LUCID: ‘See what you pay for’

Raviraj Joshi, Sagar Hotchandani, Varun Wadhwa, Bhargav Patel

Swapnil Pimpale, Mandar Naik

[#]*Pune Institute of Computer Technology, Information Technology Department*

raviraj.j1991@gmail.com, sagar.hotchandani@gmail.com,
varun.p.wadhwa52@gmail.com, brmonpara@gmail.com,
pimpale.swapnil@gmail.com, mandarn25@gmail.com

***Abstract*— Cloud Computing, one of the emerging trends provides IT services through virtualized resources to its tenants and charges them on the basis of resource utilization. Due to lack of transparency in the activities taking place on the virtual machines at service provider’s end and various security issues, users hesitate to move their infrastructure to Cloud. The users do not have a clear idea of what they are being charged for. In order to get the required information, users can track the activities on their virtual machines by using third party network monitoring tools. However these tools hog a significant amount of memory which affects the performance of other critical processes running on the VMs. Hence, we propose LUCID, a solution for monitoring network activity of VMs. LUCID will reside in Dom0 which is a controller OS of Xen Hypervisor, leading to centralized monitoring of multiple VM’s.**

LUCID provides kernel level packet capture mechanism and archiving of the captured data. Efficient Indexing Mechanism has been built to facilitate faster retrieval of this data when queried. The results are presented to the user via a SaaS interface (Software as a Service) where user can specify various queries and view corresponding results. This data helps the users to monitor network activity and also view the recorded traffic which provides the much needed transparency regarding network activity especially during the dormant period. Hence the user gets a clear idea of what he is being charged for.

***Keywords*— Xen Hypervisor, Dom0, DomU, PF_Ring, Bitmap Indexes, Cloud Computing, SaaS.**

I. Introduction

Cloud computing consists of hardware and software resources made available on the Internet as managed third party services. These services typically provide access to advanced software applications and high-end networks of server computers. There are three types of cloud services: SaaS, PaaS, and IaaS.

A. IaaS

This is essentially equivalent to providing user a "physical server box". An example of this would be going to RackSpace or SoftLayer and leasing a physical box from them. The vendor manages the networking, hard drives (if they fail), hardware of the box, virtualization O/S (if the box is virtualized).

B. PaaS

PaaS provides computing platform and solution stack as a service. PaaS is“hosted” framework/application/tools that you can leverage to build something on. An example of PaaS would be Windows Azure (excluding the VM role) services like web role, worker role, reporting services, etc.

x. SaaS

Here the unit we are gaining is business functionality. For example, Gmail is a type of a SaaS mail provider because you don't have to manage any service yourself and it's all done by the vendor (Google in this example). This is the most common form of cloud computing.

Cloud service providers provide VMs to customers (customers are unaware of virtualization and are provided with an abstraction of physical server) as a part of IaaS. LUCID primary deals with monitoring network activity of requesting customers and displaying the same to the customers using a SaaS interface. Customers desire transparency regarding activities taking place on their VMs to see what they are actually being charged for. The conventional approach is to use network monitoring tools such as tcpdump or Wireshark on each VM. However, running these on each VM degrades the performance.

II. MOTIVATION

There are a number of open source network monitoring tools available such as tcpdump [viii], Wireshark etc. which provide live capture and offline analysis and allowing captured network data to be browsed via a GUI. However, in case of cloud infrastructure these benefits come at a large cost, cost in terms of memory and performance of each VM running these tools. They hog a significant amount of main memory of the VMs and this memory requirement can become a bottleneck. This may also impact the performance of other critical processes running on each VM. Also running these tools requires root privileges which is normally not given in cloud infrastructure for security reasons.

LUCID serves as a simple and elegant solution for the above problem, providing centralized network monitoring in Hypervisor (Dom0) to monitor traffic for all VMs, with benefits like reduced rate of dropped packets and reduced response time to satisfy queries on the captured data.

