CMST SI(1) 35-44 (2010) DOI:10.12921/cmst.2010.SI.01.35-44

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **Special Issue 2010,** 35-44

Virtualization of Grid Networking Resources for Computation Mobility Support

Marcin Jarząb, Jacek Kosiński, Krzysztof Zieliński

Institute of Computer Science, University of Science and Technology,
al. Mickiewicza 30, 30-059 Kraków, Poland
{mj/jgk/kz}@agh.edu.pl
http://www.ics.agh.edu.pl

(Received: 5 July; revised: 29 October; published online: 23 November 2010)

Abstract: The paper analyses Grid networking resource virtualization technologies and requirements in the context of computational mobility. The mechanisms of network virtualization supported by Xen and xVM/Crossbow are presented and compared. A network virtualization model, consisting of three layers, is presented. Changes in virtual network infrastructures resulting from VM or zone mobility are investigated in detail. Typical migration process scenarios are presented, in addition to system commands used prior to and following migration in order to restore network communication. The concept of automating this process is also described.

Key words: virtualization, migration, network virtualization, tools

I. INTRODUCTION

Virtualization technology is playing an increasingly important role in the management of complex computer systems and resource provisioning environments such as Grids and clouds. Full exploitation of the potential of virtualization technologies requires virtualization of computational, storage and communication resources. On-demand virtual resource provisioning, required by adaptable systems supporting runtime computational mobility, introduces new, challenging requirements for the organization of communication processes. This issue can be resolved in the context of network virtualization provided by server and operating system virtualization frameworks such as Xen [5, 6] and xVM [9].

Practical usage of these technologies in a dynamic environment supporting runtime migration is not a trivial task. It requires (i) deployment of a virtual network infrastructure consisting of virtual network interfaces, bridges and routers; (ii) mapping them onto existing physical networking resources; (iii) proper connection to virtual machines (VMs) or OS containers. This sophisticated deployment process should be performed automatically, with minimum human involvement. The resulting virtual-to-physical networking resource mapping must be changed transparently whenever VM migration takes place. These changes should preserve not only the connectivity between virtualized computational resources but also the required QoS and security parameters.

The paper analyses Xen and xVM built-in network virtualization technologies and requirements in the context of computational mobility in Virtual Grids. The concepts of network virtualization supported by Xen and xVM/Crossbow are presented and compared. The process of deploying a virtual network infrastructure is described and its automation is proposed. The aspects of virtual network infrastructure changes resulting from VM mobility are investigated in detail. Suitable mechanisms implemented by the provisioning tools for VM and Zone virtual network management running within Xen or xVM virtualization technologies are proposed and described.

II. PROBLEM DEFINITION

Network virtualization needs to be considered on three layers, shown in Fig. 1. These layers are related to each other as virtualized elements of each layer are built upon the components provided by lower layers.

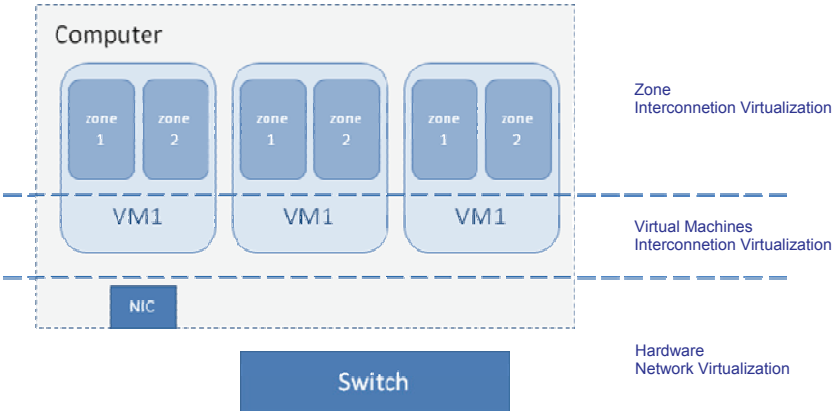

Fig. 1. Networking virtualization levels

The lowest layer – *Hardware Network Virtualization* – refers to networking hardware such as network interface cards (NIC) and switches. Virtual LAN (VLAN) technology is frequently used in this layer.

The intermediate layer – *VM Interconnection Virtualization* – involves computer paravirtualization. This technology allows coexistence of many VMs (instances of different operating systems) on a single physical machine. VMs deployed on the same computer or on different computers should be able to communicate – thus, networking virtualization should be provided for this purpose.

The topmost layer – *Zone Interconnection Virtualization* – concerns OS virtualization mechanisms supported by OpenSolaris [10] and Linux OpenVZ [11]. A single OS instance can host many containers called zones. Each zone can have its own networking resources allocated, providing isolation and QoS guarantees. Such virtualization introduces the need for similar networking virtualization technologies as in the VM Interconnection Layer but on a different level of abstraction.

Network virtualization in a dynamic environment supporting VM and zone migration remains a challenging issue. In this context, virtualization has two basic requirements:

- Provide isolation of traffic between a different group of VMs/zones – communications inside a group of VMs/zones should not be visible from the outside;
- Guarantee the requested level of communications QoS between VMs/zones inside the group – the requested flow throughput between VMs/zones should be delivered.

