TIOSA: Testing VM Interoperability at an OS and Application Level

A hypervisor testing method and interoperability survey

Alexander Lenk, Gregory Katsaros, Michael Menzel
FZI Forschungszentrum Informatik
Berlin, Germany
{lenk, katsaros, menzel}@fzi.de

Jannis Rake-Revelant
T-Labs (Research & Innovation)
Deutsche Telekom AG
Berlin, Germany
jannis.rake-revelant@telekom.de

Ryan Skipp
T-Systems International GmbH
Bellville, South Africa
ryan.skipp@t-systems.com

Enrique Castro-Leon, Gopan V P
Intel Corporation
Portland, USA
{enrique.g.castro-leon, gopan.v.p}@intel.com

***Abstract*—Virtualization is the foundation of modern, cloud-based applications. The existence of virtual machines (VM) that host the components of such applications enables their portability and scalability. VMs are used in cloud infrastructures, and with dynamic operational requirements there is a need to move VMs within and across different clouds. The successful migration of VMs from one cloud to another should not always be considered as given. The goal of this paper is to devise and implement methods and conduct functional tests towards evaluating interoperability in cloud environments. We suggest a methodology for assessing the interoperability across different systems and we conduct a survey with a series of hypervisors and operating systems.**

Keywords—ODCA, Interoperability, Hypervisor, IaaS, Virtual Machines, VM

I. INTRODUCTION

Cloud computing has become an inevitable technology for organizations due to its potential benefits and cost savings [1]. There are public cloud offerings and software to operate private clouds that make this technology available to a large group of organizations [2], [3]. For organizations that adapt cloud computing the evolution of this technology at one point will provoke the need to move software between clouds [4]. Be it between two cloud providers due to e.g. a price difference or within owned private clouds. Either way interoperability between clouds should be provided to allow for an easy migration. From a provider perspective interoperability is a chance to attract customers by advertising the benefit of no vendor lock-in [5].

From an Infrastructure-as-a-Service (IaaS) perspective, virtual machines (VM) are the smallest unit that can be migrated. VM migrations can be done either during runtime as a live-migration [6] or after stopping the VM. Every transfer of a VM, however, requires a certain level of interoperability between two clouds. Interoperability implies compatible technologies or interfaces. Beyond the basic feasibility of interoperability between two providers there are multiple quality levels to consider. While automation and reliability in the process of a VM transfer are important factors, also the quality of the result is crucial. After the migration, a VM running on the target cloud may miss features, e.g., hardware devices, network configurations or power features like suspension, originally available [7].

A testing method to assess the quality of a VM transfer, resp. the quality of interoperability between two clouds, is not available yet. Therefore, in this paper we present the TIOSA method, a testing method to evaluate the quality of manual inter-hypervisor VM migrations in private clouds[1]. As major contributions the method provides (1) a structured, replicable process model, (2) means of measurement to describe a test result and (3) an evaluation metric to make hypervisors comparable. We also propose aggregation functions that map measured quality factors to a single score. As a second contribution we conducted experiments with four well established hypervisors that show the applicability of the method and give insights regarding the interoperability qualities of current hypervisors.

The method and experiments are results of a Proof of Concept (PoC) project of the Open Data Center Alliance (ODCA). In an initial phase of the PoC project we observed that performing a VM migration is not necessarily error-free under the current state of the art. In fact, whereas in many cases the migration did not result in a working application a black-box focused evaluation method would have indicated that the migration was a success. We, hence, developed a new method to measure and evaluate results of VM migrations on an OS

[1] Parts of this paper were already published as the ODCA whitepaper "Open Datacenter Alliance: Implementing the Open Data Center Alliance Virtual Machine Interoperability Usage Model" [8]

and application level. Using the method, the outcome of a conversion is assigned a total score of "Successful", "Warning" (as partially successful but still runnable VM) or "Failure".

The PoC project team led by T-Systems, Telekom Innovation Laboratories, the FZI Research Center for Information Technology (FZI) and the Intel Corporation conducted the experiments on a private cloud testbed consisting of machines equipped with the hypervisors VMware ESXi, Citrix XenServer, KVM, and Microsoft Hyper-V. The infrastructure was provided and maintained by the Intel.