The System will provide the much needed transparency [ii] required by the users to verify what transactions they are bearing the costs for. System will provide data archiving useful in tracing past activity and post-mortem analysis of data.

III. BACKGROUND

A. Networking in Xen

We take Xen hypervisor [i] as an example to explain our whole system. Xen, originated as a research project at the University of Cambridge, is the powerful open source industry standard for virtualization .Today, Xen hypervisor is becoming the fastest and most secure infrastructure virtualization solution. It supports a wide range of guest operating systems including Windows, Linux, Solaris and various versions of Free BSD. A Xen system has multiple layers, the lowest and most privileged of which is Xen itself. Xen in turn may host multiple guest operating systems, each of which is executed within a secure virtual machine (in Xen terminology, a domain). The first domain, domain0 (Dom0) which is the Host OS while remaining guest OS's are termed DomU's. The Dom0 along with the Xen hypervisor are responsible for the whole system and management of guest OS's i.e DomU's.

Xen uses split device driver model wherein real driver exists in dom0 operating system. Dom0 is also responsible for handling administrative tasks. The split driver is designed to move data from the domU guests to the dom0, usually using ring buffers in shared memory. So to facilitate networking [vii] in guest OS, the front end interface ethN (virtual network interface creating illusion of a real one) is present which is connected to backend interface in dom0 OS named vif M.N.

So dom0 will contain the real interface peth0 and virtual backend interface vif.

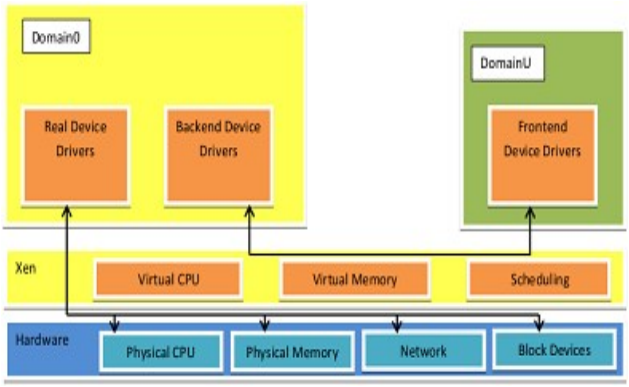


Fig.1. Split Device Driver in Xen

B. *PF_RING*

PF_RING [v] protocol handler is used to efficiently move the packets from the Network Interface Card (NIC) to a ring buffer present in the kernel space. It resides at layer 2 of the OSI model and captures raw packets from the network interface by placing hooks in kernel [vi]. The Packet Capture module is a kernel space module which will get packets from PF_RING kernel module i.e from the ring buffers of PF_RING which requires some changes to PF_RING code. Fig.2. shows how libpcap library uses PF_RING to speed up its capturing process. While writing the captured packets to disk, pcap application used fwrite () call. This involves copy of data from userspace to kernel space buffer and multiple context switches from user to kernel and kernel to user space in the process of ultimately writing data to disc. LUCID minimizes this overhead by directly writing data to disk from kernel space without bringing userspace in picture. As we are not presenting data to dom0 user, we do not bring data to userspace.

C. *Bitmap Indexes*

Bitmap Indexes are used with substantial success in the areas of databases and information retrieval. The Basic concept of Bitmap Indexes is that we need to keep ‘n’ bitmaps i.e. columns, where each column represents a value which can be taken by the attribute for which the Bitmap has been created. They are suited for applications involving network packet data for the following reasons:

(i) Maintaining Packet-traces is a read-only activity where data is only stored not modified; Bitmap Indexes are suited for such data.
(ii)Bitmap Indexes work well for numerical data and most of the fields within the packets are numerical.
(iii)Insertion rates are very high for Bitmap Indexes which is required for indexing packets arriving at high speeds.
(iv)Moreover Bitmap Indexes can be compressed and these compressed forms itself can be used to handle existence and cardinality queries without referring to the Packet trace which has been stored.

