**LỜI CẢM ƠN**

Trong suốt quá trình tìm hiểu và thực hiện đề tài “**Nghiên cứu hiệu năng mạng trên điện toán đám mây của OpenStack**”, cùng với sự cố gắng nổ lực của bản thân và rất nhiều sự quan tâm, giúp đỡ từ thầy cô, bạn bè. Đề tài nghiên cứu cơ bản đã hoàn thành.

Em xin được bày tỏ lòng biết ơn chân thành tới các thầy cô giáo trường Đại học Duy Tân nói chung và thầy cô bộ môn Kỹ Thuật Mạng nói riêng đã tận tình truyền đạt cho em những kiến thức vô cùng quý báu và động viên em trong suốt thời gian học tập tại trường.

Đặc biệt, em xin chân thành cảm ơn Ths.Võ Nhân Văn, thầy đã tận tình chỉ bảo, tạo điều kiện cho em hoàn thành đề tài nghiên cứu cũng như sữa chữa những sai sót trong suốt quá trình em thực hiện đề tài.

Mặc dù đã có nhiều cố gắng để thực hiện đề tài một cách hoàn chỉnh nhất. Song do buổi đầu mới làm quen với công tác nghiên cứu khoa học, tiếp cận với thực tế cũng như hạn chế về kiến thức và kinh nghiệm nên không thể tránh khỏi những thiếu sót nhất định mà bản thân chưa thấy được. Em rất mong được sự góp ý của quý Thầy Cô giáo và các bạn đồng nghiệp để khóa luận được hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

*Đà Nẵng, Ngày 07 Tháng 03 Năm 2016*

**Sinh viên thực hiện**

Lê Đức Trung

**LỜI CAM ĐOAN**

Tôi xin cam đoan đây là bài nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của giảng viên Ths.Võ Nhân Văn. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và không trùng lặp với các đề tài khác. Những thông tin phục vụ cho việc phân tích, nhận xét, đánh giá được chính tôi thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình. Trường Đại học Duy Tân không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*Đà Nẵng, Ngày 07 Tháng 03 Năm 2016*

**Sinh viên thực hiện**

Lê Đức Trung

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN VỀ ĐIỆN TOÁN ĐÁM MÂY 1](#_Toc449428922)

[1.1. GIỚI THIỆU CHUNG VỀ CÔNG NGHỆ ĐIỆN TOÁN ĐÁM MÂY 1](#_Toc449428923)

[1.1.1. Lịch sử 1](#_Toc449428924)

[1.1.2. Các khái niệm về điện toán đám mây 2](#_Toc449428925)

[1.2. ĐẶC ĐIỂM VÀ TÍNH CHẤT CƠ BẢN CỦA ĐIỆN TOÁN ĐÁM MÂY 4](#_Toc449428926)

[1.2.1. So sánh điện toán đám mây và điện toán lưới 4](#_Toc449428927)

[1.2.2. Các đặc tính của điện toán đám mây 8](#_Toc449428928)

[1.2.3. Các mô hình dịch vụ 9](#_Toc449428929)

[*1.2.3.1.* *Hạ tầng hướng dịch vụ (Iaas – Infrastructure as a Service)* 10](#_Toc449428930)

[*1.2.3.2.* *Nền tảng hướng một dịch vụ (PaaS – Platform as a Service)* 11](#_Toc449428931)

[*1.2.3.3.* *Phần mềm hoạt động như dịch vụ (SaaS – Software as a Service)* 13](#_Toc449428932)

[*1.2.3.4.* *Cơ bản về ba nhóm điện toán đám mây* 15](#_Toc449428933)

[1.2.4. Các mô hình triển khai 16](#_Toc449428934)

[*1.2.4.1.* *Hệ thống đám mây công cộng (Public Cloud)* 16](#_Toc449428935)

[*1.2.4.2.* *Hệ thống đám mây riêng (Private Cloud)* 17](#_Toc449428936)

[*1.2.4.3.* *Hệ thống đám mây lai (Hybrid Cloud)* 18](#_Toc449428937)

[1.3. ĐÁNH GIÁ ĐIỆN TOÁN ĐÁM MÂY 19](#_Toc449428938)

[1.3.1. Ưu điểm 19](#_Toc449428939)

[1.3.2. Nhược điểm 20](#_Toc449428940)

[1.4. LỰA CHỌN GIẢI PHÁP CHO MÔ HÌNH ĐIỆN TOÁN ĐÁM MÂY 21](#_Toc449428941)

[1.4.1. Lợi ích của việc triển khai điện toán đám mây bằng nguồn mở 21](#_Toc449428942)

[1.4.2. Các giải pháp mã nguồn mở cho mô hình điện toán đám mây 22](#_Toc449428943)

[*1.4.2.1.* *CloudStack* 22](#_Toc449428944)

[*1.4.2.2.* *Eucalyptus* 23](#_Toc449428945)

[*1.4.2.3.* *OpenNebula* 23](#_Toc449428946)

[*1.4.2.4.* *OpenStack* 23](#_Toc449428947)

[1.4.3. So sánh các giải pháp 24](#_Toc449428948)

[*1.4.3.1.* *Một số thông tin chung và đặc điểm chính* 24](#_Toc449428949)

[*1.4.3.2.* *Sự phát triển trong cộng đồng* 25](#_Toc449428950)

[CHƯƠNG 2: KIẾN TRÚC OPENSTACK 27](#_Toc449428951)

[2.1. TỔNG QUAN VỀ OPENSTACK 27](#_Toc449428953)

[2.1.1. Amazon Web Service - Nguồn cảm hứng cho sự ra đời của Openstack 27](#_Toc449428954)

[2.1.2. Lịch sử về Openstack 30](#_Toc449428955)

[2.1.3. Các phiên bản của Openstack 31](#_Toc449428956)

[2.2. CÁC THÀNH PHẦN CỦA OPENSTACK 34](#_Toc449428957)

[2.2.1. OpenStack compute 36](#_Toc449428958)

[2.2.2. OpenStack Object Storage 39](#_Toc449428959)

[2.2.3. OpenStack Image Service 41](#_Toc449428960)

[2.2.4. OpenStack Dashboard 42](#_Toc449428961)

[2.2.5. Identity Service 43](#_Toc449428962)

[2.2.6. OpenStack Networking 44](#_Toc449428963)

[2.2.7. OpenStack Metering/Monitoring 45](#_Toc449428964)

[2.2.8. OpenStack Orchestration 46](#_Toc449428965)

[2.3. TRIỂN KHAI MẠNG TRÊN OPENSTACK 47](#_Toc449428966)

[2.3.1. Nova-network 47](#_Toc449428967)

[*2.3.1.1.* *Flat Network* 47](#_Toc449428968)

[*2.3.1.2.* *Vlan Network* 48](#_Toc449428969)

[2.3.2. Neutron – Thành phần và kiến trúc 49](#_Toc449428970)

[*2.3.2.1.* *Neutron API* 50](#_Toc449428971)

[*2.3.2.2.* *Neutron plugin* 51](#_Toc449428972)

[*2.3.2.3.* *Kiến trúc Neutron* 52](#_Toc449428973)

[2.4. HIỆU SUẤT MẠNG 54](#_Toc449428974)

