OpenSAF and VMware from the Perspective of High Availability

Ali Nikzad, Ferhat Khendek
Engineering and Computer Science, Concordia University
Montreal, Canada
al_nik@encs.concordia.ca
ferhat.khendek@concordia.ca

Maria Toeroe
Ericsson Inc.
Montreal, Canada
Maria.Toeroe@ericsson.com

Abstract— **Cloud services are becoming one of the most popular means of delivering computational services to users who demand services with higher availability. Virtualization is one of the key enablers of the cloud infrastructure. Availability of the virtual machines along with the availability of the hosted software components are the fundamental ingredients for achieving highly available services in the cloud. There are some availability solutions introduced by virtualization vendors like VMware HA and VMware FT. At the same time the SAForum specifications and OpenSAF as a compliant implementation offer a standard based open solution for service high availability. In this paper, we investigate these solutions for availability through experiments, compare them according to metrics and based on the results propose architectures that combine them to provide highly available applications in virtualized environments.**

Keywords- High Availability; Virtualization; OpenSAF; SAForum; VMware;

I. Introduction

The world relies more and more on computers and the services that computer applications provide. This growth has led to higher demand and new requirements from the customers. They expect services to be available and in reach anytime, in other words, services that are highly available. Service or system availability is defined as a function of failure rate or Mean Time Between Failures (MTBF) and Mean Time To Repair. High availability which is our concern throughout this paper is at least 99.999% of availability (five nines) [1]. This allows for a maximum of 5.26 minutes of downtime in a year. To tackle this increasing demand a consortium of software and telecommunication companies, the Service Availability Forum (SAForum) [2] was created. This consortium has defined and standardized a set of middleware services.

AIS (Application Interface Specification) [3] is a set of middleware services defined by the SAForum to enable the development of highly available applications. The Availability Management Framework (AMF) [4] is one of the most important services defined in AIS. AMF manages the high availability of the services by managing the redundant software entities providing these services and shifting the workload from faulty entities to healthy ones at runtime. OpenSAF [5] is an open source SAForum compliant middleware implementation.

On the other hand, the computing world is moving toward cloud services and cloud computing, which mostly are based on virtual machines (VMs) and virtual resources. "The term virtualization broadly describes the separation of a resource or request for a service from the underlying physical delivery of that service" [6]. It is a mechanism for emulating hardware and software so that it would function exactly like a physical machine. The virtualization techniques can also be applied to other infrastructural layers like network, storage and memory. Some of the advantages of using virtualization include: improved hardware utilization, cost efficiency, power efficiency, portability and OS-hardware independence. Several virtualization solutions/products exist. Among these solutions VMware [7] has tackled the problem of availability.

The aim of this paper is to evaluate these two solutions for availability, their combinations, and to propose architectures that take advantage of the strengths of both solutions. For the purpose of our experiments we have used VMware [7] as an example of virtualization solution and OpenSAF as an SAForum compliant middleware implementation. Thus, in the proposed architectures we should benefit from both service high availability provided by OpenSAF and virtualization provided by VMware or other virtualization solutions. These architectures will certainly be useful in cloud environments.

The rest of the paper is organized as follows. In section II, we provide the necessary background information. Section III introduces the different metrics for the evaluation. In Section IV, we present the test-bed, the baseline architectures and the corresponding collected measurements followed by the analysis. In Section V we propose architectures that combine the OpenSAF solution for high-availability with VMware. In Section VI, we discuss the related work before concluding in Section VII.

II. BACKGROUND

A. SAForum and OpenSAF

The SAForum has defined a set of middleware services for enabling highly available applications and services. AMF, one of the most important services, plays the key role of keeping the service provided by an application highly available by managing and coordinating its redundant resources, and performing recovery/repair in the case of a failure. AMF detects the failure of application components and recovers its service according to the configuration provided with the application. This configuration represents the architecture of the application from the AMF perspective and describes the different entities composing it and their relations. A *component* is a logical entity that represents a set of resources. It is the smallest entity on which AMF can detect errors and carry out recovery and repair actions. Two main component categories have been defined, Service Availability *(SA)-Aware* and *non-SA-Aware*. SA-Aware components are controlled directly by AMF and they are firmly integrated with the framework. The non-SA-Aware components are not directly registered with AMF. They may be mediated through another SA-aware component called the proxy. When there is no proxy component, AMF controls only the life cycle of the component and therefore its services. The workload AMF assigns to the component is represented as a *Component Service Instance (CSI)*. A *Service Unit (SU)* is composed of a set of components combining their individual functionalities to provide a higher level service referred to as *Service Instance (SI)*, which is the aggregation of CSIs assigned to the components of the SU. At runtime AMF assigns an *HA state* to an SU and its components on behalf of an SI and its CSIs. The HA state of a component and its SU for a particular CSI and its SI can take one of these values: *active, standby, quiescing* and *quiesced*.

