

Realization of IMS Network Elements by Virtualization

Chaitra.P.Gokanvi¹, Bhagirathi.N.M²

¹Student, ²Assistant Professor

Department of Electronics and Communication Engineering
BIT, Bangalore

Abstract— IP Multimedia Subsystem (IMS) is the engineering for an incorporated system of media transmission transporters with a specific end goal to encourage the utilization of IP (Internet Protocol) for parcel correspondences over remote or wired associations. An improvement in the functionalities and introduction of new features necessitates an upgrade in the elements of IMS network. In order to facilitate traffic handling of IMS calls during the Upgrade of Network Elements, a Single Node Pair (SNP) format is implemented, such that when either of the network elements is upgraded, the other element in the SNP handles the call flows. As expected, the upgrade tends to cause few call failures, but this remains within fault tolerance limits of the system. At the end of the upgrade, the performance analysis is done and through the parameter of CPU utilization, it is seen that the system's overall performance is well within the fault tolerance levels and that post upgrade the system's performance has improved. Although IMS has been ever growing in its popularity, it is still far from being practically implemented worldwide. The major roadblocks in this regard have been the issues of scalability, elasticity and ease of deployment. Virtualization could prove to be a worthy solution to these issues as it separates the application from the underlying hardware and runs them on a separate Virtual Machine (VM).

Index Terms— IMS, SNP, CSCF, VNF, L2TD

I. INTRODUCTION

IMS is an open institutionalized engineering for getting to any sort of media administration or communication association from any sort of get to either settled or versatile like 3G versatile, WLAN, WiMAX, DSL, Multiservice Access Network (MSAN), Fiber or Ethernet, and so forth. It depends on ETSI models like SIP for interfaces between building components.

IMS is in charge of making associations amongst endorsers and system assets and subsequently assumes a key part in application enablement as an extension between supporters and application content suppliers. In 2010, GSMA embraced the One Voice over LTE or VoLTE. One Voice was made by a gathering of merchants and specialist organizations. It was expected to facilitate the utilization of presentation of voice on LTE with simple to actualize transporter interoperability. VoLTE depends on a select arrangement of IMS gauges and along these lines utilizes IMS as its session control component.

There has been an expansion in the request from the clients to benefit the administrations of a arrange design that gives them consistent gadget merging. Regardless of if the client is utilizing a Personal PC, a cell phone or a Television; one ought to have the same arrangement of administrations or applications running over these gadgets consistently whenever furthermore, according to the client's accommodation.

Virtualization of IMS is proposed as one of the conceivable answers for overcome the issues of high accessibility, adaptability, adaptability and simplicity of arrangement, versatility, and so on. These issues have ended up being significant worries in the overall arrangement of IMS. Additionally, the redesign of virtualization should be possible on comparable lines as in the exposed metal condition. It is seen that the redesign in a virtualized IMS situation is considerably speedier and more productive.

II. RELATED WORK

Configurational Management repository server deployment and CSCF deployment both can be done by using Cloud Band Application Manager. The current VNF with a single pair of CFX-LB supports 2.5 Million (TMO US traffic model). If this is scaled to 6 M subscribers we would hit the dispatcher process. High capacity VNF process can help us in scaling in and scaling out process during traffic in call flow.

III. OPENSTACK ARCHITECTURE

OpenStack is open source programming, which implies that any individual who jars get to the source code, roll out any improvements or alterations they require, and uninhibitedly share these progressions retreat to the group on the loose. It likewise implies that OpenStack has the advantage of thousands of designers everywhere throughout the world working pair to build up the most grounded, most vigorous, and most secure item that they can.

1 OpenStack cloud overview

OpenStack is an arrangement of programming devices for building and overseeing cloud computing platforms for open and private clouds. OpenStack utilizes modular design and chiefly comprises of Nova, Swift, Cinder, Glance, Neutron, Horizon, and Keystone.