[2.4.1. Truyền băng thông 54](#_Toc449428975)

[2.4.2. End-to-end delay 55](#_Toc449428976)

[2.4.3. Processing, Queuing và Propagation delay 55](#_Toc449428977)

[2.4.4. Mất gói tin 56](#_Toc449428978)

[CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM VÀ ĐÁNH GIÁ HIỆU NĂNG MẠNG NỘI BỘ CỦA OPENSTACK 57](#_Toc449428979)

[3.1. MÔ HÌNH TRIỂN THỰC NGHIỆM 57](#_Toc449428981)

[3.1.1. Mô hình triển khai 57](#_Toc449428982)

[3.1.2. Kịch bản thử nghiệm 59](#_Toc449428983)

[*3.1.2.1.* *Trường hợp 1 - Cùng mạng cùng compute node.* 60](#_Toc449428984)

[*3.1.2.2.* *Trường hợp 2 - Khác mạng cùng compute node.* 60](#_Toc449428985)

[*3.1.2.3.* *Trường hợp 3 - Cùng mạng khác compute node.* 61](#_Toc449428986)

[*3.1.2.4.* *Trường hợp 4 - Khác mạng khác compute node.* 61](#_Toc449428987)

[3.2. PHÂN TÍCH VÀ ĐÁNH GIÁ 62](#_Toc449428988)

[3.2.1. Thông lượng và tỉ lệ mất gói tin UDP 62](#_Toc449428989)

[3.2.2. Thời gian trễ 64](#_Toc449428990)

[3.2.3. Đánh giá 65](#_Toc449428991)

[*3.2.3.1.* *Giữa các trường hợp cùng node compute* 65](#_Toc449428992)

[*3.2.3.2.* *Giữa các trường hợp khác node compute* 67](#_Toc449428993)

[3.3. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 68](#_Toc449428994)

[KẾT LUẬN 69](#_Toc449428995)

[TÀI LIỆU THAM KHẢO 70](#_Toc449428996)

**DANH MỤC BẢNG - BIỂU**

[Bảng 1.1. So sánh Grid Computing với Cloud Computing 5](#_Toc449428997)

[Bảng 1.2. Cơ bản về ba nhóm điện toán đám mây 15](#_Toc449428998)

[Bảng 1.3. Thông tin tổng quát về các giải pháp đám mây mã nguồn mở 24](#_Toc449428999)

[Bảng 1.4. Tính chất nổi bật các giải pháp đám mây mã nguồn mở 24](#_Toc449429000)

[Bảng 2.1. Các phiên bản của OpenStack 32](#_Toc449429001)

[Bảng 2.2. Các thành phần chính của Openstack 35](#_Toc449429002)

**DANH MỤC HÌNH ẢNH – SƠ ĐỒ**

[Hình 1.1. Xu hướng phát triển 1](#_Toc449429003)

[Hình 1.2. Mọi thứ đều tập trung vào đám mây 3](#_Toc449429004)

[Hình 1.3. Google trends of Grid computing and Cloud computing 5](#_Toc449429005)

[Hình 1.4. Các mô hình điện toán đám mây cơ bản 9](#_Toc449429006)

[Hình 1.5. Mô hình IaaS 10](#_Toc449429007)

[Hình 1.6. Dashboard của Windows Azure 11](#_Toc449429008)

[Hình 1.7. Mô hình PaaS 11](#_Toc449429009)

[Hình 1.8. Giao diện dashboard của Heroku 12](#_Toc449429010)

[Hình 1.9. Mô hình SaaS 13](#_Toc449429011)

[Hình 1.10. Phân tầng chức năng của từng mô hình dịch vụ 14](#_Toc449429012)

[Hình 1.11. Mô hình đám mây công cộng 16](#_Toc449429013)

[Hình 1.12. Mô hình đám mây riêng 17](#_Toc449429014)

[Hình 1.13. Mô hình đám mây lai 18](#_Toc449429015)

[Hình 1.14. Số lượng chủ đề trao đổi hàng tháng 25](#_Toc449429016)

[Hình 1.15. Số lượng người tích cực tham gia trao đổi hàng tháng 26](#_Toc449429017)

[Hình 2.1. Management console AWS 28](#_Toc449429018)

[Hình 2.2. Các thành phần chính của Openstack 34](#_Toc449429019)

[Hình 2.3. Các thành phần của Nova 37](#_Toc449429020)

[Hình 2.4. Kiến trúc Logic của Swift 40](#_Toc449429021)

[Hình 2.5. Các thành phần của Glance 42](#_Toc449429022)

[Hình 2.6. Chứng thực Keystone 43](#_Toc449429023)

[Hình 2.7. Cấu hình all-in-one, sử dụng một network adapter 47](#_Toc449429024)

[Hình 2.8. VLAN network 48](#_Toc449429025)

[Hình 2.9. Mô hình triển khai Quantum (Neutron) 49](#_Toc449429026)

[Hình 2.10. Neutron API 50](#_Toc449429027)

[Hình 2.11. Neutron agents 53](#_Toc449429028)

[Hình 2.12. Cách tính End-to-end delay 55](#_Toc449429029)

[Hình 2.13. Processing Delay 55](#_Toc449429030)

[Hình 2.14. Đánh rớt ở đuôi của hàng đợi 56](#_Toc449429031)

[Hình 3.1. Mô hình triển khai 58](#_Toc449429032)

[Hình 3.2. Mô hình triển khai với 2 máy ảo cùng mạng, cùng compute node 60](#_Toc449429033)

[Hình 3.3. Mô hình triển khai với 2 máy ảo khác mạng, cùng compute node 60](#_Toc449429034)

[Hình 3.4. Mô hình triển khai với 2 máy ảo cùng mạng, khác compute node 61](#_Toc449429035)

[Hình 3.5. Mô hình triển khai với 2 máy ảo khác mạng, khác compute node 61](#_Toc449429036)

[Hình 3.6. Thông lượng UDP trong 4 trường hợp 62](#_Toc449429037)

[Hình 3.7. Thông lượng UDP trung bình trong 4 trường hợp 63](#_Toc449429038)

[Hình 3.8. Tỉ lệ mất gói tin UDP trong 4 trường hợp 63](#_Toc449429039)

[Hình 3.9. Thời gian trễ gói tin 64](#_Toc449429040)

[Hình 3.10. Thông lượng UDP giữa trường hợp 1 và trường hợp 2 65](#_Toc449429041)

[Hình 3.11. Thông lượng UDP trung bình giữa trường hợp 1 và trường hợp 2 65](#_Toc449429042)

[Hình 3.12. Tỉ lệ mất gói tin UDP giữa trường hợp 1 và trường hợp 2 66](#_Toc449429043)

[Hình 3.13. Thời gian trễ gói giữa trường hợp 1 và trường hợp 2 66](#_Toc449429044)

[Hình 3.14. Thông lượng UDP giữa trường hợp 3 và trường hợp 4 67](#_Toc449429045)

[Hình 3.15. Thông lượng UDP trung bình giữa trường hợp 3 và trường hợp 4 67](#_Toc449429046)

