TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

Đề tài:

TRIỂN KHAI CHUỖI CHỨC NĂNG MẠNG TRÊN NỀN TẢNG OPENSTACK TÍCH HỢP DPDK

Sinh viên thực hiện: ĐỖ XUÂN SƠN
ĐTTT 06 - K58

Giảng viên hướng dẫn: PGS. TS. NGUYỄN HỮU THANH

Hà Nội, 6-2019

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

Đề tài:

TRIỂN KHAI CHUỖI CHỨC NĂNG MẠNG TRÊN NỀN TẢNG OPENSTACK TÍCH HỢP DPDK

Sinh viên thực hiện: ĐỖ XUÂN SƠN
ĐTTT 06 - K58

Giảng viên hướng dẫn: PGS. TS. NGUYỄN HỮU THANH

Cán bộ phản biện:

Hà Nội, 6-2019

ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP

(Dùng cho giảng viên hướng dẫn)

Tên giảng viên đánh giá: PGS. TS. Nguyễn Hữu Thanh

Họ và tên sinh viên: Đỗ Xuân Sơn MSSV: 20133299

Tên đồ án: Triển khai chuỗi chức năng mạng trên nền tảng OpenStack tích hợp DPDK

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:
Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết và thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được	1	2	3	4	5
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4	5
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng	1	2	3	4	5
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai	1	2	3	4	5
Kỹ năng viết quyển đồ án (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phảy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định	1	2	3	4	5
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)	1	2	3	4	5
Thành tựu nghiên cứu khoa học (5) *(chọn 1 trong 3 trường hợp)*						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/Đạt giải SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế	5				
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành (VD: TI contest)	2				
10c	Không có thành tích về nghiên cứu khoa học	0				
Điểm tổng		**/50**				
Điểm tổng quy đổi về thang 10						

Nhận xét khác *(về thái độ và tinh thần làm việc của sinh viên)*

.....

.....

.....

.....

.....

.....

Ngày: … / … / 20…

Người nhận xét

(Ký và ghi rõ họ tên)

ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP

(Dùng cho cán bộ phản biện)

Giảng viên đánh giá: .....

Họ và tên sinh viên: Đỗ Xuân Sơn MSSV: 20133299

Tên đồ án: Triển khai chuỗi chức năng mạng trên nền tảng OpenStack tích hợp DPDK

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:
Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết và thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được	1	2	3	4	5
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4	5
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng	1	2	3	4	5
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai	1	2	3	4	5
Kỹ năng viết quyển đồ án (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phảy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định	1	2	3	4	5
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)	1	2	3	4	5
Thành tựu nghiên cứu khoa học (5) *(chọn 1 trong 3 trường hợp)*						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/Đạt giải SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế	5				
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành (VD: TI contest)	2				
10c	Không có thành tích về nghiên cứu khoa học	0				
Điểm tổng		**/50**				
Điểm tổng quy đổi về thang 10						

Nhận xét khác của cán bộ phản biện

.....

.....

.....

.....

.....

.....

Ngày: … / … / 20…

Người nhận xét

(Ký và ghi rõ họ tên)

LỜI NÓI ĐẦU

Trong thời đại hiện nay, việc mở rộng quy mô kinh doanh của các doanh nghiệp và các nhà mạng viễn thông trong lĩnh vực công nghệ dẫn đến nhu cầu về băng thông, chất lượng đường truyền tăng mạnh. Các nhà cung cấp dịch vụ mạng phải thiết lập và cấu hình hệ thống mạng sao cho đảm bảo dịch vụ tới khách hàng được ổn định và tin cậy nhất. Điện toán đám mây cùng với NFV ra đời trở thành giải pháp cho các doanh nghiệp tiết kiệm chi phí đầu tư vào bớt phụ thuộc và các nhà cung cấp thiết bị phần cứng chuyên dụng. Đồng thời, các nhà mạng cung cấp dịch vụ có thể khởi tạo, điều phối, di chuyển các chức năng mạng linh hoạt, dễ dàng hơn.

Trong quá trình làm đồ án này, em tìm hiểu về kiến thức điện toán đám mây, nền tảng OpenStack, công nghệ ảo hóa chức năng mạng và công nghệ DPDK (Data Plane Development Kit) để triển khai chuỗi dịch vụ mạng trong các trung tâm dữ liệu. Trong đó, em tập trung vào đo đạc hiệu năng của các chức năng mạng và các thông số như băng thông, độ trễ, tỷ lệ mất gói của chuỗi chức năng mạng trong hệ thống.

Em xin chân thành cảm ơn **PGS.TS. Nguyễn Hữu Thanh** đã tận tình giúp đỡ, tạo điều kiện để em hoàn thành đồ án tốt nghiệp này. Ngoài ra, em cũng gửi lời cảm ơn chân thành đến các thành viên Future Internet Lab đã giúp đỡ em trong suốt thời gian qua.

LỜI CAM ĐOAN

Em là Đỗ Xuân Sơn, mã số sinh viên 20133299, sinh viên lớp ĐTTT 06, khóa K58. Người hướng dẫn là PGS. TS. Nguyễn Hữu Thanh. Em xin cam đoan toàn nội dung trong đồ án "Triển khai chuỗi chức năng mạng trên nền tảng OpenStack tích hợp DPDK" là kết quả nghiên cứu của em. Những thông tin và kết quả được trình bày trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế do em làm. Các tài liệu tham khảo đều được liệt kê nguồn gốc rõ ràng, chân thực. Em xin chịu hoàn toàn trách nhiệm với những nội dung được trình bày trong đồ án này.

Hà nội, ngày 01 tháng 06 năm 2019

Người cam đoan

MỤC LỤC

DANH SÁCH KÝ HIỆU VÀ CHỮ VIẾT TẮT

Chữ viết tắt	Tiếng Anh	Tiếng Việt
NFV	Network Function Virtualization	Ảo hóa chức năng mạng
	Cloud Computing	Điện toán đám mây
IaaS	Infrastructure-as-a-Service	Cơ sở hạ tầng như một dịch vụ
PaaS	Platform-as-a-Service	Nền tảng như một dịch vụ
SaaS	Software-as-a-Service	Phần mềm như một dịch vụ
	Router	Thiết bị định tuyến
	Switch	Thiết bị chuyển mạch
IP	Internet Protocol	Giao thức Internet
API	Application Programing Interface	Giao diện lập trình ứng dụng
NIDS	Network-based	
HIDS	Host-based IDS	
NAT	Network Address Translation	Đổi địa chỉ IP của gói tin
	Firewall	Tường lửa
CPU	Central Processing Unit	Bộ xử lý trung tâm
RAM	Random Access Memory	Bộ nhớ truy cập ngẫu nhiên
VNF	Virtualized Network Function	Chức năng mạng ảo
DC	Data Center	Trung tâm dữ liệu
VM	Virtual Machine	Máy ảo
	Virtualization	Ảo hóa

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN

Trong lịch sử viễn thông đã có các thiết bị vật lý thực hiện một số loại chức năng chuyên dụng. Các thiết bị này đắt tiền và rất khó để cấu hình lại về hành vi kết nối mạng của chúng. Ví dụ điển hình là một số thiết bị vật lý được kết nối trực tiếp với nhau trong một chuỗi. Về cơ bản, Service Function Chaining (SFC) là khả năng khiến các luồng lưu lượng mạng được định tuyến qua mạng thông qua một đường dẫn khác với đường dẫn sẽ được chọn bằng cách tra bảng định tuyến dựa trên địa chỉ IP đích của gói tin. Nó thường được sử dụng cùng với công nghệ ảo hóa chức năng mạng, khi triển khai trong môi trường ảo một loạt các chức năng mạng thường được triển khai như một tập hợp các thiết bị mạng vật lý được kết nối nối tiếp nhau bằng dây cáp mạng. Qua đó, cung cấp các dịch vụ mạng khác nhau một cách linh hoạt và tiện lợi cho khách hàng.

Trong đồ án này, em tìm hiểu về kiến thức điện toán đám mây, nền tảng OpenStack, công nghệ ảo hóa chức năng mạng và công nghệ DPDK (Data Plane Development Kit) để triển khai chuỗi dịch vụ mạng trong các trung tâm dữ liệu. Đồng thời, em cũng xây dựng hệ thống chuỗi chức năng mạng ảo trên nền tảng OpenStack tích hợp DPDK. Qua đó đo đạc, đánh giá hiệu năng của các chức năng mạng và các thông số như băng thông, độ trễ, tỷ lệ mất gói của chuỗi chức năng mạng trong hệ thống.

ABSTRACT

Throughout the history of telecommunication, we have always been using a dedicated physical device for each specific work. Though showing great power and abilities in doing what they are designed for, these devices are very expensive and hard to be configured, especially when we want to connect them into a chain. Technically speaking, Service Function Chaining (SFC) allows us to drive the network traffic to our chosen path, which may be different from the original IP routing path. SFC is normally deployed together with Network Function Virtualization (NFV) technology. In virtual environment, the network functions can work as if they are physical devices with network cable connecting them but in a much more flexible way. Thus, creating various flexible and optimized function chains for different client services.

With this thesis, I will introduce the required technologies for deploying SFC in a Data Center, such as Cloud Computing, OpenStack, Network Function Virtualization and Data Plane Development Kit (DPDK). I will also construct a virtual service chain based on the combination of OpenStack and DPDK. A several tests will be done to assess the performance of the service functions and the chain.

.

PHẦN MỞ ĐẦU

a. Đặt vấn đề

Trong thời đại hiện nay, chúng ta đang chứng kiến sự ra đời của hàng loạt các công nghệ mới như công nghệ vạn vật kết nối Internet (Internet of Things), điện toán đám mây (Cloud Computing), máy học (Machine Learning), Blockchain, mạng 5G… Cùng với đó là việc mở rộng quy mô kinh doanh của các doanh nghiệp trong lĩnh vực công nghệ và các nhà mạng viễn thông, dẫn đến nhu cầu về băng thông, chất lượng đường truyền tăng lên. Sau đây nêu ra một số dự đoán về mạng toàn cầu trong giai đoạn 2017 - 2022 (dựa theo "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper"):

- Lưu lượng dữ liệu di động toàn cầu sẽ tăng gấp 7 lần từ năm 2017 đến năm 2022. Lưu lượng dữ liệu di động sẽ đạt 77.5 exabytes mỗi tháng vào năm 2022.
- Đến năm 2022 sẽ có trung bình 1.5 thiết bị di động/người. Sẽ có 12.3 tỷ thiết bị di động được kết nối vào năm 2022.
- Đến năm 2022, mạng 4G sẽ chiếm 54% kết nối, nhưng chiếm 71% tổng lưu lượng.
- Đến năm 2022, 5G sẽ chiếm 3.4% kết nối nhưng 11.8% tổng lưu lượng.
- Đến năm 2022, máy tính bảng và máy tính cá nhân sẽ được kết nối tạo ra 6.8 GB lưu lượng truy cập mỗi tháng, tăng gấp đôi so với mức trung bình năm 2017 là 3.3 GB.
- Điện thoại thông minh trung bình sẽ tạo ra 11 GB lưu lượng truy cập mỗi tháng vào năm 2022.

Tại Việt Nam, 3 nhà mạng lớn Viettel, VNPT và Mobifone đã triển khai rộng rãi 4G. Việc này đặt ra nhiều bài toán phức tạp về việc xây dựng hạ tầng phần cứng mạng bên dưới đáp ứng các nhu cầu mới. Nhu cầu cải thiện hạ tầng mạng (cả chất lượng và số lượng) là thiết yếu hơn bao giờ hết, đòi hỏi các nhà cung cấp dịch vụ mạng không ngừng mở rộng quy mô cũng như nâng cao chất lượng đường truyền. Hiện nay, các nhà cung cấp dịch vụ mạng viễn thông đa phần là mua thêm các thiết bị phần cứng chuyên dụng của các hãng như Cisco, Juniper… cho mỗi dịch vụ mạng. Cách làm này hiện nay cho thấy rất nhiều hạn chế trong triển khai và vận hành.

Trong thực tế triển khai và vận hành, những hệ thống sử dụng các thiết bị chuyên dụng có nhiều hạn chế như: giá thành thiết bị đắt đỏ, khó quản lý tập trung, kém tương thích với các thệ thống của hãng khác, tốc độ cập nhật phần mềm chậm, giấy phép sử dụng phần

mềm ngắn hạn.. Thêm vào đó, mỗi khi cần triển khai một dịch vụ mới, một chức năng mới rất tốn kém về cả thời gian lẫn tiền bạc. Với quy trình hiện nay, mỗi khi khởi tạo một dịch vụ mạng mới có thể cần tới vài ngày hay vài tuần để đưa hệ thống mới vào hoạt động. Điều đó cho thấy sẽ rất tốn thời gian và nhân lực cho mỗi dự án triển khai, đặc biệt với những dự án có thời gian sử dụng ngắn thì việc làm truyền thống là vô cùng lãng phí thời gian và tiền bạc.

Những hạn chế trên là không thể chấp nhận trong môi trường công nghệ thông tin hiện nay. Hạ tầng mạng hiện có được dự báo sẽ không thể đáp ứng kịp nhu cầu của thị trường cũng như đảm bảo lợi ích của các doanh nghiệp hay nhà cung cấp dịch vụ và khách hàng.

Vậy liệu có giải pháp nào có thể giải quyết những vấn đề đang gặp phải của các doanh nghiệp hay không?

b. Đề xuất hướng giải quyết

Giải pháp cho vấn đề này chính là ứng dụng công nghệ ảo hóa vào hạ tầng mạng tại các trung tâm dữ liệu , các nút chuyển mạch lớn; cụ thể ở đây là công nghệ "Ảo hóa chức năng mạng" (Network Functions Virtualization - NFV).

Công nghệ NFV cho phép ta tách biệt các chức năng mạng (Virtualization Network Function -VNF) như Firewall, DNS, Caching, NAT,.. khỏi các thiết bị vật lý chuyên dụng và triển khai chúng dưới dạng phần mềm trên các thiết bị phần cứng phổ thông như các máy chủ vật lý, thiết bị lưu trữ dữ liệu,... Mà các thiết bị vật lý này không phải là thiết bị phần cứng độc quyền của các nhà cung cấp, chúng được sản xuất hàng loạt theo các tiêu chuẩn chung. Qua đó sẽ giúp ta giảm chi phí đầu tư và bớt phụ thuộc và các nhà cung cấp thiết bị phần cứng chuyên dụng. Đồng thời, các nhà mạng cung cấp dịch vụ có thể khởi tạo, điều phối, di chuyển các chức năng mạng linh hoạt, dễ dàng hơn.

Trong đồ án này, em thực hiện tìm hiểu các vấn đề liên quan tới công nghệ điện toán đám mây (Cloud Computing), công nghệ ảo hóa chức năng mạng (NFV), chuỗi chức năng mạng (SFC), một số chức năng mạng như Giám sát lưu lượng mạng, Tường lửa,… Sau đó là dựng mô hình testbed chuỗi chức năng mạng ảo gồm các năng Giám sát lưu lượng mạng, Tường lửa và Hệ thống phát hiện xâm nhập để bảo vệ máy chủ web server trong trung tâm dữ liệu, xây dựng trên nền tảng Cloud OpenStack để đánh giá hiệu năng hoạt động của chuỗi chức năng mạng ảo đó.

Đồ án đi vào tìm hiểu các vấn đề chính:

- Tìm hiểu công nghệ điện toán đám mây (Could Computing) và nền tảng mã nguồn mở OpenStack.
- Tìm hiểu công nghệ ảo hóa chức năng mạng (NFV) và một số trường hợp sử dụng NFV.
- Giới thiệu một số công nghệ làm chức năng mạng ảo.
- Tìm hiểu chuỗi chức năng mạng (SFC) và các trường hợp triển khai SFC.
- Tìm hiểu về DPDK (Data Plane Development Kit) và ứng dụng của DPDK.
- Triển khai chuỗi chức năng mạng ảo (SFC) gồm Giám sát lưu lượng mạng, Tường lửa và Hệ thống phát hiện xâm nhập để bảo vệ máy chủ web server trên nền tảng Cloud OpenStack kích hoạt DPDK.
- Khảo sát hiệu năng của các chức năng mạng ảo (CPU, RAM) và các thông số chất lượng dịch vụ của chuỗi chức năng mạng (băng thông, độ trễ, tỉ lệ mất gói).

Nội dung đồ án của em gồm các phần chính như sau:

- **Chương 1:** Trình bày tổng quản cơ sở lý thuyết các công nghệ liên quan để triển khai đề tài như công nghệ điện toán đám mây (Cloud Computing), nền tảng OpenStack, công nghệ ảo hóa chức năng mạng,…
- **Chương 2:** Giới thiệu một số chức năng mạng ảo được dùng trong chuỗi chức năng mạng. Triển khai mô hình chuỗi chức năng mạng ảo gồm nhiều chức năng mạng trên nền tảng OpenStack không được tích hợp DPDK và được tích hợp OpenStack.
- **Chương 3:** Đưa ra kịch bản đo đạc hiệu năng các chức năng mạng ảo và kịch bản đo đạc các thông số đánh giá chất lượng dịch vụ của chuỗi chức năng mạng. Nêu nhận xét, đánh về những kết quả đạt được và những hạn chế của đề tài.
- **Kết luận:** Đề xuất định hướng phát triển đề tài trong tương lai.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

Trong chương này, em sẽ trình bày cơ sở lý thuyết về các công nghệ liên quan đến đề tài như điện toán đám mây, nền tảng mã nguồn mở OpenStack, công nghệ ảo hóa chức năng mạng,...

1.1. Tổng quan về công nghệ điện toán đoám mây (Cloud Computing)

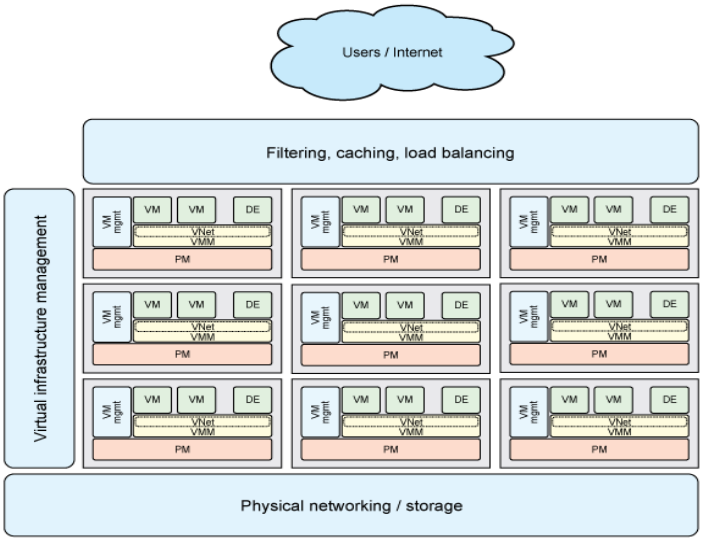
1.1.1. Giới thiệu về công nghệ điện toán đám mây

Thuật ngữ "Cloud Computing" đã được phổ biến với Amazon.com khi phát hành các sản phẩm Compute Cloud vào năm 2006. Thuật ngữ này ra đời không phải để nói về một trào lưu mới, mà là để khải quát các hướng đi của cơ sở hạ tầng thông tin đã và đang diễn ra từ mấy năm qua.

Theo định nghĩa của NIST (National Institute of Standards and Technology):

Điện toán đám mây (Cloud computing) là mô hình cho phép truy cập qua mạng để lựa chọn và sử dụng tài nguyên (ví dụ:mạng, máy chủ, lưu trữ, ứng dụng, và dịch vụ) theo nhu cầu một cách thuận tiện và nhanh chóng; đồng thời cho phép kết thúc sử dụng dịch vụ, giải phóng tài nguyên dễ dàng, giảm thiểu các giao tiếp với nhà cung cấp. [1]

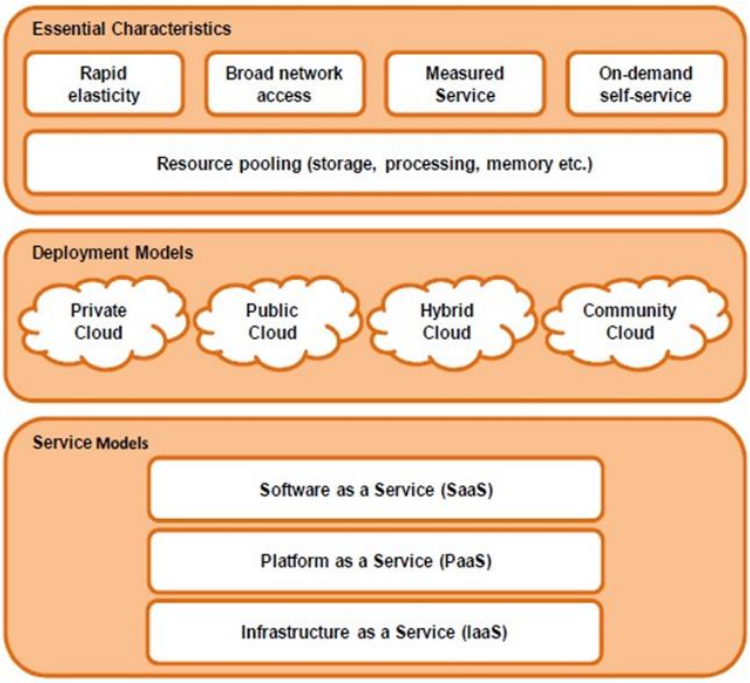
Đi sâu hơn vào kiến trúc, mỗi nút trong hệ thống điện toán đám mây chính là một máy chủ sử dụng công nghệ ảo hóa. Khi kết hợp và phối hợp quản lý các máy chủ đó một cách bài bản ta được một hệ thống cơ sở hạ tầng ảo hóa gọi là "Cloud" như hình dưới đây:



Hình 1.1. Mô hình cơ sở hạ tầng điện toán đám mây [3]

1.1.2. Các đặc trưng của công nghệ điện toán đám mây

Mô hình điện toán đám mây gồm 5 đặc tính, 3 mô hình dịch vụ và 4 mô hình triển khai.