A *Service Group (SG)* consists of a set of SUs, grouped for protecting a set of SIs. Accordingly, AMF assigns the same SI to different SUs in the SG in potentially different HA states, e.g. one SU takes the active the other the standby HA state for the SI. An SG may have one of the five different redundancy models defined in AMF. In the 2N redundancy model there is at most one SU in the SG, which takes the active HA state for all SIs protected by the SG and at most one SU with the standby HA state. in the N+M redundancy model N SUs share the active HA state assignments for the SIs and M SUs the standby HA state assignments. Each SI has one active and one standby assignment. In the N-Way redundancy models, each SU can take the active HA state for some SIs and the standby HA state for some others. As opposed to 2N and N+M, an SI can have more than one standby assignment, but in all three cases it has one active assignment. In N-Way-active redundancy model, SUs are assigned only as active. The same SI can be assigned active to multiple SUs. The No-Redundancy redundancy model also does not support the standby HA state. There is a one to one relationship between the SUs and the SIs, meaning that each SI is assigned to at most one SU and each SU protects at most one SI. A sample AMF configuration is shown in Fig. 1. The application is deployed in a cluster of three nodes.

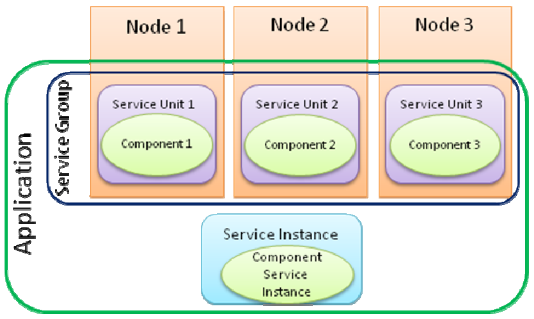

Figure 1. A sample AMF configuration

OpenSAF is an open source project and middleware, which implements several SAForum services including AMF. The OpenSAF middleware is actively supported by leading companies in the telecommunications and enterprise computing industries.

B. VMware

VMware[7] is one of the leading companies in providing virtualization solutions. The VMware products range from application level virtualization to bare-metal hypervisors. The VMware products we have used in our investigations are the following:

VMware vSphere is a cloud operating system. It leverages the power of virtualization to transform datacenters into cloud computing infrastructures.

VMware ESX and VMware ESXi are bare-metal hypervisors used in VMware vSphere. Native, bare-metal or type 1 hypervisor is installed and run directly on the host hardware and it does not need any host operating system.

VMware workstation is a non-bare-metal hypervisor which allows users to set up multiple VMs. A non-bare-metal hypervisor or hosted hypervisor is installed on a conventional operating system (the host) like Windows or Linux.

VMware vCenter Server provides unified management of all

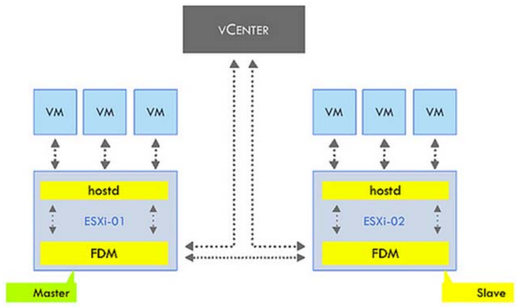

Figure 2. VMware HA [8]

vSphere hosts and VMs in the datacenter from a single console. It is also used for creating the clusters and enabling VMware availability solutions.

VMware has introduced two solutions for providing availability, VMware HA and VMware FT. Both solutions are available in VMware vSphere.

VMware HA detects failed physical hosts or failed VMs. It restarts the VMs on the same or other hosts in the event of a failure. It uses a software agent deployed on each host, along with a network heartbeat to identify when a host is offline.

VMware HA can be enabled in a cluster created in the VMware vCenter Server. When a host is added to the cluster, an agent is uploaded to the host and configured to communicate with other agents in the cluster. This agent is called Fault Domain Manager (FDM). It is responsible for tasks like communicating the host resource information, the VM states and HA properties to other hosts in the cluster. A second agent uploaded to the host is called hostd. This agent is responsible among others for powering on VMs. FDM talks directly to hostd and vCenter [8]. In a HA enabled cluster, there are two types of hosts, Master and Slave. The Master is selected through an election and the other nodes become the slaves. A VMware HA deployment example is shown in Fig. 2.

VMware FT creates an exact replica of the protected VM and keeps this replica on standby, ready to be brought into use upon a failure of the primary VM. It is built on the ESXi host platform using the vLockstep technology. The executions of the primary VM are recorded and sent to the standby through a network connection called logging channel and then replayed in the standby. The standby replays all the operations, but its outputs are dropped by the hypervisor [9].

