# Dynamic Virtual Cluster reconfiguration for efficient IaaS provisioning

Vittorio Manetti, Pasquale Di Gennaro, Roberto Bifulco,
Roberto Canonico, and Giorgio Ventre

University of Napoli Federico II, Italy
Dipartimento di Informatica e Sistemistica
Via Claudio, 21 - 80125 - Napoli, Italy
{name.surname}@unina.it

**Abstract.** Cloud computing is an emerging paradigm to provide *Infrastructure as a Service* (IaaS). In this paper we present NEPTUNE-IaaS, a software system able to support the whole lifecycle of IaaS provisioning in a Virtual Cluster environment. Our system allows interactive design of complex system topologies and their efficient mapping onto the available physical resources of a cluster. It also provides transparent VM migration features across geographically distributed datacenters, thanks to the adoption of the Service Switching paradigm. We also evaluate the effectiveness of the VM mapping procedures and compare our solution against other existing IaaS solutions.

**Key words:** Cloud Computing, IaaS, Xen, Virtual Networking.

## 1 Introduction

Cloud Computing is an innovative computing model in which "dynamically scalable and often virtualised resources are provided as a service over the Internet" [1]. Following what happened in the last century with electric power or water distribution infrastructures, Cloud Computing enables users to access computing resources on an as-needed basis, relieving them from the responsibility of buying and managing a dedicated computing infrastructure. Cloud providers, on the other hand, can take advantage of scale economies to organize and manage big datacenters, whose ICT resources can be efficiently used by partitioning and renting them to a number of customers. Depending on the abstraction level of the provided resources, Cloud Computing takes different names: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

Originally born as a cluster based network emulation system [2], NEPTUNE-IaaS is a software system developed at University of Napoli Federico II that allows interactive design of networked virtual infrastructures on geographically distributed datacenters, to help provisioning of "Infrastructures as a Service". Our system consists of an interactive client/server software system used to provide users with the possibility of describing and designing the desired virtual

infrastructure and of a set of other components that make it possible for services deployed at a given datacenter to be transparently migrated in remote datacenters for load balancing or fault/disaster recovery. NEPTUNE-IaaS is based on the use of Xen for virtualization of computing elements. Xen features are also used to multiplex the communication resources (e.g. network interfaces) available in the cluster nodes among several logically distinct virtualized nodes. Transparent migration of Virtual Machines in NEPTUNE-IaaS is implemented through the adoption of Service Switching, a novel paradigm that aims at extending the concept of virtualization to network services, by decoupling service execution environments and their physical location.

The rest of the paper is organized as follows. In Section II we present NEPTUNE-IaaS, its architecture, the web-based management application we have developed to manage the whole lifecycle of virtual infrastructures. In Section III we present *Service Switching* and its role in our system. In Section IV we present the algorithm we use to efficiently map Virtual Machines onto a cluster's physical resources. Finally, in Section V we briefly compare NEPTUNE-IaaS against two reference IaaS solutions and draw our conclusions.

## 2 NEPTUNE-IaaS

NEPTUNE-IaaS is a software system for provisioning of IaaS services. In the context of NEPTUNE-IaaS, a *Virtual Infrastructure* is a collection of Virtual Machines provided as a service to an end-user. Virtual Machines are deployed on a subset of a cluster's physical nodes and properly configured according to the user requirements in terms of computational resources, software configuration, virtual network topology, and so on. A Virtual Infrastructure presents at least one public IP address, that is used to make the infrastructure accessible from the public Internet (Entry Point). In general, public IP addresses are assigned only to a subset of the nodes of a Virtual Infrastructure. Other nodes are assigned private IP addresses and can be reached only through the Entry Point nodes. A typical Virtual Infrastructure comprises a NAT/firewall node and a set of backend service nodes, whose NICs are assigned private IP addresses. We will describe later in this paper that the necessity of supporting transparent migration of Virtual Infrastructures across geographically distributed datacenters calls for unique assignment of private IP addresses within a Service Switching domain.