The paper is structured as follows. Section II examines the current state of the art in VM interoperability and respective testing methods. Furthermore, it introduces basic concepts that build a foundation and common understanding for the main contributions. The TIOSA method is introduced in section III and the experiments and results are presented in section IV. The paper concludes the contributions and findings in section V.

II. State of the Art

In this section we describe the state of the art in VM interoperability of hypervisors and clouds, and of interoperability testing.

A. VM Interoperability

A virtual machine monitor or hypervisor runs on physical computers and is capable of managing multiple virtual machines that share the physical hardware resources. The DMTF defines three levels of VM portability [9]:

- Level 1: The VM only runs on a particular virtualization product and/or CPU architecture and/or virtual hardware selection. It is logically equivalent to a "suspend" in the source environment and a "resume" in the target environment. A live migration is possible at Level 1. However, Level 1 carries a number of operational restrictions, such as the preservation of IP addresses, limiting the applicability to virtual machines running in the same subnet and hypervisor.
- Level 2: The VM runs on a specific family of virtual hardware. Migration under Level 2 is equivalent to a shut-down in the source environment followed by a reboot in the target environment. Movement across different hypervisors is possible.
- Level 3: The VM runs on multiple families of virtual hardware. This is the most general framework for VM migration offering the greatest flexibility, essentially allowing a machine to be rebuilt to suit the target environment. This assumes advanced methodologies, such as integrated development and operations not in common practice today.

In this paper we deal with VM interoperability at Level 2, which presents the most immediate opportunity for advancing the state-of-the-art under the ODCA perspective.

B. Image Format Interoperability

A virtual machine can be packaged into a virtual machine image. A VM image contains configuration data like attached hardware devices and a hard disk image attachable as hard disk. VM images exist in many formats. There are proprietary formats, such as VMware's XML-based format [10] or Citrix's XVA format [11], and open standards, e.g., the open virtualization format (OVF) or the packaged open virtualization appliance (OVA) [9]. With attached hard disks, virtual machines are able to boot an operating system and make themselves available to software and users. Like on physical machines, a wide range of operating systems can be installed and run on virtualized hardware devices.

C. Interoperability testing

In the past, hypervisor vendors like VMware or XEN focused on the testing of virtual machines in a whole. They use a black box approach and conducted no further testing after a conversion was successful [10], [11].

In other contexts grey-box approaches like the one described in this paper are used to gather more information about a running VM than just observing it from the outside as a black-box. Wood et al. [12], [13] describe in their papers a related grey-box approach that, however, focuses not on the verification but the identification of heavy load machines.

III. The TIOSA Testing Method

The goal of testing hypervisor software regarding its interoperability capabilities is to determine if and how well VMs can be migrated from one hypervisor to another. Since state-of-the-art hypervisor software is not interoperable in an automated manner, a migration includes one or more conversion steps. Having successfully converted a VM and starting it successfully on another hypervisor does not imply a successful migration without any restrictions. Features like network devices, IP addresses, CPU cores and so on, might change or even fail during the migration process. A method to test hypervisor software interoperability must therefore provide (1) a structured, replicable process model, (2) means of measurement to describe a test result, and (3) an evaluation metric to make hypervisors comparable.

With TIOSA we introduce a virtual machine grey-box [12], [13] method for Testing Interoperability on Operating System and Application (TIOSA) level that addresses all aforementioned aspects and includes, in addition to tests on a virtualization level, functional tests on operating system and application level.

The following subsections explain the most important aspects of TIOSA. A process model is introduced that describes and structures interoperability testing processes. Furthermore, means of measurement and related metrics are introduced. Finally, evaluation metrics and aggregation functions explain how to calculate a final score of an interoperability test. These metrics are application specific, but we have provided a recommended set in the Appendix.

A. Interoperability Testing Process Modell

For each hypervisor software under test (hypervisor under test or HUT), a set of hypervisor software is to be defined that serve as import sources (hypervisor source or HS). Implicitly, each hypervisor software might be included as a HS in a test set of another HUT. Aside from the source and target hypervisors, respectively HUT and HS, the VM transferred

over the interoperability test influences test results and, hence, needs to be considered a test parameter. The execution in a different environment and configuration changes due to conversion steps has diverse effects on the operating system and software included in a VM. The operating system and installed software applications become part of the test parameters list.