III. METRICS

To evaluate the two solutions and their combinations from the availability perspective, we defined a set of metrics that we divide into two categories, qualitative and quantitative. The later require experiments and measurements while the former refer to the general characteristics of these solutions.

A. *Qualitative metrics*

Complexity of using the high availability solution: In OpenSAF for making an application SA-Aware we need to change the source code of the component. Neither VMware HA nor VMware FT requires code change. We should mention however that OpenSAF can manage Non-SA-Aware components which do not require any code change.

Redundancy Models: OpenSAF supports five different redundancy models as defined in the SAForum specification while in VMware, HA has no VM redundancy and FT is similar to the 2N redundancy model, which makes a total of two redundancy models.

Scope of failure: OpenSAF can detect failures in the hardware, the operating system and the components, while VMware only detects hardware level and VM failures with limited failure detection for operating systems.

Service continuity: Components managed by OpenSAF can provide service continuity by using, for example, the Checkpoint service for state synchronization. VMware provides service continuity with its FT solution for the case of hardware and VM failures.

Supported platforms: OpenSAF has been implemented for Linux, it only supports Linux based components while VMware supports all platforms because the FT and HA solutions are independent of the operating system running in the VM.

B. *Quantitative metrics*

Beside the qualitative metrics we have defined a number of quantitative metrics.

Reaction Time is the duration from the time when the failure happens until the time of the first reaction seen from the availability solution.

Repair Time is the period of time from the first reaction until the faulty unit is fixed.

Recovery Time also starts at the time of the first reaction until the service has been recovered.

Outage Time is the time during which the service is not provided and it starts from the time of the failure until the time the service is recovered. This time also corresponds to the sum of the reaction and the recovery times.

IV. BASELINE ARCHITECTURES AND ANALYSIS

As test-bed we used a cluster with five nodes with identical hardware. We had two switches for the inter-node connections and all nodes were connected to both switches for redundancy. For the case study we selected the VLC media player application as it has been already modified to work with OpenSAF as a SA-Aware component [10]. It was configured to

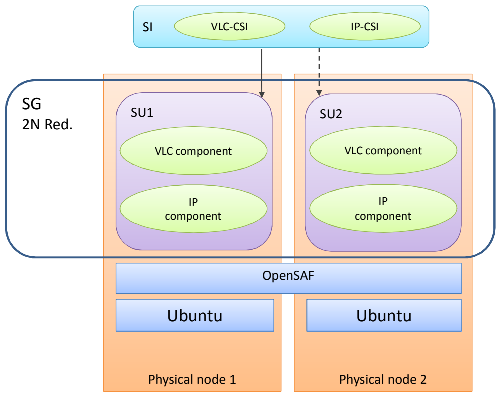

Figure 3. VLC and IP components in OpenSAF

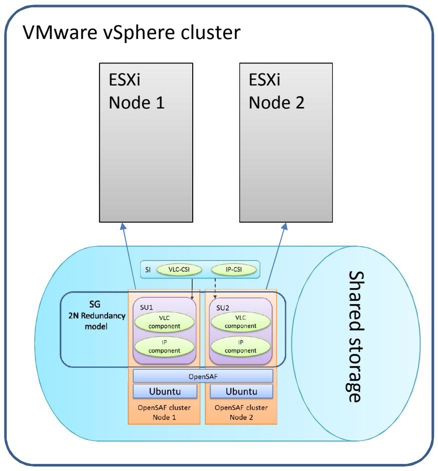

Figure 4. OpenSAF on virtual nodes

function with the 2N redundancy model as shown in Fig. 3. A possible active assignment performed at runtime by OpenSAF is shown with solid line and the standby with the dashed line.

To make VLC an SA-Aware component we made some changes to its code including check-pointing the media stream it provides to enable service continuity. We also added a new IP component, which is responsible for keeping the streaming IP address live. In our architectures, VLC can be used as an application in VMware HA, or as a non-SA-Aware or SA-Aware component managed by OpenSAF.

For the experiments we considered different failures: VLC component failure, VM failure and physical node failure. We simulated these failures by killing the VLC, or the VM processes, or force-rebooting the physical node, respectively. Since the quality of our measurements dependent highly on time synchronization, we used the NTP and PTP time synchronization protocols. VMware HA was the only VM availability mechanism we could use as VMware FT supports only a limited range of CPUs and our cluster nodes in the test-bed did not meet these requirements. All the experiments were run using VLC 1.1, OpenSAF 4.2.1, ESXi 5.0 and Ubuntu 10.04.

A. *Baseline architectures*

Our baseline architectures consisted of using OpenSAF alone, VMware HA alone and a simple combination of OpenSAF and VMware HA.