The complexity of migration is related to the issue that these requirements have to be preserved in spite of migration procedures. Therefore, migration should strike a balance between isolation and flow-related parameters. In virtualized networking infrastructures, migration can be divided into three stages:

1. Deployment of initial configuration using configuration networking commands for a selected virtualization technology. This stage should take into account the initial allocation of VMs and communication requirements.
2. Identification of network resources following VM migration, related to target computer networking interfaces and their connections via physical network devices.
3. Generation and execution of commands to be performed on the target machine or network devices to re-establish networking communications with the requested level of QoS.

The processing of stage 2 and 3 has to accompany the VM migration process in a dynamic environment. This procedure refers to the VM Connection Virtualization Layer and should be replicated in the Zone Connection Virtualization Layer. The following section discusses the procedure in more detail, in the context of each layer.

II.1. VM Connection Virtualization Layer

The complexity of steps 2 and 3 of the virtualized networking infrastructure lifecycle depends on the initial configuration built in step 1. Analysis of virtualization mechanisms supported for xVM and Xen points to the following two cases:

1. A migrating VM connected to Virtual NIC (VNIC), as presented in Fig. 2
2. A migrating VM connected to a virtual switch (EtherStub) as presented in Fig. 3 – such a switch supports only internal communications inside one computer.

In Figure 2, the simplest scenario is considered. VM1, VM2 and VM3 are initially located on the same physical

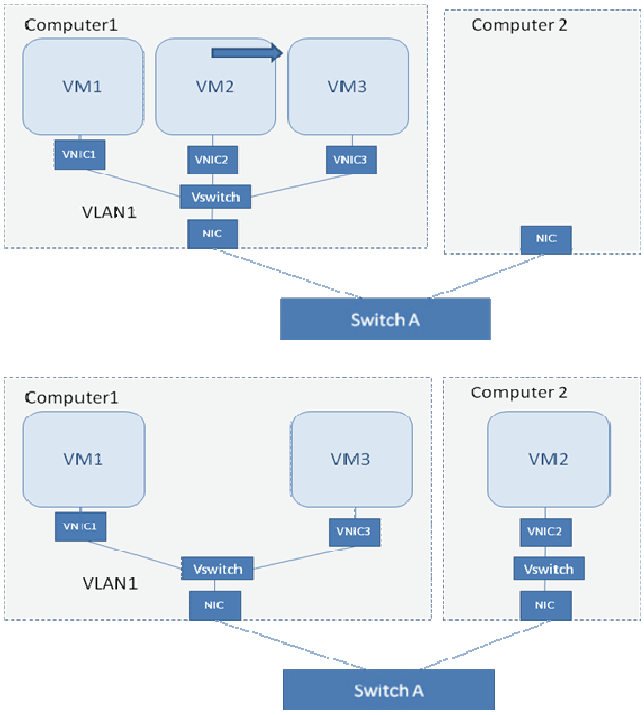

Fig. 2. VM migration process – Case 1

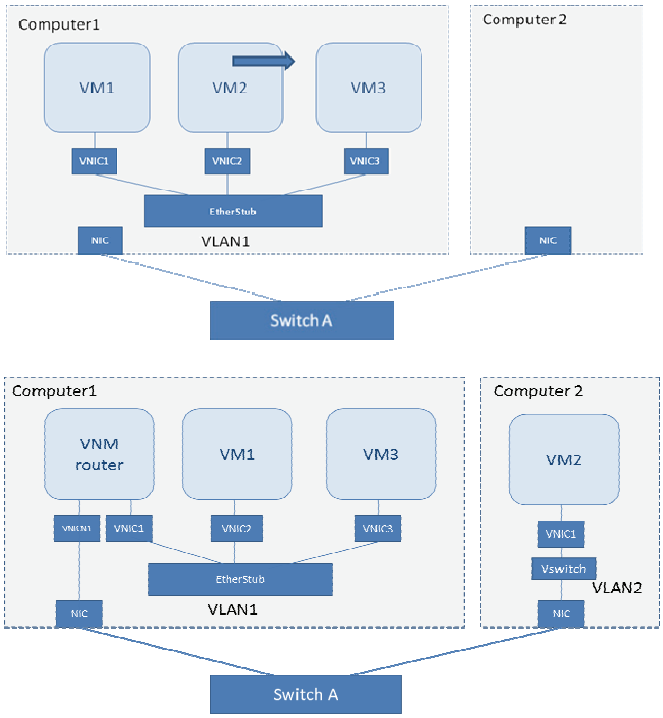

Fig. 3. VM migration process – Case 2

machine (Computer 1). Communication between these VMs is isolated from other traffic as VMs are connected to VLAN1 implemented on a single computer. A virtual switch deployed over a physical NIC is used for this purpose. It is also possible to maintain the required flow QoS between VMs. The deployment of such an initial configuration using Xen or xVM is not very challenging. A problem appears, however, when VM2 needs to migrate to another physical machine (Computer 2). Such migration should be performed at runtime and must preserve communication between VM1, VM3 and VM2 within VLAN1, as well as the flow QoS. This means that the configuration shown in Fig. 3 must be established seamlessly. On Computer 2, VNIC2 should be created over NIC and assigned to VM2.

For Case 2 (depicted in Fig. 3), the initial configuration consists of three VMs connected with EtherStub. It is a very efficient solution for connecting VMs on a single physical machine. Unfortunately, it creates difficulties during execution of steps 2 and 3. Following migration of VM3, communication between VMs can be re-established only if external communication is enabled for both VM1 and VM3. The only solution is to create a router in a dedicated VM (Network Virtual Machine), having access to NIC, and use it to connect to Computer 2. VM2 has to be connected to VNIC2 deployed over NIC on Computer 2, as depicted in Fig. 3. As a result, VM2 can reside in VLAN2 and connect to VLAN1 through the router. Both VLANs may be assigned the same ID if necessary. It is easy to notice that this migration procedure is far more involved than in Case 1.