To gain test results for all HUT-HS pairs regarding a certain VM, an identical process model shall be followed. The process model describes the course of actions to execute a test for a single HUT. Subsections 1 and 2 describe the testing sequences of the process model. The sequences are detailed steps to follow once the parameters have been defined.

At the beginning of the process, HUT and HS must first be defined as test parameters as well as a virtual machine on the HS. The test, hence points to results for the HUT regarding interoperability with the HS when transferring a virtual machine with a certain operating system (and optionally software applications). Given the test parameters the testing sequences can be followed (subsection 1 and 2). Initially an inspection of the original VM's state captures its characteristics as a basis for later comparison. An inspection includes manual and automated property tests (see tables II-IV). Next, available tools to export, convert and import the virtual machine from the HS to the HUT must be identified. The result is a set of export, conversion and import routes that allow a transfer. After applying all available routes, each resulting transferred VM on the HUT must be inspected and compared to a VM's original state. Finally, the evaluation and aggregation rules map the set of individual test results to a final classification.

In order to apply this method the following subsections give an overview over the mandatory method sequences, process steps and the expected results.

1) Sequence 1: Check Interoperability within a private cloud

Determine if a move/migration of a VM of a cloud subscriber from one hypervisor to a different one within the same cloud provider is possible.

a) Execution Steps

1. Check if the following information about VM is exposed by the hypervisors via GUI interface and/or APIs – VMware and KVM
    - Number of VMs
    - Amount and type of memory
    - Amount and type of CPU core
    - Amount and type of network interface card
    - Amount and type of disk
    - Amount of I/O required
    - Type and vendor of hypervisor
    - Firewall policies and rules

b) Expected Results

Required metadata information is available and hypervisors are in compliance with DMTF Open Virtualization Format (OVF) specification so that VMs can be migrated between hypervisors. Alternatively, if other conversion tools support non-OVF formats, a non-standard conformant path might have been chosen.

2) Sequence 2: Copy VM between two hypervisors within the private cloud

Execute the necessary steps to transfer the VM between hypervisors and evaluate according to the test sets.

a) Execution Steps

1. Get VM metadata from Source Hypervisor (e.g. VMware) (CPU, Memory, Disk space, …)
2. Perform Hypervisor Test-Set (see Appendix TABLE II.) on Source Hypervisor
3. Perform OS Test-Set (see Appendix TABLE III.) on Source Hypervisor
4. Stop the VM on Source Hypervisor
5. Export the VM in OVF Format or convert VM into destination format
6. Check whether the required resources are available on Target Hypervisor (e.g. KVM)
7. Import VM to Target Hypervisor
8. Perform Hypervisor Test-Set (see Appendix TABLE II.) on Target Hypervisor
9. Start VM on Target Hypervisor
10. Perform OS Test-Set (see Appendix TABLE III.) on Target Hypervisor
11. Compare results of Test Sets using Evaluation Rules (see Appendix TABLE IV.).

The execution steps are processed in order and the outcomes are documented in a results document. The process is also depicted in Fig. 1.

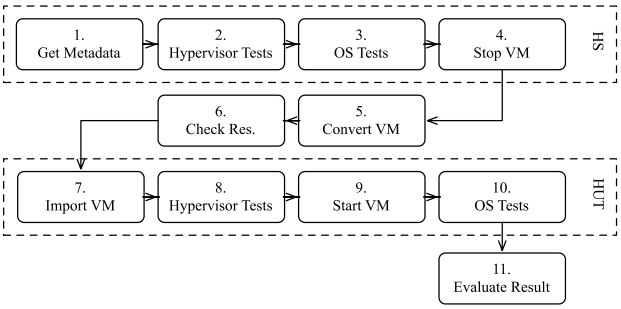

Fig. 1. TIOSA execution steps

b) Expected Results

The virtual machine has been converted correctly and can be started on the target hypervisor. All results of the tests performed in the target VM are within the tolerable margin, specified in the evaluation rules (see Tables II-IV).