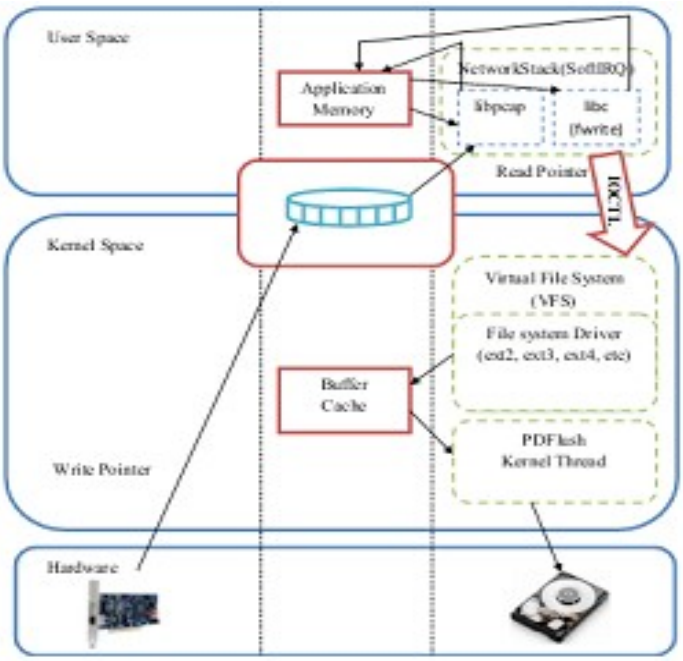


Fig.2.User Space Capture Process using PF_RING

IV. ARCHITECTURE

Cloud service user will have to specify the list of VM’s to be monitored and capture time filters to be applied which marks input to our system.

We divide our architecture into 3 modules:

(i) Kernel level packet capture

(ii) Archiving and indexing module
(iii) SaaS interface

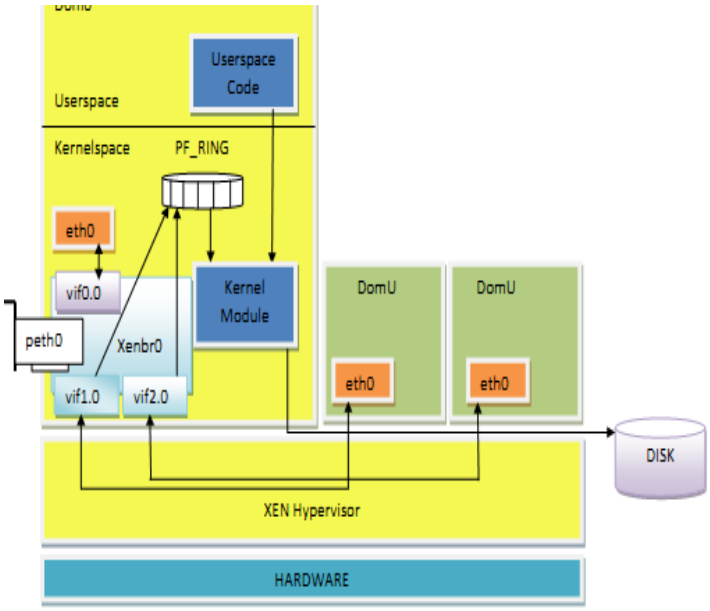
A. *Kernel Level Packet Capture*

All networking activity for guest VMs in Xen hypervisor is routed via Dom0 kernel(controller OS), so packet sniffing in Dom0 is a natural choice in virtualization environment supported by Xen. With a huge number of VMs residing on hypervisor, the amount of network traffic created will be huge. The processing speed of NIC cards used for receiving and transmitting packets will also be very high. So using pcap which is essentially a userspace library leads to substantial loss of packets. This compelled us to design a kernel space capture module. A kernel level packet capture module reduces:

1. The transactions involving kernel space to user space copy of data (prevalent in pcap).
2. The overhead involved in making system calls and user space to kernel space switch thus saving CPU time.

