Towards High-Availability for IP Telephony using Virtual Machines

Devdutt Patnaik, Ashish Bijlani
College of Computing, Georgia Institute of Technology
Atlanta, GA, USA
{devdutt.patnaik, abijlani3}@gatech.edu

Vishal K Singh
NEC Labs
Princeton, NJ, USA
vs2140@columbia.edu

***Abstract*—Virtualization technology is popular today for hosting Internet and cloud-based compute services. Recently, the IP Communications domain has seen the adoption of virtualization with enterprise telephony solutions being hosted in virtualized environments. A deployment scenario that is common in the IP communications domain is the virtual appliance. A virtual appliance encapsulates the IP Telephony application, such as a VoIP Call/Proxy Server, and the OS software, within a virtual machine. This allows for a single physical machine to host a number of virtual appliances – each being a telephony application deployed on an OS configuration for which it has been optimized. Apart from the well known benefits of virtualization such as server consolidation and improved resource utilization, a promising capability of virtualization technology is that of live virtual machine migration. In case of failure, a live virtual appliance can be efficiently and transparently migrated to a different physical machine with minimal impact on clients using the hosted IP communication service, thereby providing high-availability and fault tolerance. This work studies the feasibility of deploying IP communications infrastructure on virtualized platforms, with high-availability and reliability as a goal. We model a real IP telephony workload to understand the performance implications of such a deployment, and the effectiveness of these high-availability mechanisms. We use quantitative methods to measure performance such as call throughput, delay and packet loss. By understanding the characteristics of IP telephony workloads, we identify specific aspects that affect performance and also identify some optimal configurations. The results presented in this work will be useful for telecommunication service providers to understand the benefits and limitations of such a deployment.**

Keywords: Virtualization; High-Availability; IP Telephony workloads; Virtualization overheads.

I. INTRODUCTION

Virtualization technology has improved significantly in the recent past and is now pervasive in data centers and cloud-based service infrastructure. With the continuous evolution and improvements in virtualization technology, its adoption in various domains continues. Telecommunication service providers have begun deploying IP services on virtualized platforms thereby providing cost-savings, flexibility and ease of deployment as key benefits. However, server-consolidation comes with the risk of increased exposure to hardware failure. Hardware failure can cause disruption in service for a number of IP servers all at the same time. To solve this problem, virtualization platforms now provide a feature called live migration that supports migration of actively executing VMs from one physical machine to another with minimal disruption in service. This capability allows administrators to provide high-availability and fault-tolerance. Moreover, software applications do not need to implement failover and recovery logic. Traditionally, high-availability comes with overheads in terms of performance, resource redundancy and higher cost of deployment. IP communications workloads are sensitive to overheads that increase latency and packet-loss. It is therefore particularly interesting to measure the performance implications of hosting such applications in virtual environments when employing high-availability mechanisms.

We propose the use of virtual machine technology to deploy reliable IP Communications infrastructure. Our main contributions are:

a) We propose an architecture for the deployment of IP communications infrastructure on virtualized platforms with high-availability as a goal. This includes reliability for both signaling and media. We explore high availability and fault-tolerance provided by Xen[3] using a live migration solution.

b) We perform a feasibility analysis of such a deployment by measuring performance overheads because of reliable virtualized infrastructure, with real workloads.

c) We study failover effectiveness of the proposed architecture using quantitative metrics to measure how well the platform handles failover.

d) We present deployment configurations that achieve optimal performance for IP communications.

Our results and experimentation will be useful to telecommunication providers who are considering deployment of IP communications infrastructure on virtualization platforms and would like to understand the capabilities and implications of the underlying platform to achieve high-availability.

The rest of the paper is organized as follows: Section II discusses the background and motivation for our work. Section III introduces high-availability for virtual machines, and provides details of the approaches used such as live migration and continuous check-pointing. Section IV provides details of

proposed reliability architectures and how we modeled our experiments to measure performance of the deployment under IP telephony workloads. Section V presents the results, analysis and how we arrive at optimal settings for reasonable performance. We conclude with Section VI.