Hình 1.2. Các đặc trưng của mô hình điện toán đám mây [3]

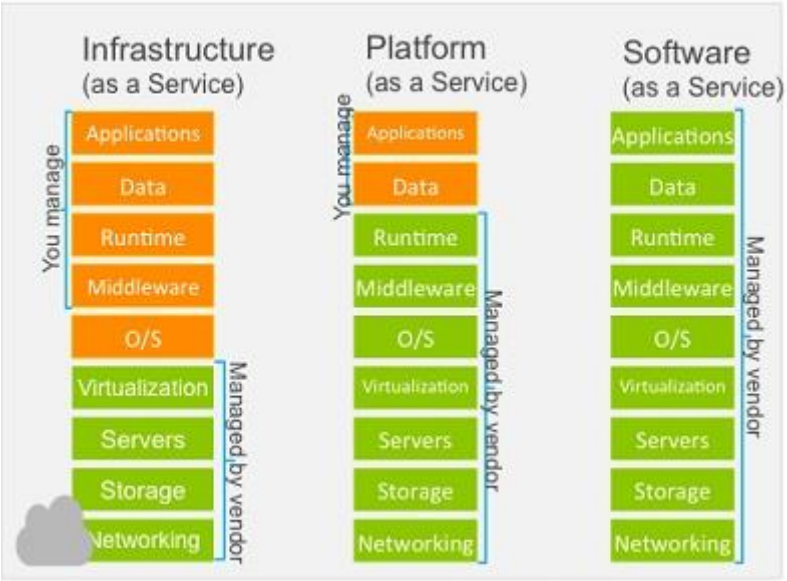
a. 5 đặc tính thiết yếu

- **On-demand self-service (Khả năng tự phục vụ):** Người dùng có thể tự cung cấp khả năng tính toán, như thời gian phục vụ và mạng lưu trữ khi cần tự động mà không yêu cầu sự tương tác của người đó với mỗi nhà cung cấp dịch vụ.
- **Broad network access (Khả năng hỗ trợ nhiều chuẩn mạng):** Hỗ trợ người dùng truy cập qua mạng từ nhiều nền tảng thiết bị như điện thoại di động, máy tính bảng, laptop và máy trạm,...) thông qua cơ chế tiêu chuẩn.
- **Resouce pooling (Khả năng tổng hợp tài nguyên):** Các tài nguyên điện toán của nhà cung cấp dịch vụ được tập hợp lại để phục vụ nhiều khách hàng, với các tài nguyên vật lý và ảo khác nhau được gán tự động và gán lại theo nhu cầu khách hàng. Ví dụ về tài nguyên bao gồm lưu trữ, xử lý, bộ nhớ, băng thông mạng,…
- **Rapid elasticity (Tính đàn hồi nhanh):** Khả năng có thể cung cấp và giải phóng tài nguyên một cách linh hoạt.
- **Measured service (Dịch vụ đô lường):** Các hệ thống cloud tự động kiểm soát và tối ưu hóa việc sử dụng tài nguyên bằng cách tận dụng khả năng đo lường. Việc sử

dụng tài nguyên có thể được theo dõi, kiểm soát và báo cáo, cung cấp sự minh bạch cho cả nhà cung cấp và khách hàng sử dụng dịch vụ.

b. 3 mô hình dịch vụ

- **Software as a Service (SaaS):** Khả năng cung cấp cho khách hàng sử dụng các ứng dụng của nhà cung cấp đang chạy trên cơ sở hạ tầng điện toán đám mây. Các ứng dụng có thể truy cập từ nhiều thiết bị khác nhau thông qua giao diện khách hàng, chẳng hạn như trình duyệt web hoặc chương trình giao diện người dùng. Trong mô hình này, khách hàng không quản lý hoặc kiểm soát cơ sở hạ tầng cloud cơ bản bao gồm mạng, máy chủ, hệ thống điều hành, lưu trữ,…
- **Platform as a Service (PaaS):** Khả năng cung cấp cho khách hàng là triển khai trên cơ sở hạ tầng cloud các ứng dụng do khách hàng tạo hoặc mua được. Khách hàng không được quản lý hoặc kiểm soát cơ sở hạ tầng đám mây cơ bản bao gồm cả mạng, máy chủ, hệ điều hành hoặc lưu trữ,... nhưng có quyền kiểm soát các ứng dụng đã triển khai và có thể cài đặt cấu hình cho môi trường lưu trữ ứng dụng đó.
- **Infrastructure as a Service (IaaS):** Khả năng cung cấp cho khách hàng là cung cấp xử lý, lưu trữ, mạng và các tài nguyên điện toán cơ bản khác, nơi khách hàng có thể triển khai và chạy phần mềm tùy ý. Khách hàng không được quản lý hoặc kiểm soát cơ sở hạ tầng cloud cơ bản nhưng có quyền kiểm soát các hệ điều hành, lưu trữ và các ứng dụng được triển khai; và có thể kiểm soát các thành phần mạng được chọn (ví dụ: tưởng lửa,…)



Hình 1.3. 3 mô hình dịch vụ của điện toán đám mây [4]

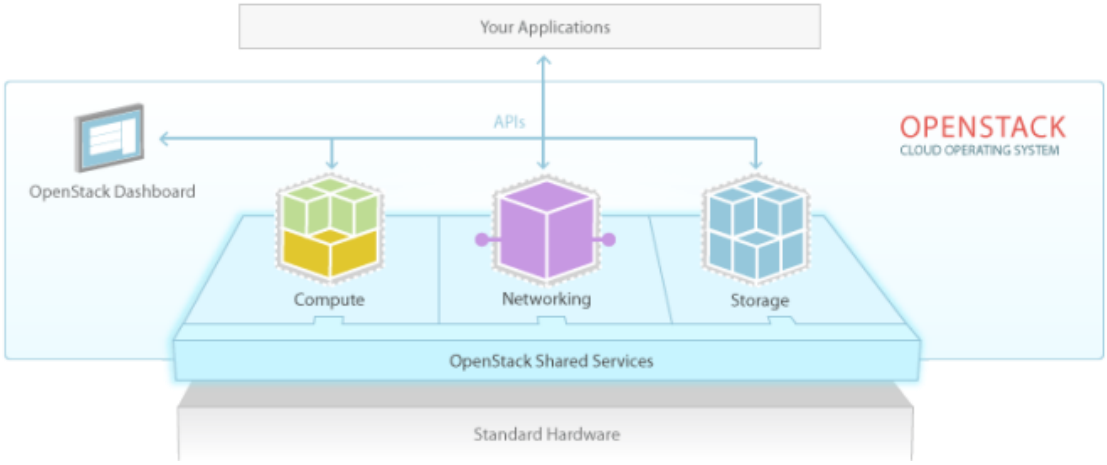
c. 4 mô hình triển khai

- **Private cloud:** Cơ sở hạ tầng cloud được cung cấp để sử dụng độc quyền bởi một tổ chức bao gồm nhiều người dùng (ví dụ: các đơn vị kinh doanh). Nó có thể được sở hữu, quản lý và vận hành bởi tổ chức, bên thứ ba hoặc một số tổ hợp của họ.
- **Community cloud:** Cơ sở hạ tầng cloud được cung cấp để sử dụng độc quyền bởi một số cộng đồng khách hàng cụ thể từ các tổ chức có chung một mối quán tâm (ví dụ: nhiệm vụ,yêu cầu bảo mật, chính sách,…) . Nó có thể được sở hữu, quản lý và vận hành bởi một hoặc nhiều tổ chức trong cộng đồng, bên thứ ba hoặc một số tổ hợp của họ.
- **Public cloud:** Cơ sở hạ tầng cloud được cung cấp để sử dụng cho khách hàng thông qua mạng internet. Nó có thể được quản lý bởi một tổ chức kinh doanh,… Ví dụ: AWS của Amazon, Azure của Microsoft,…
- **Hybrid cloud:** Cơ sở hạ tầng cloud này là sự kết hợp của hai hoặc nhiều cơ sở hạ tầng cloud riêng biệt (private, community, hoặc public).

1.2. Tổng quan về OpenStack

1.2.1. Giới thiệu về OpenStack

OpenStack là hệ điều hành điện toán đám mây kiểm soát và điều khiển các nhóm tài nguyên như tài nguyên tính toán, lưu trữ và mạng thông qua trung tâm dữ liệu, tất cả chúng được quản lý thông qua bảng điều khiển cho phép quản trị viên kiểm soát trong khi trao quyền cho người dùng của họ, cung cấp tài nguyên cho người dùng thông qua giao diện web.



Hình 1.4. Vị trí của OpenStack [4]

Hình 1.4 cho ta thấy, phía dưới là phần cứng đã được ảo hóa để chia sẻ tài nguyên cho người dùng, trên cùng là các ứng dụng mà ta sử dụng. Và OpenStack là phần ở giữa 2 phần trên.

OpenStack là một sự án mã nguồn mở dùng để triển khai private cloud và public cloud, nó bao gồm nhiều thành phần do các công ty, tổ chức, lập trình viên tự nguyên xây dựng và phát triển.

OpenStack hoạt động theo hướng mở, tức là công khai lộ trình phát triển, công khai mã nguồn,… Các phiên bản của OpenStack có chu kỳ 6 tháng, tức là 6 tháng một lần sẽ công bố phiển bản mới với các tính năng bổ sung. Tên các phiên bản được bắt đầu theo thứ tự A, B, C, … trong bảng chữ cái; đến nay - tháng 7 năm 2019, đã có 19 phiên bản được phát hành như Austin, Bexar, Cactus,… Phiên bản mới nhất hiện nay là **Stein**, dự kiến tháng 10/2019 sẽ ra mắt phiên bản **Train**.

1.2.2. Các thành phần projects trong OpenStack

Như đã giới thiệu ở trên, có thể coi OpenStack như một hệ điều hành cloud có nhiệm vụ kiểm soát các tài nguyên tính toán, lưu trữ và mạng trong hệ thống trung tâm dữ liệu, tất cả đều có thể được kiểm soát qua giao diện dòng lệnh hoặc một bảng điều khiển. OpenStack có 6 projects chính bao gồm: **NOVA, NEUTRON, SWIFT, CINDER, KEYSTONE, GLANCE**. Sau đây là thông tin về một số project quan trọng của OpenStack.

a. KEYSTONE (Identity Service)

Keystone cung cấp dịch vụ xác thực và ủy quyền cho các dịch vụ khác của OpenStack, cung cấp danh mục của các endpoints cho tất các dịch vụ trong OpenStack. Cụ thể hơn:

- Xác thực user và vấn đề token để truy cập vào các dịch vụ.
- Lưu trữ user và các tenant cho vai trò kiểm soát truy cập (cơ chế role-based access control - RBAC).
- Cung cấp catalog của các dịch vụ (và các API enpoints của chúng) trên cloud.
- Tạo các chính sách giữa user và dịch vụ.

b. NOVA (Compute service)

Nova quản lí các máy ảo trong môi trường OpenStack, chịu trách nhiệm khởi tạo, lập lịch, ngừng hoạt động của các máy ảo theo yêu cầu. Nova bao gồm nhiều tiến trình trên máy chủ, mỗi tiến trình lại thực hiện một chức năng khác nhau. Nova cung cấp REST API để tương tác với người dùng, các thành phần bên trong Nova truyền thông với nhau thông qua cơ chế truyền RPC message. Nova có thể sử dụng cơ sở dữ liệu được chia sẻ giữa tất các thành phần. Tuy nhiên, để hỗ trợ cập nhật, cơ sở dữ liệu được truy cập thông qua một object layer để đảm bảo các thành phần kiểm soát đã cập nhật có thể giao tiếp với nova-compute chạy ở phiên bản trước. Để làm điều này, nova-compute ủy quyền các yêu cầu tới cơ sở dữ liệu thông qua RPC tới một trình quản lý trung tâm, chính là dịch vụ nova-conductor. Cụ thể hơn, Nova có thể làm một số nhiệm vụ cụ thể như sau:

- Khởi tạo, thay đổi tài nguyên, dừng và truy cập đến máy ảo.
- Gán và xóa public IP.
- Gán và tháo các khối block
- Truy cập màn hình máy áo.
- Snapshot các máy ảo ở trạng thái đạng chạy hoặc ngủ.
- Nova hỗ trợ nhiều hypervisor như: KVM, VMware, Xen, Docker, etc.

Các thành phần của Nova:

- Dịch vụ nova-api: Tiếp nhận và trả lời các compute API call của người dùng cuối. Dịch vụ hỗ trợ OpenStack Compute API, Amazon EC2 API, và Admin API đặc biệt cho user thực hiện các tác vụ quản trị. Nó thực hiện một số chính sách và khởi tạo hầu hết các hoạt động điều phối, chẳng hạn như tạo máy ảo,…
- Dịch vụ nova-api-metadata: Tiếp nhận yêu cầu lấy metadata từ instances. Dịch vụ nova-api-metadata được sử dụng khi bạn chạy chế độ multi-host với nova-network.
- Dịch vụ nova-compute: Một daemon quản lý vòng đời máy ảo như tạo, hủy thông qua hypervisor APIs. Ví dụ: XenAPI cho XenServer/XCP, libvirt cho KVM hoặc QEMU, VMwareAPI cho VMware. Xử lý các tiến trính là khá phức tạp. Về cơ bản, daemon chấp nhận hành động từ queue và thực hiện một loạt các lệnh hệ thống như khởi tạo một KVM instance và cập nhật trạng thái của nó trong cơ sở dữ liệu.
- Dịch vụ nova-placement-api
- Dịch vụ nova-scheduler: Xác định máy ảo được yêu cầu từ queue sẽ đặt tại máy chủ compute nào.
- nova-conductor module: Trung gian tương tác giữa dịch vụ nova-compute và cơ sở dữ liệu. Nó loại bỏ truy cập trực tiếp vào cơ sở dữ liệu cloud được thực hiện với dịch vụ nova-compute nhằm mục đích bảo mật, tránh trường hợp máy ảo bị xóa mà không có chủ ý của người dùng.
- nova-consoleauth daemon: Ủy quyền token cho người dùng mà truy cập proxies cung cấp. Dịch vụ này phải chạy với console proxies để làm việc. Nhìn nova-novncproxy và nova-xvpvncproxy.
- nova-novncproxy daemon: Cung cấp proxy cho việc truy cập máy ảo đang chạy thông kết nối VNC.
- nova-xvpvncproxy daemon: Cung cấp proxy cho việc truy cập máy ảo đang chạy thông qua kết nối VNC. Hỗ trợ OpenStack-specific Java client.
- Queue: Một trung tâm trung chuyển các thông điệp giữa các daemons. Thường sử dụng RabbitMQ, cũng có thể thực hiện với một AMQP message queue khác là ZeroMQ.
- Cơ sở dữ liệu SQL: Lưu trữ hầu hết các trạng thái tại buil-time và run-time của cơ sở hạ tầng cloud, bao gồm: các loại instance đang có sẵn, instances đang sử dụng, các mạng có sẵn.

c. NEUTRON (Networking Service)

OpenStack Networking xử lý tạo và quản lý cơ sở hạ tầng mạng ảo, bao gồm mạng, switches, mạng con, và router cho thiết bị được quản lý bởi dịch vụ OpenStack Compute (Nova). Các dịch vụ nâng cao như firewalls hoặc virtual private networks (VPNs) có thể được sử dụng. Một số thông tin về Neutron:

- Các phiên bản trước Grizzly tên là Quantum, sau đổi tên thành Neutron
- Cung cấp kết nối mạng như một dịch vụ cho các dịch vụ khác của OpenStack.
- Cung cấp API cho người dùng để họ tạo các network của riêng mình.
- Ngoài ra nó cũng cung cấp thêm các dịch vụ mạng khác như: FWaaS (Firewall as a service), LBaaS (Load balancing as a servie), VPNaaS (VPN as a service),...

d. GLANCE (Image Service)

Glance lưu trữ và truy xuất các images của máy ảo của người dùng và các dịch vụ cloud khác. OpenStack compute sẽ sử dụng chúng để khởi tạo máy áo. Các tính năng chính:

- Người quản trị tạo sẵn mẫu để người dùng có thể tạo máy ảo nhanh chóng.
- Người dùng có thể tạo máy ảo từ ổ đĩa ảo có sẵn. Glance chuyển images tới Nova để vận hành máy ảo.
- Snapshot từ các máy ảo đang chạy có thể được lưu trữ, vì vậy máy ảo đó có thể được back up.

e. SWIFT (Object Storage Service)

SWIFT cung cấp giải pháp lưu trữ dữ liệu phi cấu trúc. Cụ thể hơn, Swift cung cấp các một số chức năng như:

- Lưu trữ và thu thập các đối tượng (các files).
- Thiết lập và chỉnh sửa metadata trên đối tượng(tags).
- Đọc, ghi các đối tượng thông qua HTTP.

f. CINDER (Block Storage Service)

- Cung cấp các khối lưu trữ bền vững (volume) để chạy các máy ảo (instances).
- Có thể gán và gỡ một volume từ máy ảo này gắn sang máy ảo khác.
- Có thể sao lưu, mở rộng các volume.

Ngoài 6 projects chính trên, còn có project **HORIZON (Dashboard Service)** cung cấp giao diện nền web cho người dùng cuối và người quản trị cloud để tương tác với các dịch vụ khác của OpenStack, ví dụ như vận hành các máy ảo, cấp phát địa chỉ IP và kiểm soát

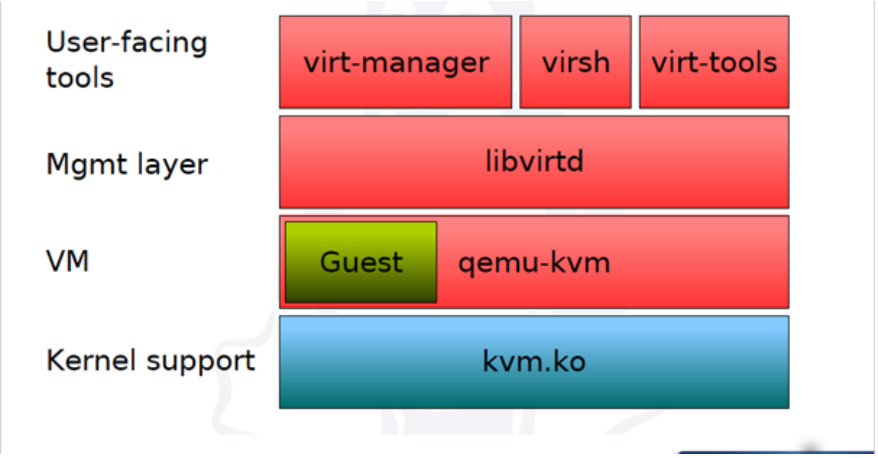
cấu hình truy cập các dịch vụ. HORIZON viết dựa trên python django framework. Một số thông tin mà giao diện người dùng cung cấp cho người sử dụng:

- Thông tin về quota và cách sử dụng
- Quản lý các volume: điều khiển khởi tạo, hủy kết nối tới các khối block.
- Images and Snapshots: tải lên các images có sẵn.

1.2.3. Kernel-based virtual machine (KVM)

KVM (Kernel-based virtual machine) là giải pháp ảo hóa cho hệ thống linux trên nền tảng phần cứng x86 có các module mở rộng hỗ trợ ảo hóa (Intel VT-x hoặc AMD-V).

Về bản chất, KVM không thực sự là một hypervisor có chức năng giả lập phần cứng để chạy các máy ảo. Chính xác KVM chỉ là một module của kernel linux hỗ trợ cơ chế mapping các chỉ dẫn trên CPU ảo (của máy ảo) sang chỉ dẫn trên CPU vật lý (của máy chủ chứa máy ảo). Hoặc có thể hình dung KVM giống như một driver cho hypervisor để sử dụng được tính năng ảo hóa của các vi xử lý như Intel VT-x hay AMD-V, mục tiêu là tăng hiệu suất cho máy ảo.



Hình 1.5. KVM Stack [5]

Hình 1.5 mô tả KVM Stack bao gồm 4 tầng:

- User-facing tools: Là các công cụ quản lý máy ảo hỗ trợ KVM. Các công cụ có giao diện đồ họa (như virt-manager) hoặc giao diện dòng lệnh như (virsh).
- Management layer: Lớp này là thư viện libvirt cung cấp API để các công cụ quản lý máy ảo hoặc các hypervisor tương tác với KVM thực hiện các thao tác quản

lý tài nguyên ảo hóa, vì tự thân KVM không hề có khả năng giả lập và quản lý tài nguyên như vậy.