1) *OpenSAF on the physical nodes*

The first architecture is the deployment of OpenSAF on the physical nodes. Since our case-study is configured with the 2N redundancy model we selected two nodes of our test-bed to host our experiments.

2) *VMware HA*

In this architecture we created a vSphere cluster using 2 ESXi nodes and enabled VMware HA on the cluster using VMware vCenter. We also added one VM with Ubuntu Linux and VLC installed on it. We put the VM image on an NFS shared storage so that it is accessible from all cluster nodes.

3) *OpenSAF on virtual nodes*

VMware HA does not detect the application failures and it is not designed for the availability of application services running in the VM. It is a solution for recovering from hardware and VM failures. OpenSAF is specialized in making application services highly available. To take advantage of the virtualization which VMware provides and the service high availability management of OpenSAF we combined these two solutions. In this first combination we deployed the OpenSAF cluster on virtual nodes rather than on physical nodes. We added a second VM to the previous architecture and had the OpenSAF middleware installed and configured on the VMs. In this architecture we could run either the SA-aware or the Non-SA-Aware version of VLC. We used this architecture with both VMware HA enabled and disabled. Fig. 4 shows this architecture where the OpenSAF configuration used in the physical nodes was applied in the VMs. We put the VM images on a shared storage so that they would be accessible for both ESXi nodes.

TABLE I. BASELINE ARCHITECTURES AND APPLICABLE FAILURE EXPERMINENTS

Architectures	**VLC component failure**	**VM failure**	**Node Failure**
OpenSAF on physical nodes with SA-Aware VLC component	√	Not applicable	√
OpenSAF on physical nodes with Non-SA-Aware VLC component	√	Not applicable	√
OpenSAF on virtual nodes with SA-Aware VLC component (with/without VMware HA enabled)	√	√	√
OpenSAF on virtual nodes with Non-SA-Aware VLC component (with/without VMware HA enabled)	√	√	√
VMware HA	Not detectable	√	√

TABLE I. summarizes the baseline architectures and the respective applicable failure types.

B. *Measurements and Analysis*

To collect the measurements, we instrumented the respective commands and code with timestamps according to our definitions. They were included in the instantiation and cleanup commands of the VLC and IP components in OpenSAF related measurements, in the system start up script, and in some parts of the VLC code associated with registering with AMF and getting the active assignment. To record the failure time we also recorded the time when we caused a failure. For example, the reaction time to a component failure in OpenSAF related architectures was measured from the failure time – killing the VLC component – to the time AMF called the VLC component's cleanup command. The recovery time was taken from this reaction time until AMF called VLC to assign the active state.

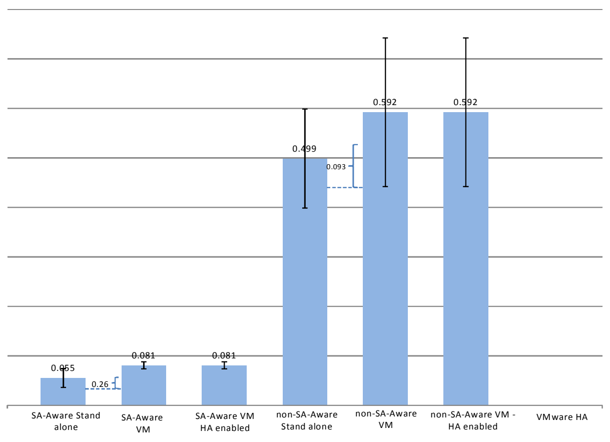

Figure 5. Outage due to VLC component failure

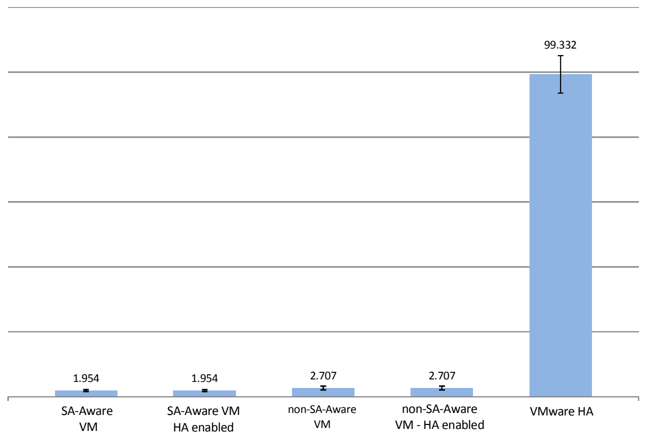

Figure 6. Outage due to VM failure

The measurements were taken in each architecture for each applicable failure type. We repeated each experiment 10 times and we determined the average. Figures 5 to 7 show the results of the experiments in seconds.