B. Means of Measurement

A major task during interoperability testing is the capturing of the initial and final state of a VM; that is before and after a transfer between HS and HUT. Tables II-IV list all test sets on a hypervisor, operating system and application level. The tables also provide a structure for capturing results. A test always results in one class of "SUCCESS", "WARNING", or "FAILURE". The test conditions are contained in the tables, too. If a condition in a row evaluates to true, then the test result is the class of the respective column. After going through each test, the set of tables reflects the state of a VM.

Parts of the tests can be automated with API calls or scripts. Hypervisor software typically provides interfaces to request a status of a VM. This interface can be leveraged to acquire the test results of the test set in table II automatically. However, an implementation for each hypervisor software and even versions is needed. Similarly, scripts can help to acquire needed data for the tests in tables III and IV. Operating systems and applications typically provide programs or configuration files to assess data.

C. Evaluation Metric and Score

After a test with multiple HSs has been finished for a HUT and a particular VM, a final result or score can be determined. A score aggregates the detailed results captured in the test set tables into a single comparable statement. The score is projected on a metric with an ordinal scale of three classifications (SUCCESS, WARNING, FAILURE) to simplify evaluation.

Aggregation rules allow determining a single score for all test results. The aggregation rules are defined per test set (or even sections within test sets) complemented by an overall aggregation rule to compute a score over all test sets. The aggregation rule can use Boolean algebra to combine single tests and evaluate them for each classification. The aggregation rules are custom and should reflect a tester's expectations.

D. Conversion Paths

A conversion of a VM from a HS to a HUT is preferably accomplished via an import program of the HUT that supports formats of the HS. However, it is not uncommon that a conversion includes multiple tools over the course. An alternative supported by a range of hypervisors is an import from a portable format such as the Open Virtualization Format (OVF). Furthermore, third-party tools and intermediary conversion steps can help in absence of official vendor support.

Most solutions support a virtual-to-virtual (V2V) conversion. In a V2V approach a virtual machine is stopped and its virtual machine image file is converted to a format supported by the target hypervisor software. In contrast, physical-to-virtual (P2V) is a more generic approach that logs into a running system (physical or virtual) as a root user and extracts data and settings into a virtual machine image.

Unlike automated VM conversion, a manual alternative is to copy the disk image part of a virtual machine only. This might involve further conversion tasks. Ultimately, a new virtual machine can be created on the target hypervisor that uses the disk image as a hard drive device.

IV. HYPERVISOR SURVEY

As we mentioned in the introduction, in this paper we present the joint work of FZI, T-Systems, T-Labs and Intel in the context of a VM interoperability proof of concept (PoC) project. The whole activity lasted five months during which several series of test cases were developed and deployed using state-of-the-art tools and technologies, in current releases or versions at that time. In the following sections we describe in more detail the test cases covering VM interoperability usages, the test-bed environment, the tools used and the findings of the experiments.

A. Experiment Setup

The first series of tests, which are described in more details in the subsection B.1 of this chapter, demonstrate VM transfers within a private Cloud infrastructure (see Figure 1).

1) Test-bed environment

The test-bed environment for the realization of this PoC was hosted in the Intel End-user Integration Center Test Lab Cloud, in an isolated network partition consisting of four server nodes and a console computer. Specifically, the environment had the following characteristics:

1. Four Dell C6220 Servers each with 2 x Intel® E5-2650 CPU @ 2.00 GHz, 64GB RAM & 280 GB Hard Disk.
2. Network switches and router as needed
3. Virtual Machines to function as management consoles
4. NFS share with 800 GB storage

The access to this environment was through the standard Web portal and associated access applets. Exclusive, out-of-band access to the hardware has been provided including low-level operations, down to setting up BIOS sheets and cycling the power in the machines. Fig. 2 illustrates the test-bed's architecture and its components.