**DANH MỤC CÁC TỪ VIẾT TẮT**

**API** Application Programming Interface

**CC** Clouding Computing

**CPU** Centranl Processing Unit

**CSMA/CD** Carrier Sense Multiple Access with Collision Detect

**DHCP** Dynamic Host Configuration Protocol

**FTP** File Transfer Protocol

**GRE** Generic Routing Encapsulation

**GUI** Graphic User Interface

**HDD** Hard Disk Drive

**IaaS** Infrastructure as a Service

**ICMP** Internet Control Message Protol

**IP** Internet Protocol

**IT** Information Technology

**KVM** Kernel Virtualization Machine

**LAB** Laboratory

**NIC** Network Interface Card

**PaaS** Platform as a Service

**RAM** Random Access Memory

**SaaS** Software as a Service

**SDRS** Storage Distributed Resources Scheduler

**UDP** User Datagram Protocol

**VCPU** Virtual Central Processing Unit

**VLAN** Virtual Local Area Network

**VM** Virtual Machine

**VNC** Virtual Network Computing

**MỞ ĐẦU**

Trong vài năm lại đây, điện toán đám mây đã tạo ra một cuộc cách mạng trong ngành công nghiệp máy tính, thay đổi cơ bản cách thức sử dụng các nguồn tài nguyên, cơ cấu vận hành cũng như việc lưu trữ, phân phối và xử lý thông tin. Đa số chúng ta đều đã và đang sử dụng một hoặc nhiều các dịch vụ ứng dụng công nghệ điện toán đám mây trong đời sống hàng ngày cũng như trong quản lý doanh nghiệp.

Điện toán đám mây là một giải pháp toàn diện cung cấp công nghệ thông tin như một dịch vụ. Nó là một giải pháp điện toán dựa trên internet ở đó cung cấp tài nguyên chia sẻ giống như dòng điện được phân phối trên lưới điện. Các máy tính trong các đám mây được cấu hình để làm việc cùng nhau và các ứng dụng khác nhau sử dụng sức mạnh điện toán tập hợp cứ như thể là chúng đang chạy trên một hệ thống duy nhất.

OpenStack, một mã nguồn mở trên nền tảng điện toán đám mây tương đối mới, tập trung vào việc cung cấp mạng như một dịch vụ (NAAS) sử dụng công nghệ ảo hóa. Openstack mang đến cho các doanh nghiệp khả năng xây dựng các đám mây riêng phục vụ cho công việc nội bộ hoặc lớn hơn là đám mây để cung cấp dịch vụ liên quan tới Cloud computing. OpenStack hứa hẹn sẽ là cơ sở hạ tầng của điện toán đám mây trong tương lại.

Vì các vấn đề nêu trên, em quyết định thực hiện đề tài “**Nghiên cứu hiệu năng mạng trên điện toán đám mây của OpenStack**”. Mục đích của đề tài là nghiên cứu hiệu năng mạng nội bộ của OpenStack dựa trên Neutron Networking. Các thông số hiệu năng mạng như thông lượng, thời gian trễ và vấn đề mất gói tin UDP giữa các máy ảo sử dụng công cụ IPERF. Từ đó những kết quả đó người quản trị mạng có thể ứng dụng vào trong thực tế. Nội dung báo cáo bao gồm 3 chương:

Chương 1: Tổng quan về điện toán đám mây.

Chương 2: Kiến trúc OpenStack.

Chương 3: Triển khai thực nghiệm và đánh giá hiệu năng mạng nội bộ của OpenStack.

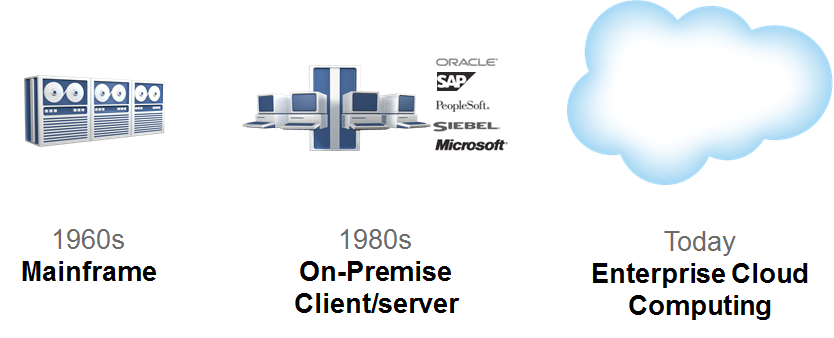
# CHƯƠNG 1: TỔNG QUAN VỀ ĐIỆN TOÁN ĐÁM MÂY

* 1. **GIỚI THIỆU CHUNG VỀ CÔNG NGHỆ ĐIỆN TOÁN ĐÁM MÂY**
     1. **Lịch sử**

Sự phát triển của công nghệ cũng như khoa học máy tính của những thập niên từ 60 – 90 đã tạo ra những cơ sở cho điện toán đám mây phát triển mạnh mẽ hơn.

Một trong những quan trọng trong lịch sử điện toán đám mây là vào năm 1999 trang Salesforce.com cung cấp các ứng dụng doanh nghiệp thông qua trang web của mình. Chính sự phát triển này đã mở đường cho các chuyên gia và các công ty phần mềm cung cấp các ứng dụng trên nền Internet.

Trong thời gian này một số công ty đã đầu tư mạnh vào lĩnh vực này. Có thể thấy đó là Amazon và Google với cung cấp dịch vụ phi lợi nhuận. Tuy nhiên để tồn tại và phát triển họ đã cải tiến rất nhiều về mô hình kinh doanh và khả năng đáp ứng dịch vụ cho khách hàng.

****

*Hình 1.1. Xu hướng phát triển*

Năm 2004, sự ra đời chính thức của mạng xã hội Facebook đã tạo ra cuộc cách mạng giao tiếp giữa người với người, mọi người có thể chia sẽ dữ liệu riêng tư của họ cho người thân bạn bè, chính điều này đã vô tình tạo ra một định nghĩa mà ta thường gọi là điện toán đám mây (Cloud Computing) cá nhân.

Vào năm 2006, Amazon đã từng bước mở rộng dịch vụ điện toán đám mây Elastic Compute Cloud (EC2) của mình, đây là hệ thống dịch vụ Web thương mại cho phép các công ty nhỏ, cá nhân thuê tài nguyên phần mềm để chạy các ứng dụng thông qua đám mây. Sau đó họ đưa ra Simple Storage Service (S3) với một mô hình mới “pay-as-you-go” (chi tiêu tùy theo khả năng) cho tất cả người dùng và các doanh nghiệp trên một hệ thống.

Năm 2007 Salesforce.com đưa ra force.com. Nền tảng như một dịch vụ (PaaS) để cho các nhà phát triển của các công ty xây dựng lưu trữ tất cả ứng dụng và trang web phục vụ hoạt động kinh doanh trên đám mây. Google Apps ra mắt năm 2009 cho phép mọi người tạo và lưu trữ dữ liệu hoàn toàn trên đám mây.

Trong 10 năm trở lại đây xu hướng sử dụng điện toán đám mây đang ngày càng phổ biến và sử dụng rộng rãi với đa dạng dịch vụ hơn. Năm 2010 Salesforce.com giới thiệu dịch vụ mà có thể sử dụng bất kỳ trên thiết bị, bất kỳ trên trên nền tảng nào được viết bằng bất kỳ ngôn ngữ lập trình nào.