Fig.3. describes the entire architecture of system. We insert our LUCID kernel module in dom0 which does the task of getting packets and dumping them straight onto disk.
Each guest OS has a virtual n/w interface (eth0) through which all network transactions occur. Hypervisor maintains a virtual backend interface (vif) in dom0 for each virtual n/w interface in guest OS to allow transfer of packets. We insert our kernel module in dom0 which does the task of getting packets by monitoring these backend interfaces and dumping them straight onto disk. A simple Xen script maps the VM's that are to be monitored to the backend virtual interfaces vif's. Now for each vif a kernel thread is created that gets packets flowing through the interface with help of layer 2 protocol handler (PF_RING, extensible to use AF_PACKET as well). Now once packets are captured, capture time filters are applied to filter out unwanted packets. This data is then dumped onto disk using indexing mechanism as described below. LUCID can work in both configurations i.e. Bridge and NAT mode on Xen.

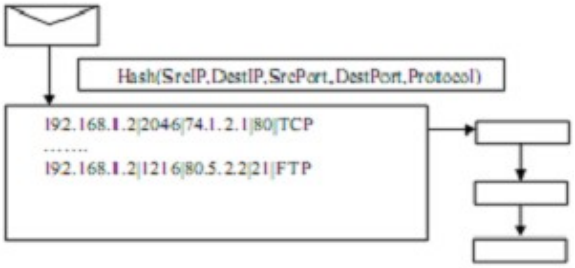
Fig.3. Design of Packet Capture Module



B. *Archiving and Indexing*

Network Monitoring involves extremely high-speed streams of structured data. It refers to passive collection of packet information such as headers, payloads, flow-identifiers, etc useful for purposes like security, network management and visibility into network interactions which is our ultimate motive. The most important feature is the ability to find the "Needle in the haystack" i.e. locate small number of Packets within a stream of uninteresting data. We group the incoming packets according to Flow Records. Flow Records have many common fields like Source IP, Destination IP, Source Port, Destination Port and Protocol, which designate a Flow. Using these parameters we create Flow Records which contain a link to the packets associated with the Flow. Such Flow Records are maintained for each Customer + Virtual Machine Pair.

They are then grouped in blocks which have a certain threshold for number of Flow Records it can hold. On reaching the threshold two operations occur in parallel on a block of flow records:



1.] Compressed Data Archiving: Flow Record blocks are compressed and stored onto the disk. Compression scheme [iii] is evaluated based on not only how efficient the compression rate is, but also how fast is the data decompressed. We choose the LZO (Lempel-Ziv-Oberhumer) compression scheme which is one of the fastest algorithms present taking into account decompression rate.

2.] Index Creation: The Indexing Mechanism chosen for facilitating faster retrieval of the archived data is the Bitmap Index. Indexes are also created based on the columnar data present in Flow Records and only for the most queried attributes like Source and Destination IP Addresses and Ports and Layer3 Protocol. Using Bitmap Indexes not only can we achieve fast insertion time, but also perform inexpensive bitwise AND/OR operations between columns. The Bitmap Indexes further are compressed using various Run Length Encoding (RLE) Schemes. Hence Bitmaps