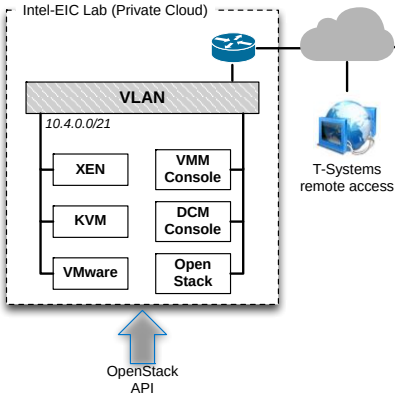

Fig. 2. PoC environment set-up

2) Test-bed component stack and tools

For the implementation of this interoperability PoC we had to set-up and use several hypervisors, Guest OSs and VM monitoring consoles in combination. In the following list we present the solutions that were available in the provided test-bed:

- Hypervisors
  - VMware ESXi 5.0.0, Citrix Xen Server 6.0.2, KVM (Cent OS6.3), Microsoft Hyper-V (Windows Server 2008 R2)
- Guest OS:
  - CentOS 6.3 64 bit, Ubuntu 12.04 64 bit, Microsoft Windows Server 2008 R2 64 bit
- Virtual Machine Monitor consoles
  - VMware vSphere Client v5.0.0, Citrix Xen Center 6.0.2, Virtual Machine Manager 0.9.4 for KVM , Microsoft Hyper-V Manager 6.1

The experimentation with several VM conversion tools was very important in order to perform all identified test series. The documentation of the versions of each tool and component is of major importance, because the results of the test cases are directly related with the functionality and features of those tools. In TABLE I. we present the specific tools that we used throughout this PoC. For our testing we used the latest publicly available version of the tools at the time of the testing. Some of the results may differ with newer tool or hypervisor versions.

Hypervisor management tools commonly offer a user interface to remotely configure a hypervisor and manage virtual machines instantiated on the hypervisor machine. A common virtual machine management operation involves adding and removing virtual machine instances, as well as resource assignments and access to machine instances via console interfaces. Beyond that, import and export functionality allows migrating virtual machines from or to compatible hypervisors. However, exports are usually offered only to hypervisor products of the same vendor.

TABLE I. VM CONVERSION TOOLS:

VM Converter Tool	Version
VMware vSphere Client	5.0.0 Build 455964
Citrix XEN Center	6.0.2 (build 53158)
Red Hat KVM Virtual Machine Manager	0.9.4
Red Hat RHEV Manager	3.0.7_0001_2.el6_3
Microsoft Hyper-V Manager	6.1.7601.17514
VMware vCenter Converter Standalone	5.0.0 build-470252
Citrix XEN Convert	2.3.1.2654
Microsoft Virtual Machine Converter	1.0.4619.17079

B. *Experiment Results*

The testing of VM interoperability presents a combinatorial challenge. In order to keep this project within the available resource constraints, the team selected four hypervisor environments and three operating systems in common use today, namely VMware, Citrix Xen, KVM and Microsoft Hyper-V among the hypervisors and CentOS 6.2, Ubuntu 12.0.4 and Microsoft Windows 2008 R2 64-bit as the operating systems.

Experimental results are a function of the operating system images being run under a virtual machine. Tests have been conducted for all the combinations possible for the selected source (HS) and target hypervisors (HUT) with each of the three selected, very common operation systems CentOS 6.3, Ubuntu 12.0.4 and Windows Server 2008 R2. The HSs and HUTs, and the VMs with all three selected operating systems are thus the determined parameters. The following describes the outcome of runs conducted with the TIOSA method for the aforementioned parameter combinations. For each operating system image selected, each table captures the results with HUTs in columns and HSs in rows.

For private-to-private Cloud conversions, the technical team tested all 12 possible conversion combinations between the hypervisors for the pre-mentioned OS images. For each of the OS images tested there were three test sequences with TIOSA:

1. A pretest to assess whether the conversion is possible
2. The process of converting a VM image from the source hypervisor to the target hypervisor
3. Attempting to run the translated image to the target environment

The results of executed test cases must be repeatable and, hence, every test case demands at least 3 execution cycles and according evaluations showing stable, identical results.

1) *CentOS 6.3*

For the CentOS 6.3 images most VM conversions were successful except those with KVM as target hypervisor (Fig. 3). Even after the successful conversion most of the machines were not runnable in the target environment due to OS related problems, except for the VMware to Citrix Xen conversion.