Since VMware HA did not detect component failures, Fig.5 has no measurement in the last column. The first three columns represent the measurements for the SA-Aware VLC component in the three different deployments. The next three columns represent the measurements for the non-SA-Aware component in the same deployments. They show a significant increase of the outage time. The first reason of the difference is in the way the fault is detected. The SA-Aware components have a very tight coupling with AMF; they link to the AMF library which results in faster failure detection. In the Non-SA-Aware component, there is only a passive monitor which is implemented as checking on the process ID of the VLC process in the OS. The second reason is related to the recovery, which is faster in the SA-Aware version. The SA-Aware VLC is pre-instantiated on the standby node so that it only needs to take over the active assignment. For the Non-SA-Aware component, the instantiation is done when it is assigned as active. We also see a difference between using the component in a physical node and in a virtual node, which indicates a 15% and 30% overhead of the VM layer. We have the similar delays in the measurements of the physical node failure (see Fig. 7). Huber et al also reports in [11] that the performance, using VMs is less compared to physical hosts. All these support the assumption that the delay we experience is the result of the difference in the performance of a VM and a physical machine and it is inevitable.

Fig. 6 illustrates the measurements for the service outage time due to VM failure. As VM failure is not applicable to deployments on stand-alone nodes we did not include them in the chart. The huge difference between the VMware HA and OpenSAF based deployments is quite obvious. In VMware HA there is no standby, so to recover it can only restart the failed node on the same or another ESXi host. One of the important factors in VMware HA outage time is the way that it handles failures. VMware HA detects a failure by missing heartbeats. If this is the case, it checks for the I/O activity from the VMware Tools process in the VM and if both (heartbeats and I/O activity) are missing, the VM is declared dead and VMware HA restarts the VM. High sensitivity VM monitoring means 30 seconds interval and can only be increased.

The physical node failure outage times in the different deployments are shown in Fig. 7. The outage in the last column (having the VMware HA as the only availability solution) is far longer than in the other architectures which use OpenSAF. The outage is 89 seconds in VMware HA vs. less than 3 seconds with OpenSAF. VMware HA determines host failure using the heartbeats between master and slave hosts. If the heartbeat is missed VMware HA uses data-store heart-beating to resolve any network partitioning and if the host failure is confirmed it restarts the VMs of the failed host at another location. As we mentioned, comparing the virtualized deployments with their stand-alone equivalents, we see that again there is a delay when the virtualization layer comes into the picture.

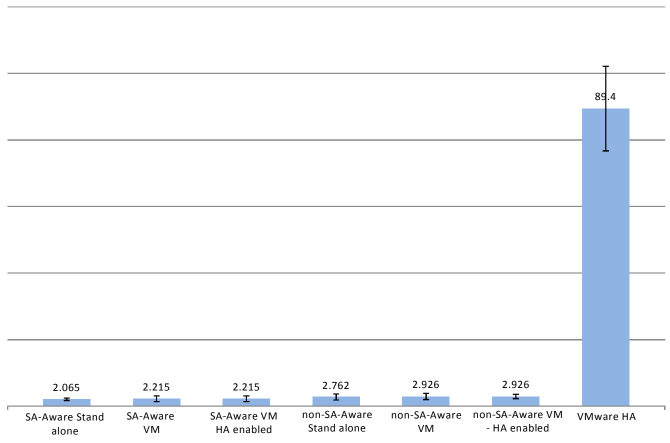

Figure 7. Outage due to physical node failure

TABLE II. REPAIR OF THE FAILED ELEMENT IN DIFFERENT DEPLOYMENTS

	Repair for Failed component	**Repair for Failed VM**	**Repair for Failed Node**
OpenSAF on Standalone machine	Yes	-	No
OpenSAF in VM	Yes	No	No
VMware HA	No	Yes	Yes(restarting the VM on another host)
OpenSAF in VM + HA	***Yes***	***Yes***	***Yes***(restarting the VM on another host)

According to the definition of high availability which allows for only 5.26 minutes of down time, it is unlikely that the VMware HA solution can be considered high available if the VM needs to be restarted at least five times a year; it would already use up this time budget. It can neither detect application failures which happen more often. On the other hand, the VMware FT supports very limited hardware and it does not support VMs with multiprocessors. According to our experiments, one can conclude that the VMware HA cannot compete with the service high availability management provided by OpenSAF.

One measurement we did not cover in the previous charts is the repair time. The failed element is not repaired in all the architectures. TABLE II shows the differences and the repair capabilities of the different deployments. Accordingly only the combined architecture supports the repair of all failure types. The repair of a failed SA-Aware component takes between 0.136 to 0.243 seconds in stand-alone and virtual nodes, respectively, while the repair of a VM or node takes around 100 seconds in the VMware HA enabled deployments. Repairing a failed element is important for high availability because during this time the service is not protected and there is a possibility of failure. Hence it would be nice to reduce the repair time as well.