II.2. Zone Connection Virtualization Layer

The scenarios described in the previous section can be implemented in the Zone Connection Virtualization Layer using OpenSolaris with the Crossbow library. This bases on the assumption that the VNIC allocated to OpenSolaris VM may be used in exactly the same way as a physical NIC, meaning that other VNICs and virtual switches can be defined inside the virtual machine and subsequently allocated to zones. This assumption will be validated in Section IV.

Under this assumption, communication between zones is organized in a similar way as between VMs. Two cases similar to the ones described in Section II.2 can be distinguished:

1. Migrating zone connected to Virtual NIC (VNIC) built over NIC, as depicted in Fig. 4;
2. Migrating zone connected to a virtual switch (EtherStub) as shown in Fig. 5 – this type of switch supports only internal communications inside one computer.

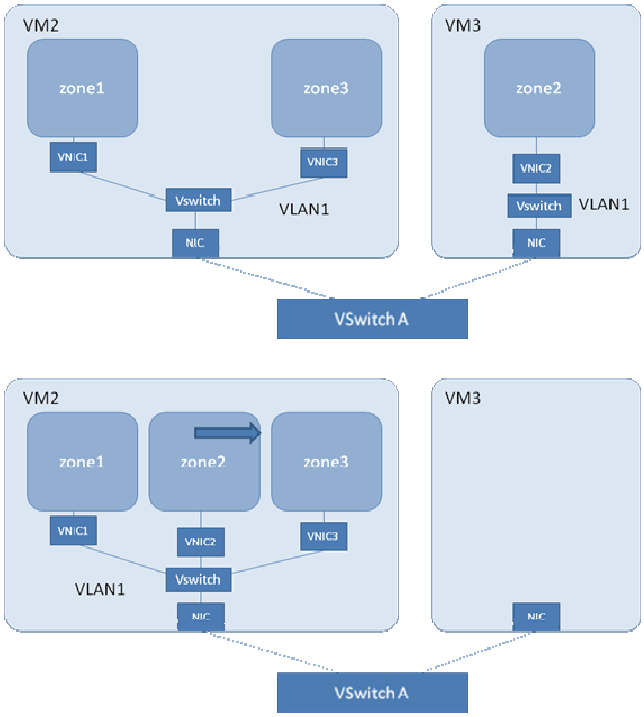

Fig. 4. Zone migration – Case 1

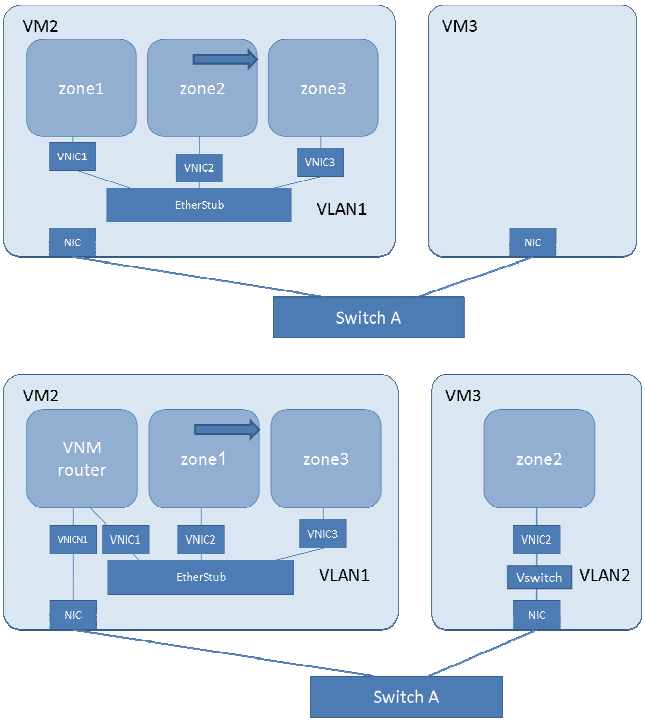

Fig. 5. Zone migration – Case 2

It is necessary to point out that runtime cross-VM zone migration is not yet supported. Live migration can only occur for specific VM instances. In such a situation, the zone must be stopped prior to migration, and attached to the guest OS hosted by the target VM machine. An important issue is that the internal communication configuration for VM zones must be preserved during VM migration, ensuring that the existence of zones within a VM is completely transparent from the point of view of the migration process. A problem appears if a zone needs to be migrated from one VM to another.

Figures 4 and 5 show that activities performed in steps 2 and 3 of the migration procedure are very much the same as the ones involved in VM migration. The case presented in Figure 5 requires a dedicated zone, created to enable the deployment of a VNM router.

III. NETWORK VIRTUALIZATION SUPPORT MECHANISMS

In the previous section we described the basic concepts of VM and OS containers in the context of network virtualization. The presented requirements are technology-neutral. In this section, network virtualization techniques for xVM and Xen implementation will be presented.

Enabling local network communications for VMs can proceed in many ways. Choosing the right solution depends on its availability in the applied virtualization technology. Commonly used technologies presented in this paper include Xen and xVM. They differ with respect to network communications.