- Virtual machine: Chính là các máy ảo người dùng tạo ra. Thông thường, nếu không sử dụng các công cụ như virsh hayvirt-manager, KVM sẽ sử được sử dụng phối hợp với một hypervisor khác điển hình là QEMU.
- Kernel support: Chính là KVM, cung cấp một module làm hạt nhân cho hạ tầng ảo hóa (kvm.ko) và một module kernel đặc biệt hỗ trợ các vi xử lý VT-x hoặc AMD-V (kvm-intel.ko hoặc kvm-amd.ko)

Hệ thống ảo hóa KVM hay đi liền với QEMU. Về mặt bản chất, QEMU là một emulator. QEMU có khả năng giả lập tài nguyên phần cứng, trong đó bao gồm một CPU ảo. Các chỉ dẫn của hệ điều hành tác động lên CPU ảo này sẽ được QEMU chuyển đổi thành chỉ dẫn lên CPU vật lý nhờ một translator là TCG(Tiny Core Generator) nhưng TCG hiệu suất ko cao.

Do KVM hỗ trợ ánh xạ CPU vật lý sang CPU ảo, cung cấp khả năng tăng tốc phần cứng cho máy ảo và hiệu suất của nó nên QEMU sử dụng KVM làm accelerator tận dụng tính năng này của KVM thay vì sử dụng TCG.

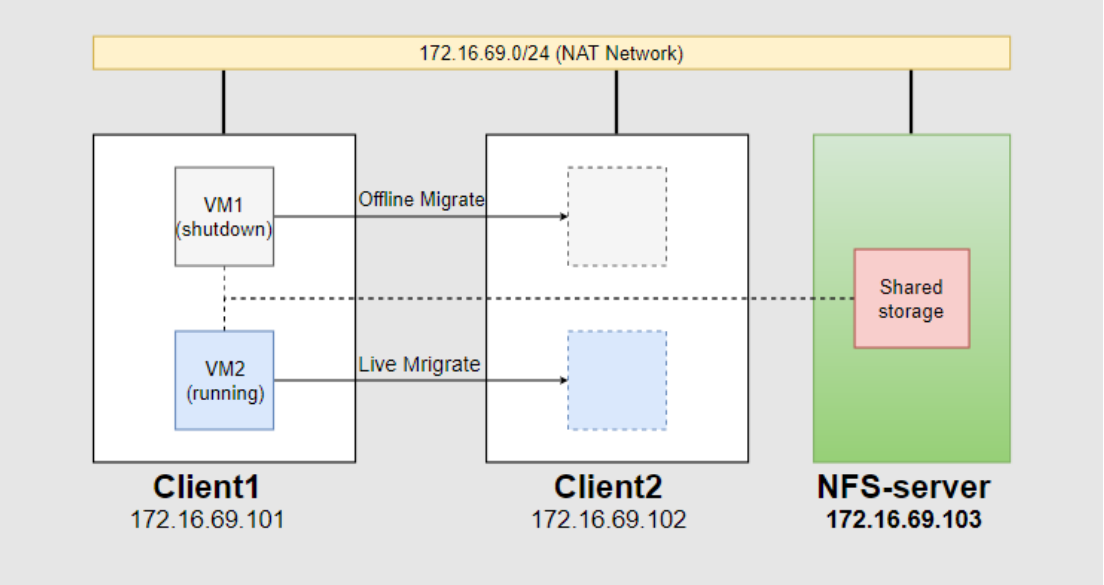
Migrate là chức năng được KVM-QEMU hỗ trợ, nó cho phép di chuyển các máy ảo từ một máy chủ vật lý này sang máy chủ vật lý khác và không ảnh hướng để máy ảo đang chạy cũng như dữ liệu bên trong nó.

Migrate giúp cho nhà quản trị có thể di chuyển các máy ảo trên máy chủ đi để phục vụ cho việc bảo trì và nâng cấp hệ thống, nó cũng giúp nhà quản trị nâng cao tính dự phòng, và cũng có thể làm nhiệm vụ cân bằng tải cho hệ thống khi một máy chủ quá tải.

Migrate có 2 cơ chế:

- Cơ chế Offline Migrate: là cơ chế cần phải tắt máy ảo đi thực hiện việc di chuyển image và file xml của máy ảo sang một máy chủ khác.
- Cơ chế Live Migrate: đây là cơ chế di chuyển guest khi máy ảo vẫn đang hoạt động, quá trình trao đổi diễn ra rất nhanh các phiên làm việc kết nối hầu như không cảm nhận được sự gián đoạn nào. Quá trình Live Migrate được diễn ra như sau: Bước đầu tiên của quá trình Live Migrate 1 ảnh chụp ban đầu của máy ảo trên máy chủ 1 được chuyển sang máy chủ 2. Trong trường hợp người dùng đang truy cập tại máy chủ 1 thì những sự thay đổi và hoạt động trên máy chủ 1 vẫn diễn ra bình thường, tuy nhiên những thay đổi này sẽ được ghi nhận. Những thay đổi trên máy chủ 1 được đồng bộ liên tục đến máy chủ 2. Khi đã đồng bộ

xong thì máy ảo trên máy chủ 1 sẽ offline và các phiên truy cập trên máy chủ 1 được chuyển sang máy chủ 2.



Hình 1.6. Offline Migrate và Linve Migrate với KVM

Hình 1.6 mô tả trực quan 2 cơ chế của Migrate. Trong đó, với cơ chế Offline Migrate VM1 cần tắt sau đó mới thực hiện migrate. Còn với cơ chế Live Migrate, VM2 đang chạy vẫn thực hiện migrate được.

1.3. Tổng quan về công nghệ ảo hóa chức năng mạng (Network Functions Virtualization - NFV)

Network Functions Virtualization (NFV) là việc ảo hóa các chức năng mạng như Firewall, NAT, Load blancer,… để đạt được tính linh động cao cũng như giảm chi phí giá thành và bớt phụ thuộc vào các nhà cung cấp thiết bị phần cứng mạng.

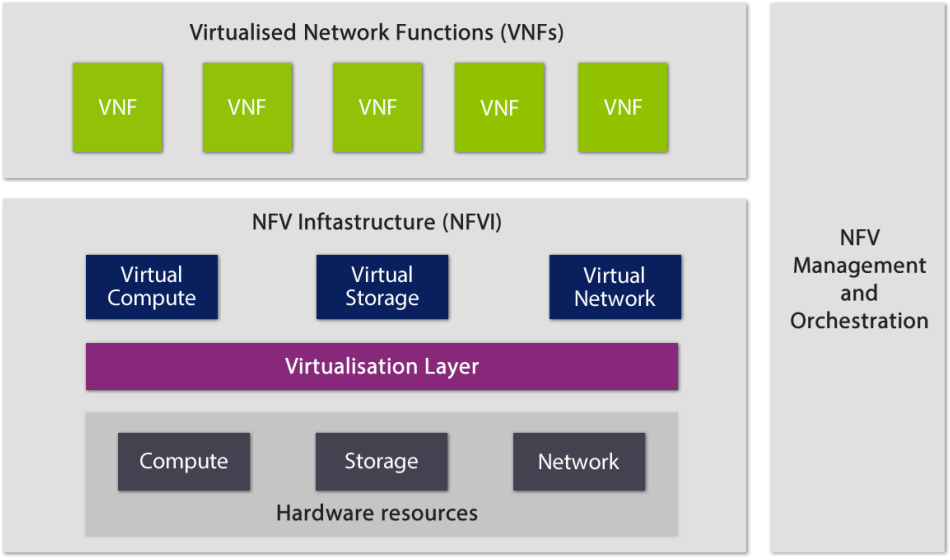
Trong hệ thống mạng truyền thống, nếu nhà cung cấp dịch vụ muốn mở rộng quy mô kinh doanh, hay mở rộng hệ thống mạng hiện tại, thì cần phải lắp đặt hàng loạt các thiết bị đầu cuối như router, switch, monitor,…

NFV ra đời giúp các nhà cung cấp dịch vụ mạng giải quyết các vấn đề của hệ thống truyền thống bằng cách ảo hóa các chức năng mạng bằng cách sử dụng các phần mềm được cài đặt trên các máy ảo hoạt động trên các máy chủ vật lý. Qua đó, hạ tầng mạng chuyển từ việc sử dụng toàn bộ thiết bị phần cứng chuyên dụng sang sử dụng một phần thiết bị phần cứng chuyên dụng và một phần sử dụng các mảy chủ vật lý.

Định nghĩa về NFV bắt nguồn từ các nhà cung cấp dịch vụ - những người đang tìm kiếm giải phát để thúc đẩy nhanh hơn việc triển khai các dịch vụ mạng mới, thu về lợi nhuận. Những hạn chế của các thiết bị phần cứng đòi hỏi họ phải áp dụng các công nghệ ảo hóa vào hệ thống mạng của họ. Vì chung mục đích như vậy, nhiều nhà cung cấp dịch vụ đã hợp tác với nhau và thành lập nên ETSI (European Telecommunications Standards Institute - 1988). Trong đó ETSI ISG NFV (ETSI Industry Specification Group for Network Functions Virtualization), là nhóm có nhiệm vụ phát triển các yêu cầu và kiến trúc để áp dụng ảo hóa cho nhiều chức năng trong hệ thống mạng viễn thông. ETSI ISG NFV vận hành từ tháng 1 năm 2013, mang 7 nhà mạng viễn thông hàng đầu đến với nhau: AT&T, BT, Deutsche Telekom, Orange, Telecom Italia, Telefonica, và Verizon. Ngoài ra còn có sự tham gia của 52 nhà mạng, các nhà cung cấp thiết bị viễn thông, IT vendors khác. Không lâu sau đó, cộng đồng ETSI ISG NFV mở rộng với 230 công ty, bao gồm nhiều nhà cung cấp dịch vụ toàn cầu.

Kiến trúc NFV ở mức high level gồm 3 miền làm việc chính:

- VNF - Virtualised Network Functions: là các chức năng mạng hay thiết bị mạng ảo triển khai dưới dạng phần mềm trên hạ tầng NFV
- NFVI - NFV Infrastructure: bao gồm các tài nguyên vật lý và công nghệ ảo hóa hỗ trợ để cung cấp tài nguyên triển khai VNF.
- NFV Management and Orchestration: thực hiện điều phối, quản lý vòng đời các tài nguyên phần cứng và tài nguyên phần mềm hỗ trợ hạ tầng ảo hóa, quản lý vòng đời các VNFs.

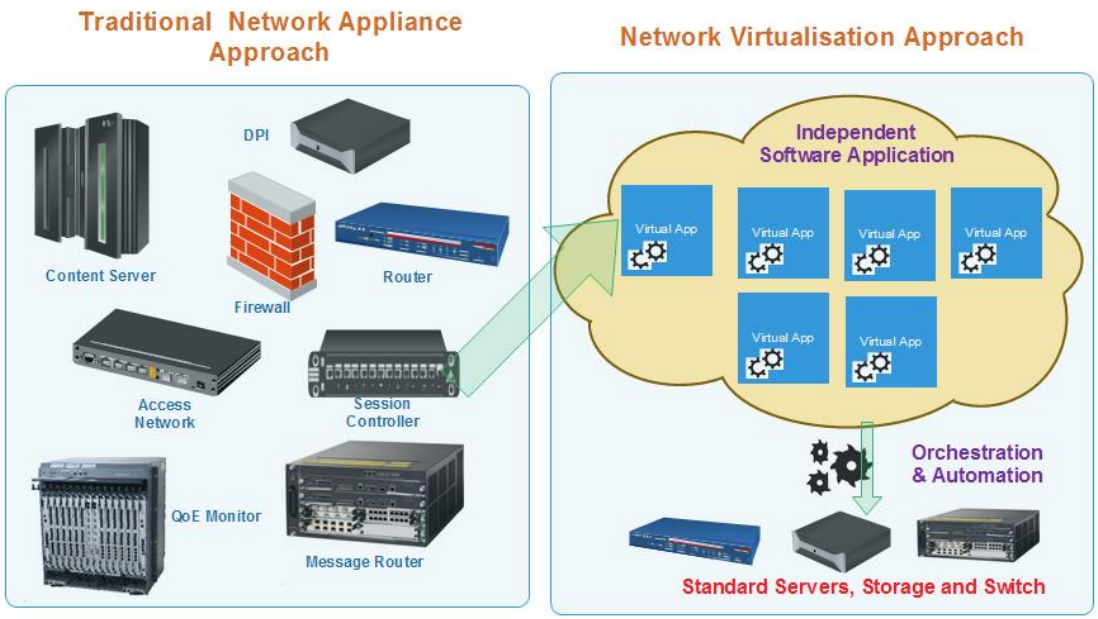


Hình 1.7. Kiến trúc NFV [6]

Những lợi ích của NFV:

- **Giảm chi phí phần cứng:** Chi phí thấp hơn do chỉ sử dụng phần cứng phổ thông, đồng thời chủ động được về phần mềm.
- **Tăng khả năng mở rộng, thay thế phần cứng:** Dễ dàng thay thế do chỉ sử dụng các thiết bị phần cứng phổ thông.
- **Tăng khả năng quản trị, thay thế, năng cấp phần mềm:** Do cơ chế nguồn mở và có nhiều hãng cung cấp phần mềm điều khiển.
- **Tăng khả năng điều khiển luồng dữ liệu:** Dễ dàng, linh động, đặc biệt là khi kết hợp với công nghệ SDN (Sofware defined Networking).
- **Mở rộng hệ sinh thái:** Dễ dàng tương tác với các thành phần của các hãng khác thông qua các chuẩn giao tiếp chung được ETSI (European Telecommunications Standards Institute) đặt ra.

Ngoài những lợi ích trên thì NFV có một số nhược điểm như hiệu năng, độ ổn định thấp do sử dụng phần cứng phổ thông,…



Hình 1.8. Công nghệ ảo chức năng mạng [6]

Hình 1.8 cho thể thấy được các chức năng như Firewall, DPI (Deep Packet Inspection), Access Network,… đã được ảo hóa thành phần mềm tương ứng và cài đặt trên những máy ảo. Những máy ảo này nằm trên những máy chủ vật lý được cài đặt phần mềm ảo hóa.

Công nghệ ảo hóa chức năng mạng (NFV) có rất nhiều ứng dụng thực tế. Sau đây là 3 trường hợp phổ biến sử dụng NFV:

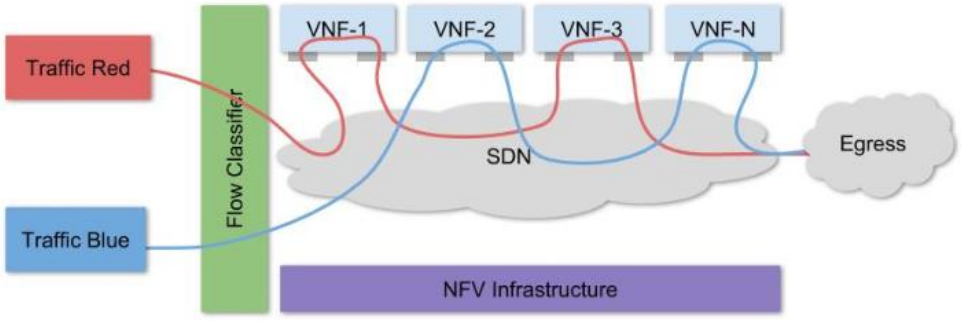
1. Ảo hóa chức năng mạng (Hạ tầng như một dịch vụ): Các nhà cung cấp dịch vụ mạng cung cấp cho khách hàng hạ tầng ảo hóa chức năng mạng, ví dụ như dịch vụ điện toán đám mây.
2. Ảo hóa chức năng mạng như một dịch vụ (NFV as a Service): Các nhà cung cấp dịch vụ mạng đầu tư, vận hành các thiết bị mạng chuyên dụng (Router, Firewall,…) khá tốn kém tiền bạc và nhân sự. Vì vậy, các nhà cung cấp mạng viễn thông sử dụng công nghệ ảo hóa chức năng mạng CPE dưới dạng điện toán đám mây theo mô hình Software as a Service.
3. Virtual Network Platform as a Service: Ảo hóa chức năng mạng tăng độ linh hoạt và giảm chi phí thiết lập và quản lý tài nguyên. Các nhà cung cấp dịch vụ mạng sẽ cung cấp nền tảng theo mô hình PaaS.

1.5. Tổng quan về chuỗi chức năng mạng (Service Function Chaining - SFC)

1.5.1. Giới thiệu về Service Function Chaining

Trong lịch sử viễn thông đã có các thiết bị vật lý thực hiện một số loại chức năng chuyên dụng. Các thiết bị này đắt tiền và rất khó để cấu hình lại về hành vu kết nối mạng của chúng. Ví dụ điển hình là một số thiết bị vật lý được kết nối trực tiếp với nhau trong một chuỗi.

Về cơ bản, SFC là khả năng khiến các luồng lưu lượng mạng được định tuyến qua mạng thông qua một đường dẫn khác với đường dẫn sẽ được chọn bằng cách tra bảng định tuyến dựa trên địa chỉ IP đích của gói tin. Nó thường được sử dụng cùng với công nghệ ảo hóa chức năng mạng khi triển khai trong môi trường ảo một loạt các chức năng mạng thường được triển khai như một tập hợp các thiết bị mạng vật lý được kết nối nối tiếp nhau bằng dây cáp mạng.



Hình 1.9. Chuỗi chức năng mạng [7]

Hình 1.9 có 2 luồng dữ liệu đỏ và xanh da trời trong mạng. Với luồng dữ liệu đỏ, lưu lượng đi qua VNF-1 và VNF-3 sau đó đi ra ngoài chuỗi dịch vụ mạng. Với luồng dữ liệu xanh da trời, lưu lượng đi qua VNF-2 và VNF-N sau đó đi ra ngoài chuỗi dịch vụ mạng.

1.5.2. Những khái niệm thông dụng trong SFC

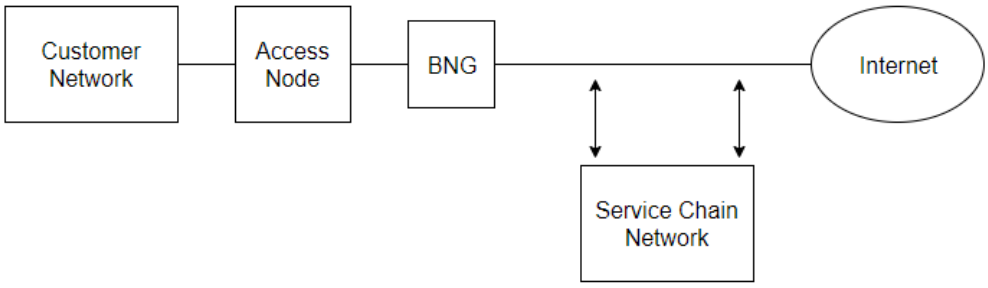
- Endpoint (EP): Là thiết bị hoặc ứng dụng ở đầu cuối. Ví dụ như điện thoại di động, laptop hoặc máy tính bàn,...
- Service Function (SF): Một chức năng mạng chịu trách nhiệm xử lý các gói tin nhận được. Ví dụ như monitor, firewall,…
- Classifier (CF): Là phần tử chịu trách nhiệm lựa chọn, phân loại lưu lượng cũng như các chuỗi dịch vụ mạng, dựa trên các chính sách và chuyển tiếp lưu lượng nhận được trên chuỗi dịch vụ mạng.

1.5.3. Các kịch bản sử dụng của SFC trong thực tế

Chuỗi chức năng mạng (SFC) có thể được triển khai trong nhiều tình huống như mạng băng thông rộng, mạng di động, và trung tâm dữ liệu. Phần này mô tả một tập hợp các kịch bản để triển khai SFC.

1) Service Function Chaining trong mạng băng thông rộng cố định (Fixed Broad Networks)

Trong mạng băng thông rộng, người dùng có thể được truy cập vào mạng thông qua các công nghệ khác nhau, thường bao gồm DSL, Ethernet và PON. Dù công nghệ trung cập là gì, kiến trúc của mạng truy cập cũng bao gồm Access Nodes (ANs) và Broadband Network Gateways (BNGs), trong đó AN thường là thiết bị cung cấp truy cập vào mạng cho khách hàng có quyền truy cập vào BNG là nút IP đầu tiên và cung cấp ủy quyền, xác thực và tính toán cho thuê bao.



Hình 1.10. Service Function Chain in Fixed Broadband Networks

2) Service Function Chain trong mạng di động (Mobile Networks)

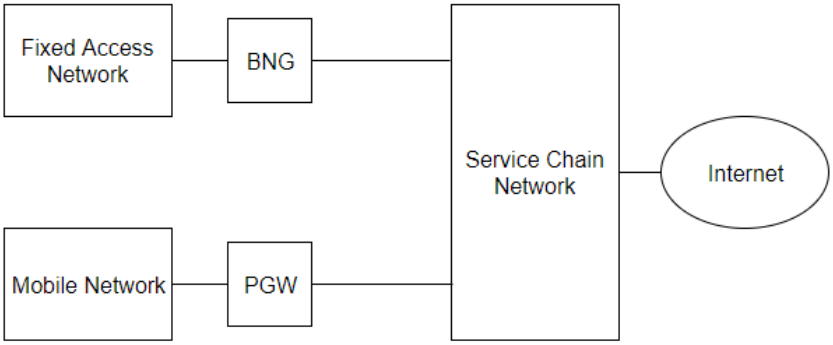
Lưu lượng truy cập từ Internet đi qua một hoặc nhiều Service Functions. Ví dụ như các chức năng dịch vụ giá trị gia tăng như dịch vụ kiểm soát (Parental Control) hoặc Tường lửa (Firewall),DPI (Deep Packet Intrustion),... Các thuê bao có thể chủ động đăng kí hoặc hủy bỏ các dịch vụ này bất kì lúc nào mà không cần tới sự can thiệp với nhà cung cấp dịch vụ.