Với sự phát triển của công nghệ ngày nay ngày càng nhiều thiết bị thông minh và nhu cầu về mọi mặt của người dùng trong doanh nghiệp ngày càng lớn thì xu hướng điên toán đám mây là xu hướng tất yếu.

Tại Việt Nam trong những năm gần đây khái niệm điện toán đám mây đã và đang được giới thiệu rộng rãi. Một số doanh nghiệp trong lĩnh vực Công nghệ Thông tin đã nhìn thấy xu hướng này và đang cung cấp các giải pháp điện toán đám mây cho Doanh nghiệp, tổ chức và người dùng cá nhân. Dịch vụ được sử dụng nhiều nhất ở Việt Nam hiện nay là hệ thống website, dịch vụ lưu trữ dữ liệu, các ứng dụng cơ bản như email, lịch và một số phần mềm quản lí dự án, tài chính, nhân sự… Đã cho thấy một mô hình kinh doanh mới ở Việt Nam kinh doanh cùng điện toán đám mây.

* + 1. **Các khái niệm về điện toán đám mây**

Có rất nhiều khái niệm về điện toán đám mây:

Theo **Wikipedia**: Điện toán đám mây là một mô hình điện toán gồm các tài nguyên điện toán thường được ảo hóa, có khả năng tùy biến linh hoạt và được cung cấp dưới dịch vụ thông qua Internet.

Theo **Defininitioncloudcomputing.com**: Điện toán đám mây là một kiến trúc phân tán có khả năng tập trung hóa các nguồn lực máy chủ dựa trên nền tảng có thể mở rộng nhằm cung cấp các tài nguyên điện toán theo nhu cầu.

Theo **IBM**: Điện toán đám mây là một mô hình cho phép truy cập thông qua mạng, từ bất kỳ nơi nào, một cách tiện lợi và theo nhu cầu, đến một kho dùng chung các tài nguyên tính toán có thể cấu hình được (như mạng máy tính, máy chủ, lưu trữ, ứng dụng và dịch vụ). Các tài nguyên này có thể được cấp phát hoặc trả về một cách nhanh chóng với nổ lực quản lý hoặc tương tác với nhà cung cấp dịch vụ ở mức tối thiểu.

Theo **Ian** **Foster**: “Một mô hình điện toán phân tán có tính co giãn lớn mà hướng theo co giãn về mặt kinh tế, là nơi chứa các sức mạnh tính toán, kho lưu trữ, các nền tảng (platform) và các dịch vụ được trực quan, ảo hóa và co giãn linh động, sẽ được phân phối theo nhu cầu cho các khách hàng bên ngoài thông qua Internet”.



*Hình 1.2. Mọi thứ đều tập trung vào đám mây*

Điện toán đám mây bao gồm: Phần mềm hoạt động như dịch vụ (SaaS: Softwareas a service), nền tảng như một dịch vụ (Paas: Platform as a Service), dịch vụ Web và những xu hướng công nghệ mới. Chúng đều dựa vào mạng Internet để đáp ứng nhu cầu sử dụng của người dùng. Những ví dụ tiêu biểu về điện toán đám mây là Salesforce.com và Google Apps. Chúng cung cấp những ứng dụng thương mại trực tuyến, được truy cập thông qua trình duyệt web, trong khi dữ liệu và phần mềm được lưu trên đám mây.

Điện toán đám mây thường bị nhầm lẫn với điện toán lưới (Grid Computing) (một loại hình điện toán phân tán được tạo bởi các mạng máy tính nhỏ hoặc các cặp máy tính, hoạt động phối hợp với nhau để thực hiện các chức năng rất lớn), điện toán theo nhu cầu (Utility Computing) (khối những tài nguyên máy tính, như các bộ xử lý và bộ nhớ, trong vai trò một dịch vụ trắc lượng tương tự với các công trình hạ tầng kỹ thuật truyền thống) và điện toán tự trị (Autonomic Computing) (các hệ thống máy tính có khả năng tự quản lý).

Trên thực tế, việc triển khai các cơ sở hạ tầng cho điện toán đám mây dựa trên các đặc điểm của điện toán lưới, điện toán theo nhu cầu và điện toán tự trị. Điện toán đám mây có thể được xem như là giai đoạn tự nhiên tiếp theo từ mô hình điện toán lưới.

* 1. **ĐẶC ĐIỂM VÀ TÍNH CHẤT CƠ BẢN CỦA ĐIỆN TOÁN ĐÁM MÂY**
     1. **So sánh điện toán đám mây và điện toán lưới**

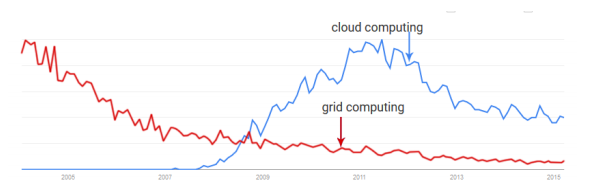
Điện toán đám mây và điện toán lưới có khả năng mở rộng, được thực hiện thông qua cân bằng truyền tải của các ứng dụng chạy độc lập trên một loạt các hệ thống hoạt động kết nối thông qua dịch vụ web. Phân phối CPU và băng thông mạng khi cần thiết, dung lượng lưu trữ hệ thống điều chỉnh theo số lượng người sử dụng, số lượng dữ liệu chuyển giao dựa trên thời gian cụ thể.

Hai loại tính toán liên quan đến đa nhiệm vụ và nhiều bên thuê dịch vụ, đó là nhiều người sử dụng có thể thực hiện các nhiệm vụ khác nhau, truy cập một hoặc nhiều trường hợp ứng dụng. Nó có thể làm giảm chi phí cơ sở hạ tầng và cải thiện khả năng chịu tải cao thông qua các nguồn tài nguyên lớn của người sử dụng chia sẻ.

Điện toán đám mây là một gợi ý cho tương lai, là thời điểm chúng ta không tính toán trên các máy tính cục bộ mà thực hiện tính toán trên các tiện ích tập trung được điều hành bởi thành phần thứ ba (third party).

Khi xem xét các định nghĩa về “đám mây”, “lưới”, chúng ta dễ dàng thấy rằng định nghĩa của “đám mây” có điểm trùng lấp với các định nghĩa của “lưới”. Điều này không phải là một vấn đề đáng ngạc nhiên, bởi vì “đám mây” không ra đời một cách riêng lẻ hay độc lập mà nó dựa trên nền tảng của các công nghệ trước đó.

Theo thống kê của Google Trends về chỉ số tìm kiếm liên quan đến điện toán lưới và điện toán đám mây đến tháng 3/2015. Điện toán đám mây có xu hướng tìm kiếm nhiều hơn, còn điện toán lưới vẫn ở mức ổn định từ 2012 đến nay.



*Hình 1.3. Google trends of Grid computing and Cloud computing*

Sự khác biệt giữa hai công nghệ này nằm trong các nhiệm vụ được tính toán trong từng môi trường tương ứng. Điện toán lưới và điện toán đám mây thông được so sánh qua các đặc trưng.