In Xen network communications are based on pinned virtual interfaces [1-4] – every virtual interface has its equivalent on the list of interfaces of the physical host and is directly connected via a point-to-point virtual link. Communication between virtual machines is possible through additional mechanisms, e.g. routing techniques, switching/bridging or network address translation. Such an approach provides extensive configuration capabilities, since by using routing, switching and NATs it is possible to build any network topology using virtual computers. In addition, it is also possible (in the case of point-to-point links) to set filtration (firewall), queuing or traffic shaping (QoS).

These built-in technologies are not able to realize LAN-like communication of VMs located on different nodes in distinct LAN networks. In the case of WAN communications (host computers located in different LAN segments) and live migration of virtual machines, it is necessary to provide mechanisms which dynamically alter/update the layout of the virtual topology, changing its physical equivalent.

The development of a network topology layout and managing its configuration (e.g. by specifying bandwidth restrictions) in the Linux OS is facilitated by a set of system commands whose effect changes in the structure of the kernel. The network topology definition obtained, for example, from a graphical layout and configuration design tool, is mapped to a sequence of system commands. Suitable components should be implemented for translating configurations from the input format (usually XML) to system commands with proper arguments. Execution of these commands on physical resource nodes results in the creation of a virtual network. The implementation of the virtual Ethernet switch (from the Virtual Distributed Ethernet project) can be used as a mechanism for constructing virtual networks from Xen virtual machines belonging to different LANs. The operation relies on creating virtual switches on nodes connected via tunnels (VDE plug with VDE cables).

Migration support for Xen VMs implies changes in the layout of physical topologies used for routing traffic in the virtual network. Migration of VM execution requires the physical host to support a VDE virtual switch representing the LAN network to which the migrating machine belongs. The network configuration update which follows (or accompanies) Xen VM migration calls for a new connection to be established between the logical (virtual) network interface and its virtual switch while removing the corresponding interface from the virtual switch interface list on the source host (this is done automatically upon VM removal).

Migration in virtual networks created using VDE components requires updating connection parameters between the virtual switch and the virtual interface (representing the VM Ethernet interface in the logical interface pool of a physical host's network configuration). In practice, this means reconfiguring the list of interfaces for the group, which contains, for example, the tap interface of the VDE switch (removal of mapping (1), addition of mapping (2) and reconfiguration of connection (3-4) in Fig. 6).

Sun xVM Server is a Xen-based hypervisor that uses OpenSolaris as the base operating system providing native support for network virtualization called Crossbow [11, 12] 10]. It offers network virtualization and resource control by creating virtual stacks for Internet services based on application protocols (FTP, NFS or HTTP) and network protocols (TCP, UDP).

The core Crossbow components virtualize a physical NIC into multiple Virtual NICs (VNICs) that can be assigned to virtualized OS domains such as xVM domains, zones (OS Containers) and LDOM domains. Each Virtual NIC can be assigned its own priority and bandwidth on a shared NIC without incurring performance degradation. In Crossbow, physical resources (hardware rings and interrupts) of a NIC are allocated to a specific VNIC, thus enabling independent scheduling based on the load of a given VNIC and the classification of packets. Network traffic for one VNIC can be completely isolated from other types of traffic and assigned custom limits for the bandwidth it can use to ensure that QoS contracts are maintained.

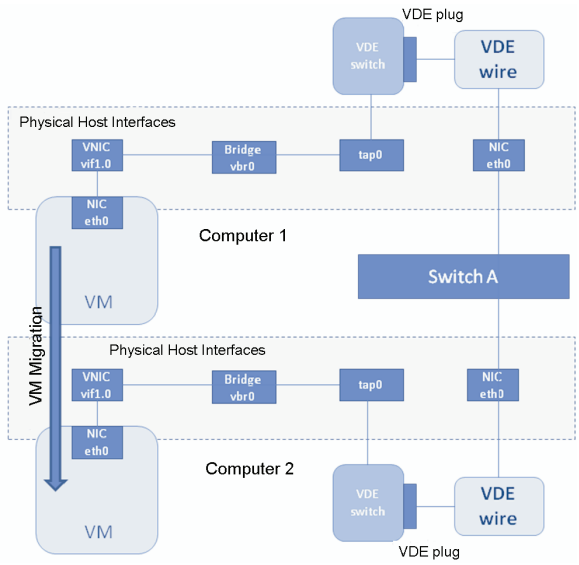

Fig. 6. VM migration in the VDE virtual network

Two types of virtual switches exist, both providing the same semantics as physical switches. When at least two VNICs are created on top of a physical NIC, the MAC layer automatically creates a virtual switch. It is also possible to set up pseudo-Ethernet NICs called etherstubs, independent from the underlying hardware and managed by the system administrator. Such etherstubs allow the creation of VNICs and provide virtual networks on a machine without actually using any hardware resources.

To prevent NIC failure, system administrators can use link aggregations supported by the IPMP (IP Multipathing) technology, which provides high availability and better throughput when several physical interfaces are grouped into one link. Such link aggregations are used for the definition of VNICs.

In the case of xVM domains or Solaris Containers (also called zones), Crossbow enables each such unit to be assigned a virtual stack instance along with one or more VNICs. This approach is used for the definition of network virtual machines (NVMs) which abstract network functionality – virtual routing, IP filtering or load balancing within a VM or a zone. Such a configuration can be prepared as an appliance (configuration template) stored in a repository and provisioned on demand.