3) Mạng chuỗi dịch vụ chung

Từ hai trường hợp sử dụng trên, chúng ta có thể thấy sự tương đồng trong chuỗi dịch vụ mạng. Mặc dù mạng băng thông cố định và mạng băng thông rộng được triển khai riêng rẽ, nhưng đối với các nhà khai thác tích hợp chạy cả 2 loại mạng này, rõ ràng có lợi khi cung cấp dịch vụ cho cả hai mạng từ một mạng chuỗi dịch vụ chung.

Ngoài việc tối ưu hóa tài nguyên, mạng chuỗi dịch vụ chung cũng có thể cho phép chuyển đổi dịch vụ từ mạng này sang mạng khác. Ví dụ: một khách hàng đang truy cập Facebook trên điện thoại di động của mình thông qua 4G, sau khi về nhà, khách hàng đó có thể chuyển sang sử dụng mạng WiFi ở nhà. Trong trường hợp này, việc cung cấp dịch vụ không bị gián đoạn khi sử dụng mạng chuỗi dịch vụ chung.

SFC có thể được sử dụng như một công cụ để quải quyết tốt hơn như cầu hội tụ (bao gồm điều chỉnh việc cung cấp dịch vụ cho các rằng buộc của mạng truy cập hoặc khả năng của các thiết bị được kết nối).



Hình 1.11. Common Service Chain Network

Hình trên minh họa một mạng chuỗi dịch vụ được chia sẻ bởi cả 2 mạng băng thông rộng cố định và mạng di động.

4) Chuỗi chức năng dịch vụ phân tán

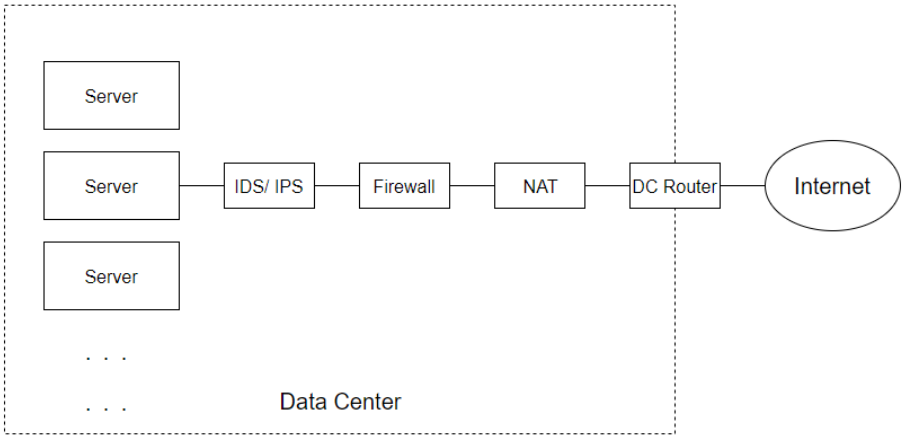
Bên cạnh các trường hợp sử dụng triển khai được nêu ra ở trên, SFC không nhất phải triển khai ở một vị trí duy nhất mà cũng có thể được phân tán qua một số phần của mạng (ví dụ: trung tâm sữ liệu) hoặc thâm chí sử dụng chức năng dịch vụ được đặt gần phía khách

hàng (ví dụ: một số chức năng bảo mật). Nhiều tên miền SFC có thể được kích hoạt trên cùng một tên miền hành chính.

5) Chuỗi chức mạng trong trung tâm dữ liệu

Trong trung tâm dữ liệu, ta có thể khai chuỗi chức năng dịch vụ để cung cấp các dịch vụ giá trị gia tăng.

Hình dưới minh họa kịch bản có thể xảy ra trong trung tâm dữ liệu, chức năng mạng được đặt giữa máy chủ và bộ định tuyến mạng. Từ máy chủ đến internet, có các chức năng dịch vụ như IDS/IPS, Firewall, NAT được xếp tuần tự.



Hình 1.12. Service Function Chain in Data Center

Hình 1.12 mô tả một chuỗi dịch vụ mạng gồm IDS/IPS, Firewall và NAT, yêu cầu truy cập đến Server phải đi qua DC Router, sau đó lần lượt đi qua chức năng NAT, chức năng Firewall, sau đó là chức năng IDS/IPS rồi mới đi vào Server.

1.6. Tổng quan về Data Plance Developmetnt Kit - DPDK

1.6.1. Giới thiệu về DPDK

Data Plane Development Kit (DPDK) bao gồm các thư viện để tăng tốc công việc xử lý gói tin đang chạy trên nhiều kiến trúc CPU.

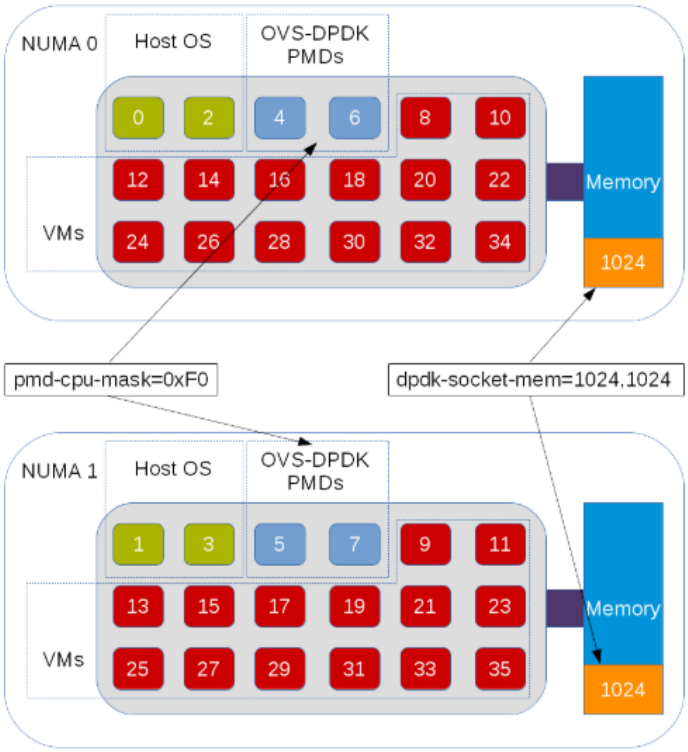
Trong thế giới mà mạng đang trở thành nền tảng cho cách mọi người giao tiếp; hiệu suất, thông lượng và độ trễ ngày càng quan trọng đối với các ứng dụng như cơ sở hạ tầng mạng, router, load balancers, firewall, video streaming, VoIP, và nhiều hơn thế nữa. Bằng cách cho phép xử lý gói tin nhanh, DPDK làm cho ngành viễn thông có thể di chuyển các ứng dụng hiệu suất cao đến cloud.

DPDK được tạo ra trong năm 2010 bởi Intel và được cung cấp theo giấy phép mã nguồn mở. Kể từ đó, cộng đồng đã không ngừng phát triển về số lượng người đóng góp, bản vá và các tổ chức đóng góp, với 5 bản phát hành chính đã hoàn thành bao gồm đóng góp từ hơn 160 cá nhân và 25 tổ chức khác nhau. DPDK hiện hỗ trợ tất cả các kiến trúc CPU và NIC chính từ nhiều nhà cung cấp, điều này làm cho nó phù hợp với các ứng dụng cần phải di động trên nhiều nền tảng.

Các loại CPU và card mạng được DPDK hỗ trợ:

- DPDK ban đầu hỗ trợ CPU Intel x96, và hiện được mở rộng cho IBM Power 8, EZchip TILE-GX và ARM.
- Các loại NIC được DPDK hỗ trợ: AMD (axgbe), Broadcome (bnxt), Cisco (enic), Intel (e1000, ixgbe, i40e, ice, fm10k, ipn3ke, ifc), Software NICs (af_packet, af_xdp, tap, pcap, ring),…

Hiện nay, người ta thường tích hợp DPDK và Open vSwitch để nâng cao hiệu năng chuyển tiếp gói tin trong các trung tâm dữ liệu.

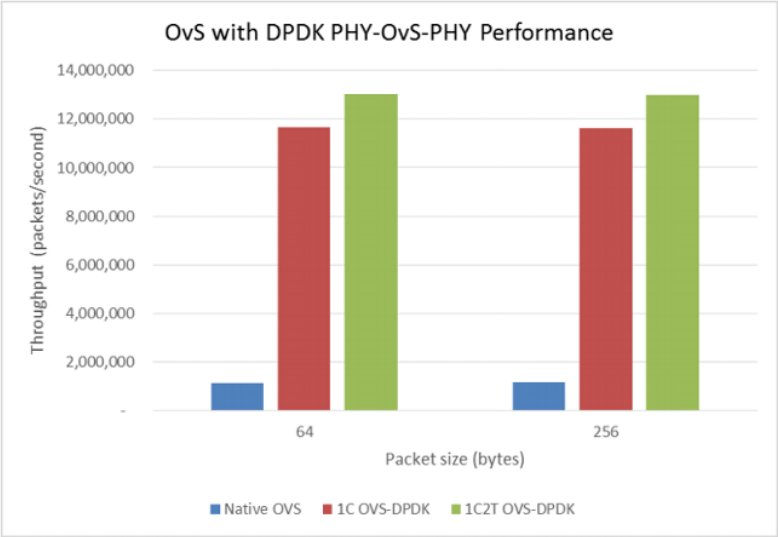


Hình 1.13. Sử dụng DPDP để phân bổ tài nguyên [8]

Hình 1.13 mô tả sự phân chia tài nguyên CPU và bộ nhớ trên một máy chủ gồm 2 socket (tương ứng với NUMA 0 và NUMA 1). Trong đó, phân chia cho Open vSwitch sử dụng 4

CPU (CPU 4,5,6,7) và 2048 MB RAM. Máy ảo được phân chia sẻ dụng 28 CPU, còn lại máy chủ sử dụng 4 CPU.

1.6.2. Hiệu năng giữa Open vSwitch và Open vSwitch có kết hợp DPDK



Hình 1.14. Biểu đồ hiệu năng giữa Open vSwitch và Open vSwitch tích hợp DPDK [8]

Hình 1.14 cho thấy hiệu năng của Open vSwitch có kích hoạt DPDK tốt hơn hẳn Open vSwitch chính gốc.

Theo bài viết trên trang chủ Intel, thông lượng gói tin/s của Open vSwitch có tích hợp DPDK tăng gần 10 lần so với Open vSwitch nguyên gốc; đối với công nghệ Intel Hyper-Threading được kích hoạt, hiệu năng tăng gần 12 lần.

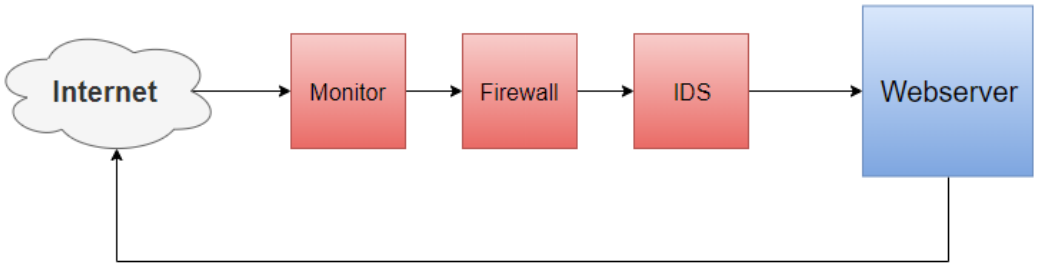
1.7. Kết luận

Chương 1 này, em đã trình bày tổng quan về công nghệ điện toán đám mây (Cloud Computing), phần mềm mã nguồn mở OpenStack, công nghệ ảo hóa chức năng mạng (Network Functions Chaining), chuỗi chức năng mạng (SFC) và các kịch bản sử dụng trong thực tế, Data Plane Development Kit (DPDK). Qua đó, em lựa chọn triển khai chuỗi chức năng mạng ảo trong trung tâm dữ liệu; cụ thể đồ án này là bảo vệ máy chủ web server.

Chuỗi chức năng mạng ảo gồm 3 thành phần chính là chức năng giám sát mạng (Monitoring), tưởng lửa (Firewall) có tác dụng chặn, lọc dữ liệu ở tầng 3 và 4; chức năng hệ thống phát hiện xâm nhập NIDS có khả năng phát hiện mã độc.

CHƯƠNG 2. TRIỂN KHAI CHUỖI CHỨC NĂNG MẠNG ẢO

Chương 2 này trình bày cách thức triển khai chuỗi chức năng mạng ảo đã đề cập ở chương 1. Ta triển khai sử dụng nền tảng OpenStack để triển khai chuỗi chức năng mạng có mô hình logic như hình bên dưới.



Hình 2.1. Mô hình chuỗi chức năng mạng mức logic

Hình trên mô tả mô hình chuỗi chức năng mạng gồm 3 chức năng mạng gồm:

- **Monitor** (giám sát lưu lượng mạng) dùng phần mềm **Ntopng**.
- **Firewall** (tưởng lửa) dùng phần mềm **Iptables**.
- **IDS** (hệ thống phát hiện xâm nhập) sử dụng phần mềm **Suricata**.

Ngoài ra còn có máy chủ Webserver sử dụng phần mềm **Nginx**.

Người dùng sẽ truy cập đến máy chủ Webserver thông qua mạng Internet, lúc này lưu lượng sẽ được chuyển hướng đi qua các chức năng mạng Monitor, Firewall, sau đó là IDS rồi cuối cùng mới đến máy chủ Webserver.

Tại chức năng mạng Monitor, người quản trị mạng có thể thông qua giao diện web của chức năng này để xem lưu lượng đi của mạng, đồng thời phát hiện lưu lượng tăng giảm bất thường hay không,…

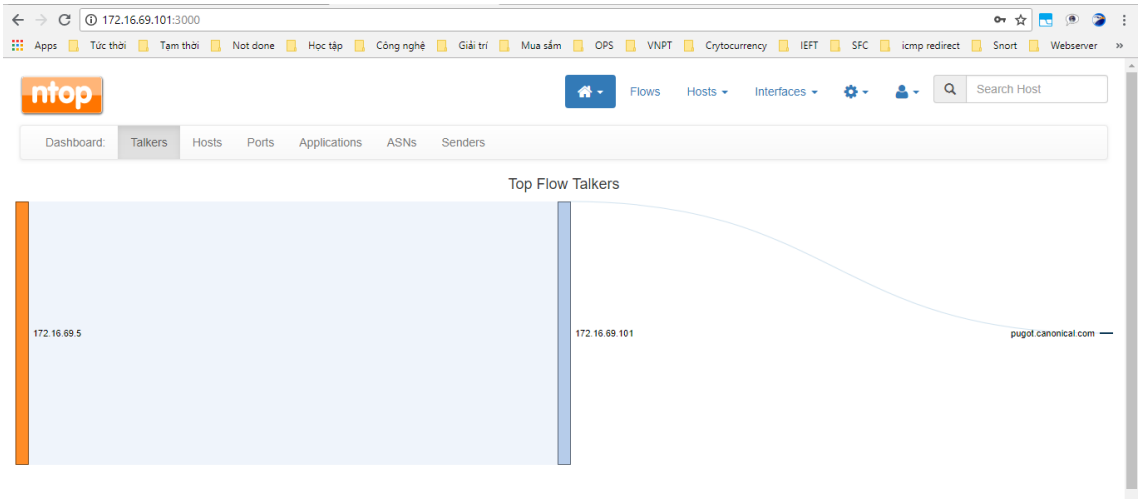
Chức năng Firewall và IDS sẽ kết hợp thành một bộ công cụ hoàn chỉnh để ngăn chặn các cuộc tấn vào máy chủ Webserver.

2.1. Giới thiệu một số công nghệ sử dụng làm các chức năng mạng

2.1.1. Giám sát lưu lượng mạng (Monitoring) - Ntopng

Giám sát mạng là việc sử dụng một hệ thống để liên tục theo dõi một mạng máy tính, xem xét coi có các thành phần hoạt động chậm lại hoặc không hoạt động và thông báo cho quản trị viên mạng (qua email, tin nhắn SMS hoặc các báo động khác) trong trường hợp mạng không hoạt động hoặc có các rắc rối khác.

Ntopng là công cụ mã nguồn mở được sử dụng để giám sát các giao thức mạng khác nhau trong hệ thống mạng của bạn. Ntopng cung cấp giao diện web thân thiện với người dùng để lấy thông tin về lưu lượng và trạng thái hệ thống mạng.



Hình 2.2. Giao diện web của Ntopng

Công cụ Ntopng cho phép người dùng theo dõi được lưu lượng mạng, đồng thời thống kê cho người dùng địa chỉ IP nguồn, địa chỉ port nguồn, giao thức truy cập mạng của các gói tin. Ntopng cũng có chức năng giám sát và báo cáo thông lượng trực tiếp, đỗ trễ của mạng và ứng dụng; thống kê TCP (truyền lại, truyền gói không theo thứ tự, gói bị mất). Đồng thời, Ntopng cũng hỗ trợ xuất dữ liệu theo dõi.

Ntopng hỗ trợ đa số các nền tảng phổ biến như Unix (Linux), Windows x64,… Hiện nay, nó có 3 phiên bản:

- **Community**: đây là phiên bản dành cho cộng đồng và hoàn toàn miễn phí.
- **Professional**: đây là phiên bản dành cho doanh nghiệp nhỏ, phiên bản này có mất phí.
- **Enterprise**: đây là phiên bản dành cho doanh nghiệp và có tính phí.

2.1.2. Tưởng lửa (Firewall) - Iptables

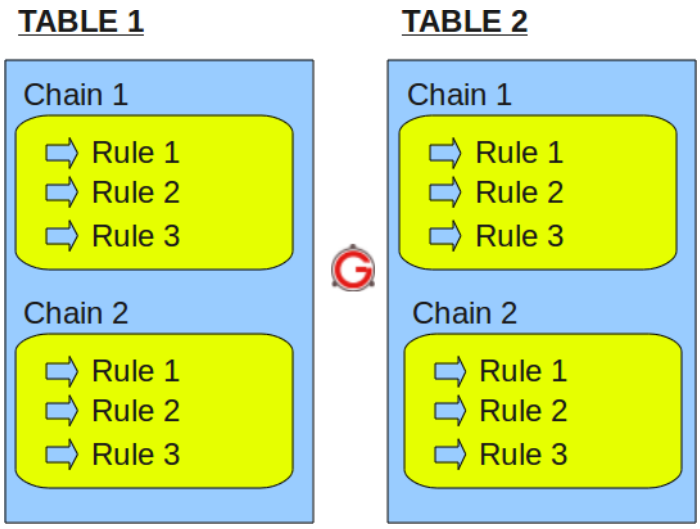
Iptables là chương trình tiện ích user-space cho phép người quản trị hệ thống cấu hình tables được cung cấp bởi Linux kernel firewall (được thực hiện bởi các modules Netfilters khác nhau) và chains và rules nó lưu trữ. Modules kernels và chương trình khác nhau được sử dụng cho các giao thức khác nhau: iptables áp dụng cho IPv4, ip6tables áp dụng cho IPv6. Netfilter/Iptables được tích hợp vào Linux kernel 2.3.x. Trước iptables, Linux dùng ipchains trong Linux kernel 2.2.x và ipfwadm trong Linux kernel 2.0.x. Iptables yêu cầu quyền ưu tiên được vận hành và phải được thực hiện bởi người dùng root.

Iptables thuộc loại tưởng lửa Stateful Packet Filtering. Với mọi packet đi vào mà bộ lọc có thể biết được quan hệ của chúng như thế nào đối với packet đi trước hoặc đi sau nó, ví dụ như các trạng thái bắt tay 3 bước trước khi thực hiện 1 phiên kết nối giao thức TCP/IP (SYN, SYN/ACK, ACK) gọi là firewall có thể phân biệt được trạng thái của các packet. Với loại firewall này, chúng ta có thể xây dựng các quy tắc lọc để có thể ngăn chặn được ngay cả các kiểu tấn công phá hoại như SYN flooding hay Xmas tree...

Hơn thế nữa Iptables còn hỗ trợ khả năng giới hạn tốc độ kết nối đối với các kiểu kết nối khác nhau từ bên ngoài, cực kỳ hữu hiệu để ngăn chặn các kiểu tấn công từ chối phục vụ (DoS) mà hiện nay vẫn là mối đe doạ hàng đầu đối vói các website trên thế giới. Một đặc điểm nổi bật nữa của Iptables là nó hỗ trợ chức năng dò tìm chuỗi tương ứng (string pattern matching), chức năng cho phép phát triển firewall lên một mức cao hơn, có thể đưa ra quyết định loại bỏ hay chấp nhận packet dựa trên việc giám sát nội dung của nó. Chức năng này có thể được xem như là can thiệp được đến mức ứng dụng như HTTP, TELNET, FTP... mặc dù thực sự Netfilter Iptables vẫn chỉ hoạt động ở mức mạng (lớp 3 theo mô hình OSI 7 lớp).

Kiến trúc của Iptables:

- Iptables chứa nhiều tables.
- Tables chứa nhiều chains. Chains có thể được tích hợp sẵn hoặc do người dùng định nghĩa.
- Chains chứa nhiều rules. Rules được định nghĩa cho các gói tin.



Hình 2.3. Kiến trúc Iptables [9]

Iptables có 4 tables được xây dựng sẵn.