*Bảng 1.1. So sánh Grid Computing với Cloud Computing*

|  |  |  |
| --- | --- | --- |
| Bảng so sánh Grid Computing và Clouding Computing | | |
| Đặc trưng | Grid computing | Cloud computing |
| Mục đích sử dụng | Hợp tác chia sẽ các nguồn tài nguyên | Sử dụng dịch vụ |
| Sức mạnh tính toán | Tính toán mạnh hơn điện toán đám mây, sử dụng khả năng tính toán của internet | Sử dụng khả năng tính toán đám trong nội bộ của “đám mây” |
| Hệ điều hành | Bất kỳ hệ điều hành tiêu chuẩn nào | Một máy ảo có nhiều hệ điều hành chạy |
| Khả năng cộng tác | Theo tiêu chuẩn lưới mở | Dựa vào dịch vụ Web |
| Lưu trữ | Lưu trữ nhiều hơn “đám mây”, dùng các giao thức để tìm kiếm các tài nguyên thích hợp trên mạng để lưu trữ | Khả năng lưu trữ ít hơn “lưới”, dùng các trung tâm dữ liệu trong việc lưu trữ |
| Tốc độ truyền dữ liệu (trao đổi các tài nguyên trong lúc thực thi) | Tốc độ chậm hơn “đám mây”, tốc độ thường là mega byte | Nhanh hơn “lưới”, việc trao đổi tài nguyên thường thực hiện bằng đường truyền nội bộ, được xây dựng để kết nối giữa các trung tâm dữ liệu. Tốc độ có thể lên đến hàng giga byte |
| Khả năng mở rộng | Có khả năng mở rộng. Khi có nhu cầu sử dụng thêm tài nguyên thì hệ thống sẽ tìm trên mạng xem hiện có tài nguyên nào đáp ứng phù hợp nhu cầu của mình | Có khả năng mở rộng, co lại dễ dàng và nhanh (theo nhu cầu sử dụng) |
| Phạm vi ứng dụng | Chủ yếu hướng tới khoa học | Chủ yếu hướng tới thương mại, quan tâm đến việc phục vụ nhu cầu của khách hàng thông qua việc cung cấp các dịch vụ theo nhu cầu khách hàng |
| Quản lý tài nguyên | Phân tán | Tập trung/ Phân tán |
| Tài nguyên | Việc sử dụng tài nguyên thông qua việc tìm kiếm các tài nguyên trên internet, người dùng không thể cấu hình tài nguyên theo ý muốn | Cung cấp tài nguyên theo dạng tài nguyên thống nhất, người dùng được phép cấu hình tài nguyên theo nhu cầu sử dụng |
| Quản lý người dùng | Tổ chức phân cấp và ảo hóa là nền tảng | Tập trung hoặc có thể ủy nhiệm cho bên thứ ba |
| Cấp phát/ Lập lịch | Phân tán | Tập trung/ Phân tán |
| Những ứng dụng | DDGrid (Drug Discovery Grid), MammoGrid, Geodise | Cloudo (Google apps, Amazon Web Service, …), RoboEarth, Panda Cloud, Antivirus |
| Các công cụ hổ trợ | Nimrod-G, Gridbus, Legion | Cloudera, CloudSim, Zenoss |

Đối với điện toán lưới, một nhiệm vụ lớn được chia thành các nhiệm vụ nhỏ hơn và phân phối đến các máy chủ khác nhau. Khi nhiệm vụ hoàn thành, kết quả được gửi lại cho máy chính, sau đó sẽ cung cấp một đầu ra duy nhất. Ý tưởng chính của công nghệ điện toán lưới là tận dụng năng lực của tất cả các máy tính đang có (kể cả các máy tính đã cũ, lạc hậu) để tạo ra một siêu máy tính ảo khổng lồ thông qua một giải pháp phần mềm đệm (middleware) mà không "cướp quyền" điều khiển của các thành viên trong “lưới”. Còn điện toán đám mây là hình thức phát triển của điện toán lưới với mức giá thấp cạnh tranh trong việc cung cấp tài nguyên theo yêu cầu giúp làm giảm chi phí đầu tư cho hệ thống, cơ sở hạ tầng và giảm chi phí cho nhân lực về công nghệ thông tin.

Điện toán đám mây và điện toán lưới khác nhau trong phương pháp mà họ sử dụng. Việc so sánh giữa “lưới” và “đám mây” trong thời điểm hiện tại chỉ phản ánh được phần nào các ưu điểm và khuyết điểm hiện có của chúng. Nhưng cả hai cộng đồng này vẫn đang không ngừng phát triển để hoàn thiện. Sử dụng các công nghệ này như thế nào tùy thuộc vào đáp ứng nhu cầu công việc của người sử dụng.

* + 1. **Các đặc tính của điện toán đám mây**

Đại bộ phận hạ tầng cơ sở của điện toán đám mây hiện nay là sự kết hợp của những dịch vụ đáng tin cậy được phân phối thông qua các trung tâm dữ liệu (data center) được xây dựng trên những máy chủ với những cấp độ khác nhau của các công nghệ ảo hóa. Những dịch vụ này có thể được truy cập từ bất kỳ đâu trên thế giới, trong đó đám mây là một điểm truy cập duy nhất cho tất cả các máy tính có nhu cầu của khách hàng. Các dịch vụ thương mại cần đáp ứng yêu cầu chất lượng dịch vụ từ phía khách hàng và thông thường đều đưa ra các mức thỏa thuận dịch vụ (Service level agreement). Các tiêu chuẩn mở (Open standard) và phần mềm mã nguồn mở (open source software) cũng góp phần phát triển điện toán máy chủ ảo.

Như vậy, trước đây để có thể triển khai một ứng dụng (ví dụ một trang Web), bạn phải đi mua/thuê một hay nhiều máy chủ (server), sau đó đặt máy chủ tại các trung tâm dữ liệu (data center) thì nay điện toán đám mây cho phép bạn giản lược quá trình mua/thuê đi. Bạn chỉ cần nêu ra yêu cầu của mình, hệ thống sẽ tự động gom nhặt các tài nguyên rỗi (free) để đáp ứng yêu cầu của bạn. Chính vì vậy, có thể kể đến một vài lợi ích cơ bản của điện toán đám mây như sau:

**Sử dụng các tài nguyên tính toán động (Dynamic computing resources)**: Các tài nguyên được cấp phát cho doanh nghiệp đúng như những gì doanh nghiệp muốn một cách tức thời. Thay vì việc doanh nghiệp phải tính toán xem có nên mở rộng hay không, phải đầu tư bao nhiêu máy chủ thì nay doanh nghiệp chỉ cần yêu cầu "Hey, đám mây, chúng tôi cần thêm tài nguyên tương đương với 1 CPU 3.0 GHz, 128GB RAM…" và đám mây sẽ tự tìm kiếm tài nguyên rỗi để cung cấp cho bạn.

**Giảm chi phí**: Doanh nghiệp sẽ có khả năng cắt giảm chi phí để mua bán, cài đặt và bảo trì tài nguyên. Rõ ràng thay vì việc phải cử một chuyên gia đi mua máy chủ, cài đặt máy chủ, bảo trì máy chủ thì nay bạn chẳng cần phải làm gì ngoài việc xác định chính xác tài nguyên mình cần và yêu cầu.