II. Background And Motivation

Reliability for Enterprise IP telephony has been studied and solutions have been proposed. The work by K. Singh et. al. [5] discusses approaches to achieve failover and scalability using a 2-stage architecture where the first stage uses DNS records for load distribution. The request arrives at one of N primary servers based on a load distribution policy specified using DNS SRV records. The first stage then routes requests to a second stage cluster using a hashing algorithm. The second stage comprises of clusters of proxy servers where each cluster handles a certain space of the hash-table. Each cluster has a primary server and a secondary server where each server maintains a separate database. In this architecture, reliability and failover within a cluster uses techniques such as IP address takeover, client's abilities to register with multiple call servers, or database replication where the primary database is synchronized with the secondary by sending updates. All of these techniques are not transparent to the client or the server components. Support is required to be added to the clients or the servers for the failover to work. If the primary were to fail before a new client REGISTER message was updated in the secondary database, the client will be unable to make calls until a REGISTER refresh is sent. This work does not address media server reliability which is used in many deployments (e.g., media relay). It also does not replicate the in-memory state of the proxy servers which maybe important in some situations. Our proposed deployment using virtual machines can be used in conjunction with their proposed architecture to provide even higher guarantees in a completely transparent manner and with significant cost-savings.

A. Fessi et al [6] propose a hybrid approach for SIP server reliability where they use SIP proxy servers as the primary service for locating clients, and fall back on a peer-to-peer lookup-up approach when the primary fails. They however do not provide any results from their evaluation. They also do not cover media servers or replication of in-memory state.

D. Sisalem et al [7] show that more than 40% of failures in IP communication services are caused due to hardware and software failures. We primarily address hardware failures and operating system failures.

In our proposed deployment, we aim to address the limitations stated above. Additionally, this work studies the performance implications of such a deployment. Prior work done in [4] has shown that virtualized hosting impacts performance. IP telephony workloads have not been studied in a virtualized environment with reliability features. While Cully et. al. [1] study workloads such as SPECweb and Linux kernel compilation, real-time workloads have not been studied. Our goal is to study the performance implications of such a hosting. We identify the benefits and drawbacks of such a deployment. We address the study of reliability for both signaling and media servers, and by using experimentation, identify deployment configurations that provide optimal performance.

III. High Availability Using Virtual Machines

As virtualization technology continues to mature and improve, it continues to provide benefits over and above server consolidation such as security, live VM migration etc. A background in virtualization is provided in [3][4]. Due to the narrow interface between the VM and the VMM, a VM can be efficiently and transparently migrated to another physical host. The entire in-memory state can be transferred in a consistent manner. This applies to network connections and application-level state. Disk replication is done using shared storage. The clients that were serviced by the virtualized server VM continue to use the service after the VM has migrated. This can be achieved with small or negligible down times. IP endpoints that are involved in call-setup or are currently in active calls continue to function without any significant degradation in quality of service. This migration technique offered by the VMM can be exploited to provide high-availability and fault-tolerance. These high-availability solutions are currently available on virtualization platforms such as VMware and Xen. VMware provides this using vSphere. Xen provides this capability using a solution called Remus[1]. We chose to evaluate the open-source Xen VMM [3] for our experiments.

We discuss how XEN's high-availability solution works. A pair of servers run in an active-passive configuration. The basic stages of this approach are:

1) Stage 0 – Pre-copy

In this stage, when the primary VM is started, a secondary VM is created which is a paused replica of the memory state of the primary VM. This copy of the primary is incrementally updated during the subsequent checkpoint stages. The secondary VM does not execute on the backup host until a failure occurs. This consumes a relatively small amount of the backup host's resources allowing it to concurrently protect VMs running on multiple physical hosts in an N-to-1 configuration.

2) Stage 1 - Checkpointing

In this stage, using an epoch-based system, the execution of the primary VM is bounded by brief pauses in execution during which the changed state of the VM is atomically captured. The down-time during this stage is a function of the memory in use by the VM and its writable working set [2], which is the set of pages repeatedly dirtied during VM execution. Using the approach of pipelined checkpointing, performance gains can be achieved. This checkpointing is done once per epoch.

3) Stage 2 – Transmit Buffered State

The checkpoint buffer from Stage 1 is transmitted and stored in memory on the secondary host. A point to note is that after Stage 1 is completed, the primary is allowed to continue execution speculatively. Therefore runtime performance is achieved. This allows the primary server to remain productive while synchronization with the secondary server is performed asynchronously.