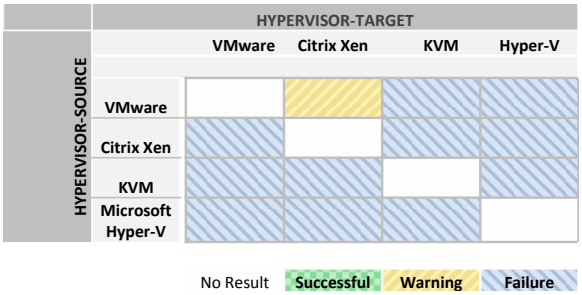

Fig. 3. CentOS 6.3 Results

2) *Ubunbtu 12.04*

The Ubuntu 12.04 test series (Fig. 4) were relatively better than the previous ones, having successful conversion from Citrix Xen to Hyper-V, successful with some warnings conversion for VMware to Citrix, KVM to VMware and Hyper-V and Hyper-V to VMware and Citrix. The remaining combinations were not operational.

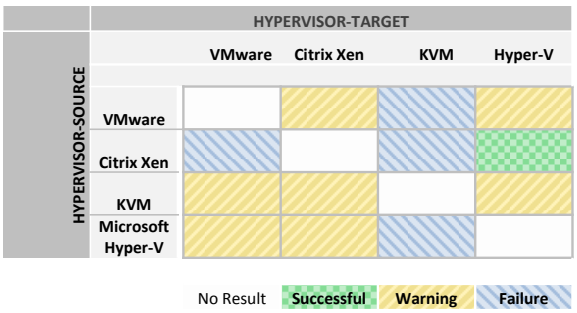

Fig. 4. Ubuntu 12.04 Results

3) *Windows Server 2008 R2*

For Windows Server 2008 R2 images, most VM conversions were successful except from Xen and Hyper-V to KVM, and from KVM to VMware. The specific result combinations can be seen in Fig. 5.

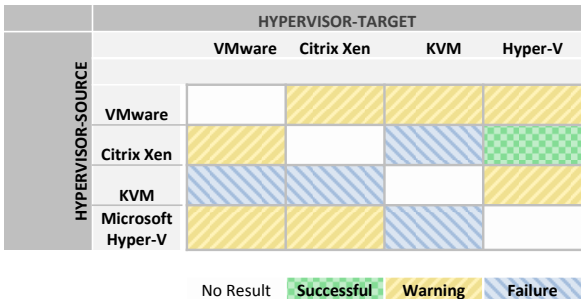

Fig. 5. Windows Server 2008 R2

Most of the cases required manual creation of the VM on the target hypervisor from the exported images, or migration of running VM using specific tools.

V. CONCLUSION

VM interoperability is an absolute precondition for truly realizing the often expressed benefits of virtualized clouds such as the ability to balance resources through fungible pools of resources, business continuity and load balancing by leveraging distributed publicly available resources. To the knowledge of the participants of this project, this experiment is the first of its kind leveraging a broad spectrum of hypervisor environments, guest operating systems and publicly available conversion tools.

Yet, in spite of the extant number of visionary articles, and industry espousing the benefits of virtualized clouds, the outcome of the experiments described in this study is sobering at a first blush: For the 36 possible conversion paths between hypervisors within a private cloud in this study, 2 were successful, 15 went through with warnings and 19 had failures (see Fig. 6). Of the failures, 8 were due to missing tool support by KVM, 8 were due to Cent OS related problems, and 3 due to commercial hypervisor related issues. Several migration paths that succeeded with warnings required virtual machines to be created or disk images attached manually.

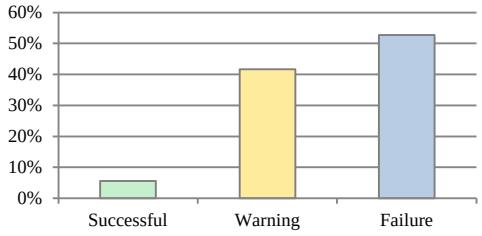

Fig. 6. Overall results

The results may sound alarming and can indeed be interpreted as a signal to further extend support for virtual machine interoperability in hypervisors and private clouds. However, it may be noted that the tests followed our new extensive testing methodology that comprises of tests on the operating system and hypervisor levels with automated test scripts running within the virtual machines before and after migration, meticulously verifying each evaluation rule defined for the project. With our survey we showed that our method gives more insight into the migration process than just observing the virtual machine as a black-box. The discussions with the vendors after the survey also showed that our testing was done correctly and was useful for them in understanding what their customers need during a migration.