V. PROPOSED ARCHITECTURES COMBINING OPENSAF AND VIRTUALIZATION

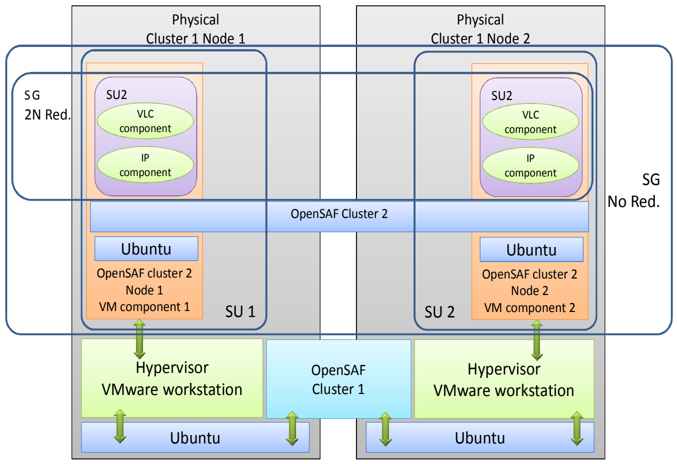

Figure 8. VM availability management in non-bare-metal hypervisor

Our previous virtualization based deployments used VMware HA to manage the availability of VMs. With respect to service outage we saw that it was outperformed by OpenSAF, but it did not repair failed VMs. In addition not all virtualization solutions provide similar mechanism for managing the VM life cycle. Hence, in this section we propose architectures which take further advantage of the different solutions strength and fix their weaknesses. For this we try to use tools and libraries available in other virtualization solutions as well, so that they can be used with other virtualization solutions. In these architectures the main goal is to manage the life cycle of the VM by OpenSAF to reduce not only the outage, but also the repair time of the VM in case of failures, while the services running in the VMs continue providing the services. The first architecture targets non-bare-metal hypervisors while the other targets bare-metal hypervisors.

A. VM availability management in non-bare-metal hypervisor

Most of the hypervisor vendors provide APIs and CLI commands. They can be used to develop tools to control some of the functions the hypervisor provides like managing the VM life cycle. For example VMware provides the VIX APIs. Using these CLI commands we can start, stop, pause and resume the VMs in a hypervisor. Also the non-bare-metal hypervisors (e.g. VMware Workstation) expose a process ID for each VM running in a host operating system. So, the main idea behind this architecture is OpenSAF AMF uses the hypervisor CLI commands to control the lifecycle of the VMs and passive monitoring to monitor the VM process in the operating system. Fig. 8 illustrates this architecture.

There are two different and independent OpenSAF clusters in this configuration. The VMs are considered as non-SA-Aware -non-proxied components in the first cluster. The started VMs form a second cluster running OpenSAF and the target application. When a VM fails, the VM process dies along with it and OpenSAF of the first cluster detects the failure through the passive monitoring. The passive monitoring is started from within the CLI script OpenSAF uses to instantiate the VM component. When a failure is detected OpenSAF runs the cleanup CLI script associated with the VM component and if successful, it re-starts the VM. In the second cluster, OpenSAF detects the VM failure as a node failure, so it fails over any active assignment served by the VM to the other VMs.

We deployed this architecture using VMware Workstation 9.1 as the non-bare-metal hypervisor and installed it on two of our test-bed cluster nodes running Ubuntu 10.04. We also installed OpenSAF 4.2.1 on the same nodes so that it could control the life cycle of the VM running in the VMware Workstation on the same node. Inside the VMs we installed the same Ubuntu 10.04 operating system along with the OpenSAF 4.2.1 and the SA-Aware version of our case study application, the VLC media player. We configured two different and independent OpenSAF clusters which were not aware of each other. The OpenSAF cluster deployed on the physical nodes was responsible for the life cycle of the VMs and the other OpenSAF cluster residing inside the VMs was responsible for the VLC media player as we have seen for VMware HA. From the redundancy model perspective in the first cluster we used the no-redundancy redundancy model and in the second we used the same 2N redundancy model as before.