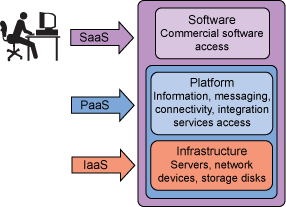
**Giảm độ phức tạp trong cơ cấu của doanh nghiệp**: Doanh nghiệp sản xuất hàng hóa mà lại phải có cả một chuyên gia IT để vận hành, bảo trì máy chủ thì quá tốn kém. Nếu khoán ngoài được quá trình này thì doanh nghiệp sẽ chỉ tập trung vào việc sản xuất hàng hóa chuyên môn của mình và giảm bớt được độ phức tạp trong cơ cấu.

**Tăng khả năng sử dụng tài nguyên tính toán**: Một trong những câu hỏi đau đầu của việc đầu tư tài nguyên (ví dụ máy chủ) là bao lâu thì nó sẽ hết khấu hao, tôi đầu tư như thế có lãi hay không, có bị outdate về công nghệ hay không. Khi sử dụng tài nguyên trên đám mây thì bạn không còn phải quan tâm tới điều này nữa.

* + 1. **Các mô hình dịch vụ**

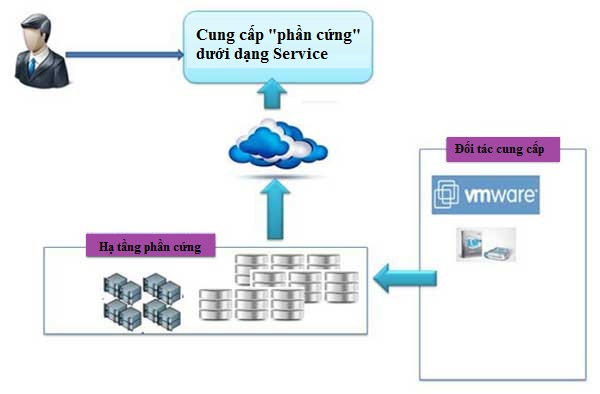
Các nhà cung cấp dịch vụ điện toán đám mây cung cấp các dịch vụ của họ theo ba mô hình cơ bản:

* Cơ sở hạ tầng như một dịch vụ (IaaS)
* Nền tảng như một dịch vụ (PaaS)
* Phần mềm như một dịch vụ (SaaS)
* Ngoài ra, một số dịch vụ hỗ trợ khác



*Hình 1.4. Các mô hình điện toán đám mây cơ bản*

* + - 1. ***Hạ tầng hướng dịch vụ (Iaas – Infrastructure as a Service)***



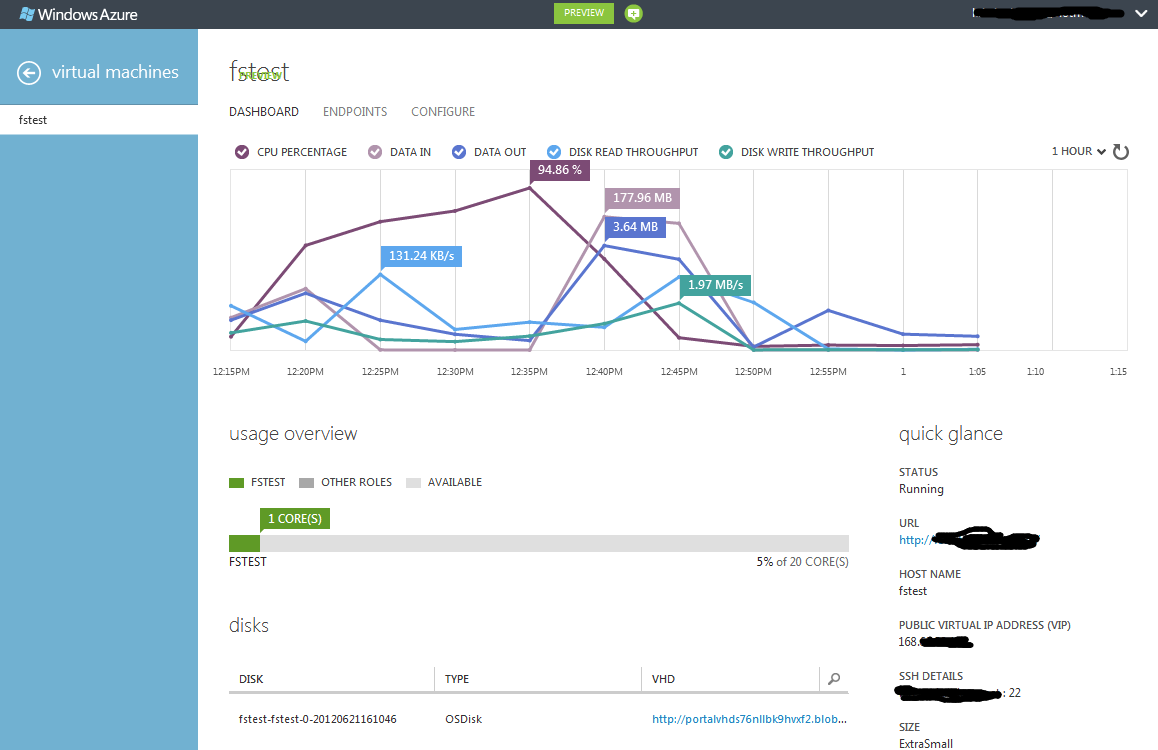
*Hình 1.5. Mô hình IaaS*

Infrastructure as a service (IaaS) là tầng thấp nhất của Điện toán đám mây, nơi tập hợp các tài sản vật lý như các phần cứng máy chủ, hệ thống lưu trữ và các thiết bị mạng, được chia sẻ và cung cấp dưới dạng dịch vụ IaaS cho các tổ chức hay doanh nghiệp khác nhau. Cũng giống như dịch vụ PaaS, ảo hóa là công nghệ được sử dụng rộng rãi để tạo ra cơ chế chia sẻ và phân phối các nguồn tài nguyên theo yêu cầu.

IaaS cung cấp các máy tính (máy vật lý hoặc máy ảo) và các tài nguyên khác. Một Hypervisor (bộ phận quản lý máy ảo) đảm nhận chạy các máy ảo. Một tập hợp các Hypervisor trong hệ thống clou có thể cung cấp một lượng lớn các máy ảo, có thể mở rộng hoặc thu hẹp mô hình theo yêu cầu của khách hàng. IaaS thường hổ trợ các tài nguyên khác như file ảnh, lưu trữ, tường lửa, cân tải, địa chỉ IP, mạng nội bộ (VLANs) và các bộ phận mềm. Nguồn tài nguyên phần cứng trong các trung tâm dữ liệu của nhà cung cấp dịch vụ.