To achieve higher degrees of scalability and resource efficiency, Virtual Infrastructures are instantiated by allocating multiple Virtual Machines onto each of the cluster's real nodes (*node multiplexing*). Likewise, multiple virtual links are multiplexed onto the same shared physical link by associating each virtual link endpoint to a different virtual NIC (*link multiplexing*). Multiple fully isolated Virtual Infrastructures can be concurrently hosted by NEPTUNE-IaaS in the same datacenter, providing users with the illusion of having allocated a dedicated infrastructure.

### 2.1 NEPTUNE-IaaS Architecture

A cluster managed by NEPTUNE-IaaS (Figure 1) is composed of three components: i) a set of worker nodes providing computational resources used to reproduce emulated networks, ii) a centralized repository providing storage space to worker nodes and iii) a front-end node, *Neptune Manager*. By NEPTUNE-IaaS we intend the whole collection of system software, of which the management software running in the Neptune Manager front-end is the most relevant part. All the physical components of the cluster are connected by two switched LANs, one for "control traffic" (e.g. node configuration) and another for "operational traffic" (i.e. traffic generated by users' applications).

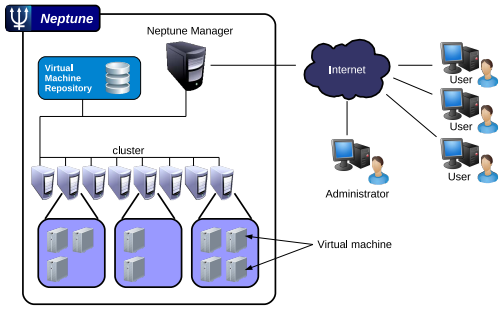


**Fig. 1.** NEPTUNE architecture.

### 2.2 Virtual Infrastructure life-cycle

A Virtual Infrastructure life-cycle can be described by a Finite State Machine (Figure 2). A Virtual Infrastructure life-cycle begins with the definition of a virtual network topology. Once the topology is defined, the infrastructure can be allocated onto the cluster's physical nodes. On user demand, a running Virtual Infrastructure can be either suspended for future reallocation or definitively terminated. Allocation of infrastructures onto the cluster is made under control of system administrators, who need to explicitly accept users requests. Once accepted, an infrastructure's topology allocation process starts. Such allocation process is automatic, involving tasks like virtual nodes mapping on cluster's physical nodes and IP addresses assignments.

To define a Virtual Infrastructure, users can either write a topology description in a custom XML format, defining nodes'properties (NICs, RAM, software configuration, etc.) and links'properties (bandwidth, end points, etc.) or use an interactive graphic tool embedded into the web user interface (Figure 3). The tool assists the description of any node or link property suggesting available choices to the user. It is also possible for users to select pre-defined topologies for fast infrastructure definition. To define virtual nodes software configuration,

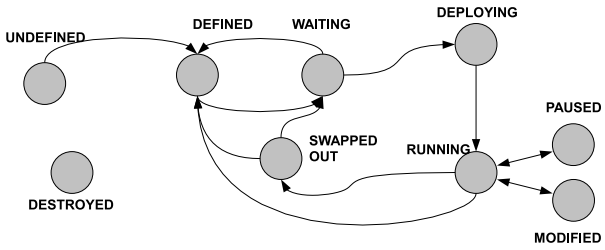


**Fig. 2.** Virtual Infrastructure lifecycle.

users can access a "Virtual Nodes Template Images Repository" and select a VM template for each of the virtual nodes. VM templates can be modified and saved as new templates for reuse.

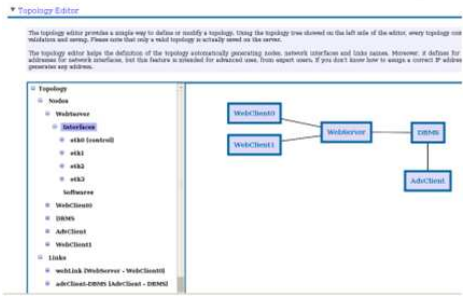

**Fig. 3.** Interactive editor.

### 2.3 Implementation details

Node multiplexing is implemented in NEPTUNE-IaaS by means of Xen [3]. Our current implementation relies on the libvirt virtualization API [4], making it feasible supporting different virtualization technologies in the future. The NEPTUNE-IaaS Management Node is responsible of managing Virtual Machines lifecycle.