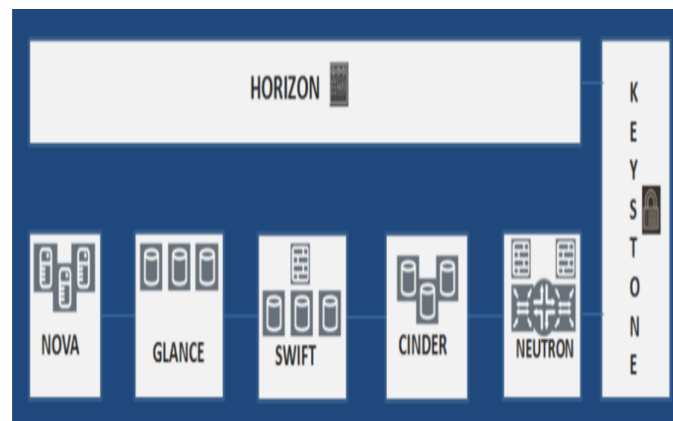


Fig 1. Openstack Architecture

- **Nova:** Nova is the essential figuring motor behind OpenStack. It is utilized for sending and overseeing expansive quantities of virtual machines and different occasions to deal with registering undertakings
- **Swift:** Swift is a capacity framework for articles and documents.
- **Cinder:** Cinder is a square stockpiling part, which is more undifferentiated from the traditional notion of a PC having the capacity to get to particular areas on a plate drive.
- **Neutron:** Neutron gives the systems administration capacity to OpenStack. It guarantees that each of the segments of an OpenStack sending can speak with each other rapidly and proficiently.
- **Horizon:** Horizon is the dashboard behind OpenStack. It is the main graphical interface to OpenStack.
- **Keystone:** Keystone gives character administrations to OpenStack. It is basically a focal rundown of the majority of the clients of the OpenStack cloud, mapped against the greater part of the administrations given by the cloud, which they have authorization to utilize.
- **Glance:** Glance gives picture administrations to OpenStack. For this situation, "pictures" alludes to images (or virtual duplicates) of hard plates. Look enables these pictures to be utilized as layouts while sending new virtual machine occasions.
- **Heat:** Heat is the fundamental venture in the OpenStack Orchestration program. It implements an organization motor to dispatch different composite cloud applications in view of layouts as content records that can be dealt with like code Existing solution

A.CSCF VNF

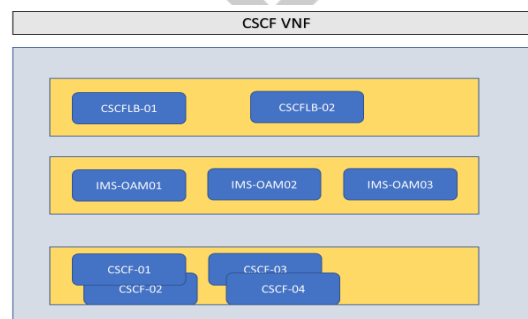


Fig 2. CSCF VNF structure

The cluster has 2 CFX-LB VMs in 1 (Active) + 1 (Standby) configuration. Both the VMs are up and running but the IPs are hosted on the Active VM. The external IPs are hosted on the CFX-LB VM and they terminate the connections from Peers. The CFX-LB VM hosts the LB processes like IPDP, DID and DNS PH.

The CFX-Role VMs are deployed in pairs (1 [Active] + 1 [Active]). There are at max 14 pairs of role VM deployed in the cluster. The CFX-Role VMs host the P/S/I Role processes.

IV. PROPOSED WORK

The current VNF with a single pair of CFX-LB supports 2.5 Million (TMO US traffic model). If this is scaled to 6 M subscribers we would hit the dispatcher process (IPDP) throughput bottle neck on the LB machine. There are different ways in which this issue can be resolved. The chosen solution based on two level load balancing