Another factor to be considered is that, given that there was little prior research for VM interoperability, we designed our evaluation rules to err on the side of rigor, and for instance changes in resource memory size or IP Address changes or firewall and routing rules or inability to pause and un-pause in the migrated VMs were flagged as warnings. Most applications will continue running under these circumstances, and therefore it might be possible to relax some of these requirements. For instance, if an application is known to be impervious to IP-changes, the operator may select to ignore this. However, the team decided that these conditions should be relaxed only after a thorough discussion in the industry and consensus has been built about what the actual practice should be. It is entirely possible that the tools in this project were not used in the manner intended by the supplier. For instance the tools may have been designed for a "once or twice in a lifetime" condition. This is reflected in the degree of manual intervention required to make them function.

The results indicate that in spite of the promise of the cloud, the road to interoperability where IT processes can be extended seamlessly to the public cloud is not a reality yet. The project participants would like to encourage a healthy dialog in the industry to advance the state of the art to the point that interoperability becomes a second order consideration, allowing users to focus on the business problems at hand instead. Also under the current state of the art, virtual machine conversion tools specific to the hypervisors have to be used. Intermediate conversion to or from an interoperable format such as OVF is sometimes offered. There is no guarantee that the two-step conversion will work in all cases. The upshot is that no universal translator exists, and hence any attempt to carry out migrations across a nontrivial multiplicity of hypervisors will have to use a patchwork of conversion tools, each with particular idiosyncrasies. For business and technical reasons, it might not be realistic to require vendors to supply universal translators. Our recommendation is to encourage the industry to establish consensus for consistent behaviors in translation tools, in such a way that when operators need to use more than one, the tools will behave in a self-consistent manner.

For the future, the ODCA plans to use the testing method presented in this paper for monitoring the progress of hypervisor interoperability in the industry.

VI. REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST Spec. Publ.*, vol. 800, p. 145, 2011.

[2] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the Cloud? An architectural map of the Cloud landscape," in *Software Engineering Challenges of Cloud Computing, 2009. CLOUD'09. ICSE Workshop on*, 2009, pp. 23–31.

[3] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang, "Comparison of open-source cloud management platforms: OpenStack and OpenNebula," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, 2012, pp. 2457–2461.

[4] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," presented at the CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization, 2011, pp. 32–38.

[5] R. Cowan, "Tortoises and hares: choice among technologies of unknown merit," *Econ. J.*, vol. 101, no. 407, pp. 801–814, 1991.

[6] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. De Laat, J. Mambretti, I. Monga, B. Van Oudenaarde, S. Raghunath, and P. Yonghui Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Gener. Comput. Syst.*, vol. 22, no. 8, pp. 901–907, 2006.

[7] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehörster, and A. Brinkmann, "Non-intrusive virtualization management using libvirt," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2010, pp. 574–579.

[8] A. Lenk, M. Menzel, D. Müller, J. Rake-Revelant, R. Bederke, R. Skipp, M. Tadikonda, S. Govindan, E. Castro-Leon, Gopan P.V, and G. Katsaros, "Open Datacenter Alliance: Implementing the Open Data Center Alliance Virtual Machine Interoperability Usage Model," presented at the FORECAST 2013, San Francisco, 2013, Available: http://www.opendatacenteralliance.org/docs/VM_Interop_PoC_White_Paper.pdf

[9] Distributed Management Task Force, Inc. (DMTF), "Open Virtualization Format White Paper Version 1.0.0 DSP2017." 06-Feb-2009.

[10] VMware, "VMware vCenter Converter Standalone User's Guide - vCenter Converter Standalone 5.1." [Online]. Available: http://www.vmware.com/pdf/convsa_51_guide.pdf. [Accessed: 29-Oct-2013].

[11] "CTX133505 - How to Convert VMware Virtual Machines to XenServer 6.0 and later - Citrix Knowledge Center." [Online]. Available: http://support.citrix.com/article/CTX133505. [Accessed: 29-Oct-2013].

[12] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration.," in *NSDI*, 2007, vol. 7, pp. 229–242.

[13] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Comput. Networks*, vol. 53, no. 17, pp. 2923–2938, Dec. 2009.