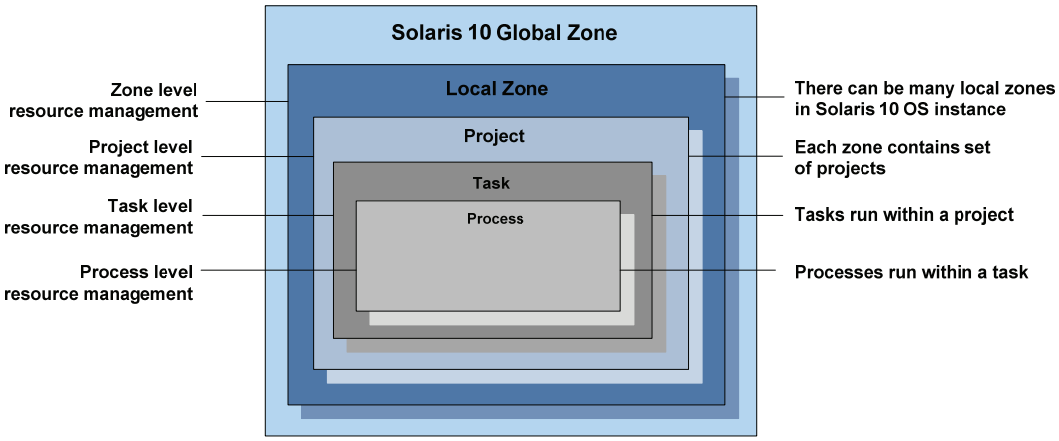

Fig. 7. Solaris virtualization technology overview

All these components can be integrated in complex virtual networks called virtual wires (vWire), spanning several virtualized physical machines connected through a physical network. These vWires are separated with VLANs and support dynamic reconfiguration and migration of virtual machines.

OpenSolaris also offers an advanced environment for hierarchical virtualization (Fig. 7) and enables specifying resource consumption limits (managed by the Solaris Resource Manager) at each of the following levels:

- Zone: Virtualized OS instance;
- Project: Identifies running workloads;
- Task: Workload component;
- Process: Running processes within a task.

Virtual Machine domains created on top of the xVM hypervisor can apply both Crossbow and Solaris Resource Manager (SRM) [13, 15] technologies to provide advanced capabilities for networking and computational resource management. These techniques are very efficient and easy to use, thus providing a very good foundation for systems that must be scalable and secure. Moreover, OpenSolaris includes many modern facilities that provide a fully integrated stack, even more comprehensive than any recent AIX, HP-UX, Linux or Windows release. In addition to the Solaris Zones virtualization technology, we can enumerate the following mechanisms [14]:

- *Zeta File System* (ZFS): Integrates volume management functions; however instead of a volume manager there is the concept of a storage pool, where capacity is provided on demand and can be dynamically resized. Administrative tasks are very easy to perform with the available tools, even by users who are not experienced in volume management. Installation directories for VMs and Containers can be created on top of ZFS. It is possible to create snapshots which can be sent to remote locations, as well as add extra capacity (if needed).
- *Process Rights Managemen*t: Extends common UNIX security functions with a set of privileges, allowing processes to acquire the required access rights and making superuser rights more granular. Individual processes are granted only those permissions which they absolutely need to work. Installation of VMs or Containers, and operations related to network configuration can be defined within a security profile and assigned to specific users who do not require superuser privileges.
- *Service Management Facilities* (SMF): A unified model of services and management activities. Services are represented as objects with defined dependencies. In case of errors (administrator error, bug or hardware error) services are restarted in a dependency-based order. SMF also facilitates the Predictive Seal-Healing feature. Grid workloads performing computations managed by SMF are described via XML descriptors that handle boot-up, access management and self-healing.
- *Extended Accounting*: Enables gathering statistics at the process, task and network flow levels for a given container. System administrators can analyze these statistics in order to determine the cost of running user applications.
- OpenSolaris can be used as a guest operating system in other virtualization solutions such as VMWare or Xen, thus providing QoS management for network, CPU and memory resources used by computations on a given virtual machine instance. The ability to run workloads can be further extended by introducing the Open HA Cluster [16, 17] which ensures the clustering of Grid services in a protected, highly-available cluster environment.

Table 1. Comparison of Crossbow and VDE network virtualization technologies

Concept	OpenSolaris xVM/Crossbow	Linux XEN with VDE
Virtualization level	Solaris Zone, xVM virtual machine	XEN Virtual Machine
Virtual NICs (VNICs)	Logical networking interface, which can be created and assigned to a VM. Administrator can choose any VNIC name ending with a digit	Logical interface created automatically by XEN hypervisor upon VM bootup. VNIC name uses the following schema: 'eth' + VM ID + '.' + 'interface ID' (e.g. eth10.1 – second VM interface with id 10)
Virtual Switching (public)	Set of VNICs created over a physical network interface	VDE Switch instance connected with a physical interface. Multiple VDE switches can be connected together with so-called 'vde_cables'
Virtual Switches (isolated)	Etherstubs – software switches to which VNICs can be assigned. VNICs must be created over an etherstub instance. Provides connectivity between virtual machines/zones on the same physical host	VDE Switch instance without real network connectivity. VNICs can be assigned directly or via Tun/Tap Linux interface bridged with VNIC
Flow	Communication stream identified by the following parameters: services (protocol + remote/local ports), transport protocol name (TCP, UDP, SCTP, etc.), remote and local IP addresses or subnets, DSCP labels. The following properties can be set for each flow: bandwidth limits, priorities, CPU pins	Similar behavior can be simulated via the Linux kernel traffic shaping configuration (e.g. with 'ip', 'iptables' and 'tc' tools). Independent of hypervisor and VM configuration
vWire	A virtual network machine can be combined with VNICs and virtual switches to create a fully virtualized network	Two or more VDE switch plugs connected via network by third-party universal communication software (such as VPN daemons or the basic "NetCAT" tool)

Table 1 summarizes the basic features of network and VM virtualization technologies. The presented features support the concept of introducing OpenSolaris and the xVM hypervisor as a base Grid platform for virtualization and hosting of computations. xVM and Crossbow ensure that the hardware stack is well integrated, efficient and easier to manage by system administrators.