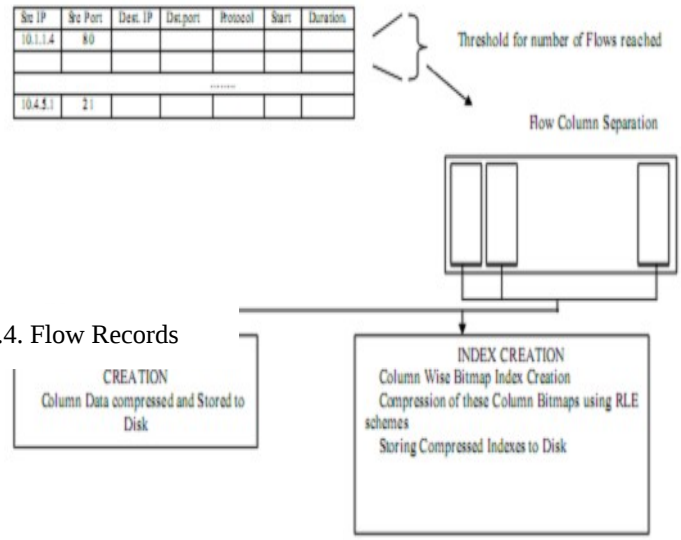


Fig.4. Flow Records

Fig.5. Storage Process i.e. Data Archive and Index Creation

3.] Query Processor

The following steps demonstrate the Query execution process [iv]:

1.] Attributes specified in the query are determined and corresponding compressed Bitmap Indexes are retrieved.
2.] These Bitmap Indexes are subjected to Boolean AND/OR operations providing required Flow Record Positions which satisfy the criteria.
3.] Based on Flow Record Positions, portions of archived data are decompressed.
4.] More complex queries can be executed on the reduced set of packets returned by the above process.
Explanation of the above process is presented by an example. Suppose the query is to retrieve all the packets contacted on port 140 with the source IP in the range 10.2.0.0/16. The corresponding query formed for it is:
Find all packets having source IP address range 10.2.*.* at the destination port 140.

The Bitmap Indexes for Src IP Byte 1, Src IP Byte 2 and Dest. Port are retrieved and from those the required columns i.e. 10, 4 and 140 respectively are fetched. The Result will be the set of Flow Records which satisfy the query and based on the Flow Records Positions only the required blocks of the Archive are decompressed. More complex queries can be executed on the set of packets returned by these Flow records and finally furnishing the results to the User.

C. SaaS Interface

Once the User Query has been processed and the results retrieved, we present this information to the user. The user will be provided with set of filters so that he can look at the data he wants to. The filtering can take place according to various parameters like Source IP, destination IP, Port, Timestamp, Protocol, Packet length, etc. provisions for graphs based on various

parameters can also be made. This will help in better interpretation of the collected data.

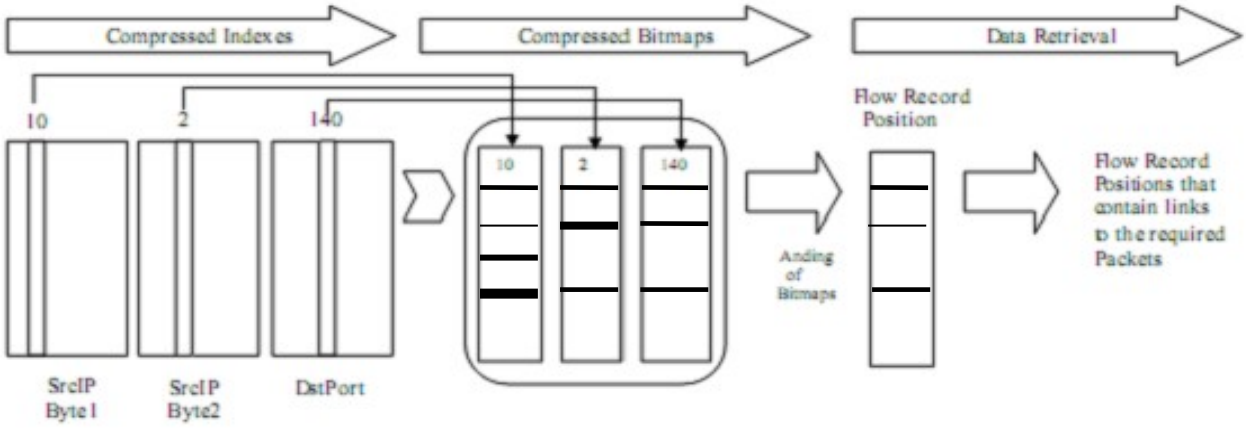


Fig.6. Query Processing and Data Retrieval

V. Applications

LUCID performs the task of Data archiving and Indexing on behalf of the user thus providing a useful service in addition to reducing his effort to maintain this data.
Captured data can be useful in:

- Tracing past activity and performing Postmortem analysis of data for mapping intrusion activities.
- Validating claims of Service Level Agreement (SLA) violations.
- Performing traffic analysis task that requires access to a historical packet repository.