Mapping of virtual nodes onto the cluster physical nodes is described by an allocation map which can be generated either manually by a system administrator or automatically, by means of a software module implementing a Lin-Kernighan derived optimization algorithm (described in Section 4).

When a virtual network is to be deployed on the physical cluster, Neptune Manager distributes Virtual Machine template instances to the physical cluster nodes. This distribution process is composed of two phases for each virtual node:

1) raw copy of the virtual machine image file containing VM template, and 2) VM creation on the target virtual machine monitor. During this last phase, virtual hardware resources are provided to the virtual node according to node definition provided by the Virtual Infrastructure topology description.

A major problem when dealing with the creation of virtual links is the need to assign IP addresses to both ends of virtual links, according to a general IP addressing scheme. NEPTUNE-IaaS provides an algorithm that automatically assigns subnets to links and IP addresses to their end-points. Furthermore, since several infrastructures can be running on the same shared infrastructure, this algorithm also ensures non overlapping of address spaces used by different infrastructures.

## 3 The Service Switching Paradigm

Service mobility is a key feature for new generation networks. In distributed service hosting environments, service mobility allows satisfaction of requirements like: efficient management of available resources, computational load balancing, service continuity even in presence of critical conditions. Service Switching aims at extending the concept of virtualization to network services by decoupling service execution environments and their physical location [5]. Service instances in a Service Switching environment may be dynamically migrated across geographically dispersed datacenters, to achieve more efficient utilization of both network and computing resources. The Service Switching paradigm allows creation and management of *Service Execution Environments* across different datacenters with minimal impact on service continuity.

The architectural implementation of the Service Switching paradigm is centered around a main component, that we call *Service Switch*. Such a component is a network node that, in addition to the plain packet and/or flow switching capabilities, has more advanced features, including the ability to forward packets towards migrated Service Execution Environments. Service Switches can be located both at the edges of a network and in its core. Deployment of Service Switches in the core of the network of course requires cooperation of Internet Service Providers, but allows faster reconfiguration and migration of services.

Our current implementation of the Service Switching model relies on a combination of system-level virtualization technologies and of the Mobile IP model. In the following we firstly introduce a brief description of Mobile IP, and then the Service Switching architecture customized for the NEPTUNE-IaaS context.

IP version 4 assumes that the IP address of a node uniquely identifies its point of attachment to the Internet: a node must be located on the network indicated by its IP address in order to receive datagrams which are destined to it. IP Mobility Support (or Mobile IP) provides a mechanism which allows Mobile Nodes to change their point of attachment to the Internet without changing their IP address [6]. This mechanism relies on two intermediary entities: the *Home Agent* and the *Foreign Agent*. The role of the Home Agent is to maintain current location information of the mobile node, and to re-transmit all the

packets addressed to the Mobile Node through a tunnel to the Foreign Agent to which the Mobile Node is currently registered. The role of the Foreign Agent, in turn, is to deliver datagrams to the Mobile Node.

Service Switching allows services to be deployed at different geographic locations, each of which hosts a cluster of physical machines. A physical cluster is connected to the Internet through a special router, that we call *Edge Service Switch*. In the context of NEPTUNE-IaaS we are interested in transparently migrate a collection of related Virtual Machines (a Virtual Infrastructure, according to the definition we gave in Section 2). When a Virtual Infrastructure is deployed for the first time, it is associated to one of the available datacenters. This allocation choice assigns one or more public IP addresses to the Entry Points of the Virtual Infrastructures. These IP addresses will be kept for the entire lifecycle of the Virtual Infrastructure, even in case of migration. Such IP addresses are referred to as the Virtual Infrastructure's *Home Addresses*. The Edge Service Switch located at the edge of the datacenter in which the Virtual Infrastructure is initially deployed, will be referred to as the Virtual Infrastructure's *Home Service Switch*. An Edge Service Switch not only behaves as a normal IP edge router, forwarding incoming packets to the VMs hosted in the cluster and outgoing packets to a next hop router according to its current routing table, but it also implements specific traffic flow readdressing mechanisms to support service migration. Such mechanisms have been derived as extensions of the classical Mobile IP model. A generic end user terminal accessing a service will be referred to as *Correspondent Node*.