IV. DEPLOYMENT – CASE STUDY

The case study bases on a sample configuration of virtual machines connected in the virtualized network topology. Our approach must provide connectivity between VMs running on different physical hosts in the same local network, and maintain this connectivity following VM migration. The scenario assumes a dedicated VNIC, ensuring the isolation of each VM group as an important function of network virtualization.

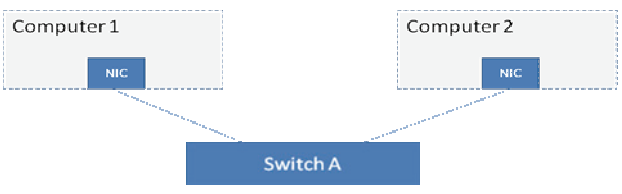

Fig. 8. Case study topology – initial setting

The scenario topology is presented in Fig. 8. It consists of two hardware nodes running OpenSolaris OS on which a set of VMs is deployed. The initial configuration of Crossbow provides a simple mechanisms for assigning bandwidth limits to VM guests (domUs). The VM definition with the use of libvirt [8] commands is as follows:

```
#virt-install --nographics --paravirt --ram 1024 –
name
 domu-vm1-osol -f /dev/zvol/dsk/rpool/zvol/domu-vm1-
osol
-root -l /var/installs/osol-0906-x86.iso –network
bridge=elxl0,capped-bandwidth=50M
```

When a Xen domU is defined, Crossbow implicitly creates a VNIC and assigns it to the VM. In our scenario, VNIC is automatically created on top of elxl0 NIC and can be verified by issuing the following command:

```
# dladm show-vnic
   LINK OVER SPEED MACADDRESS MACADDRTYPE VID
   vnic1 elxl0 100 2:8:20:53:f0:b9 random 0
   xvm1_0 elxl0 50 0:16:3e:79:d6:21 fixed 0
```

The VNIC can be created before domU, with given bandwidth restrictions, and then assigned to VMs by including the required statements in the VM XML configuration file. From the level of the VM OS, the configuration of the virtual instance can be accessed using the following command:

```
#dladm show-phys
```

```
LINK MEDIA STATE SPEED DUPLEX DEVICE
xnf0 Ethernet up 1000 full xnf0
```

In OpenSolaris running on the VM, this VNIC is visible as a normal 'physical' operating system interface. Due to this fact, two levels of Crossbow network virtualization can be introduced. The following commands create a VNIC on top of this interface using the dom0 configuration. In addition, further bandwidth allocation can be achieved by assigning limits to individual interfaces, within the scope of the main interface limit.

```
# dladm create-vnic -p maxbw=10M -l xnf0 vnic1
# dladm create-vnic -p maxbw=20M -l xnf0 vnic2
#dladm show-vnic
   LINK OVER SPEED MACADDRESS MACADDRTYPE VID
   vnic1 xnf0 10 2:8:20:1d:1a:44 random 0
   vnic2 xnf0 20 2:8:20:6a:97:9f random 0
```

The created VNICs can then be assigned to Solaris zones. In addition, each VNIC can be bound to CPUs assigned to its proper zone. Traffic management is performed by Crossbow and although it can be CPU-intensive, it does not affect running computations. The above commands result in the configuration shown in Fig. 9.

```
#dladm set-linkprop cpus=0,1 vinc1
#dladm set-linkprop cpus=2,3 vinc2
```

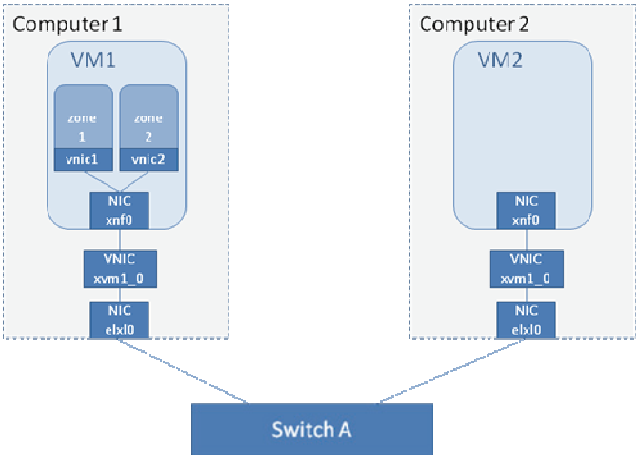

Fig. 9. Case study configuration prior to zone migration

Another scenario deals specifically with zone migration for specific VM instances. Unfortunately, the current implementation of Solaris Containers does not support live migration. This means that the zone must be halted prior to migration (running processes are stopped). Once migration has been performed, source and destination VMs are booted up again. The following scenario assumes a zone running within the ZFS file system, which enables "versioned" snapshots to be stored in an NFS-based repository.

```
#zlogin zone1 shutdown -y -i 0
#zoneadm -z zone1 detach
#zfs snapshot pool1/zones/zone1@@snap
#zfs send
pool1/zones/zone1@ver.1>/nfs_snap/zone1@ver.1
```