4) Stage 3 – Synchronization

Once the checkpoint buffer has been received on the secondary host, it is synchronized with earlier state, and the checkpoint is acknowledged to the primary.

5) Stage 4 – Output Release

In order to permit speculative execution and still ensure consistency in case of failure during a checkpoint phase, the network output resulting from the current speculative execution round is not released to clients outside the primary server. Once stage 4 is completed, the buffered network output is released. While this approach affects latency sensitive workloads, it ensures stronger consistency guarantees by not making system state externally visible until the corresponding system state that produced it has been replicated.

This approach uses a heartbeat mechanism, wherein the failure to receive a heartbeat from the primary host results in the activation of the backup VM on the secondary host. The backup VM is unpaused, and using an unsolicited ARP reply, the network traffic from clients is redirected to it. This ensures seamless fail-over.

IV. Reliability Architecture for IP Telephony using Virtual Machines

K. Singh et al [5] proposed an architecture for failover and load-sharing for SIP servers. We propose taking a similar approach but with virtualized hosting of servers within the second stage cluster, with high-availability capabilities enabled. The benefits of such a deployment are efficient failure detection of the primary using the heartbeat mechanism, automatic and transparent failover to the secondary, and protection capabilities for both signaling and media server state. At the same time this deployment can utilize the benefits of server consolidation. Figure 1 shows a stateless SIP server (which is selected from a pool of servers using DNS SRV) distributing load between two primary SIP/Media servers collocated on the same VM. The primary VMs are fail-safe by copying memory state on the back up VM which is in paused state and will be triggered once a heartbeat is missed. Primary servers can be added elastically as load increases. Single physical server can host many back up VMs for many primary servers (n x 1 or n x m back up).

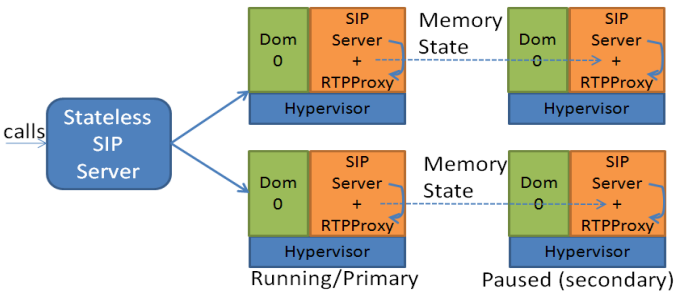

Figure 1 Deployment Model

Cherkasova et. al. [14] shows that I/O affects the domain 0 CPU utilization using web server workloads. Even for VoIP, this needs to be considered for consolidation decisions. Here, we investigated how much overhead is incurred because of media and signaling traffic on domain 0 under both server consolidation only (Xen, No Remus) as well as reliability with consolidation settings (Xen with Remus), with both UDP and TCP traffic (for signaling).

A. Reliability vs Performance

Remus operates in 2 modes: Net and No-Net. In "Net" mode Remus buffers the network output during the time when the checkpoint is transmitted to the secondary server. This is done to ensure that the client view of the primary server is consistent in case there is a failure when the checkpoint is being performed on the secondary. Since the network output has not been released, a hardware failure will simply lead to these buffered packets being discarded, and the secondary will take over from the previous snapshot of the primary. While this provides for high consistency guarantees, it results in additional latency which is directly proportional to the checkpoint interval. The "No-Net" mode does not buffer network output between 2 successive checkpoint operations. This has the benefit that latency sensitive workloads will see significant performance improvements compared to the "Net" configuration. The side-effect of this mode is that while we aim to minimize delays, when failure and subsequent fail-over occurs, there is a chance that there will be some inconsistency in the client view of the primary server and the state in the secondary server after take-over occurs. Typically this state inconsistency is fairly negligible in the case of stateless servers and is bounded by the checkpoint frequency that Remus is configured to use. It is typically less than 100ms. We evaluated both deployment modes to evaluate implications on reliability and performance.