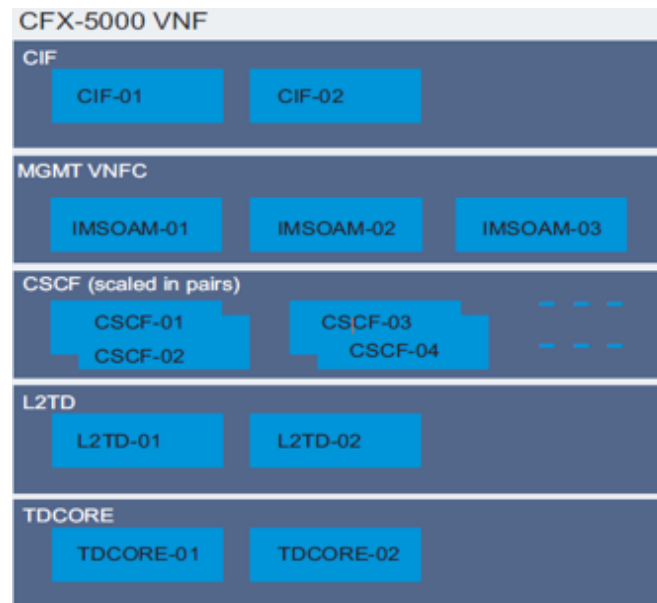


Fig 3. High Capacity VNF structure

The CFX VNF will consist of the following

- 1 L2 Traffic Dispatcher VMs deployed in 1 + 1 (Active/Standby) mode.
- 2 Traffic Dispatcher VMs deployed in 1+1 (Active/Active) mode.
- 3 Traffic Dispatcher Core VMs deployed in Active mode
- 4 Traffic Dispatcher Diameter VM deployed in 1+1 (Active/Standby) mode.
- 5 Central Interface and Functions (CIF) VM deployed in 1+1 (Active/Standby) mode.
- 6 Central Data store Interface (CDI) VM deployed in 1+1 (Active/Standby) mode.
- 7 This VM is introduced as part of FRS **Error! Reference source not found..**
- 8 N CSCF server (CFX-Role) 1+1 (Active) mode.
- 9 3 OAM unit VM in 1+1+1 (Active/Standby/Standby) mode.

A. L2Traffic Dispatcher VM : The L2Traffic Dispatcher VM shall host the external IPs of the VNF and provide a single point of contact for the peers. It will perform L2 based load balancing for the SIP and H.248 traffic and forward it to Traffic Dispatcher VMs. It is the current LB VM – (Dispatcher processes for SIP, Diameter, H.248 and DNS) + DPDK based L2 dispatcher for Core traffic.

B. TrafficDispatcher Core VM (new VM): This Traffic Dispatcher VM will host IpDp-Core, IpDp-Ic, IpDp-ISC, IpDp-H.248, ICM, IPFD and any other process related to the core dispatching functionality. The VMs are running in Active mode. The network connections will terminate on these VMs and existing lskpmc based traffic dispatching will be done by these VMs. For core traffic the traffic distribution at the L2Traffic Dispatcher can be done based On Load on the Traffic Dispatcher VMs

DEPLOYMENT PROCEDURE

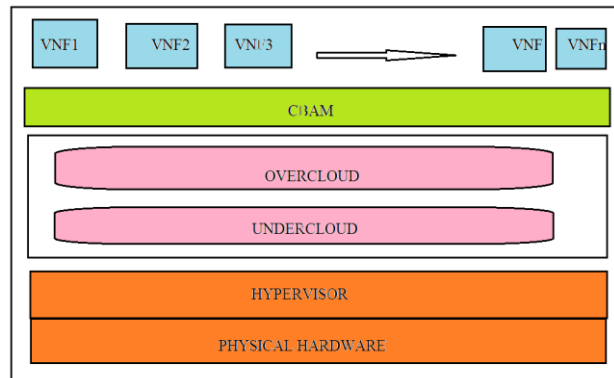


Fig 4. CBAM Deployment hierarchy

A. HYPERVISOR INSTALLATION

we are at present support Nokia Airframe Hardware , and HP additionally conceivable in future

1. Hypervisor should be installed on dedicated hardware
2. Download and save the ISO for the installation.
3. Connect to the remote console of the blade and mount the image.
4. Reboot the server and boot from the Virtual CDROM.
5. select the memory space required for software installation.
6. Configure and Enable the Network on the Public Port, add DNS server ip and hostname and Vlan addresses
7. For Ipv6 interfaces add a valid external IPv6 address & gateway to the interface VLAN you just configured.
8. Once installation finishes, reboot the nodes.

B. CLOUD BAND INFRASTRUCTURE SOFTWARE

a. Undercloud install procedure

1. Get the latest successful Undercloud artifact
2. Extract the CloudBand Infrastructure Software installer
3. Copy user configure file and edit network address, vlan, gateway and ip addresses
4. Other default values should be changed according to requirement.
5. Run the install server using `./install_undercloud.py` in `cbis-install` server path.

b. Overcloud installation procedure

- 1) In configure file add RAM allocation ratio, CPU allocation ratio & vlan details
- 2) Provide the host groups Controller nodes, OVS compute node, SR-IOV performance compute nodes, DPDK performance compute nodes & Storage Node
- 3) Number of controller hosts & compute hosts should be 3 or more.
- 4) Login to the Undercloud VM with user `stack`
- 5) The `user_config.yaml` file is automatically transferred from the physical Undercloud server to the Undercloud VM at `/home/stack/` as part of the Undercloud VM installation process.
- 6) Fill the mandatory values like network address, gateway, `ip_range_start`, `ip_range_end`, `vlan ip` in user configure file
- 7) Run the script source `~/stackrc` on the Undercloud VM, as `stack` user
- 8) Generate the Overcloud templates configuration folder. This folder will be used later for the hardware inspection and overcloud deployment.
- 9) Generate a template

Hardware scan

1. Replace the IPs and host groups according to your deployment design and setup IPs.
2. Create input file `hosts.yaml`. Run the following `hwscan` command