Once *zone1* is detached, the ZFS snapshot must be imported on the destination VM instance. Additionally, a VNIC needs to be created (VNICs are not automatically created in Solaris Containers). The zone can then be migrated and set up for servicing user requests.

```
#zfs receive pool2/zones/zone1 < /nfs_snap/zone1@ver.1"
/nfs_snap/zone1@ver.1
#dladm create-vnic -p maxbw=10M -l xnf0 vnic1
#zonecfg -z zone1
#zonecfg:zone1> create -a /zones/zone1
#zonecfg:zone1> exit
#zoneadm -z zone1 attach
#zoneadm -z zone1 boot
```

Such an infrastructure is a complex system with many elements that must be effectively managed. The constructed software tools must therefore enable *network automation* for control and management of vWire provisioning over a physical network. In addition to fulfilling computational mobility requirements, they must also support flexible resource management. The proposed solution must offer a unified stack of tools, exposing a uniform interface for managing and monitoring network resources, CPU and memory.

V. SOLUTION

The previous sections presented aspects related to migration (VM, Solaris Containers) within a virtual network. This process can be complex, especially when – apart from the need to preserve network topology – it becomes necessary to preserve QoS parameters or communication isolation. Hence, automation appears desirable. We propose to add components that perform such changes automatically without administrator supervision (an example involving zone migration was presented in Section IV).

Such an automatic process requires support for mechanisms enabling the construction of network topologies with specific layouts and, more importantly, influencing the parameters of virtual connections (QoS guarantees or restrictions).

The capabilities offered by existing network virtualization components include:

- Creating a virtual network topology by connecting VM virtual network interfaces;
- Defining primary connection parameters (bandwidth, etc.);
- Creating dedicated network services, such as firewalls and routers based on preconfigured OS images, to implement VNM functionality;

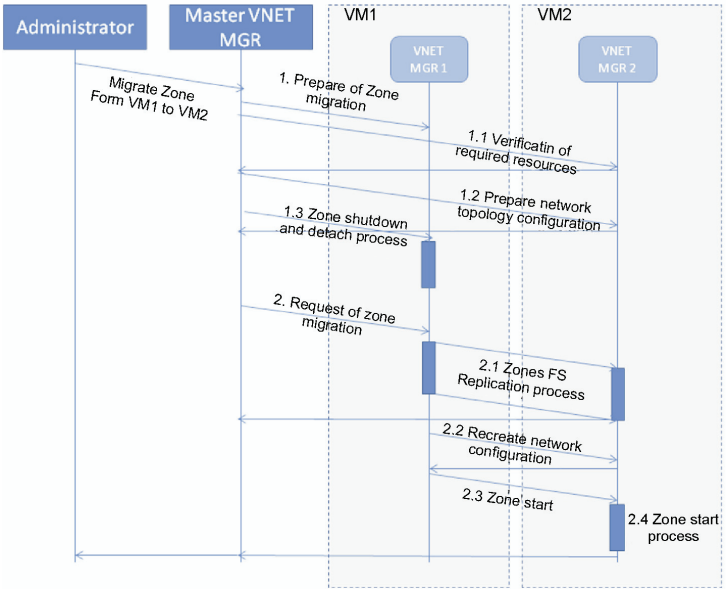

Fig. 10. Sequence of operations related to zone migration

- Recreating virtual network topology following VM or zone migration.

The key aspect of the discussed solution, presented in Section IV, is the ability to alter the physical topology layout while preserving the parameters of virtual connections. This operation is performed using VM migration techniques and can be executed autonomously or by the operator.

To automate virtual network creation and reconfiguration processes, it is necessary to implement a set of agents (Master VNET Manager and VNET Managers). These agents implement a set of commands which the administrator would otherwise have to execute (a sample command set was presented in Section IV). They can operate on the VM host operating system or in a global zone. The initial virtual network configuration is passed to agents in the XML format, and describes both the topology and QoS requirements. This configuration needs to be created by the administrator. Software agents based on JMX (Java Managed Extensions [18] technology provide an interface for buiding and modifying virtual network topologies [19]. The master agent is responsible for coordination and execution of tasks on many physical hosts. Configuration involves defining the topology and access parameters for agents running on virtual machines.

Figure 10 presents a sample sequence of invocations generated in the process of recreating a virtual topology during zone migration. Each component (VM, zone) has an initial topology and configuration of QoS parameters. This configuration is recreated within the network configuration of the target host (step 2.2 in Fig. 10). Hence, prior to migration, it is necessary to verify if the given QoS guarantees can be met by the hardware and configuration of networking components on the target host (step 1.2). Communication between agents (including operations and event delivery) can be accomplished using the JMX technology.

VI. SUMMARY

Migration of virtualized resources involves substantial complexity related to reconfiguration of virtualized networking infrastructures. Existing libraries, such as Crossbow or VDE, offer basic support for manual reconfiguration; however, automation of this process requires dedicated software agents managing the virtualized network infrastructure during the migration process.

In the scope of the VM Connection Layer, migration involves similar reconfiguration requirements as for the Zone Connection Layer. Thus, the same software agents on both layers could perform virtualized networking infrastructure automation.