Các nhà cung cấp IaaS trên thế giới: Amazon EC2, Windows Azure Virtual Machines, DynDNS, Google Compute Engine, Rackspace Cloud, HP Cloud, …

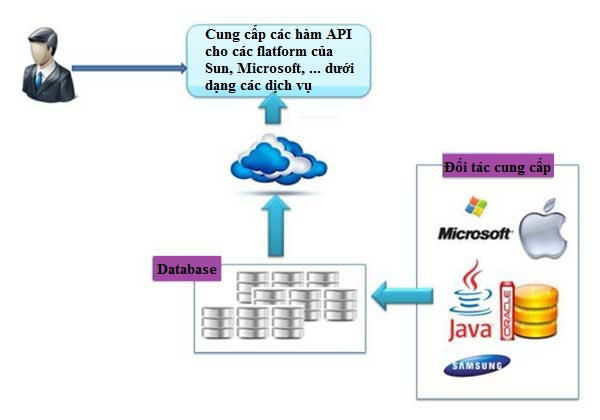
Ví dụ về IaaS:

**

*Hình 1.6. Dashboard của Windows Azure*

Windows Azure Virtual Machines cho phép tạo các máy ảo từ kho các file ảnh sẵn có hoặc sử dụng file ảnh do người dùng cung cấp. Người sử dụng có toàn quyền điều khiển máy ảo đã tạo. Với Dashboard, người dùng có thể theo dõi tình trạng của máy ảo qua biểu đồ thời gian thực.

* + - 1. ***Nền tảng hướng một dịch vụ (PaaS – Platform as a Service)***

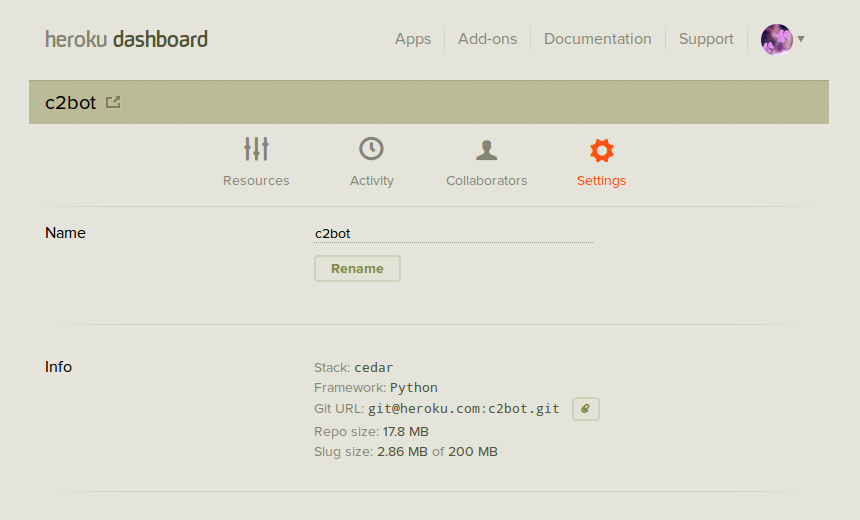


*Hình 1.7. Mô hình PaaS*

Đây cũng là một biến thể của SaaS nhưng mô hình này là một nhánh của điện toán đám mây (cloud computing), mang đến môi trường phát triển như một dịch vụ: người sử dụng xây dựng ứng dụng chạy trên cơ sở hạ tầng của nhà cung cấp và phân phối tới người sử dụng thông qua máy chủ của nhà cung cấp đó. Người sử dụng sẽ không hoàn toàn được tự do vì bị ràng buộc về mặt thiết kế và công nghệ.

PaaS đưa ra một nền tảng điện toán bao gồm cả hệ điều hành, môi trường lập trình, cơ sở dữ liệu và máy chủ web. Các nhà phát triển ứng dụng có thể phát triển, chạy các giải pháp phần mềm của họ trên nền tảng cloud mà không cần lo đến chi phí và sự phức tạp của việc mua và quản lý hạ tầng phần cứng và phần mềm bên dưới. Các tổ chức cung cấp PaaS: Heroku, Google App Engine, Windows Azure.

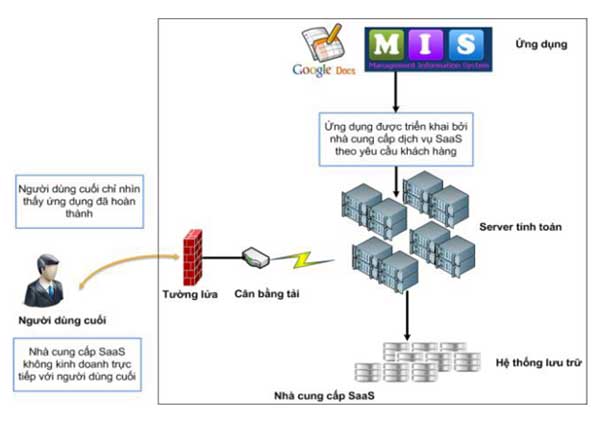
Ví dụ về PaaS: Heroku



*Hình 1.8. Giao diện dashboard của Heroku*

Heroku cung cấp nền tảng hổ trợ môi trường lập trình các ngôn ngữ Ruby, Java, Node.js Scala, Clojure, Python và PHP cùng với các cơ sở dữ liệu như PostgreSQL, MongoDB, Redis, … trên nền tảng hệ điều hành Debian và Ubuntu. Các nhà phát triển phần mềm đưa mã nguồn vào hệ thống của Heroku tự động thực hiện. Lượng tài nguyên ứng dụng sử dụng được đo và ghi lại, nhà phát triển sẽ trả tiền cho lượng tài nguyên đó.

* + - 1. ***Phần mềm hoạt động như dịch vụ (SaaS – Software as a Service)***



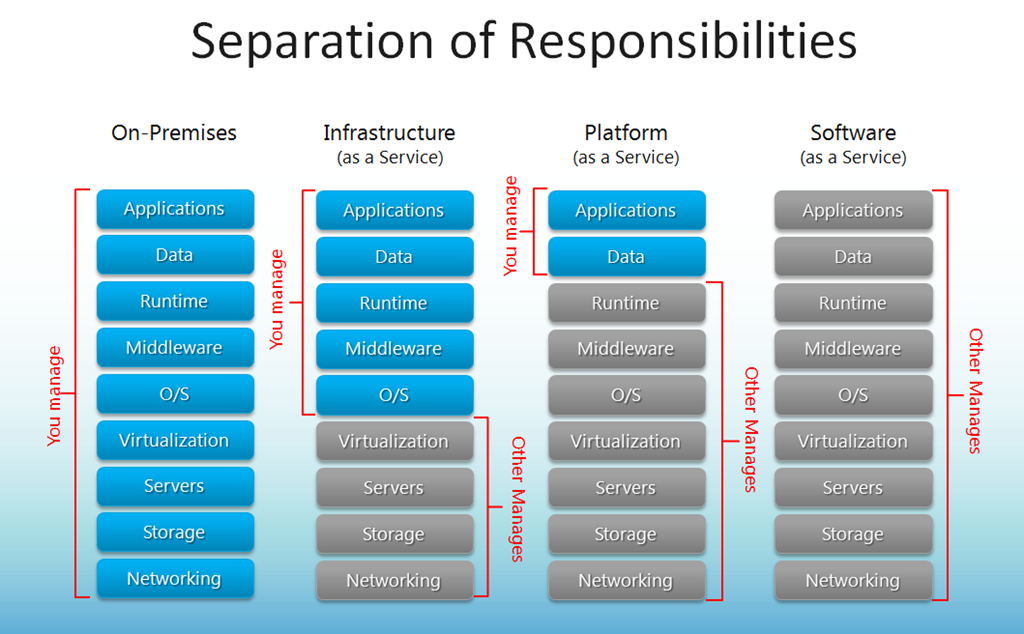
*Hình 1.9. Mô hình SaaS*

Phần mềm hoạt động hướng dịch vụ hoặc gọi tắt là Phần mềm dịch vụ - SaaS – Software as a Service, là mô hình triển khai phần mềm, một nhánh của điện toán đám mây, theo đó các nhà cung cấp phần mềm như là các dịch vụ theo yêu cầu cho khách hàng. Theo định nghĩa của hãng nghiên cứu toàn cầu IDC là: "phần mềm hoạt động trên web, được quản lý bởi nhà cung cấp và cho phép người sử dụng truy cập từ xa".