B. IP Telephony Workloads

SIPstone [12] provides us with design considerations for implementing SIP performance benchmarks. SIPp [10] gives us the capability to implement these benchmarks and to characterize IP telephony workloads. In order to accurately model real-world workloads in telephony environments, we used carrier-grade servers and clients. The key components of an enterprise telephony system are the signaling servers, the media servers and the SIP clients. We used OpenSips[8], which is a high performance SIP proxy server. The primary role of this server was to handle client registrations and perform call control. Media server can be used for media relay or as voice mail server, IVR server, and transcoder or conference mixers. We used RTPProxy [9] serving as a media relay for our experiments. OpenSIPS can work together with RTPProxy. The goal was to be able to understand the effects of basic media processing in virtualized environment. We used SIPp [10] for generating SIP and RTP traffic. SIPp can be used to program various call scenarios, with the ability to stream media, following call-setup. SIPp allowed us to vary the workload and obtain statistics for increasing load.

V. Experiments And Analysis

In this section, we first describe the hardware and software platform used for the experiments. Then we discuss the experimental configuration and the results.

A. Hardware and Software Platform

The choice for the hardware platform was based on real-world hardware configurations that are typically chosen for deploying telephony infrastructure and VoIP applications such as media and signaling servers. The server hardware ran on an Intel Core 2 Quad processor with each core running at 2.5 GHz, 8 Gbytes of RAM and two 1 Gb NIC cards. 4MB of L2 cache was available. The open-source Xen 3.2.1 was used as the Virtual Machine Monitor for our experiments. The SIP Proxy and the Media server were co-located in the same guest VM also called DomU. The guest VM ran paravirtualized Linux 2.6.18. The guest VM was given 2 vCPUs. Since the machine had 4 physical cores, by default Xen assigns 4 vCPUs to privileged domain or Dom0, so that it has the ability to use all the cores. The guest VM was assigned 1GB of memory. Dom0 was given the residual memory after the other VMs had been assigned their memory quotas.

B. Experimental Configurations

Our experiments were targeted at first understanding the impact of hosting IP telephony applications on virtualized servers with high-availability capabilities. We experimented with various loads with Remus configurations to get a good idea of performance and reliability metrics. We ran our experiments with the in-memory database configuration for SIP server. This was done to explore the performance overhead by the reliability mechanism given that Remus provides high guarantees for memory consistency on failure. We also disabled authentication. This section provides details of the experiments carried out. We used the "Net" mode (section IV A) for the first set of experiments since it guarantees against loss which is desirable for user-registration and call-setup operations. The checkpointing interval for Remus was set at 100ms.

1) User Registration Performance:

We first performed experiments with increasing loads for user registrations i.e., SIP REGISTER messages. For these experiments we first ran experiments without high-availability enabled i.e., without Remus protection. The experiments were carried out using UDP and TCP as underlying protocols. We also used a single TCP connection for all REGISTERs as well as a new TCP connection for each REGISTER message. Figure 2 shows the CPU overheads of the virtualized platform without Remus. This was done to validate the findings of [14] and see the overheads with SIP signaling workloads.

We plot the CPU overheads on Dom0 and the guest VM (which is hosting the Call Server) with increasing load i.e., REGISTER requests per sec. The experiments shown are for TCP and UDP as the underlying protocol. For TCP experiments, we collected data when using a single TCP connection for all registrations (TCP_1 in graph) and when using individual TCP connections for each registration (TCP_n).

From Fig 2, we can see that UDP and TCP_1 behave similarly in terms of overhead on Domain 0. TCP_1 causes more CPU utilization on the guest VM (call server application VM) which is as expected. Also we can see that with one TCP connection per register, the overhead on Domain 0 increases by almost 5 times. We note that the number of packets for SIP REGISTER when using 1 connection per register increases because of TCP handshake. Even for UDP based registration, we see that overhead on Domain 0 is proportional to number of registrations per second. Finer model needs to be built to exactly measure the overhead with number of packets.

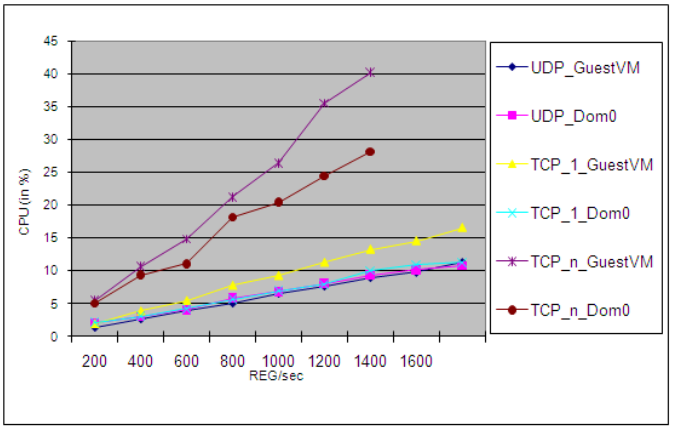

Figure 2 REGISTER Signaling performance w/o Remus

The next experiment was carried out by enabling high-availability using Remus and increasing REGISTER loads.