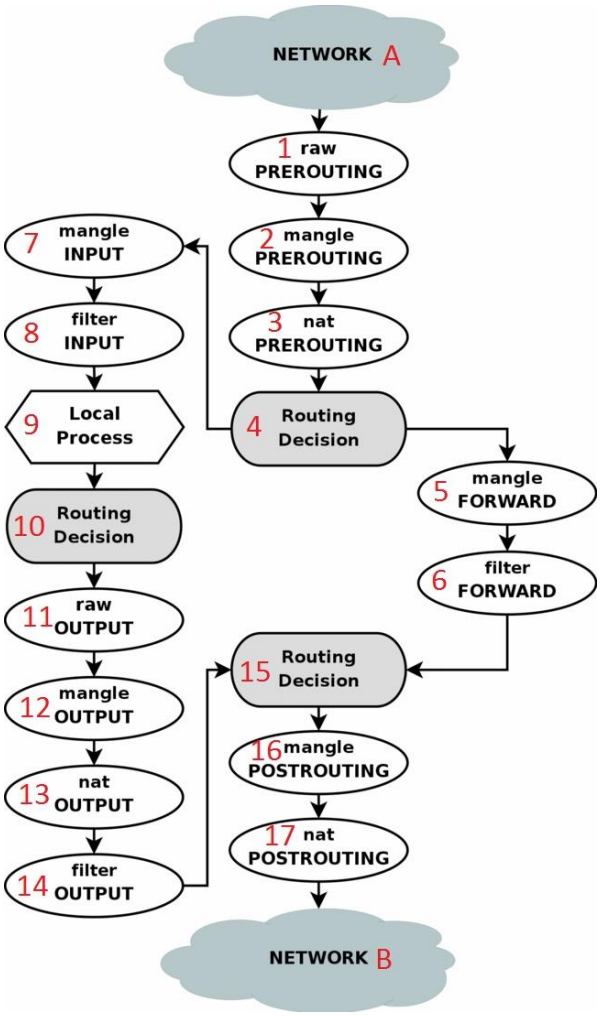
- Filter tables
  - Filter là table mặc định cho iptables. Nếu bạn không xác định tables, bạn sẽ sử dụng filter tables.
  - Bảng này được sử dụng để lọc gói tin.
  - Filter tables của iptables có các chains INPUT, OUTPUT và FORWARD.
- NAT tables
  - Bảng này được sử dụng cho chức năng NAT trên các gói tin khác nhau.
  - NAT tables của iptables có các chains PREROUTING, POSTROUTING và OUTPUT.
- Mangle table: Mangle tables của Iptables dành cho những thay đổi gói tin đặc biệt. Điều này thay đổi bit QoS của TCP header.
- Raw table: Bảng raw chủ yếu chỉ được sử dụng cho một điều, và đó là để thiết lập một đánh dấu trên gói tin rằng họ không nên được xử lý bởi các hệ thống theo dõi kết nối.

Giải thích các chain của Iptables:

Bảng 2.1. Ý nghĩa các chain trong Iptables

Chain	Ý nghĩa
INPUT	những gói tin đi vào hệ thống
OUTPUT	những gói tin đi ra từ hệ thống
FORWARD	những gói tin đi qua hệ thống (đi vào một hệ thống khác
PREROUTING	sửa địa chỉ đích của gói tin trước khi nó được routing bởi bảng routing của hệ thống (destination NAT hay DNAT).
POSTROUTING	ngược lại với Pre-routing, nó sửa địa chỉ nguồn của gói tin sau khi gói tin đã được routing bởi hệ thống (SNAT).

Quá trình xử lý gói tin của Iptables:



Hình 2.4. Quá trình xử lý gói tin của Iptables [9]

1) Nếu gói tin đi từ ngoài đến

Đầu tiên gói tin từ mạng A đi vào hệ thống firewall sẽ phải đi qua table **Mangle** với chain là **PREROUTING** (với mục đích để thay đổi một số thông tin của gói tin trước khi đưa qua quyết định dẫn đường) sau đó gói tin đến table **NAT** với với chain **PREROUTING** tại đây địa chỉ đích của gói tin có thể bị thay đổi hoặc không, qua bộ **Routing** và sẽ quyết định xem gói tin đó thuộc firewall hay không:

- **TH1**: gói tin đó là của firewall: gói tin sẽ đi qua table **Mangle** và đến table **Filter** với chain là **INPUT**. Tại đây gói tin sẽ được áp dụng **rule** và ứng với mỗi rule cụ thể sẽ được áp dụng với **target**. Sau đó đến **Local Prcoess** (chính là ứng dụng, VD: chương trình server/client).
- **TH2**: gói tin không phải của firewall sẽ được đưa đến table **Mangle** với chain **FORWARD** đến **bảng filter** với chain **FORWARD**. Đây là chain được sử dụng rất nhiều để bảo vệ người sử dụng mang trong lan với người sử dụng internet các gói tin thỏa mãn các **rule** đặt ra mới có thể được chuyển qua giữa các card mạng với nhau, qua đó có nhiệm vụ thực hiện chính sách với người sử dụng nội bộ nhưng không cho vào internet, giới hạn thời gian,...và bảo vệ hệ thống máy chủ đối với người dung internet bên ngoài chống các kiểu tấn công. Sau khi đi qua card mạng, gói tin phải đi lần lượt qua table **Mangle** và **NAT** với chain **POSTROUTING** để thực hiên việc chuyển đổi địa chỉ nguồn với target **SNAT** & **MASQUERADE**.

2) Nếu gói tin đi là của Firewall đi ra

- Gói tin xuất phát từ **Local Prcoess** (chính là ứng dụng, VD: chương trình server/client). **Routing decision** được sử dụng, địa chỉ nguồn cần sử dụng, giao diện đi ra là gì, và những thông tin khác cần thu thập.
- Sau đó là quá trình xử lý gói tin sẽ đi đến table **Mangle** tiếp đến là table **NAT** với chain **OUTPUT** và table **FILTER** với chain **OUTPUT** được áp dụng một số rule.
- Sau đó đi lần lượt qua các table **Mangle** với chain **POSTROUTING** cuối cùng đi đến table **NAT** với chain **POSTROUTING** để thay đổi địa chỉ nguồn nếu cần thiết.

2.1.3. Phát hiện xâm nhập (IDS) - Suricata

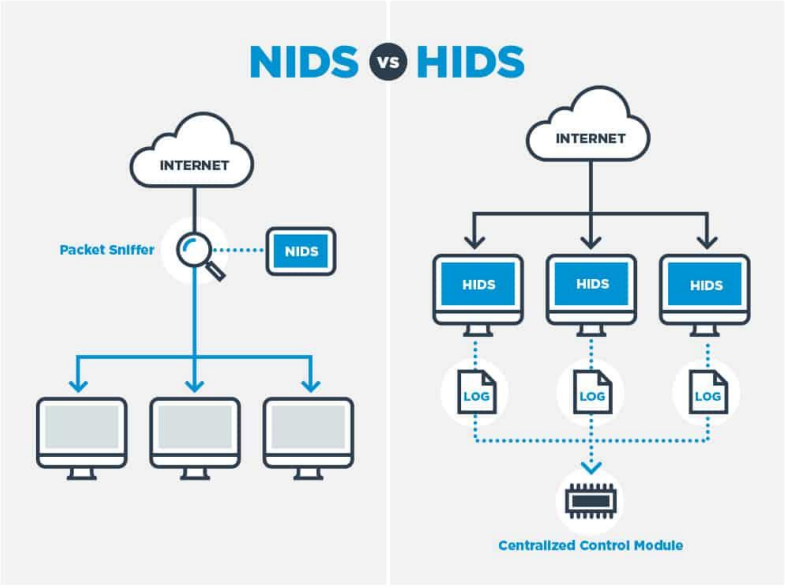
Hệ thống phát hiện xâm nhập - IDS là một hệ thống giám sát lưu lượng mạng nhằm phát hiện hiện tượng bất thường, các hoạt động trái xâm nhập phép và hệ thống. IDS có thể phân biệt được những tấn công từ bên trong (nội bộ) hay tấn công từ bên ngoài (từ các tin tặc). IDS phát hiện dựa trên các dấu hiệu đặc biệt về các nguy cơ đã biết (giống như cách các phần mềm diệt virus dựa vào các dấu hiệu đặc biệt để phát hiện và diệt virus) hay dựa trên so sánh lưu thông mạng hiện tại với baseline (thông số đo đạt chuẩn của hệ thống có thể chấp nhận được ngay tại thời điểm hiện tại) để tìm ra các dấu hiệu khác thường.

Tính năng quan trọng nhất của hệ thống phát hiện xâm nhập - IDS là:

- Giám sát lưu lượng mạng và các hoạt động khả nghi.
- Cảnh báo về tình trạng mạng cho hệ thống và nhà quản trị.
- Kết hợp với các hệ thống giám sát, tường lửa, diệt virus tạo thành một hệ thống bảo mật hoàn chỉnh.

Phân loại IDS (hệ thống phát hiện xâm nhập)

- Network-based IDS (NIDS): Hệ thống phát hện xâm nhập mạng. Hệ thống sẽ tập hợp gói tin để phân tích sâu bên trong mà không làm thay đổi cấu trúc gói tin.
- Hosted-based (HIDS): Hệ thống phát hiện xâm nhập trên máy chủ. Theo dõi các hoạt động bất thường trên các máy chủ riêng biệt. HIDS được cài đặt trực tiếp trên các máy chủ cần theo dõi.



Hình 2.5. NIDS và HIDS [10]

Suricata là công cụ phát hiện xâm nhập mạng, miễn phí và là mã nguồn mở, nhanh chóng và mạnh mẽ. Công cụ Suricata có khả năng phát hiện xâm nhập (IDS), ngăn chặn xâm nhập (IPS), giám sát an ninh mạng (NSM).

Suricata kiểm tra lưu lượng mạng bằng cách sử dụng các quy tắc mạnh mẽ và rộng rãi để phát hiện các mối đe dọa phức tạp.

Suricata hỗ trợ đa luồng, tức là nó có thể sử dụng nhiều lõi CPU cùng một thời điểm cho phép xử lý nhiều sự kiện mà không phải gián đoạn các yêu cầu.

Hiện nay, Suricata phiên bản ổn định mới nhất là 4.1.4, được phát hành ngày 30 tháng 4 năm 2019, hỗ trợ các nền tàng Linux/Mac/FreeBSD/Unix/Windowns. Phiên bản mới nhất của Suricata là 5.0.0 đang trong giai đoạn thử nghiệm, cũng được phát hành ngày 30 tháng 4 năm 2019, hỗ trợ các nền tàng Linux/Mac/FreeBSD/Unix/Windowns. Tuy nhiên, người dùng vẫn có thể sử dụng các phiên bản cũ hơn bằng cách tải tại trang web https://www.openinfosecfoundation.org/downloads/.

2.1.4. Cân bằng tải (Load Balancer) - HAProxy

HAProxy (High Availability Proxy) là phần mềm cân bằng tải cho TCP/HTTP mã nguồn mở, nó có thể chạy trên Linux, Solaris và FreeBSH. Cách sử dụng phổ biến nhất của nó là cải thiện hiệu năng và tính tin cây của môi trường máy chủ bằng cách phân phối công việc trên nhiều máy chủ (VD: web, application, database). Nó được sử dụng trong nhiều môi trường hiệu năng cao bao gồm Github, Imgur, Instagram và Twitter.

HAProxy được thiết kế với mục tiêu chuyển tiếp dữ liệu, kiến trúc của nó được tối ưu hóa để di chuyển data nhanh nhất có thể với các hoạt động ít nhất có thể. Như vậy nó thực hiện một mô hình lớp cung cấp các cơ chế bỏ qua ở mỗi cấp đảm bảo dữ liệu không đạt được cấp cao hơn khi không cần thiết.

Hầu hết quá trình xử lý được thực hiện trong kernel, và HAProxy làm hết sức mình để giúp kernel thực hiện công việc nhanh nhất có thể bằng cách đưa ra một số gợi ý hoặc bằng cách tránh một số thao tác nhất định khi nó đoán chúng có thể được nhóm lại sau.

Kết quả là, các số liệu điển hình cho thấy 15% thời gian xử lý dành cho HAProxy so với 85% trong kernel trong chế độ TCP close hoặc chế độ HTTP close và khoảng 30% cho HAProxy so với 70% cho hạt nhân ở chế độ HTTP keep-alive.

HAProxy chỉ yêu cầu tệp thực thi và tệp cấu hình haproxy để chạy. Đối với việc ghi nhật ký, chúng tôi rất khuyên bạn nên có một syslog daemon được cấu hình đúng cách và ghi nhật ký tại chỗ. Các tệp cấu hình được phân tích cú pháp trước khi bắt đầu, sau đó

HAProxy cố gắng ràng buộc tất cả cổng logic đang lắng nghe, và từ chối bắt đầu nếu bất cứ điều gì không thành công. Quá khứ thời điểm này nó không thể thất bại nữa. Điều này có nghĩa rằng có không có runtime fail và nếu nó chấp nhận để bắt đầu, nó sẽ làm việc cho đến khi nó được dừng lại.

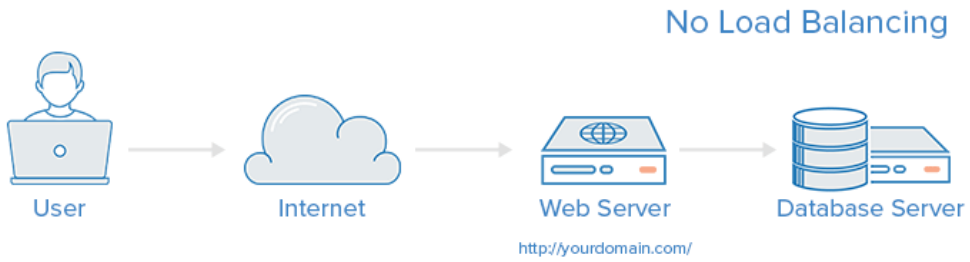
Khi HA được bật, nó thực hiện:

- kiểm tra tình trạng máy chủ định kỳ (được gọi là health checks)
- trao đổi thông tin với các nodes haproxy khác

Các loại cân bằng tải:

- Không có cân bằng tải

  Môi trường ứng dụng web đơn giản với không có cân bằng tải.

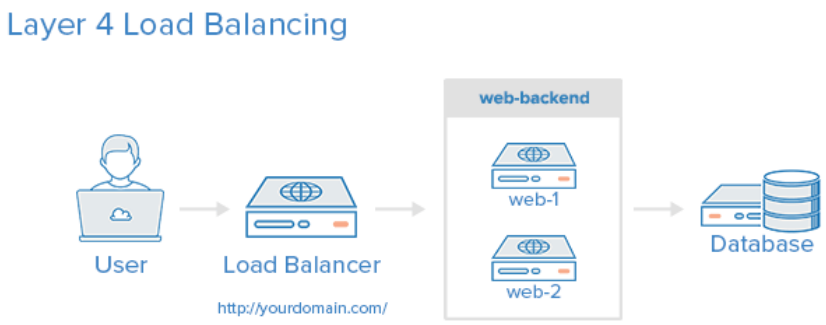


Hình 2.6. Không có cân bằng tải [11]

  Trong ví dụ này, người dùng kết nối trực tiếp với web server của bạn, tại yourdomain.com và không có cân bằng tải. Nếu máy chủ web đơn của bạn bị hỏng, người dùng sẽ không còn có thể truy cập máy chủ web của bạn nữa. Ngoài ra, nếu nhiều người dùng đang cố gắng truy cập máy chủ web của bạn cùng một lúc và không thể xử lý tải, họ có thể có trải nghiệm chậm hoặc họ hoàn toàn không thể kết nối.

- Cân bằng tải tại tầng 4 (tầng vận chuyển)

  Cách đơn giản nhất để cân bằng tải lưu lượng mạng tới nhiều máy chủ là sử dụng cân bằng tải tại tầng 4. Cân bằng tải theo cách này sẽ chuyển tiếp lưu lượng người dùng dựa trên địa chỉ IP và port (ví dụ: nếu yêu cầu đến là yourdomain.com/anything, lưu lượng sẽ được chuyển tiếp đến backend để xử lý tất các request cho trên yourdomain.com port 80).

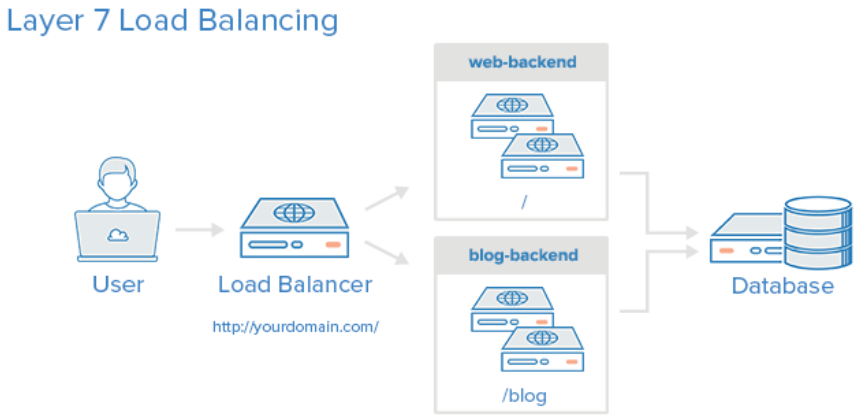


Hình 2.7. Cân bằng tải tại tầng 4 [11]

Người dùng truy cập cân bằng tải, được chuyển tiếp yêu cầu đến nhóm web-backend của các máy chủ backend. Các máy chủ backend nào được chọn sẽ phản hồi trực tiếp các yêu cầu của người dùng. Nói chung, tất cả các mảy chủ trong web-backend đều phải phân phối nội dung giống nhau - nếu không, người dùng có thể nhận được nội dung không phù hợp. Lưu ý rằng cả 2 máy chủ web đều kết nối với cùng một máy chủ cơ sở dữ liệu.

- Cân bằng tải tại tầng 7 (tầng ứng dụng)

Một cách phức tạp hơn để cân bằng tải lưu lượng mạng là sử dụng cân bằng tải tại tầng 7. Sử dụng tầng 7 cho phép cân bằng tải chuyển tiếp yêu cầu đến các máy chủ backend khác nhau dựa trên nội dung của yêu cầu người dùng. Chế độ cân bằng tải cho phép bạn chạy nhiều mảy chủ ứng dụng web trong cùng tên miền và port.



Hình 2.8. Cân bằng tải tại tầng 7 [11]

VD: Nếu người dùng request đến yourdomain.com/blog, chúng ta có thể forwarded đến *blog-backend*, là tập hợp các server chạy trên ứng dụng *blog*. Một số

yêu cầu được chuyển tiếp đến *web-backend*, có thể đang chạy ứng dụng khác. Cả 2 backend đều sử dụng cùng máy chủ cơ sở dữ liệu.

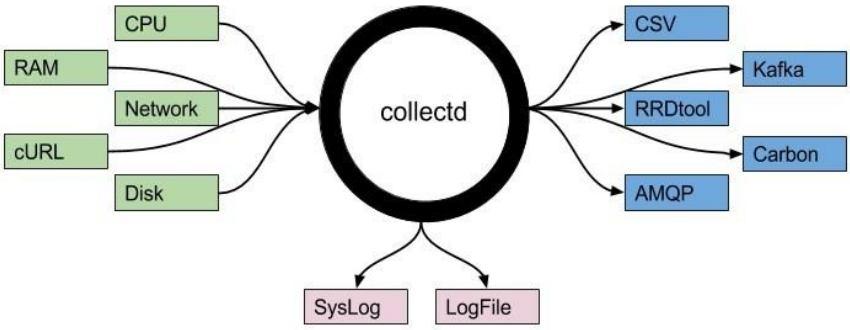
2.1.5. Collectd, Graphite và Grafana

a. Collected

Collected là daemon thu thập số liệu về hiệu năng của ứng dụng và hệ thống; cung cấp các cơ chế để lưu trữ giá trị bằng nhiều cách, ví dụ như lưu trong các file RRD.

Collected tập hợp dữ liệu từ các nguồn khác nhau, vd: hệ điều hành, ứng dụng, logfiles và các thiết bị mở rộng, và lưu trữ thông tin hoặc cung cấp thông tin qua mạng. Những thống kê này được sử dụng để theo dõi hệ thống (monitor system), tìm các điểm thắt nút cổ chai (performance bottlenecks) - (tức là phân tích hiệu năng) và dự đoán tải của hệ thống.

Hiểu đơn giản, collectd là công cụ có khả năng theo dõi các chỉ số chung như memory được sử dụng, CPU được sử dụng, network traffic,..



Hình 2.9. Chức năng Collectd [12]

Ngoài việc thu thập thông tin hệ thống chuẩn, collectd cũng có 1 hệ thống plugin mở rộng chức năng của nó. Điều này có nghãi là bạn có thể dàng theo dõi phần mềm phổ biến như Apache, Nginx, iptables, memcache, MySQL, PostgreSQL, OpenVPN và nhiều hơn nữa.