VI. Results

To compute the results we used an AMD machine with 4GB RAM, 160 GB Hard disk (5400 rpm), 1.8GHz Dual Core Processor and a 10 MB Ethernet card. It had Xen Hypervisor running with Fedora 13 as Dom0 and 2 VMs running CentOS. All results were computed by replaying the same capture file using tcpreplay in both the VM's.

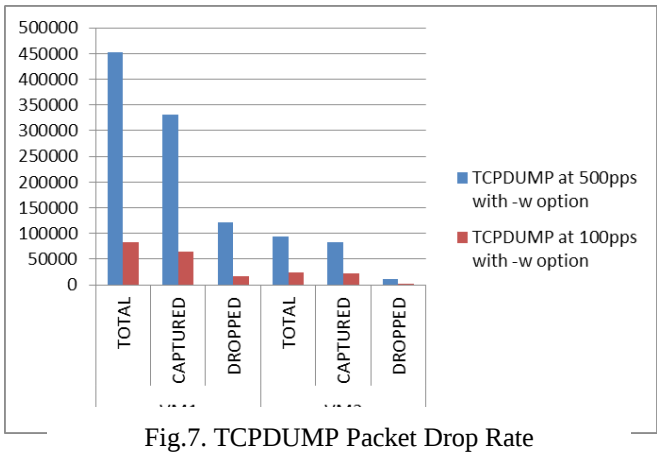


Fig.7. TCPDUMP Packet Drop Rate

then captured using LUCID, Wireshark and TCPDUMP all running in Dom0

.

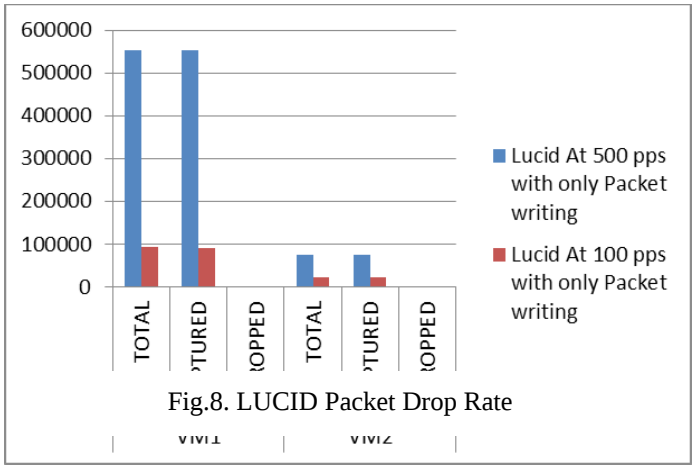


Fig.8. LUCID Packet Drop Rate

Fig.7. illustrates packet drop rate of TCPDUMP and Fig.8. illustrates packet drop rate of LUCID.

Fig.9. illustrates packet drop rate of LUCID when indexing is enabled.