The proposed migration procedure can be performed successfully only if the target computer provides the requested resources. This introduces the need for integration with a suitable resource monitoring system, which will be the focus of further study.

Acknowledgment

This research is supported by the PL-Grid Project; POIG.02.03.00-00-007/08-00.

References

[1] A. Ganguly, A. Agrawal, P.O. Boykin, R.J. Figueiredo. *IP over P2P: Enabling Self-configuring Virtual IP Networks for Grid Computing*. In Proceedings of 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS). Rodos, Grece, Apr 2006.

[2] Xuxian Jiang, Dongyan Xu, *VIOLIN: Virtual Internetworking on Overlay Infrastructure*. Parallel and Distributed Processing and Applications 3358, 937-946 (2004).

[3] Tsugawa Maur´ıcio, J.A.B. Fortes. A Virtual *Network (ViNe) Architecture for Grid Computing*. In Proceedings of 20th International Parallel and Distributed Processing Symposium (IPDPS-2006), p. 10 (2006).

[4] A. Sundararaj, P. Dinda, *Towards virtual networks for virtual machine grid computing*. In Proceedings of the 3rd USENIX Virtual Machine Research and Technology Symposium. San Jose, CA, USA, May 2004.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield. *Xen and the Art of Virtualization*. In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pp. 164-177. ACM, New York, NY, USA (2003). doi:http://doi.acm.org/10.1145/945445.945462

[6] D. Chisnall, *The Definitive Guide to the Xen Hypervisor (Prentice Hall Open Source Software Development Series)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[7] S.J. Vaughan-Nichols, N*ew Approach to Virtualization Is a Lightweight*. Computer 39 (11) 12-14 (Nov. 2006).

[8] Libvirt toolkit virtualization platforms API website, http://libvirt.org/

[9] Oracle xVM hypervisor website, http://hub.opensolaris.org/-bin/view/Community+Group+xen/WebHome

[10] N.A. Solter, J. Jelinek, G. Jelinek, D. Miner, *OpenSolaris Bible*. John Wiley and Sons (2009).

[11] N. Droux. *Crossbow for Cloud Computing. Solaris Kernel Networking*. SUN Microsystems (May 2009).

[12] S. Tripathi, K. Belgaied, N. Droux, Crossbow: Network Virtualization & Resource Partitioning – Crossbow Technical Paper, http://opensolaris.org/os/project/crossbow/-Docs/Crossbow_WP.pdf

[13] M. Lageman, *Solaris Containers – What They Are and How to Use Them*.

[14] G. Haff, *Solaris Rises, Illuminata Research Note report*, http://www.sun.com/software/solaris/solaris_rises_report.pdf, 2004.

[15] System Administration Guide: Solaris Containers-Resource Management and Solaris Zones (http://docs.sun.com/app/docs/doc/817-1592)

[16] R. Elling, *Designing Enterprise Solutions with Sun Cluster 3.0*. Prentice Hall PTR, (December 2001).

[17] J.S. Bozman, *Addressing Virtualization and High Availability Needs with SUN Solaris Cluster*. IDC White Paper – Sponsored by SUN Microsystems, October 2009.

[18] Sun Microsystems, Java™ management extensions instrumentation and agent specification, v1.2 (JSR003), Maintenance Release, October 2002

[19] K. Bałos, M. Jarząb, D. Wieczorek., K. Zieliński, *Open interface for autonomic management of virtualized resources in complex systems – construction methodology*, Future Generation Computer Systems 24 (5) 390-401 (2008).

MARCIN JARZĄB. Received his M.Sc. in Computer Science, at the University of Science and Technology (AGH – UST) in Kraków, Poland, in 2002. He worked as a software consultant at Consol Solutions and Software from 2000-2002, participating in many projects for Telco companies. He was an intern at Sun Labs in the latter half of 2003, investigating the application of the Multi-tasking Java Virtual Machine to the J2EE environment. His research interests include the tuning and performance evaluation of distributed systems, design patterns, frameworks, lightweight virtualization technologies, and architectures of autonomic computing environments. Currently he works in SOA research project performed by IT-SOA Consortium in Poland and in a project releated with Polish national grid initiative – PL-Grid.

JACEK KOSIŃSKI. He is a PhD research assistant at the department of Computer Science, AGH University of Science Technology Kraków since 2000. His main interests focus on computer networks, operating systems and virtualization. He is an author of several papers in these areas. He is a Cisco CCNP instructor. He has participated in national and international research projects, mainly EU-founded. Currently he works in SOA research project performed by IT-SOA Consortium in Poland and in a project releated with Polish national grid initiative – PL-Grid. He works on usage of virtualization techniques in areas related with resources management.

KRZYSZTOF ZIELIŃSKI. He is a full professor and head of Institute of Computer Science at AGH-UST. His interests focus on networking, mobile and wireless systems, distributed computing, and service-oriented distributed systems engineering. He is an author of over 200 papers in this area. He has been Project/Task Leader numerous EU-funded projects, like e.g.: PRO-ACCESS, 6WINIT, Ambient Networks. He served as an expert with Ministry of Science and Education. Now he is leading SOA oriented research performed by IT-SOA Consortium in Poland. In this area his research interest concerns: Adaptive SOA Solution Stack, Services Composition, Service Delivery Platforms and Methodology. His a member of IEEE, ACM and Polish Academy of Science Computer Science Chapter.