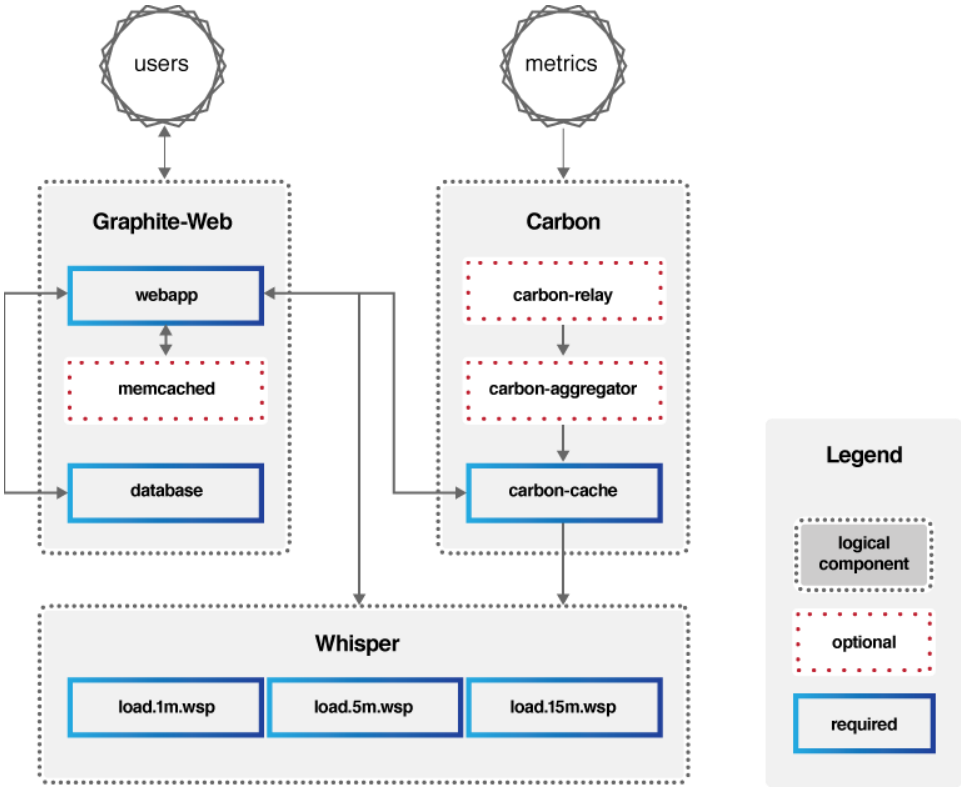
b. Graphite

Graphite là công cụ theo dõi các thông số như CPU, memory, ổ cứng, mạng,.. được dùng trong các doanh nghiệp được viết bởi Chris Davis tại Orbitz vào năm 2006. Năm 2008, Orbitz cho phép Graphite được phát hành dưới giấy phép mã nguồn mở Apache 2.0.

Graphite không phải agent thu thập dữ liệu cho bạn, tuy nhiên có 1 số công cụ trên mạng có thể gửi dữ liệu tới graphite. Graphite làm 2 việc:

- Lưu trữ time-series data
- Render đồ thị của dữ liệu theo yêu cầu

Graphite có thể kết hợp với các công cụ thu thập dữ liệu như Collectd, StatsD,…



Hình 2.10. Kiến trúc Graphite [13]

Graphite gồm 3 thành phần:

- Carbon: là twisted daemon lắng nghe dữ liệu chuỗi thời gian được gửi đến.
- Whisper: thư viện cơ sở dữ liệu đơn giản để lưu trữ dữ liệu chuỗi thời gian (tương tự thiết kế trong RRD).
- Graphite webapp: 1 Django webapp render đồ thị theo yêu cầu và hiển thị cho người dùng.

Carbon là dịch vụ mạng lắng nghe dữ liệu chuỗi thời gian được gửi đến. Nó lưu trữ các số liệu tạm thời trong bộ nhớ buffer-cache trong một thời gian ngắn trước khi ghi gửi chúng đến ổ cứng dưới định dạng cơ sở dữ liệu Whisper.

Khi ai đó đề cập đến Carbon là thành phần của Graphite, hầu như đều nói về dịch vụ carbon-cache. Nhưng thật ra, Carbon có 3 daemon riêng biệt gọi chung là Carbon:

- carbon-cache listener

- carbon-replay
- carbon-aggregator

Carbon-cache daemon là quy trình làm việc khó nhất trong chuỗi thời gian. Nó được viết bằng Python và dựa trên thưu viện Twitsed. Nhưng nó không phải đơn giản là 1 bộ nhớ cache thông thường, carbon-cache là dịch vụ mạng chấp nhập số liệu được gửi từ máy khách đến tạm thời lưu trữ các dữ liệu trong bộ nhớ (cache) và viết chúng đến ổ cứng như tệp cơ sở dữ liệu Whisper.

Nó chấp nhận các số liệu của giao thức TCP hoặc UDP hoặc sử dụng giao thức Python's pickle (một định dạng tuần tử hiệu quả).

Nếu bạn hỗ trợ 1 hệ thống phân tán, Graphite thậm chí còn hỗ trợ bus thông báo AMQP. Bởi vì, Graphite cần truy cập vào các số liệu ở cả ổ cứng và bộ nhớ, thì cache daemon cũng bao gồm cổng truy vấn để ứng dụng web Graphite có thể thăm dò các số liệu quan trọng mà chưa được ghi lên ổ cứng.

Carbon-relay được thiết kế để relay số liệu từ 1 Carbon đến các tiến trình Carbon khác. Có 2 nhiệm vụ chính là carbon-relay được thiết kế:

- chuyển tiếp các số liệu đến carbon daemon khác
- sao chép số liệu thành nhiều bản sao

Điều này mở ra con đường cho các trường hợp sử dụng mà bạn cần phải thêm tính năng cân bằng tải.

Whisper là định dạng tệp cơ sở dữ liệu có kích thước cố định với sự hỗ trợ cho datapoints chính xác đến từng giây. Whisper được thiết kế tương tự như thiết kế trong RRD. Máy chủ cài whisper thường dùng ổ cứng SSD để khả năng truy xuất dữ nhanh hơn.

RRD (Round Robin Database) là một hệ thống lưu trữ chuyên dụng gọi là Round Robin Database cho phép lưu trữ một lượng lớn các thông tin theo thời gian như nhiệt độ, băng thông mạng và giá cổ phiếu và ghi dữ liệu vào ổ cứng liên tục. Ví dụ: nếu chúng ta lấy mẫu tín hiệu ở khoảng 5 phút thì khoảng thời gian 24 giờ sẽ có 288 điểm dữ liệu (24 giờ * 60 phút / giờ chia cho 5 phút cho mỗi mẫu). Để tiết kiệm không gian, chúng ta có thể thu thập dữ liệu cũ bằng chức năng hợp nhất, thực hiện một số tính toán trên nhiều điểm dữ liệu để kết hợp nó vào một điểm duy nhất trong một khoảng thời gian dài hơn. Hãy tưởng tượng rằng chúng tôi lấy trung bình của 288 mẫu vào cuối mỗi 24 giờ. Trong trường hợp đó, chúng tôi chỉ cần 365 điểm dữ liệu để lưu trữ dữ liệu cho cả năm. Nhưng RRD không chấp nhận các dữ liệu bất thường, nếu số liệu A với mốc thời gian là 5:05:00 đến sau số liệu B

với mốc thời gian 5:05:30, số liệu đến trước sẽ bị loại bỏ hoàn toàn bởi RRD (điều này được thiết kế lại về sau).

Whisper giải quyết thiết kế thiếu sót này một cách đặc biệt, đơn giản hóa cấu hình và bố trí thời gian lưu trữ với mỗi tệp cơ sở dữ liệu.

Graphite webapp là một Django webapp render đồ thị theo yêu cầu và hiển thị cho người dùng.

Một số khái niệm cần biết:

- Time series data
  - Time series data là 1 dãy giá trị được thu thập ở khoảng cách đều đặn.
  - VD: Nhiệt độ mỗi phút trong 1 ngày, % CPU tại mỗi giây của 1 máy chủ,…
- Time series databases: là hệ thống phần mềm được tối ưu hóa để lưu trữ và truy xuất dữ liệu chuỗi thời gian.
- Roll-up trong tổng hợp dữ liệu

  VD sau sẽ cho bạn hiểu về roll-up.

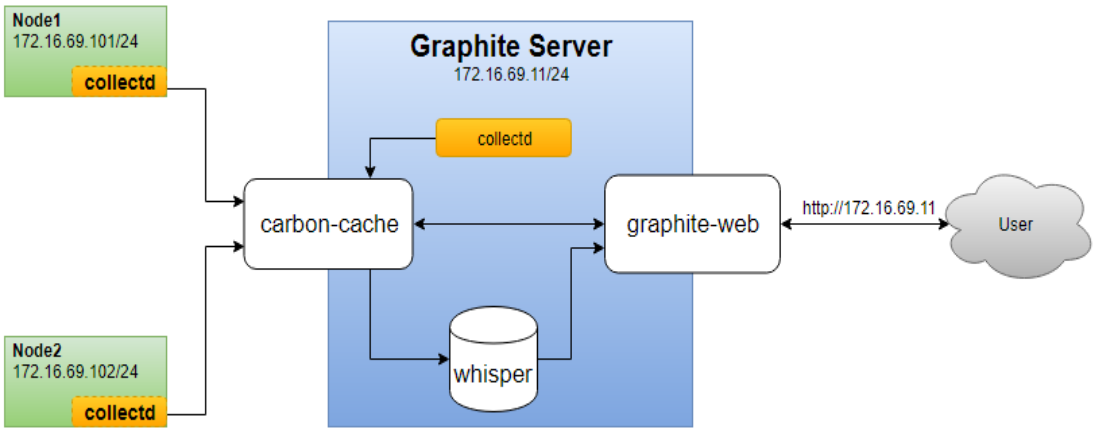
  Nếu bạn gửi số liệu mỗi 30s và nhìn nó trong chế độ "last 30 minutes" thì bạn đang xem xét trong chế độ "raw resoluation", do đó có thể lên đến 60 điểm. Cùng chỉ số khi xem ở chế độ "last 60 minutes" thì sẽ hiện thị 120 điểm.

  Đối với cửa sổ trong thời gian lớn hơn 60 phút, chúng tôi có thể "roll up" dữ liệu trong 1 khoảng thời gian nhất định bằng cách sử dụng phương pháp được chỉ định trong thuộc tính metric (average, sum, min, max hoặc count). Khi xem dữ liệu ở quy mô lớn bạn có thể đang xem giá trính tính toán này.

c.Grafana

Grafana là một bộ mã nguồn mở sử dụng trong việc phân tích các dữ liệu thu thập được từ máy chủ và hiện thị một các trực quan dữ liệu thu thập được ở nhiều dạng khác nhau.

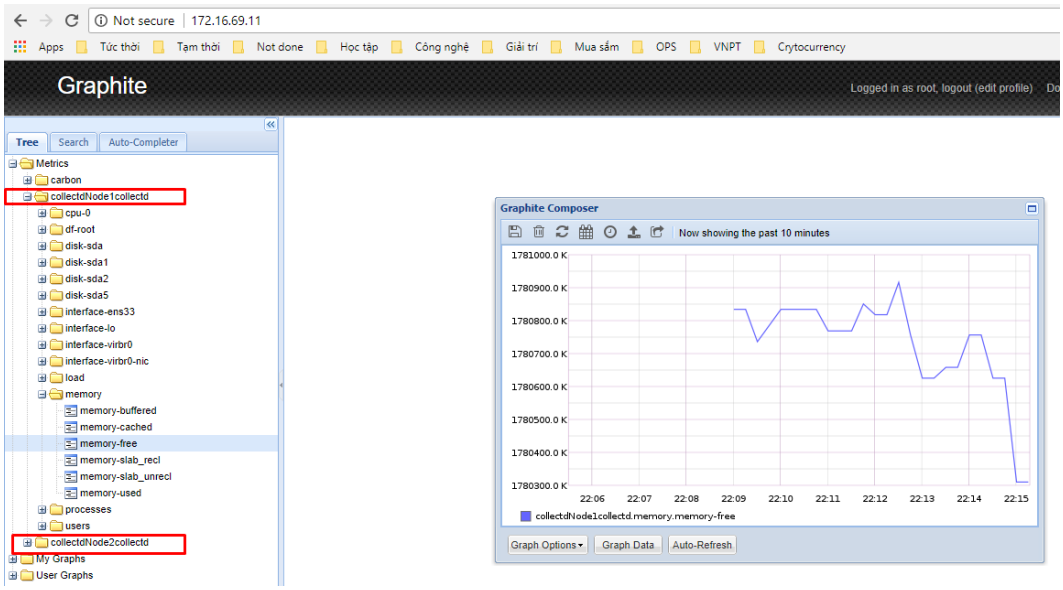
d. Mô hình kết hợp sử dụng Collectd, Graphite



Hình 2.11. Mô hình kết hợp sử dụng Collectd, Graphite

Hình 2.11 là mô hình kết hợp sử dụng Collectd và Graphite trong môi trường mạng.

Trong đó, tại Node1 và Node2, ta cài công cụ Collectd để thu nhập dữ liệu, sau đó dữ liệu sẽ được gửi về cho 1 máy chủ khác cài công cụ Graphite, cụ thể ở đây là gửi về cho dịch vụ carbon-cache. Tiếp theo Graphite sẽ đưa ra đồ thị về thông số cần đo từ các số liệu nhận được. Người dùng có thể truy cập vào giao diện web của Graphite để xem đồ thị, từ đó quan sát và đánh giá được hiệu năng sử dụng tài nguyên của Node1 và Node2.

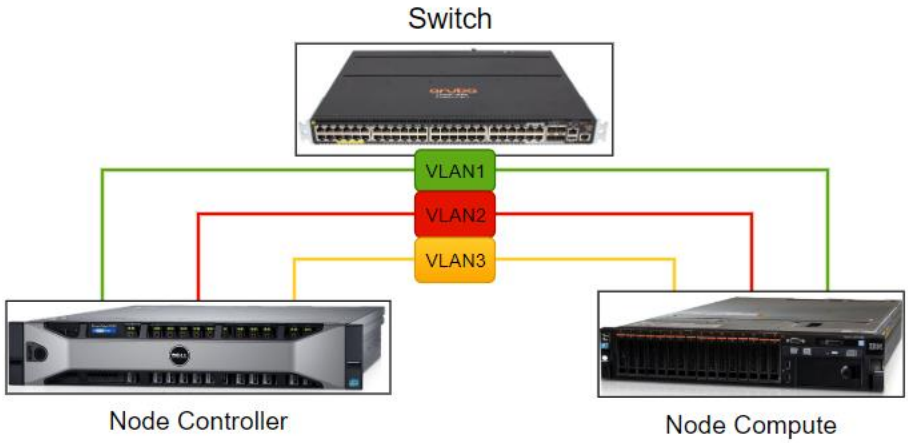


Hình 2.12. Giao diện Graphite

Hình 2.12 cho ta thấy được giao diện Graphite, tương đối trực quan và dễ quan sát được các thông số như CPU, RAM,…

2.2. Mô hình testbed vật lý

Testbed được triển khai trên 2 server vật lý và 1 switch.



Hình 2.13. Mô hình testbed thực tế

Hình 2.13 là mô hình thực tế triển kahi testbed, gồm 1 mảy chủ DELL, 1 máy chủ IBM và 1 switch Aruba. Switch Aruba được chia 3 VLAN là 1, 2 và 3.

Phân hoạch địa chỉ IP và thông số phần cứng các máy chủ.

PHÂN HOẠCH CÁC ĐỊA CHỈ IP CHO MÁY CHỦ						THÔNG SỐ VỀ PHẦN CỨNG				
Hostname	Inteface	IP Address	Netmask	Gateway	DNS	RAM (GB)	CPU	DISK1 (GB)	DISK2 (GB)	Info
Controller	eth0	192.168.101.201	255.255.255.0	192.168.101.1	8.8.8.8	64	24	300GB	300GB	Server DELL RAID 0 -> 600 GB
	eth1	10.10.10.201	255.255.255.0							
	eth2	10.10.20.201	255.255.255.0							
Compute	eth0	192.168.101.211	255.255.255.0	192.168.101.1	8.8.8.8	64	32	1000GB	1000GB	Server IBM RAID 1 -> 1000 GB
	eth1	10.10.10.211	255.255.255.0							
	eth2	10.10.20.211	255.255.255.0							

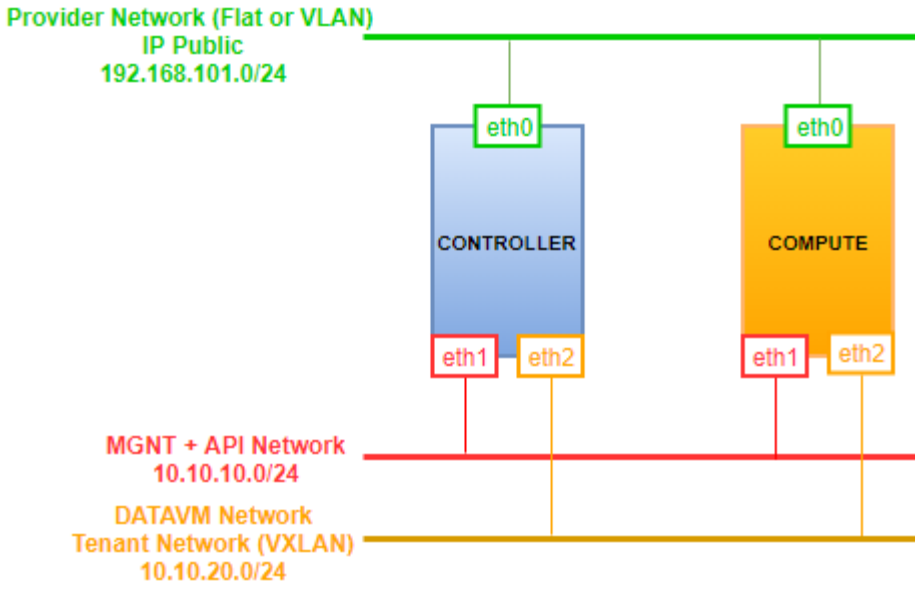
Hình 2.14. Phân hoạch địa chỉ IP và thông số phần cứng các máy chủ

Hình 2.14 thể hiện địa chỉ IP, gateway, DNS, thông số phần cứng của node Controller và Compute. Trong đó node Controller được cài trên máy chủ DELL, còn node Compute được cài trên máy chủ IBM.

Node Controller và node Compute có 3 giao diện mạng lần lượt là eth0, eth1 và eth2 với các thông số địa chỉ IP như trên hình.

2.3. Triển khai nền tảng OpenStack

Ở đây, em sử dụng OpenStack phiên bản Pike, sau đây là mô hình cài đặt OpenStack gồm 2 node: Controller và Compute.



Hình 2.15. Mô hình cài đặt OpenStack mức logic

Trong đó, 2 node Controller và Compute đều cài đặt hệ điều hành Ubuntu Server 16.04. Trên mỗi node cài đặt:

- **Node Controller**: Node Controller cung cấp API, lập lịch và chia sẻ các dịch vụ khác cho cloud. Node Controller có bảng điều khiển, lưu trữ images và dịch vụ xác thực. Trên node này, ta cài đặt các project Keysont, Glance, Nova, Neutron, Horizon.
- **Node Compute**: Node Compute là nơi chứa các máy ảo. Node này cài đặt các project Neutron và Nova.
- **Hạ tầng mạng gồm 3 dải mạng**:
    - *Dải Provider Network*: cung cấp kết nối Internet cho toàn bộ các máy ảo.
    - *Dải MNGT + API Network*: cung cấp kết nối giữa các node trong OpenStack. Ví dụ: truyền dẫn lệnh từ node Controller gửi sang node Compute.
    - *Dải DATAVM Network*: cung cấp kết nối giữa các máy ảo nằm trên các node khác nhau.

2.4. Tích hợp DPDK và OpenStack

Ta cần phân chia tài nguyên CPU và RAM trên từng node cho Open vSwitch và máy ảo sao cho hợp lý để đạt được hiệu năng cao nhất. Ở testbed này, ta chỉ định cho Open vSwitch sử dụng 4 CPU và 4G RAM của máy chủ, chỉ định 24 CPU cho máy ảo dùng.

Tiếp theo đó, ta thiết lập và cài đặt DPDK trên từng node của mô hình OpenStack. Sau đó ta chỉnh sửa một số tệp cấu hình của DPDK và OpenStack sao phù hợp với nhau.

2.5. Khởi tạo các máy ảo làm chức năng mạng

Khởi tạo các máy ảo với các câu lệnh:

```
openstack server create --image ubuntu16 --flavor medium1 --user-data myfile \
      --nic port-id=P1_ID Ntopng

openstack server create --image ubuntu16 --flavor medium2 --user-data myfile \
      --nic port-id=P2_ID Iptables

openstack server create --image ubuntu16 --flavor medium3 --user-data myfile \
      --nic port-id=P3_ID Suricata

openstack server create --image ubuntu16 --flavor medium4 --user-data myfile \
      --nic port-id=P4_ID WebServer

openstack server create --image ubuntu16 --flavor medium5 --user-data myfile \
      --nic port-id=P5_ID Graphite
```

Thông số tài nguyên các máy ảo như sau:

Bảng 2.2. Thông số tài nguyên máy ảo triển khai chức năng mạng

Máy ảo	CPU	RAM (GB)	IP	Floating IP
Ntopng	2	2	10.1.0.11	
Iptables	2	2	10.1.0.12	
Suricata	2	2	10.1.0.13	
Webserver	3	3	10.1.0.14	192.168.101.9
Grafana	1	1	10.1.0.16	

Ta thấy riêng mảy chủ Webserver được gán Floating IP (đây là IP public) để người dùng có thể truy cập đến Webserver thông qua mạng Internet.