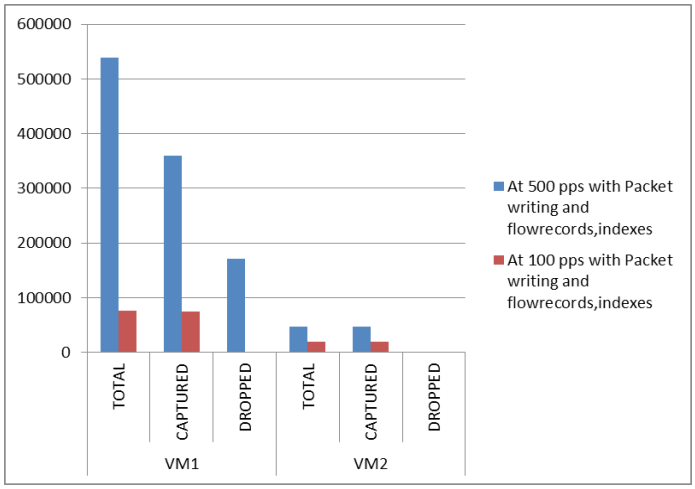


Fig.9. LUCID Packet Drop Rate with Indexing

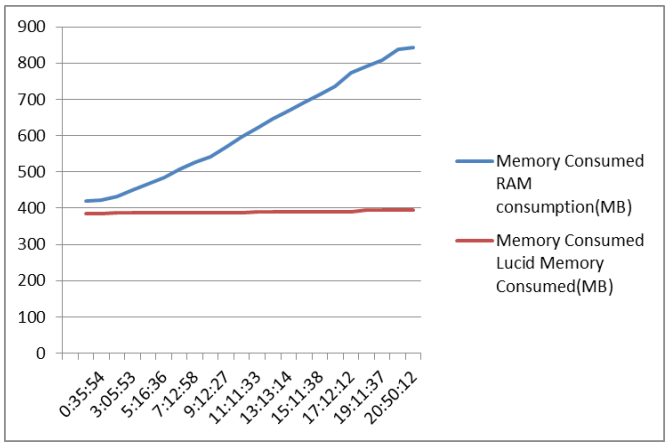


Fig.10. Wireshark vs. LUCID Memory

It is completely evident from the graphs that while writing packets to disk without indexing enabled packet drop rate is reduced to 0 due to kernel level packet capture, whereas significant drop rate occurs in case of TCPDUMP.

However there is some amount of drop when indexing is enabled as bitmap and flowrecord files were written to disk.

Fig.10. compares memory consumption of Wireshark and Lucid. In case of Wireshark memory requirement goes on increasing as it keeps packets in RAM whereas in case of LUCID, RAM requirement is almost constant. The results would have been worse for TCPDUMP and Wireshark had they been run within Guest OS which will ideally be the case.

VII. Conclusion and Future Work

This paper presents LUCID, a virtualization specific network monitoring tool(specific to cloud infrastructure) to capture network packet of all VMs at a centralized location. LUCID aims to outdo the problems associated with the use of current desktop specific network monitoring tools in virtualization environment. By having a kernel level packet capture module we reduce packet drop rate and memory utilization of capture process as evident from our results. We also provide efficient information retrieval mechanisms based on queries supplied by user. In future, this system can be scaled for a distributed environment.

References

(i) Xen and the art of Virtualization, P. Barham; B. Dragovic; K. Fraser; S. Hand; T. Harris; A. Ho; R. Neugebauer; I. Pratt and A. Warfield, 19th ACM Symposium on Operating Systems Principles, Oct 2003.

(ii) Network Security for Virtual Machine in Cloud Computing, Hanqian Wu; Yi Ding; Winer, C.; Li Yao; 2010 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT).

(iii) NET-Fli: On the fly compression, archiving and indexing on streaming network traffic; Francesco Fusco ; Marc Ph.Stoecklin ; Michail Vlachos : 2010 36th International conference On Very Large Data Bases.

(iv) pcapIndex: An index for network packet traces with legacy compatibility ; F Fusco ; X.Dimitropoulos; M Vlachos; L Deri (to appear) 2012 ACM SIGCOMM Computer Communication Review.

(v) Improving Passive Packet Capture: Beyond Device Polling, L. Deri, Proc. of SANE 2004.

(vi)www.ecsl.cs.sunysb.edu/elibrary/linux/network/LinuxKernel.pdf

(vii)http://wiki.kartbuilding.net/index.php/Xen_Networking

(viii)http://www.tcpdump.org/

(ix) The Definitive Guide to the Xen Hypervisor – David Chisnall.

(x) Understanding the Linux kernel-Daniel P. Bovet, Marco Cesati.