APPENDIX: EVALUATION RULES

TABLE II. HYPERVISOR TEST

ID	*Test*	*Success*	*Warning*	*Failure*
1.1	**VM Lifecycle**			
1.1.a	Launch	successful	-	otherwise
1.1.b	Reboot	equivalent to source or successful	-	otherwise
1.1.c	Pause	equivalent to source or successful	-	otherwise
1.1.d	Un-Pause	equivalent to source or successful	-	otherwise
1.1.e	Suspend	equivalent to source or successful	-	otherwise
1.1.f	Resume	equivalent to source or successful	-	otherwise
1.1.g	Terminate	Successful	-	otherwise
1.2	**VM Disk Management**			
1.2.a	Resize Volume	equivalent to source or successful	-	otherwise
1.2.b	Attach Volume	equivalent to source or successful	-	otherwise
1.2.c	Detach Volume	equivalent to source or successful	-	otherwise
1.3	**Injection**			
1.3.a	Inject network	equivalent to source or successful	-	otherwise
1.3.b	Inject file	equivalent to source or successful	-	otherwise
1.4	**VM Access**			
1.4.a	Serial Console	equivalent to source or successful	-	otherwise
1.4.b	Graphical Console	equivalent to source or successful	-	otherwise
1.5	**VM Network Management**			
1.5.a	VLAN Networking	equivalent to source or successful	-	otherwise
1.5.b	Flat Networking	equivalent to source or successful	-	otherwise
1.5.c	Hypervisor Firewall Rules	equivalent to source or successful	-	otherwise
1.5.d	Routing	equivalent to source or successful	-	otherwise

TABLE III. OPERATING SYSTEM (OS) AND APPLICATION TEST SET

ID	*Test*	*Success*	*Warning*	*Failure*
2.1	**OS metadata**			
2.1.a	Kernel Version	no change	revision change	version change or arch change
2.2	**Check resources**			
2.2.a	CPU	no change	+-10%	higher deviation
2.2.b	MEM	no change	+-10%	higher deviation
2.2.c	Disk	same mount points, for all size not changed	different mount points, for all size not changed	further deviations
2.3	**Connectivity**			
2.3.a	Network interfaces	no change	changed	missing
2.3.b	IP Address	no change	changed	missing
2.3.c	Firewall Rules	no change	rule is changed	rule is missing
2.3.d	Routing	no change	route is changed	route is missing
2.3.e	ICMP private IP	3 packages sent, 0% loss, < 3000ms	3 packages sent, < 33% loss, < 6000ms	higher loss or latency
2.3.f	ICMP public IP	3 packages sent, 0% loss, < 3000ms	3 packages sent, < 33% loss, < 6000ms	higher loss or latency
2.3.g	DNS reverse lookup private IP	successful	-	not successful
2.3.h	DNS reverse lookup public IP	194.25.2.129 -> dns.isp.t-ipnet.de	-	otherwise
2.3.i	Remote Shell Process running	no change	-	changed
2.3.j	Remote Shell Port available	no change	-	changed
2.4	**Check file system and user management**			
2.4.a	Add dummy user	successful	-	failed
2.4.b	Write test file to home dir	successful	-	failed
2.4.c	Read test file from home dir	successful	-	failed
2.4.d	Delete test file from home dir	successful	-	failed
2.4.e	Change dummy user password	successful	-	failed
2.4.f	Delete dummy user	successful	-	failed

TABLE IV. OVERALL EVALUATION SET

ID	*Test*	*Success*	*Warning*	*Failure*
3.1	**Hypervisor**			
3.1.a	Hypervisor Aggregation rule	For all tests (1.x) the result is “Successful”	otherwise	Test 1.1.a (Launch) result is “Failed” or Test 1.1.g (Terminate) result is “Failed”
3.2	**OS**			
3.2.a	OS Aggregation rule	For all tests (2.x) the result is “Successful”	otherwise	For at least one test (2.x) the result is “Failed” or at least one test could not be performed
3.3	**Overall**			
3.3.a	Overall Aggregation rule	For all results (3.1-3.2) the value is “Successful”	For at least one result (3.1-3.2) the value is “Warning”	For at least one result (3.1-3.2) the value is “Failed”