Making the simplistic assumption that a Virtual Infrastructure presents a unique Entry Point, in order to access a given service, a Correspondent Node sends packets to this latter, using the VI's Home Address as IP Destination Address. Incoming packets will be processed by the VM's Home Service Switch. In case a Virtual Infrastructure had to be migrated to a different datacenter, the Virtual Infrastructure's Home Service Switch creates an entry in its *Mobility Binding Table* (MBT in short) that contains information about the Entry Point of the migrated Virtual Infrastructure. The MBT keeps the association between the VI's Home Address and the corresponding *Care-of Address*. Such Care-of Address is the IP address of the Edge Service Switch associated to the datacenter hosting the migrated Virtual Infrastructure, that we may call the Virtual Infrastructure's *Foreign Service Switch*. Migration of a Virtual Infrastructure is performed through a procedure that consists in updating the Home Network's MBT and in managing the migration of all the VMs belonging to the Virtual Infrastructure. Concerning the dataceneter that hosts the migrated Virtual Infrastructure, apart from the configuration of the Foreign Service Switch, no other settings are needed. Migrated VMs keep using their own VI's Home Address as IP source address for outgoing packets, and Correspondent Nodes, being unaware of the migration, keep sending packets to the Virtual Infrastructure's Home Address. Once these packets reach the Home Service Switch, this latter forwards them to the Foreign Service Switch, by encapsulating such packets in a point-to-point tunnel (figure 4). The Foreign Service Switch, in turn, de-tunnels

the incoming packets and delivers them to the migrated VM. As it happens in the Mobile IP scheme, reverse traffic is sent by the migrated VM directly to the Correspondent Nodes.

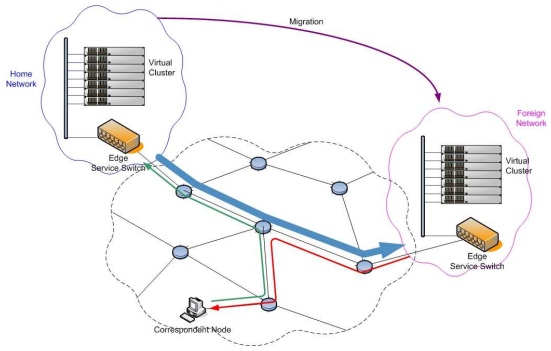


**Fig. 4.** Tunneling mechanism implemented on the edge.

## 4 Optimal VM allocation in a datacenter

One of the key steps in the Virtual Infrastructure deployment process is the mapping of Virtual Machines onto the physical resources of the target datacenter. This problem is known in literature as the *network testbed mapping problem*[7]. Due to its complexity, the challenge is to find a *good* solution in *acceptable* computational times. Our approach to manage complexity consists in splitting the mapping problem in two sub-problems: *topology partitioning* and a *partition mapping*.

Several graph partitioning algorithms have been proposed in the literature. An algorithm that provides good results with reasonable times of calculation is the Lin-Kernighan (LK) heuristic algorithm [8]. Theoretical complexity of LK is $O(n^2 \log n)$. We implemented this algorithm in JAVA to evaluate its applicability to cluster environments and to assess its performance. A first test was performed to estimate the solver execution time while varying number of nodes in the graph. Size of the matrix was varied between 100x100 and 1000x1000 with steps of 100. The graph was been partitioned into subsets of cardinality equal to 5 while non-zero elements incidence for considered matrix were 2%. Computational times represented in Figure 5 were calculated by using a system equipped with 2 GB of RAM and an Intel CPU T2250 running at 1.73 GHz.