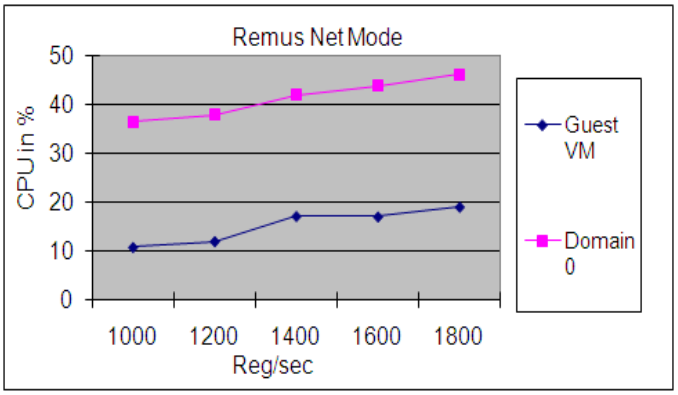

Figure 3 REGISTER Signaling performance with Remus

From Figure 3, we can observe that with Remus we have higher CPU overheads due to the processing needed to perform the continuous and asynchronous checkpointing to the secondary host (back up VM). With increasing loads from 1000 to 1800 registrations per second, the CPU utilization for the Dom0 goes from 36.6 to 46.3 and for Guest VM from 10.9 to 19.2 percent. Overall, we observe that CPU overhead is incurred and increases with in-memory state. We forced a failure during this experiment when we ran the 1400 registrations/second with UDP and found that the secondary VM takes over seamlessly with 100% success of all registrations as per the statistics from SIPp. This shows that with Remus Net mode, for signaling we can get all registrations seamlessly transferred to secondary VM.

2) Call-Setup Performance with Media:

In this set of experiments we use UDP to setup calls with media streams being relayed by the Media Server. For this

experiment we set-up 800 simultaneous calls and measured call completion results and losses in media packets. This configuration created significant load on the servers with CPU utilization reaching 77% on Dom0 and 33% on the guest domain. We were able to perform 100% failover of all calls (signaling) in progress. We noticed losses in media streams. This was due to heavy load of 800 streams. We then scaled the experiments for media from 100 streams to 800 streams and noticed losses. This is shown in the form of box plots in Figure 4. The experiments show the *delays* in milliseconds and *% loss*, as load increases – 100, 200, 400, 600 and 800 streams. We plot the max, min, median, first(q1) and third quartile(q3). The results show that for smaller workloads of 100 streams, Remus provides good reliability and performance.