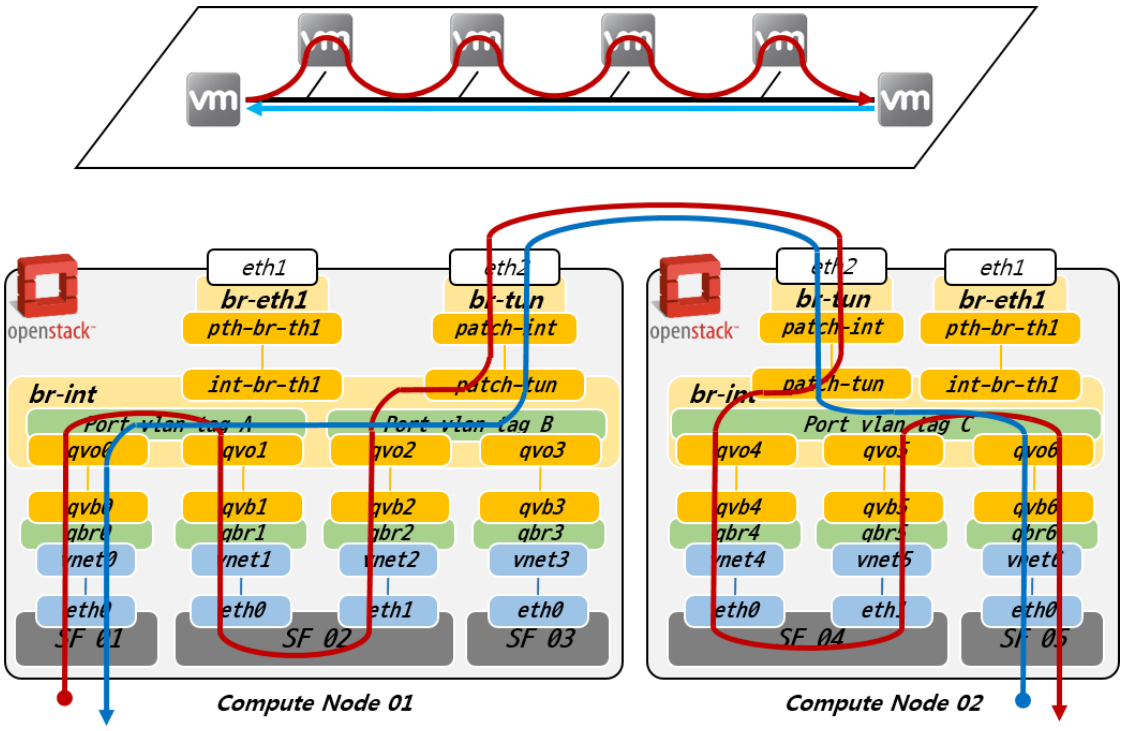
2.6. Triển khai chuỗi dịch vụ mạng dựa trên nền tảng OpenStack

2.6.1. Giới thiệu về Networking SFC trong OpenStack

Networking SFC là một dự án cung cấp API và hỗ trợ triển khai chuỗi chức năng dịch vụ trong neutron trên nền tảng OpenStack.

Chuỗi chức năng dịch vụ là một cơ chế để ghi đè dựa trên đích của gói tin. Nó liên quan về mặt khái niệm với định tuyến dựa trên chính sách trong các mạng vật lý. Nó thường được sử dụng cùng với các chức năng bảo mật mặc dù nó có thể được sử dụng cho một phạm vi rộng hơn của các tính năng. Về cơ bản SFC là khả năng khiến các luồng gói tin mạng định tuyến qua mạng thông qua một đường dẫn khác với đường dẫn sẽ được chọn bằng cách tìm kiếm bảng định tuyến trên địa chỉ IP đích của gói. Nó thường được sử dụng cùng với ảo hóa chức năng mạng khi tái tạo trong môi trường ảo, một loạt các chức năng mạng thường được triển khai như một tập hợp các thiết bị mạng vật lý được kết nối nối tiếp bằng dây cáp.

Một ví dụ rất đơn giản về chuỗi dịch vụ sẽ là một chuỗi buộc tất cả lưu lượng truy cập từ điểm A đến điểm B phải đi qua tường lửa mặc dù tường lửa không thực sự nằm giữa điểm A và B theo quan điểm của bảng định tuyến.

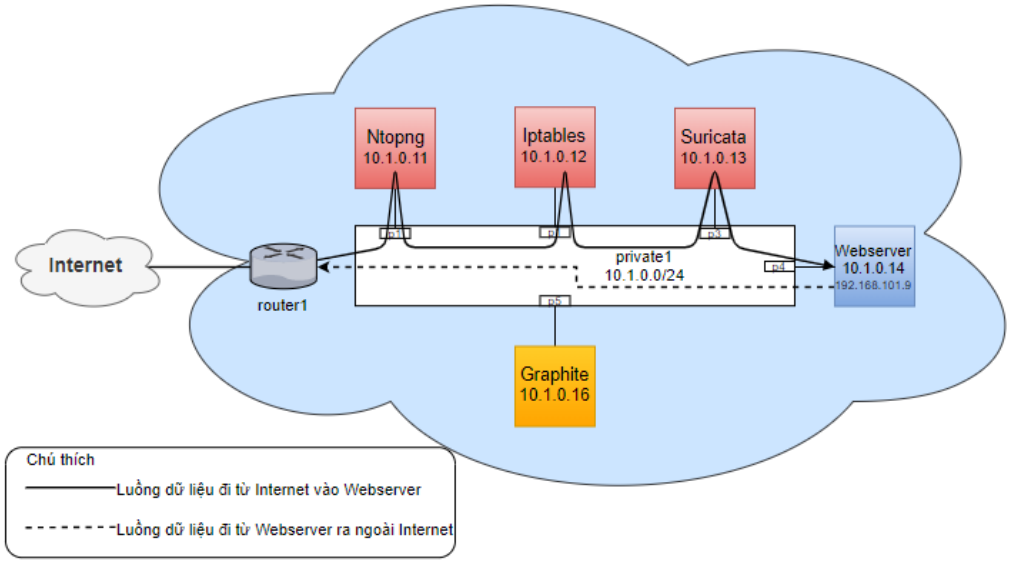


Hình 2.16. Networking SFC trong OpenStack [7]

Hình 2.16 cho thấy chuỗi chức năng có thể gồm nhiều chức năng (được cài trên máy ảo) nằm trên các máy chủ khác nhau.

2.6.2. Mô hình chuỗi chức năng mạng thực tế trong OpenStack

Mô hình luồng dữ liệu của chuỗi chức năng mạng như sau:



Hình 2.17. Mô hình chuỗi chức năng mạng thực tế

Hình 2.17 mô tả chuỗi chức năng mạng được cài đặt trên OpenStack. Tất cả các chức năng mạng được cài đặt trên máy ảo sử dụng hệ điều hành Ubuntu Server 16.04, kết nối với dải mạng ảo tên "private1" có dải IP là 10.1.0.0/24.

Ta thiết lập luồng đi của dữ liệu sử dụng các câu lệnh như sau:

- Tạo flow-classifier
  - FC1

```
neutron flow-classifier-create \
 --ethertype IPv4 \
 --source-ip-prefix 0.0.0.0/0 \
 --destination-ip-prefix 10.1.0.14/32 \
 --protocol icmp \
 --logical-source-port p1 FC1
```

- FC2

```
neutron flow-classifier-create \
 --ethertype IPv4 \
 --source-ip-prefix 0.0.0.0/0 \
 --destination-ip-prefix 10.1.0.14/32 \
 --protocol tcp \
 --logical-source-port p1 FC2
```

- FC3

```
neutron flow-classifier-create \
 --ethertype IPv4 \
 --source-ip-prefix 0.0.0.0/0 \
 --destination-ip-prefix 10.1.0.14/32 \
 --protocol udp \
 --logical-source-port p1 FC3
```

- Tạo port-pair

```
neutron port-pair-create --ingress=p2 --egress=p2 PP1
neutron port-pair-create --ingress=p3 --egress=p3 PP2
```

- Tạo port-pair-group

```
neutron port-pair-group-create --port-pair PP1 PG1
neutron port-pair-group-create --port-pair PP2 PG2
```

- Tạo port-chain

```
neutron port-chain-create --port-pair-group PG1 --port-pair-group PG2 --
flow-classifier FC1 --flow-classifier FC2 --flow-classifier FC3 PC1
```

2.7. Kết luận

Chương 2 này đã trình bày về quá trình triển khai testbed trên nền tảng OpenStack khá chi tiết và đầy đủ. Phần tiếp theo em trình bày về kịch bản và kết quả đo hiệu năng của các chức năng mạng ảo, cũng như băng thông, đỗ trễ, tỉ lệ mất gói trong mạng.

CHƯƠNG 3. KỊCH BẢN ĐO ĐẠC VÀ ĐÁNH GIÁ

Chương 3 này sẽ tập trung vào việc đo đạc hiệu năng của các chức năng mạng ảo, cũng như các thống số băng thông, độ trễ, tỷ lệ mất gói của hệ thống mạng và đưa ra nhận xét về hệ thống.

3.1. Kịch bản

Kịch bản bao gồm:

1. Đo đạc hiệu năng (mức độ tiêu thụ tài nguyên) của các chức năng mạng ảo Ntopng, Suricata trong khi hoạt động.
2. Đo đạc các thông số để đánh giá chất lượng dịch vụ mạng bao gồm: băng thông, độ trễ và tỉ lệ mất gói khi lưu lượng đi qua chuỗi chức năng mạng tăng dần trên nền tảng OpenStack không được tích hợp DPDK và được tích hợp DPDK.

Ta sử dụng công cụ Bonesi để phát hàng loạt gói tin đến máy chủ Webserver, lúc này luồng dữ liệu gồm các gói tin sẽ lần lượt đi qua các chức năng mạng của chuỗi chức năng mạng (Ntopng, Iptables, Suricata), sau đó mới Webserver. Qua đó ta có thể đo được hiệu năng của chức năng mạng, cũng như đo được các thông số để đánh giá chất lượng dịch vụ mạng.

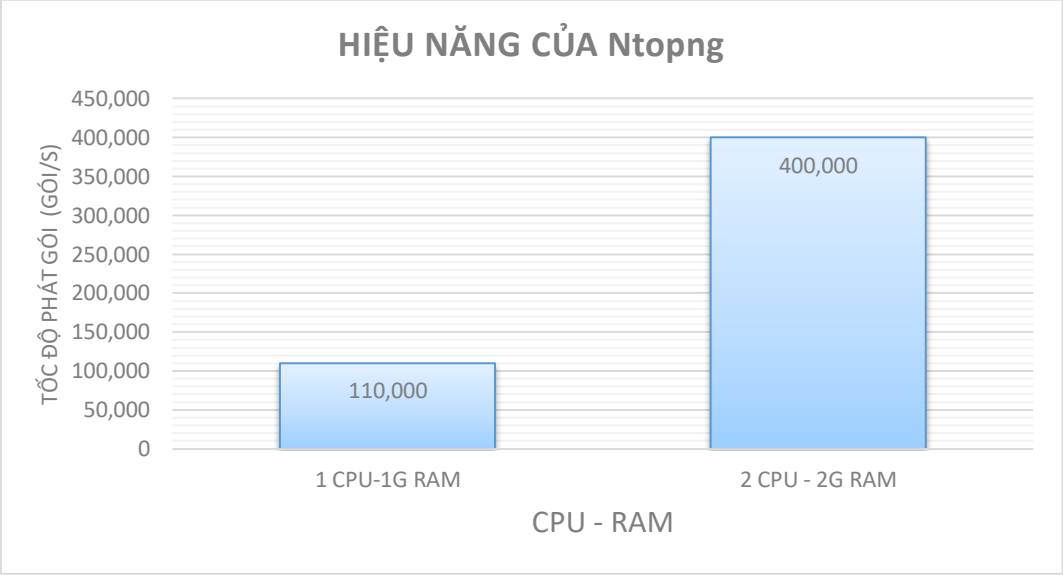
Ta sử dụng công cụ Graphite (có kết hợp Grafana) để đo đạc hiệu năng của Ntopng và Suricata. Graphite kết hợp Grafana cho phép giám sát tiêu thụ tài nguyên trên các mảy ảo thông qua giao diện web.

Hình 3.1. Giao diện web của Grafana

3.2. Kết quả đo hiệu năng của Ntopng và Suricata

3.2.1. Hiệu năng của Ntopng

Hiệu năng của chức năng mạng Ntopng



Biểu đồ 3.1. Hiệu năng của Ntopng

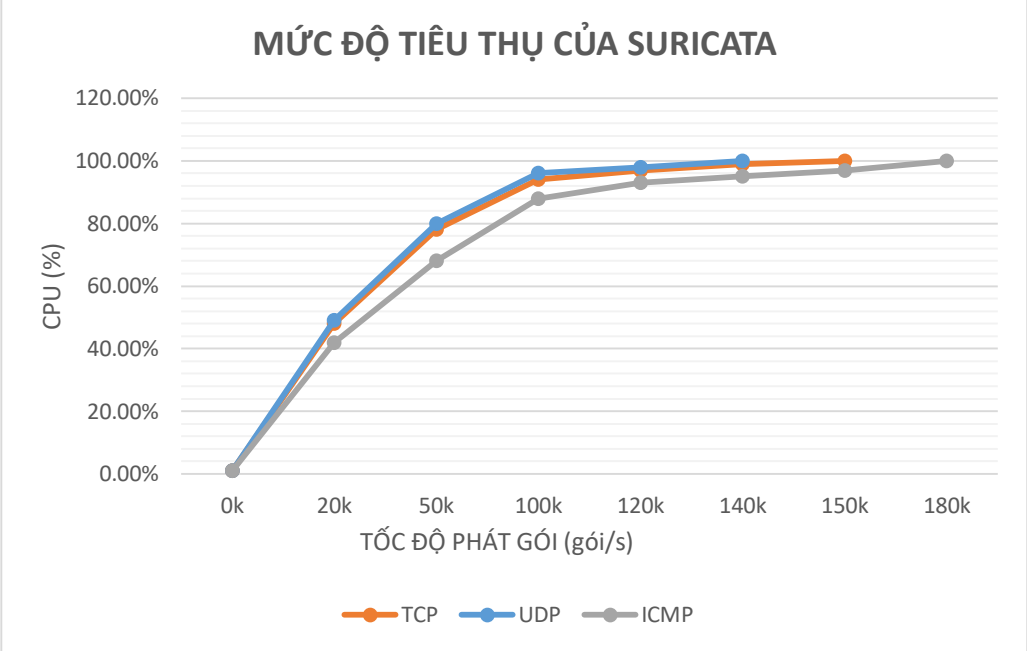
Biểu đồ 3.1 cho thấy với thông số tài nguyên 1 CPU và 1G RAM, Ntopng chỉ xử lý được 110000 gói tin/s, nhưng khi ta tăng thông số tài nguyên nên 2 CPU và 2G RAM, Ntopng đã xử lý được 400000 gói tin/s.

Với thông số tài nguyên 2 CPU - 2G RAM Ntopng xử lý được 400000 gói tin/s, vậy khi tăng thông số tài nguyên lên hơn nữa, Ntopng hoàn toàn có thể tiếp nhận và xử lý được hàng triệu gói tin/s. Với hiệu năng như vậy, chức năng mạng Ntopng có thể đáp ứng được lượng lưu lượng mạng trong thực tế, ví dụ trong trường hợp tại một thời điểm, máy chủ web tiếp nhận hàng chục nghìn yêu cầu từ khách hàng, lúc này sẽ có hàng trăm nghìn gói tin qua Ntopng, Ntopng có thể xử lý được lượng lưu lượng này mà không gây ra tắc nghẽn trong mạng.

3.2.2. Hiệu năng của Suricata

Ở đây, ta cài đặt Suricata với 10000 rules, thông số tài nguyên của Suricata sử dụng là 2 CPU và 2G RAM.

Mức độ tiêu thụ CPU của Suricata:



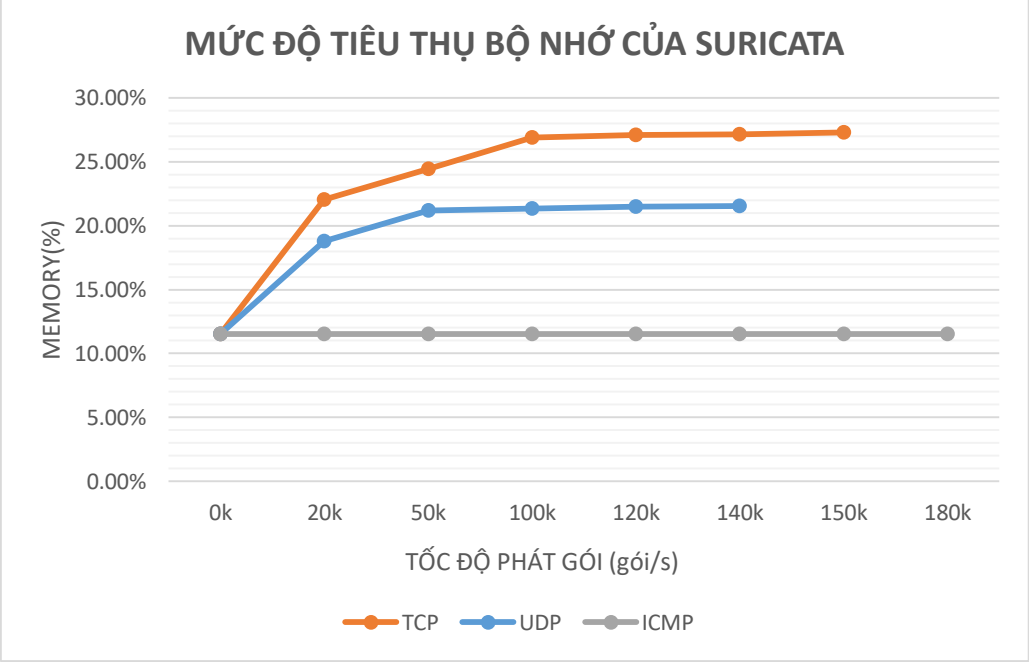
Biểu đồ 3.2. Hiệu năng sử dụng CPU của Suricata

Biểu đồ 3.2 cho thấy mức độ tiêu thụ CPU tăng dần khi tải tăng dần.

Đối với gói tin UDP, mức độ sử dụng CPU là 100% khi tốc độ phát gói là 140000 gói/s; đối với gói tin TCP, mức độ sử dụng CPU là 100% khi tốc độ phát gói là 150000 gói tin/s; còn đối với gói tin ICMP, mức độ sử dụng CPU là 100% khi tốc độ phát gói là 180000 gói tin/s.

Suricata xử lý gói tin TCP và UDP tiêu tốn CPU nhiều, xử lý gói tin ICMP là tiêu tốn CPU thấp hơn. Lý do của vấn đề này là, trong 10000 rules của Suricata, số lượng rules dành ICMP chỉ hàng vài chục, trong khi số lượng rules dành cho TCP và UDP lên tới vài nghìn. Thêm vào đó, ICMP là bản tin có phần dữ liệu là nhỏ không đáng kể, còn phần dữ liệu của TCP và UDP là tương đối lớn, đòi hỏi Suricata phải xử lý nhiều hơn, gây tốn CPU hơn.

Mức độ tiêu thụ bộ nhớ của Suricata



Biểu đồ 3.3. Hiệu năng sử dụng bộ nhớ của Suricata

Biểu đồ 3.3 cho thấy mức độ tiêu thụ bộ nhớ tăng dần khi tải tăng dần đối với gói tin TCP và UDP, mức độ tiêu thụ bố giữ nguyên đối với gói tin ICMP.

Với gói tin ICMP, khi tải đầu vào là 20000 gói tin/s , bộ nhớ sử dụng là 11.5 %; đến khi tải đầu vào là 180000 gói tin/s, bộ nhớ vẫn sử dụng là 11.5 %. Lý do, ICMP là bản tin có phần dữ liệu là nhỏ không đáng kể và số lượng rules dành ICMP trong Suricata chỉ vài chục. Vì vậy tiến trình xử lý những gói tin ICMP diễn ra rất nhanh và chuyển tiếp chúng đi vào mạng ngay khi xử lý xong, không lưu lại tại bộ nhớ.

Với gói tin UDP, khi tải đầu vào là 20000 gói tin/s , bộ nhớ sử dụng là 18.8 %; đến khi tải đầu vào là 140000 gói tin/s, bộ nhớ sử dụng tăng lên là 21.55 %. Lý do, UDP là bản tin có phần dữ liệu khá lớn. Vì vậy tiến trình xử lý những gói tin UDP diễn ra chậm hơn, Suricara cần phân tích dữ liệu chứa trong mỗi gói tin xem chúng chứa mã độc hay không, vì vậy cần lưu các gói tin chưa kịp xử lý trong bộ nhớ, điều này dẫn đến bộ nhớ sử dụng tăng lên khi tải đầu vào tăng.