Our algorithm implementation requires that once found a minimum cost solution, the procedure is restarted with a new initial solution. After running 5 iterations the algorithm stops and returns the minimum cost solution. This test highlights the relationship between the iteration $i^*$ at which the optimal solution is found with the size and density (arcs/nodes ratio) of the matrix.

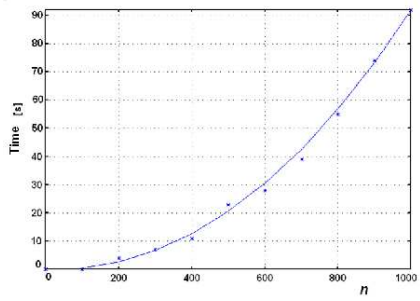


**Fig. 5.** LK best mapping solution times

| | arc/nodes=4 | arc/nodes=6 | arc/nodes=8 |
|---|---|---|---|
| Matrix 20x20 | 100run | 100run | 100run |
| Matrix 100x100 | 100run | 100run | 100run |
| Matrix 400x400 | 100run | 100run | 100run |

**Table 1.** Tests organization

Virtual links and physical links bandwidths have been respectively fixed at 10 and 100. Matrices are generated randomly and before subjecting a matrix to the solver, it is verified that each node has at least one connection and that the sum of the costs associated to all the outgoing arcs from one same node does not exceed 90% of the physical connections bandwidth. Tests organization is shown in Table 1, while tests results are shown in Table 2.

| | i*=1 | i*=2 | i*=3 | i*=4 | i*=5 |
|---|---|---|---|---|---|
| Matrix 20x20 arc/nodes=4 | 47 | 21 | 18 | 7 | 7 |
| Matrix 20x20 arc/nodes=6 | 52 | 21 | 13 | 8 | 6 |
| Matrix 20x20 arc/nodes=8 | 43 | 27 | 14 | 6 | 10 |
| Matrix 100x100 arc/nodes=4 | 23 | 21 | 21 | 23 | 12 |
| Matrix 100x100 arc/nodes=6 | 23 | 21 | 28 | 20 | 8 |
| Matrix 100x100 arc/nodes=8 | 18 | 16 | 26 | 18 | 22 |
| Matrix 400x400 arc/nodes=4 | 18 | 24 | 16 | 18 | 24 |
| Matrix 400x400 arc/nodes=6 | 20 | 24 | 18 | 16 | 22 |
| Matrix 400x400 arc/nodes=8 | 22 | 28 | 14 | 12 | 24 |

**Table 2.** Tests results

Results for the case 4 arcs/node and 6 arcs/node are further shown in Figure 6 and in Figure 7.

This test demonstrates that for matrices of small size (20x20), our solver returns in almost 50% of the cases the least-cost solution at the first iteration.

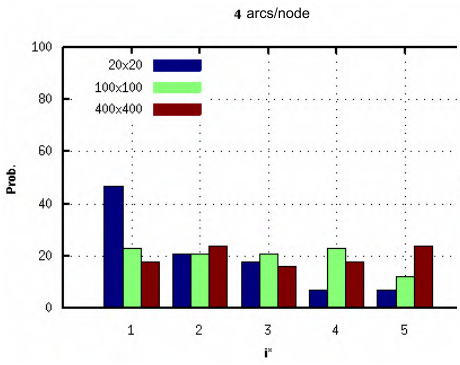


**Fig. 6.** Four-arcs/node case

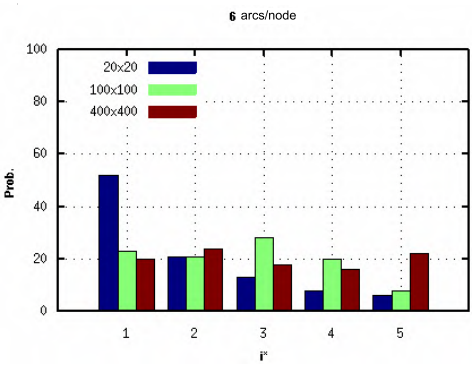


**Fig. 7.** Six-arcs/node case

When the matrix increases in size (100x100 and 400x400), the probability of finding good solutions at the first iteration is lower. In these cases, better results could be obtained by running more iterations, but the rapid increase of computational times does not encourage this approach. The Lin-Kernighan algorithm does not guarantee that it is always possible to find an admissible solution, so it could happen that the found solution does not meet the admissibility constraints. However, in our tests, the solver always returned an acceptable solution.