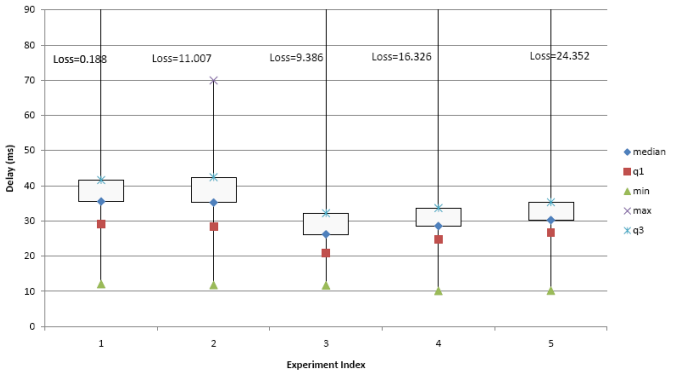

Figure 4 Media stream plots in Net Configuration

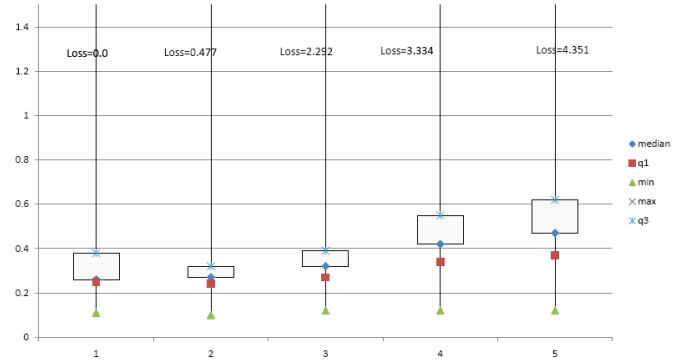

Figure 5 Media stream plots in No-Net Configuration

From the results in Figure 4, we observe that due to overheads of continuous checkpoints and virtualization related overheads, there were significant losses when using Net mode. No-Net mode (Figure 5) allows for higher performance with weaker consistency leading to possibility of packet loss during failover.

The results for the same experimental configurations from Figure 4 gave significantly better results as shown in Figure 5. ***We conclude that using No-Net mode for media streams gives us a balance between performance(loss and delay) and reliability(failover) while still being able to migrate 100% of all calls in progress (using TCP) which is a significant result.***

We can achieve approximately 2% losses for 400 media streams and 3% for 600 streams. From experiments in [4], [17], these losses can be further reduced by tweaking scheduler parameters such as weights for Dom0 since Dom0 is usually the bottleneck for I/O intensive workloads.

3) Summary of Experiments :

From the experiments conducted in Section V, we can conclude the following:

1. Our proposed virtualized deployment provides high guarantees for signaling while providing for good performance and throughput. From our signaling experiments we observed that 100% user registrations and call-setup requests were successful with the help of minimal support in the clients for retry mechanisms.
2. Migration of user registration and call-setup operations was successful with SIPp reporting 100% success in spite of induced failure during these operations.
3. While signaling operations were completed reliably in the Net configuration, high delays and losses were observed for media streams.
4. No-Net mode for the Media server deployment provides significant improvement in performance: loss and delay reduces significantly. While the No-Net configuration performs better for media, it may not provide call completion guarantees during the fail-over operation for signaling.
5. From 3 and 4 above, an optimal deployment could be to use a Net mode deployment for the Signaling server and No-Net mode deployment for Media servers.
6. At this point, Remus only protects a single VM on any host machine (This limitation will be addressed in future releases).Therefore, if we require overall guarantees of full-system protection, as well as high performance, it is recommended that we use separate physical hosts to deploy the Signaling and Media VMs. In this deployment configuration, we should enable Net mode protection for the Signaling server and No-Net mode protection for the Media Server. When hosted separately we can also increase individual load on the Signaling and Media VMs. This deployment gives the best overall results yielding optimal performance, reliability and deployment cost.

C. Positioning of Signaling and Media Servers

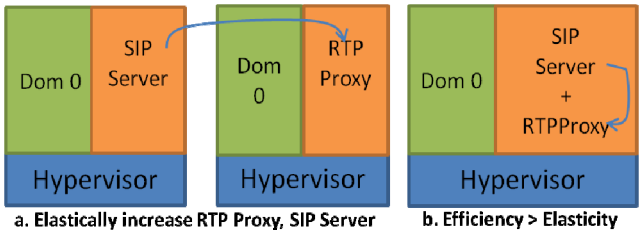

Figure 6 Positioning SIP Server, RTPProxy on VMs

The SIP Proxy server and the Media server can be placed within the same VM (b in Fig 6) or in separate VMs (a in Fig

6). The operator can place Signaling and Media servers on separate physical machines where each server is separately protected by using Remus, Signaling server in 'Net Mode' and Media server in 'No Net Mode'. In this way, the operator can elastically increase the number of SIP server or Media server independent of each other. The disadvantage of this approach is possible cost of additional machines because of I/O overheads incurred by domain 0 because of inter-vm communications. Hence, this elasticity to handle load is achieved at the cost of efficiency. On the other hand, collocating SIP and RTPProxy server is efficient but it is difficult to provision for dynamic workloads. Our experimental approach can be used in making these decisions based on the performance of each configuration.