TABLE III. COMPARISON OF MEASUREMENTS FOR FAILURE OF THE SA-AWARE VLC COMPONENT IN DIFFERENT ARCHITECTURES

	Repair	**Outage**
OpenSAF with no virtualization	0.136	0.055
OpenSAF with ESXi (VMware HA manages the VMs)	0.243	0.081
OpenSAF with VMware Workstation(OpenSAF manages the VMs)	0.848	0.592

TABLE IV. COMPARISON OF MEASUREMENTS FOR VM FAILURE IN DIFFERENT ARCHITECTURES

	Repair	**Outage**
ESXi without OpenSAF (VMware HA manages the VMs)	27.166	99.332
OpenSAF with ESXi (VMware HA manages the VMs)	*107.90*	1.953
OpenSAF with VMware Player (OpenSAF manages the VMs)	*3.73*	3.505

In this deployment, OpenSAF controls the life cycle of the VMs using the "vmrun" command included in the VIX API libraries [12]. We used the start and stop commands respectively for instantiating and terminating the VM, which is configured with these commands as a non-SA-aware-non-proxied component. When the cluster on the physical nodes is started, OpenSAF starts the VMs and when the VMs are up, OpenSAF of this second cluster inside the VMs starts the VLC components and assigns them the active and standby states.

In this setup we experimented with component failure as described earlier by killing the active VLC component as well as with VM failure. In the latter case we killed the process of the VM which hosted the active VLC component. This was perceived as a component failure in the first cluster and resulted in the restart the failed VM. On the other hand, in the second cluster it was detected as a node (VM) failure and OpenSAF failed over the service to the standby VLC component, which resumed the service. Once the failed node was restarted, AMF assigned the standby assignment to its VLC component.

We took the same measurements as for the baseline architectures. TABLE III compares the measurements of the failure of the SA-Aware VLC media player in the different architectures.

TABLE IV shows that the VM repair time is reduced drastically from more than 100 seconds with VMware HA to less than 4 seconds. Note that without OpenSAF it takes VMware HA about 60 seconds to detect the failure and 30 to repair the VM while with OpenSAF it is OpenSAF which reacts to the failure and hence reduces the outage, and the repair is counted from this moment even though it is VMware HA which performs the VM repair. When OpenSAF performs the VM re-start as well, the repair time is improved considerably.

The drawback of this architecture is the increased service outage which we attribute to the additional layer of the host operating system and to the difference between the hypervisors used (ESXi vs. Workstation). We assumed that if we could use ESXi, the delay introduced in this deployment would disappear and we would have the same functionality and better repair time. Hence, we came up with the next architecture.

B. VM availability management in bare-metal hypervisor

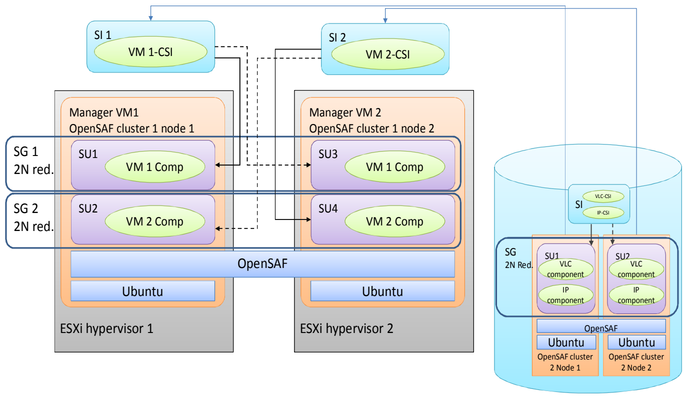

Figure 9. VM availability management in bare-metal hypervisor

In this architecture we have the same two VM images on the shared storage and the two ESXi hypervisors as we had in our baseline architecture of OpenSAF in virtual nodes (Fig. 4). The VMs provide the video streaming service so we call them "Service VMs". Previously, we managed the life cycle of these VMs using VMware HA and we want to replace that by OpenSAF. For this reason we created two new VMs which we call "Manager VMs", which run OpenSAF that manages the following configuration: We defined two SIs each with a CSI which correspond to the two "Service VMs". They can be provided by four components configured in two different SGs with the 2N redundancy model. Each component resides in a different SU meaning we have four SUs (Fig. 9). The SUs are ranked so that the SI assignments are distributed evenly between the two nodes. The components are sets of scripts for starting and stopping the "Service VMs" using the "virsh" command of the libvirt library. Hence, they are non-SA-aware-non-proxied components. The health of the "Service VMs" is checked by external active monitors started with the VMs. They also use libvirt library to get the health status of the VM and report it to AMF. The "Manager VMs" form a separate cluster from the "Service VMs". A possible runtime arrangement is shown in Fig. 9. Manager VM1 contains SU1 with VM1 component having the active assignment for Service VM1 and SU2 with VM2 component having the standby assignment for Service VM2. Manager VM2 contains the SU3 with VM1 component with standby assignment for Service VM1 and SU4 with VM2 component having the active assignment for Service VM2. At the same time the VLC component in Service VM1 has the active assignment and the one in Service VM2 the standby. The Service VMs are started in the hypervisor, which also runs the Manager VMs.