## 5 Related Work and Conlcusions

In the last few months the term "Cloud computing" is transforming from a buzzword into real world engineering solutions and commercial products. In this paper we mention two established solutions that have some features in common with NEPTUNE-IaaS: Amazon EC2 and Eucalyptus.

Amazon's *Elastic Compute Cloud* (EC2) [9] is an IaaS commercial system that first introduced the utility computing model, where computation, storage and bandwidth resources are rent on an as-needed basis. As well as NEPTUNE-IaaS, EC2 is based on Xen. Users select an Amazon Machine Image (AMI), including the machine's software configuration from a set of AMIs proposed by Amazon, or create a new one from scratch. To each AMI instance (i.e. a Xen Virtual Machine) is associated an "instance type" that defines the resources of the machine in terms of CPU, RAM, HD. Resources are paid on a consumption basis: a machine is paid for each hour of activity, bandwidth is paid per-gigabyte of traffic and so on. Amazon provides two ways to access EC2 services: via a web interface or through web services. A complete set of tools and programming libraries are provided to access these service.

Eucalyptus [10] is an open-source cloud-computing framework, built to be interface-compatible with Amazon EC2: users can interact with Eucalyptus using same tools and interfaces that they use with Amazon EC2. Because the main goal of Eucalyptus is to provide a common open-source framework that enables

researchers to do experiments and studies, even by replacing or modifying the implementation of system modules, the system is based on three components, each with a well defined Web-service interface. The software architecture is hierarchical: the base level is composed by Instance Managers (IM), responsible to manage virtual machines running on top of a physical machines, the middle layer contains Group Managers (GM), each of which manages a set of IMs residing on the same physical subnet. The top layer is the Cloud Manager (CM), that manages all the GM making high-level scheduling decisions and represents the entry-point to Eucalyptus for users as well as for administrators.

NEPTUNE-IaaS has some features in common with both EC2 and Eucalyptus, but also some important differences. In particular, we want to highlight that NEPTUNE-IaaS provides tools to interactively design virtual networked infrastructures and supports transparent and efficient migration of infrastructures across geographically dispersed datacenters. NEPTUNE-IaaS is an ongoing project, whose future development include more complex management procedure to handle migration of complex virtual infrastructures in a reliable way. Integration of NEPTUNE-IaaS with storage services, such as those provided by Amazon's S3 are also being investigated.

## 6 Acknowledgements

The research leading to these results has been partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°224263-OneLab2.

## References

1. Wikipedia: Cloud computing. `http://en.wikipedia.org/wiki/Cloud_computing`
2. Di Gennaro, P., Bifulco, R., Canonico, R., Ventre, G.: Neptune: Network emulation for protocol tuning and evaluation. Poster presented at (SIMUTOOLS09), Rome, March 2009.
3. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Procs. of the 19th ACM Symposium on Operating Systems Principles, SOSP'03. (2003) 164–177
4. RedHat, et others: LibVirt virtualization API. `http://www.libvirt.org`
5. Manetti, V., Canonico, R., Ventre, G., Stavrakakis, I.: System-level virtualization and mobile ip to support service mobility. In: Proceedings of the International Workshop on Design, Optimization and Management of Heterogeneous Networked Systems (DOM-HetNetS'09). (September 2009)
6. Perkins, C., et al.: IP mobility support (1996)
7. Ricci, R., Alfeld, C., Lepreau, J.: A solver for the network testbed mapping problem. SIGCOMM Comput. Commun. Rev. **33**(2) (2003) 65–81
8. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell system technical journal **49**(1) (1970) 291–307
9. Amazon: EC2 web site. `http://aws.amazon.com/ec2/`
10. Eucalyptus Systems: Eucalyptus web site. `http://www.eucalyptus.com/`