Hardware Introspection

1. Perform hardware inspection
2. Import the hosts.yaml file to Ironic
3. Configure the hosts
4. Start introspection
5. Verify hardware inspection

Deploy Openstack

- The Openstack deployment is based on the set of generated templates and hardware scan:
- If you run the `openstack overcloud deploy` command with parameters that overlap with the parameters in the `user_config.yaml` file the parameter in the CLI command will take over. Be sure to not configure different values for the same parameters in both the `user_config.yaml` and the Openstack deployment command.
- If you need to use the command `openstack cloudband overcloud deploy`, run it with `--help` to understand the possible arguments.
- It is mandatory to run the `openstack cloudband overcloud deploy` command from the folder `/home/stack` in the Undercloud VM
- Post deploy, the actually used deploy command can be found (for later debug usage) in: `/var/log/cloudband/overcloud_installation.log` under the Undercloud VM.
- During deployment process, the cloudband client allows you to check the installation progress using the `openstack cloudband overcloud status` command.

C. CMREPO DEPLOYMENT PROCEDURE

1. Creating extension and input json file
2. Application creation using CBAM
3. Uploading Repo file to CBAM catalogue
4. Creating and Modifying VNF
5. Instantiating VNF

D. CSCF DEPLOYMENT PROCEDURE

1. Uploading configuration data and software to Repo Server
2. Any route added manually in `IP_DYNAMIC_ROUTING` is non-persistent across reboots, service network restart and offline parameter update.
3. Copy the filled Excel sheet in the CMRepo to any of the location.
4. The application and the platform schema files need to be copied to the CMRepo along with the Excel sheet, otherwise parsing of the TPD fails. Copy the schema file to any location on the CMRepo and perform the following steps:
 - Add the Platform schema file
5. Execute the `cmcli` tool to parse the TPD. First parse the Common TPD followed by the DUSpec TPD.
6. Parse load traffic input using `cmcli` command.
7. The `swcli.py` CLI command must be used with appropriate options to prepare DU for installation of APS and HFs. Execute the `swcli.py` tool as root user.
8. For initial preparation of DU with CSCF APS, FEE APS (if enabled), and Hot fix.
5. Creating extension and input json files
6. Uploading CSCF package
7. Creating and Modifying CSCF VNF
8. Instantiating CSCF VNF

Conclusion: Availability is high in Openstack than hardware component. It should be feasible for the Virtualization administration framework to start scale out (adding more VMs to handle expanding movement) of serving VMs in view of expanding activity request. It should be workable for the Virtualization administration framework to start scale in (expelling VMs from movement preparing) of serving VMs in light of diminishing activity drift. No change in charging infrastructure. The CSCF Role processes create the ticket files (temporarily) on CSCF Servers VMs. Charging Transfer Function transfers the tickets to the Active CIF VM. Either the CDFS pulls the tickets from the Active CIF or the CIF pushes the tickets to the CDFS. A single IP address based interface shall be provided to the CDFS. The current VNF with a single pair of CFX-LB supports 2.5 Million (TMO US traffic model). If this is scaled to 6 M subscribers we would hit the dispatcher process. High capacity VNF process can help us in scaling in and scaling out process during traffic in call flow

REFERENCES

- [1] Haitao Meng, "A Preliminary Research on Security Issues in IP Multimedia Subsystem", 9th IEEE International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2012, DOI: 10.1109/FSKD.2012.6234000, pp. 2560 – 2564
- [2] Isam Abdalla and S. Venkatesan, "Notification based S-CSCF load Balancing in IMS Networks", IEEE Wireless Telecommunications Symposium (WTS), 2011, DOI: 10.1109/WTS.2011.5960852, pp. 1-5.

- [3] A. Vrábel, R. Vargic and I. Kotuliak, "Subscriber Databases and their Evolution in Mobile Networks from GSM to IMS", 49th International Symposium ELMAR-2007, DOI: 10.1109/ELMAR.2007.4418811, pp. 115-117.
- [4] Plarent Tirana and Deep Medhi, "Distributed Approaches to S-CSCF Selection in an IMS Network", IEEE Conference on Network Operations and Management Symposium (NOMS), 2010, DOI: 10.1109/NOMS.2010.5488465, pp. 224 – 231.
- [5]<http://www.wseas.us/e-library/conferences/2009/vouliagmeni/ACCMM/ACCMM1-42.pdf>. "An IMS-based Virtualized Architecture: Performance Analysis"
- [6]<http://www.dit.upm.es/~egarcia/pdf/HPOVUA06.pdf>. "Design and Implementation of an IP Multimedia Subsystem (IMS) Emulator Using Virtualization Techniques".