Consider a scenario that the Service VM1 dies. The external active monitor reports it to AMF, where AMF first cleans-up the VM1 component in SU1 by calling its stop script, and then it fails over its CSI to the standby VM1 component in SU3. Thus it becomes active and instantiates the Service VM1 again by calling its start script. As a result both Service VMs are hosted in ESXi hypervisor 2. Meanwhile since Service VM1 hosted the active VLC component for the streaming service it is failed over to the VLC component in Service VM2 because the second OpenSAF cluster detects the node failure; so the service continues. Once VM1 component becomes available OpenSAF will switch back the SI representing Service VM1 to its preferred node. This way we avoid the situation that one ESXi node holds both Service VMs for extended period of time. We could configure the VM components to be restarted rather than failed over, but in this case the outage will be the entire VM restart time – approximately 4 seconds, while our earlier measurements predict that the failover takes only 0.081 seconds. An added benefit of this architecture is that it is not limited to VMware ESXi and can be deployed in the hypervisors which supports libvirt like Linux KVM, Xen, etc.

VI. Related work

Most of the work done for high availability in virtualized environment targets VM and hardware failures and does not cover application failures. HAVEN [13] uses the VM states defined by DMTF with other additional new states. They defined a set of HAVEN levels 1-4 (availability levels) which are achieved by a set of recovery state transitions. Loveland et al [14] try to use virtualization for cost reduction and resource consolidation of traditional HA approaches like active/active and active/passive. In [15] Braastad introduced a service for providing high availability using virtualization. In this work, an add-on to Heartbeat – an open source software package used to create high availability clusters – is developed, allowing Heartbeat to seamlessly migrate the VMs between the physical nodes when shut down gracefully. The emphasis is on graceful failures and the solution does not support uncontrolled failures like power, hardware or network failures. Remus [16] uses concepts similar to VMware FT. It is implemented for Xen and creates an exact standby of the primary VM. It focuses on fail-stop failures and does not support the recovery of software failures or non-fail-stop conditions.

VII. Conclusion

In this paper our goal was to address highly availability in a virtualized environment. We selected OpenSAF, which is an open source high availability solution and VMware as the virtualization solution, and in particular VMware HA. Our initial experiments showed that VMware HA was not as responsive to failures as OpenSAF, which is essential for service high availability. Also, the two solutions handle different sets of failures. To combine their features and take advantage of their strengths we designed two new architectures that used non-bare-metal and bare-metal hypervisors to address the identified shortcomings. In these architectures we used OpenSAF for two purposes: To manage the availability of the VMs and to make the service running in the VMs highly available. So far, we have implemented and experimented with the architecture that uses a non-bare-metal hypervisor. The results showed some improvement in comparison to the baseline architectures, but other characteristics suffered from the change of the hypervisor. We believe that this can be eliminated by a different setup, which again allows the use of a bare-metal hypervisor. We are in the process of implementing this solution.

Acknowledgement

This work has been partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Ericsson Research.

References

[1] M. Toeroe and T. Francis, *Service Availability: Principles and Practice*. Wiley, 2012.

[2] Service Availability^{TM} Forum, "Official homepage" [Online]. Available: http://www.saforum.org.

[3] Service Availability^{TM} Forum, "Service Availability Interface, Overview, SAI-Overview-B.05.02"

[4] Service Availability^{TM} Forum, "Application Interface Specification, Availability Management Framework, SAI-AIS-AMF-B.04.01"

[5] The Open Service availability Framework, "Official homepage", [Online]. Available: http://www.opensaf.org.

[6] VMware Inc., "Virtualization Overview", pp. 1–11, 2006.

[7] VMware Inc., "Official homepage" [Online]. Available: http://www.vmware.com.

[8] D. Epping, et al., *VMware vSphere 5 Clustering technical deepdive*. 2011.

[9] D. J. Scales, et al., "The Design of a The Design of a Practical System for Practical Virtual System for Machines Fault-Tolerant Virtual Machines", *SIGOPS Operating Systems Review*, vol. 44, no. 4, pp. 30–39, 2010.

[10] A. Kanso, et al., "Integrating Legacy Applications for High Availability□: a Case Study"*, proceedings of IEEE HASE* 2011, pp. 83–90.

[11] N. Huber, et al., "Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments", *proceedings of CLOSER 2011*.

[12] VMware Inc., "Using vmrun to Control Virtual Machines", 2012.

[13] E. Farr, et al., "A Case for High Availability in a Virtualized Environment (HAVEN)", *proceedings of ARES,* March 2008.

[14] S. Loveland, et al. "Leveraging virtualization to optimize high-availability system configurations", *IBM Systems Journal*, vol 47, no. 4, 2008.

[15] E. Braastad, "Management of high availability services using virtualization", University of Oslo, 2006.

[16] B. Cully, et al, "Remus□: High Availability via Asynchronous Virtual Machine Replication", in *proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008.