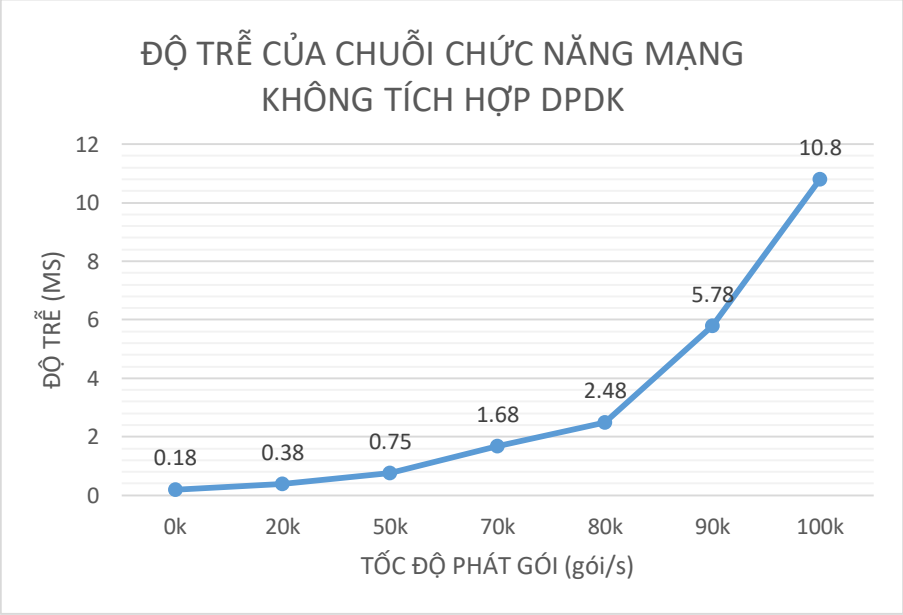
Với gói tin TCP, khi tải đầu vào là 20000 gói tin/s , bộ nhớ sử dụng là 22 %; đến khi tải đầu vào là 150000 gói tin/s, bộ nhớ sử dụng tăng lên là 27.2 %. Lý do, TCP là bản tin có phần dữ liệu khá lớn. Vì vậy tiến trình xử lý những gói tin TCP diễn ra chậm hơn, Suricara cần phân tích dữ liệu chứa trong mỗi gói tin xem chúng chứa mã độc hay không, vì vậy cần

lưu các gói tin chưa kịp xử lý trong bộ nhớ, điều này dẫn đến bộ nhớ sử dụng tăng lên khi tải đầu vào tăng.

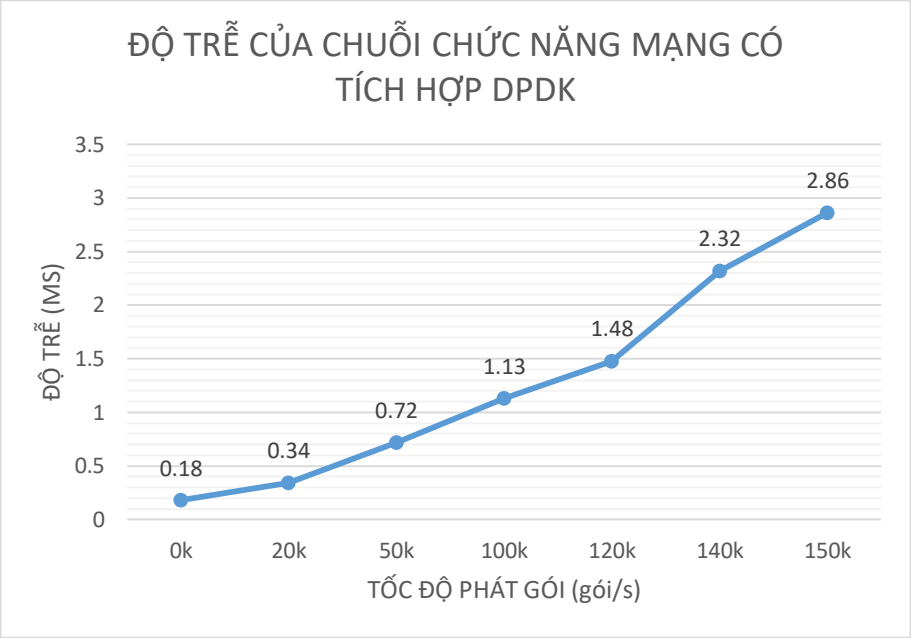
3.3. Kết quả đo băng thông, độ trễ và tỷ lệ mất gói của chuỗi chức năng mạng

Băng thông của chuỗi chức năng mạng có thể đạt được thấp nhất 1Gb/s và hơn thế nữa. **Băng thông** của chuỗi chức năng mạng không những phụ thuộc vào hiệu năng của các chức năng mạng, hiệu năng của Open vSwitch mà còn phụ thuộc vào băng thông mạng trong OpenStack và hạ tầng mạng vật lý bên ngoài. Qua đo đạc cho thấy, **băng thông** của chuỗi chức năng mạng có thể đạt được thấp nhất là 1Gb/s (do điều kiện dựng testbed chỉ có card mạng 1 Gb/s).

Độ trễ của chuỗi chức năng mạng không tích hợp DPDK và có tích hợp DPDK:



Biểu đồ 3.4. Độ trễ của chuỗi chức năng mạng không tích hợp DPDK



Biểu đồ 3.5. Độ trễ của chuỗi chức năng mạng có tích hợp DPDK

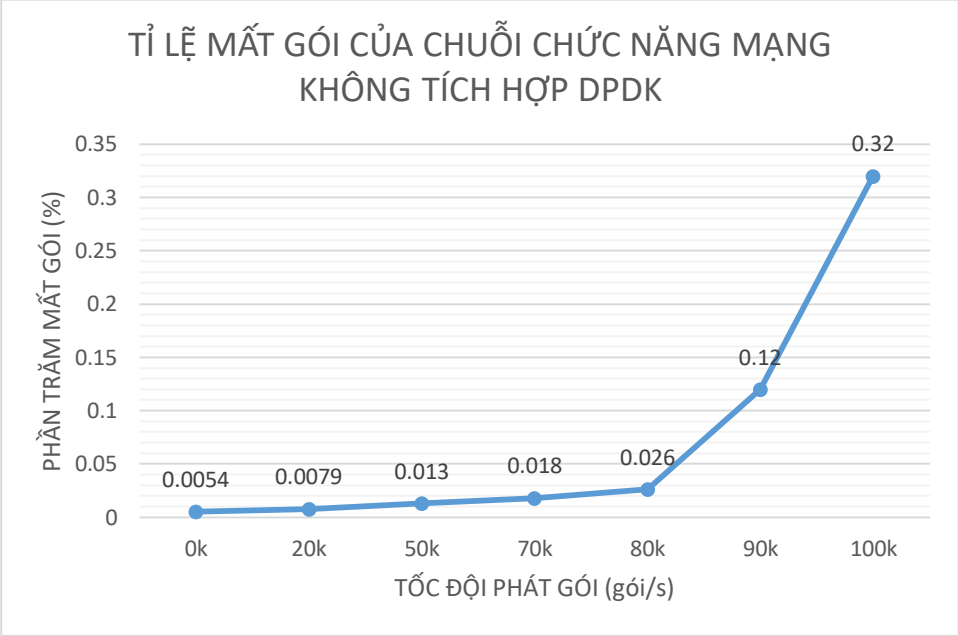
Biểu đồ 3.4 và 3.5 đều cho thấy độ trễ của chuỗi chức năng trong mạng tăng dần khi tốc độ phát gói tăng.

Với biểu đồ 3.4, khi không có lưu lượng đầu vào (tốc độ phát gói bằng 0) thì độ trễ là 0.18 ms; khi tốc độ phát gói là 80000 gói tin/s thì độ trễ là 2.48 ms; khi tốc độ phát gói là 100000 gói tin/s thì độ trễ tăng lên khá nhiều là 10.8 ms.

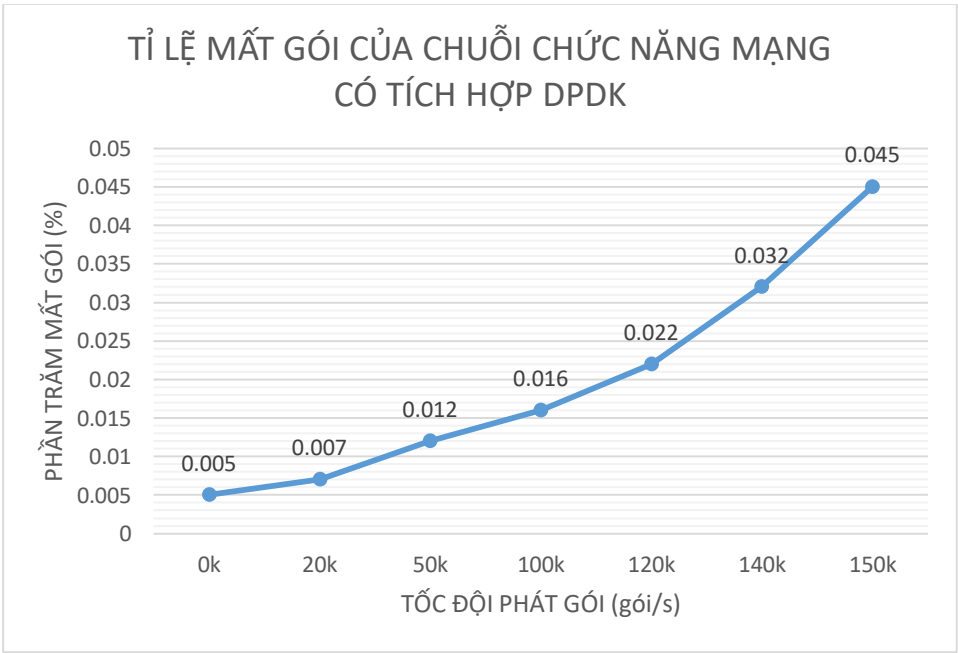
Với biểu đồ 3.5, khi không có lưu lượng đầu vào (tốc độ phát gói bằng 0) thì độ trễ cũng là 0.18 ms; khi tốc độ phát gói là 100000 gói tin/s thì độ trễ là 1.13 ms; khi tốc độ phát gói là 150000 gói tin/s thì độ trễ là 2.86 ms.

Ta có thể nhận thấy ở biểu đồ 3.4, bắt đầu từ tốc độ phát gói > 80000 gói tin/s thì độ trễ đột ngột tăng lên nhiều bất thường , điều này xảy ra là do với tốc độ phát gói đó, Open vSwitch không tích hợp DPDK bị quá tải, không xử lý kịp lưu lượng dữ liệu qua nó. Còn với biểu đồ 3.5, khi tốc độ phát gói tăng lên 100000 gói tin/s hay thậm chí 150000 gói tin trên thì độ trễ tăng khá ít, điều này là vì ở đây Open vSwitch đã được tính hợp DPDK khiến khả năng xử lý gói tin của switch tăng mạnh.

Tỷ lệ mất gói của chuỗi chức năng mạng không tích hợp DPDK và có tích hợp DPDK:



Biểu đồ 3.6. Tỷ lệ mất gói của chuỗi chức năng mạng không tích hợp DPDK



Biểu đồ 3.7. Tỷ lệ mất gói của chuỗi chức năng mạng có tích hợp DPDK

Biểu đồ 3.6 và 3.7 đều cho thấy tỷ lệ mất gói của chuỗi chức năng trong mạng tăng dần khi tốc độ phát gói tăng.

Với biểu đồ 3.6, khi không có lưu lượng đầu vào (tốc độ phát gói bằng 0) thì tỷ lệ mất gói rất thấp chỉ là 0.0054%; khi tốc độ phát gói là 80000 gói tin/s thì tỷ lệ mất gói là 0.026%; khi tốc độ phát gói là 100000 gói tin/s thì tỷ lệ mất gói tăng lên khá nhiều là 0.32%.

Với biểu đồ 3.7, khi không có lưu lượng đầu vào (tốc độ phát gói bằng 0) thì tỷ lệ mất gói rất thấp chỉ là 0.005%; khi tốc độ phát gói là 100000 gói tin/s thì tỷ lệ mất gói là 0.016%; khi tốc độ phát gói là 150000 gói tin/s thì tỷ lệ mất gói tăng là 0.045%.

Ta có thể nhận thấy ở biểu đồ 3.6, bắt đầu từ tốc độ phát gói > 80000 gói tin/s thì tỷ lệ mất gói đột ngột tăng lên nhiều bất thường , điều này xảy ra là do với tốc độ phát gói đó, Open vSwitch không tích hợp DPDK bị quá tải, không xử lý kịp lưu lượng dữ liệu qua nó. Còn với biểu đồ 3.7, khi tốc độ phát gói tăng lên 100000 gói tin/s hay thậm chí 150000 gói tin trên thì tỷ lệ mất gói tăng khá ít, điều này là vì ở đây Open vSwitch đã được tính hợp DPDK khiến khả năng xử lý gói tin của switch tăng mạnh.

3.4. Nhận xét, đánh giá

Nhận xét:

- Từ những kết quả trên, ta thấy mô hình chuỗi chức năng mạng đã được triển khai thành công và sử dụng ổn định trong thực tế. Luồng lưu lượng trước khi tới máy chủ Web server đã được điều khiển để đi qua các chức năng giám sát mạng (Ntopng), tường lửa (Iptables) và hệ thống phát hiện xâm nhập (Suricara).
- Mức độ tiêu thụ CPU, bộ nhớ của các chức năng mạng ảo tăng khi tải tăng.
- Mô hình triển khai chuỗi chức năng mạng trên nền tảng OpenStack nếu không được tích hợp DPDK thì hiệu năng tương đối thấp, luồng lưu lượng xử lý được trong mạng là tương đối nhỏ. Vì vậy trong thực tế, các nhà cung cấp mạng đều triển khai chuỗi chức năng mạng trên nền tảng OpenStack tích hợp DPDK, qua đó nâng cao hiệu suất và luồng lưu lượng qua chuỗi.
- Băng thông của chuỗi chức năng mạng khá cao, có thể đạt mức lớn hơn 1Gb/s. Nguyên nhân chưa thể kiểm tra được băng thông lớn hơn là do tài nguyên vật lý khi thực hiện triển khai testbed còn hạn chế, không có card mạng 10Gb/s. Độ trễ trong mạng khá nhỏ và tỉ lệ mất gói thấp.
- Nhìn chung, chuỗi chức năng mạng hoạt động khá ổn định, giúp các nhà cung cấp dịch vụ mạng tiết kiệm chi phí phần cứng và bớt lệ thuộc vào các nhà cung cấp thiết bị phần cứng chuyên dụng.

Kết quả đạt được:

- Có hiểu biết tương đối đầy đủ về nền tảng cloud OpenStack, Network Functions Virtualization và Service Function Chaining.
- Có hiểu biết về Data Plane Development Kip - DPDK.
- Triển khai thành công nền tảng OpenStack gồm các projects Keystone, Glance, Nova, Neutron, Horizon.
- Tích hợp OpenStack và DPDK để nâng cao hiệu năng xử lý gói tin của Open vSwitch.
- Triển khai thành công chuỗi chức năng mạng gồm 3 chức năng: giám sát lưu lượng mạng, tưởng lửa và hệ thống phát hiện xâm nhập.

Mô hình vẫn còn hạn chế ở số điểm như sau:

- Do tài nguyên hạn chế, testbed chỉ dựng 2 máy chủ vật lý nên mô hình OpenStack tương đối nhỏ.
- Mô hình với 3 chức năng năng Ntopng, Iptables và Suricata còn khá nhỏ, có thể thêm một số chức năng khác như cân bằng tải,…
- Do mô hình chỉ gồm 1 máy chủ vật lý làm Compute, em vẫn chưa thử nghiệm migrate chức năng mạng ảo trong khi chúng hoạt động.
- Thiết lập luồng lưu lượng của chuỗi chức năng mạng là luồng tĩnh dẫn đến không linh hoạt khi thực hiện thay đổi luồng dữ liệu trong mô hình.
- Chuỗi chức năng mạng ảo ở đây là các chức năng mạng ảo được cài đặt trên máy ảo (môi trường ảo là KVM-QEMU), chưa triển khai được chuỗi chức năng mạng mà các chức năng mạng được cài trên container (Docker).

KẾT LUẬN

Qua quá trình thực hiện đồ án, em đã có thêm nhiều kiến thức về các công nghệ ảo hóa, điện toán đám mây (Cloud Computing), nền tảng mã nguồn mở OpenStack, ảo hóa chức năng mạng (NFV) và chuỗi chức năng mạng (SFC) và rất nhiều kiến thức liên quan.

Đồ án của em đã triển khai thành công nền tảng OpenStack tích hợp DPDK, trên môi trường đó triển khai chuỗi chức năng mạng ảo với các thông số chất lượng dịch vụ như băng thông, độ trễ, tỷ lệ mất gói,… ở mức chấp nhận được.

Dựa vào những kết quả đạt được và hạn chế còn tồn đọng trong đồ án, em sẽ tiếp tục phát triển đề tài này trong tương tai theo hướng:

- Tích hợp Docker và OpenStack, qua đó triển khai các chức năng mạng trong các container thay vì máy ảo để nâng cao hiệu năng. Sau đó so sánh hiệu năng của chuỗi chức năng mạng triển khai bằng máy ảo và chuỗi chức năng mạng được triển khai bằng container.
- Xây dựng hệ thống NFV sử dụng Tacker trên nền tảng OpenStack để đảm bảo tính sẵn sàng cao.
- Tìm hiểu và học hỏi xem nhiều công nghệ triển khai NFV ngoài Tacker.
- Tích hợp SDN vào nền tảng OpenStack, qua đó kết hợp SDN với NFV để có thể tạo luồng lưu lượng trong chuỗi chức năng mạng một cách linh động và trực quan nhất.
- Nghiên cứu và triển khai nhiều mô hình chuỗi chức năng mạng khác có ứng dụng thực tế cao.
- Đo đạc thêm nhiều tham liên quan đến chất lượng dịch vụ mạng, từ đó đưa ra cái nhìn khái quát hơn về chất lượng dịch vụ của chuỗi.
- Mô hình hóa chuỗi chức năng mạng và đưa ra thuật toán giải quyết các vấn đề tồn động. Tính toán độ trễ trong mạng sử dụng mô hình hàng đợi và các công thức liên quan.

BẢNG ĐỐI CHIẾU THUẬT NGỮ ANH-VIỆT

Tiếng Anh	Tiếng Việt
Virtualization	Ảo hóa
Network Funtion Virtualization	Ảo hóa chức năng mạng
Virtualized Network Function	Các thiết bị mạng ảo hóa
Server	Máy chủ
Cloud Computing	Điện toán đám mây
Use case	Các ứng dụng trong thực tế
Instance	Một thể hiện cụ thể như máy ảo, router ảo, switch ảo,…
Testbed	Mô hình thử nghiệm
Webserver	máy chủ web
Firewall	Tưởng lửa
Monitor	Giám sát mạng

TÀI LIỆU THAM KHẢO

[1] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf/ truy cập lần cuối ngày 05/06/2019.

[2] https://artifacts.opnfv.org/sfc/docs/design/design.pdf/ truy cập lần cuối ngày 05/06/2019.

[3] https://www.slideshare.net/mirantis/openstack-architecture-43160012/ truy cập lần cuối ngày 05/06/2019.

[4] https://www.unixarena.com/2015/08/openstack-architecture-and-components-overview.html/ truy cập lần cuối ngày 05/06/2019.

[5] https://developers.redhat.com/blog/2015/03/24/live-migrating-qemu-kvm-virtual-machines/ truy cập lần cuối ngày 05/06/2019.

[6] https://www.networksecuritydaily.com/2017/11/13/network-function-virtualization-nfv/ truy cập lần cuối ngày 05/06/2019.

[7] https://uni2u.tistory.com/tag/OpenStack%20SFC/ truy cập lần cuối ngày 05/06/2019.

[8] https://software.intel.com/en-us/articles/open-vswitch-with-dpdk-overview/ truy cập lần cuối ngày 05/06/2019.

[9] https://www.thegeekstuff.com/2011/01/iptables-fundamentals/ truy cập lần cuối ngày 05/06/2019.

[10] https://www.comparitech.com/net-admin/network-intrusion-detection-tools/ truy cập lần cuối ngày 05/06/2019.

[11] https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts/ truy cập lần cuối ngày 05/06/2019.

[12] https://collectd.org/documentation.shtml/ truy cập lần cuối ngày 05/06/2019.

[13] Graphite Tool. [Online]. Available: https://graphiteapp.org/

[14] https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html truy cập lần cuối ngày 05/06/2019.

[15] http://www.interdynamix.com/2018/05/14/introduction-service-function-chaining/ truy cập lần cuối ngày 05/06/2019.

[16] https://en.wikipedia.org/wiki/Cloud_computing truy cập lần cuối ngày 05/06/2019.

[17] https://tools.ietf.org/html/draft-liu-sfc-use-cases-08 truy cập lần cuối ngày 05/06/2019.

[18] https://tools.ietf.org/html/draft-ietf-sfc-use-case-mobility-01 truy cập lần cuối ngày 05/06/2019.

[19] https://docs.openstack.org/networking-sfc/latest/ truy cập lần cuối ngày 05/06/2019.

[20] https://wiki.debian.org/Hugepages truy cập lần cuối ngày 05/06/2019.

[21] https://developers.redhat.com/blog/2017/06/28/ovs-dpdk-parameters-dealing-with-multi-numa/ truy cập lần cuối ngày 05/06/2019.

[22] OpenStack. [Online]. Available: https://docs.openstack.org/

[23] DPDK tool. [Online]. Available: http://doc.dpdk.org/

[24] Ntopng Tool. [Online]. Available: https://www.ntop.org/

[25] Suricata Tool. [Online]. Available: https://suricata-ids.org/

[26] Haproxy Tool. [Online]. Available: http://www.haproxy.org/

[27] Nginx Tool. [Online]. Available: https://www.nginx.com/

[28] Grafana Tool. [Online]. Available: https://grafana.com/

[29] BoNeSi Tool. [Online]. Available: https://github.com/Markus-Go/bonesi/ truy cập lần cuối ngày 05/06/2019.