for building such VPNs is to allow for the network programmability and the network function virtualization. This leads us to more research on software-defined networking (SDN)-assisted techniques in the development of efficient cloud technologies.

In line with our contribution, development of resource provisioning techniques that can acquire resources from multiple clouds can also be explored. In addition, dynamic resource provisioning policies are required for judiciously adding resources to the Aneka cloud based on application requirements such as deadline or budget. Future directions also include new algorithms for innovative provisioning

[5] JClouds, Available: `https://jclouds.apache.org/`.

algorithms honoring user requirements such as privacy and the location of sensitive data. This can be further extended to support resource allocation techniques in the context of various other programming models such as Map-Reduce model.

9 Summary and Conclusions

In this chapter, we outlined our experiences of building a virtual private network (VPN) for a hybrid cloud environment in our Aneka platform using Azure VPN services. We described the benefits of a hybrid cloud solution and its connectivity issue for resources residing behind a NAT or firewalls. We discussed how virtual private networks (VPNs) resolve this issue. We presented our Aneka Platform-as-a-Service (PaaS) for building a hybrid cloud environment. We explored through Azure VPN solutions and we presented how Azure point-to-site VPN connection can be used for building an Aneka hybrid cloud. To demonstrate the effectiveness of the VPN for hybrid cloud solutions, we provided details of a case study hybrid cloud for running a parameter sweep application from the biology domain. Our conclusion is that the establishment of VPN overlays over hybrid environments provides a feasible solution for connectivity issue which is more accessible as many more cloud providers offer VPN services. We also identified a number of open issues that form the basis for future research directions.

Acknowledgements We thank Australian Research Council (ARC) Future Fellowship and the Australia-India Strategic Research Fund (AISRF) for their support of our research. We also thank Microsoft for providing access to the Azure IaaS infrastructure.

References

[1] Marcos D. Assuno, Rodrigo N. Calheiros, Silvia Bianchi, Marco A.S. Netto, and Rajkumar Buyya. Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 7980:3 – 15, 2015. Special Issue on Scalable Systems for Big Data Management and Analytics.

[2] Mohamed Ben Belgacem and Bastien Chopard. A hybrid hpc/cloud distributed infrastructure: Coupling EC2 cloud resources with HPC clusters to run large tightly coupled multiscale applications. *Future Generation Computer Systems*, 42:11 – 21, 2015.

[3] M. Brock and A. Goscinski. Execution of compute intensive applications on hybrid clouds (case study with mpiblast). In *Proceedings of the Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 995–1000, July 2012.

[4] Roberto Brunetti. *Windows Azure Step by Step*. Microsoft Press, 2011.

[5] R. Buyya and D. Barreto. Multi-cloud resource provisioning with aneka: A unified and integrated utilisation of microsoft azure and amazon ec2 instances. In *2015 International Conference on Computing and Network Communications (CoCoNet)*, pages 216–229, Dec 2015.

[6] Rajkumar Buyya and Amir Vahid Dastjerdi (eds.), editors. *Internet of Things: Principles and Paradigms*. Morgan Kaufmann, Burlington, Massachusetts, USA, May 2016.

[7] Rodrigo N. Calheiros, Christian Vecchiola, Dileban Karunamoorthy, and Rajkumar Buyya. The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. *Future Generation Computer Systems*, 28(6):861 – 870, 2012.

[8] F. J. Clemente-Castell, B. Nicolae, K. Katrinis, M. M. Rafique, R. Mayo, J. C. Fernndez, and D. Loreti. Enabling big data analytics in the hybrid cloud using iterative mapreduce. In *Proceedings of the 8th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, pages 290–299, Dec 2015.

[9] Marcos Dias de Assunção, Alexandre di Costanzo, and Rajkumar Buyya. A cost-benefit analysis of using cloud computing to extend thecapacity of clusters. *Cluster Computing*, 13(3):335–347, 2010.

[10] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communication of the ACM*, 51(1):107–113, January 2008.

[11] Huber Flores, Satish Narayana Srirama, and Carlos Paniagua. A generic middleware framework for handling process intensive hybrid cloud services from mobiles. In *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '11, pages 87–94, New York, NY, USA, 2011. ACM.

[12] Bahman Javadi, Jemal Abawajy, and Rajkumar Buyya. Failure-aware resource provisioning for hybrid cloud infrastructure. *Journal of Parallel and Distributed Computing*, 72(10):1318 – 1331, 2012.

[13] Georg Lackermair. Hybrid cloud architectures for the online commerce. *Procedia Computer Science, World Conference on Information Technology*, 3:550 – 555, 2011.

[14] Gabriel Mateescu, Wolfgang Gentzsch, and Calvin J. Ribbens. Hybrid computingwhere HPC meets grid and cloud computing. *Future Generation Computer Systems*, 27(5):440 – 453, 2011.

[15] M. Mattess, C. Vecchiola, and R. Buyya. Managing peak loads by leasing cloud infrastructure services from a spot market. In *Proceedings of the 12th IEEE International Conference on High Performance Computing and Communications (HPCC)*, pages 180–188, Sept 2010.

[16] Mihaela-Andreea Vasile, Florin Pop, Radu-Ioan Tutueanu, Valentin Cristea, and Joanna Koodziej. Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. *Future Generation Computer Systems*, 51:61–71, 2015.

[17] Christian Vecchiola, Rodrigo N. Calheiros, Dileban Karunamoorthy, and Rajkumar Buyya. Deadline-driven provisioning of resources for scientific appli-

cations in hybrid clouds with aneka. *Future Generation Computer Systems*, 28(1):58 – 65, 2012.

[18] X. Xu and X. Zhao. A framework for privacy-aware computing on hybrid clouds with mixed-sensitivity data. In *Proceedings of the IEEE International Symposium on Big Data Security on Cloud*, pages 1344–1349, Aug 2015.

[19] H. Yuan, J. Bi, W. Tan, and B. H. Li. Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds. *IEEE Transactions on Automation Science and Engineering*, 14(1):337–348, Jan 2017.