VI. CONCLUSIONS

Server consolidation using virtualization is gaining significant adoption in Internet hosting centers and cloud-computing service centers. In addition to benefits of virtualization such as server-consolidation and greener solutions, virtualization technology is evolving at a fast pace, hence providing valuable services such as high-availability and fault-tolerance. In this work we evaluated the feasibility of bringing high-availability to the Enterprise IP Telephony domain when the services are hosted in virtual environments. We show that virtualization can be combined with existing scalability and reliability architectures [5] for both signaling and media servers. Using an experimental and iterative approach we identified the benefits and limitations of such a deployment. We conclude that currently available solutions such as Remus can be tuned appropriately for reasonable performance. In particular we observe that (I) Media VMs are heavily penalized when configured to run in network buffering mode (Net Mode). At the same time they are more tolerant to minor packet losses during fail-over. Therefore they are better candidates to execute with network buffering disabled. Also hosting them on a separate physical host can deliver additional performance. (II) Signaling VMs are more tolerant to latency but cannot afford to incur packet losses as this results in reduces QoS for end-users. Therefore such VMs greatly benefit from the network buffering mode (Net Mode), as it provides higher reliability and stronger consistency guarantees during fail-over. (III) We conclude that virtualization solutions provide high availability for IP telephony and live migration of calls can be achieved with 100% guarantees with respect to signaling. (IV) We measure the overhead on the domain 0 because of signaling and media workload and believe that fine tuned model is needed to appropriately provision the guest VM and domain 0, as well as to make consolidation (co-location and VM migration decisions).

We used experiments to propose the best configurations keeping the balance between performance, reliability and deployment-cost. We believe that the results of this work will be very useful for telecommunication solution providers who wish to deploy their applications in virtualized environments and wish to exploit the benefits of high-availability features using commodity hardware.

REFERENCES

[1] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson and A. Warfield, Remus: High availability via asynchronous virtual machine replication. In Proceedings of the 5th Conference on Symposium on Networked Systems Design & Implementation (San Francisco,CA, USA, 2008), USENIX Association.

[2] C. Clark, K. Fraser, S. Hand, J. G, Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines. In Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation (Berkeley,CA, USA, 2005), USENIX Association.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In Proceedings of the Symposium on Operating Systems Principles (SOSP), Oct. 2003.

[4] D. Patnaik, A.S. Krishnakumar, P. Krishnan, N. Singh, S. Yajnik, "Performance Implications of Hosting Enterprise Telephony Applications on virtualized multi-core platforms," IPTComm 2009.

[5] K Singh, H Schulzrinne, Failover and Load Sharing in SIP Telephony, International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Philadelphia, PA, July 2005.

[6] Ali Fessi, Heiko Niedermayer, Holger Kinkelin, and Georg Carle. A cooperative sip infrastructure for highly reliable telecommunication services. In IPTComm '07: Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications, New York, NY, USA, 2007.

[7] Sisalem, D., Ehlert, S., Geneiatakis, D., Kambourakis, G., Dagiuklas, T. Markl, J., Rokos, M., Boltron, O., Rodriquez, J., Liu, J., "Towards a Secure and Reliable VoIP Infrastructure", CEC Project No. COOP-005892, April 30, 2005.

[8] OpenSIPS http://opensips.org/

[9] RTPProxy http://www.rtpproxy.org/

[10] SIPp http://sipp.sourceforge.net/

[11] http://transnexus.blogspot.com/2008/09/openser-and-rtpproxy-performance-test.html

[12] SIPstone - Benchmarking SIP Server Performance

[13] Characterization & Analysis of a Server Consolidation Benchmark – VEE '08, Padma Apparao, Ravi Iyer, Xiaomin Zhang, Don Newell and Tom Adelmeyer.

[14] L. Cherkasova and R. Gardner. Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. In USENIX Annual Technical Conference, Apr. 2005.

[15] E. Ackaouy. [Xen-devel] New CPU scheduler w/ SMP loadbalancer.http://lists.xensource.com/archives/html/xen-devel/2006-05/msg01315.htm%l.

[16] H. Raj and K. Schwan. High performance and scalable i/o virtualization via self-virtualized devices. In Proceedings of the 16th International Symposium on High Performance Distributed Computing, June 2007.

[17] Lee, M., Krishnakumar, A. S., Krishnan, P., Singh, N., and Yajnik, S. Supporting soft real-time tasks in the xen hypervisor. VEE (2010).