SaaS là một mô hình phân phối phần mềm, trong đó các ứng dụng sẽ được cung cấp, quản lý, vận hành bởi một nhà cung cấp dịch vụ, người dùng có thể truy cập qua mạng. Toàn bộ ứng dụng và dữ liệu người dùng đều được lưu trữ bởi nhà cung cấp dịch vụ. Người dùng thường truy cập SaaS qua trình duyệt Web. Ví dụ: Gmail, Box.net, Mediafire, …

Ví dụ về SaaS: Google Drive

Người dùng đăng kí sử dụng dịch vụ. Với tài khoản đã đăng ký, Google cung cấp cho người dùng một lượng lưu trữ và khả năng truy cập ứng dụng Google Drive trực tuyến. Với ứng dụng web này, người dùng có thể soạn thảo, lưu trữ, chia sẻ các file văn bản, bảng tính tự như sản phẩm Word, Excel chạy trên các máy tính của Microsoft Office. Người dùng chỉ cần có một client để truy cập và sử sụng ở bất cứ nơi đâu.



*Hình 1.10. Phân tầng chức năng của từng mô hình dịch vụ*

Hình 1.10 chỉ rõ chức năng của mỗi mô hình đảm nhận. Ở mỗi dịch vụ, nhà cung cấp sẽ quản lý các phần có tô màu xám, người dùng sẽ được quản lý các phần tô màu xanh. Với IaaS, người dùng sẽ không cần lo lắng về hạ tầng mạng, khả năng lưu trữ, máy vật lý và công nghệ ảo hóa mà chỉ cần quản lý hệ điều hành và các ứng dụng chạy trên nó. Với PaaS, phần quản lý của người dùng thu hẹp lại chỉ còn dữ liệu và các ứng dụng của họ, mọi thứ khác đều do nhà cung cấp dịch vụ quản lý và vận hành. Đến mô hình SaaS thì người dùng không còn quản lý bất cứ tài nguyên nào, tất cả những gì họ có là quyền truy cập đến dịch vụ, ngay cả dữ liệu cá nhân của người dùng cũng do nhà cung cấp nắm giữ và quản lý.

Ba mô hình nói trên thực hiện cung cấp các dịch vụ ở ba tầng khác nhau. Các dịch vụ tầng trên được xây dựng trên các dịch vụ tầng thấp hơn để nâng cao khả năng mở rộng của toàn bộ hệ thống.

Ngoài ba mô hình trên, còn có các mô hình khác như Database as a service (DbaaS), Network as a service (NaaS) …

* + - 1. ***Cơ bản về ba nhóm điện toán đám mây***

*Bảng 1.2. Cơ bản về ba nhóm điện toán đám mây*

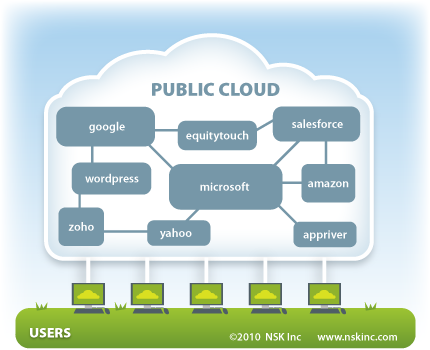
|  |  |  |  |
| --- | --- | --- | --- |
|  | Đặc điểm | Lợi thế | Bất lợi |
| IaaS | - Luôn độc lập về nền tảng.  - Chia sẻ chi phí cơ sở hạ tầng và do đó làm giảm chi phí này.  - Các thỏa thuận ở mức dịch vụ .  - Trả tiền theo mức sử dụng, tự điều chỉnh quy mô. | - Tránh được chi phí vốn cho phần cứng và nguồn nhân lực.  - Giảm rủi ro lợi tức đầu tư .  - Rào cản thấp khi tham gia vào.  - Điều chỉnh quy mô tự động hóa và trơn tru. | - Năng suất và hiệu quả kinh doanh phụ thuộc nhiều vào khả năng của nhà cung cấp.  - Chi phí dài hạn có tiềm năng lớn hơn.  - Sự tập trung hóa đòi hỏi các biện pháp an ninh khác hoặc mới. |
| PaaS | - Dùng cơ sở hạ tầng đám mây.  - Cung cấp các phương pháp quản lý dự án nhanh. | - Triển khai phiên bản trơn tru. | - Sự tập trung hóa đòi hỏi các biện pháp an ninh khác hoặc mới |
| SaaS | - Các thỏa thuận ở mức dịch vụ .  - Giao diện người dùng do các ứng dụng máy khách nhẹ (thin-client) cung cấp.  - Các thành phần điện toán đám mây qua các API.  - Được ghép lỏng; theo mô đun.  - Khả năng tương tác theo ngữ nghĩa. | - Tránh được chi phí vốn cho phần mềm và phát triển tài nguyên.  - Giảm rủi ro lợi tức đầu tư.  - Cập nhật lặp nhiều lần và trơn tru. | - Sự tập trung hóa đòi hỏi các biện pháp an ninh khác hoặc mới |

* + 1. **Các mô hình triển khai**

Trong thực tế, các hệ thống đám mây được triển khai rất đa dạng để phù hợp với nhu cầu sử dụng của từng tổ chức cụ thể. Nhìn chung, ta có thể chia hệ thống đám mây ra thành ba loại như hình sau:

* + - 1. ***Hệ thống đám mây công cộng (Public Cloud)***

Public Cloud dành cho nhiều người sử dụng được sở hữu bởi một công ty nào đó kinh doanh dịch vụ cho người dùng cuối. Public Cloud có nhiều dạng và tồn tại dưới nhiều hình thức như là Windows Azure, Microsoft Office 365 và Amazon Elastic Compute Cloud… Bạn cũng có thể tìm thấy các dịch vụ với quy mô nhỏ hơn với những dịch vụ khách hàng phù hợp với nhu cầu cá nhân.



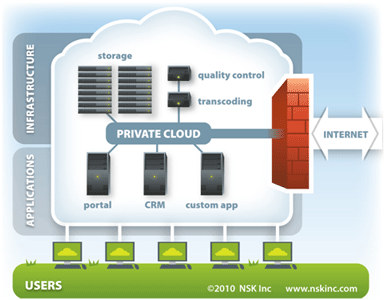
*Hình 1.11. Mô hình đám mây công cộng*

Ưu điểm lớn nhất của Public Cloud chính là nó luôn được sẵn sàng để sử dụng nhanh chóng. Một ứng dụng kinh doanh mới nhất có thể được triển khai chỉ trong vòng vài phút. Doanh nghiệp không cần đầu tư vào hệ thống hạ tầng IT nội bộ để vận hành và đưa ra giải pháp nữa. Điều này có làm cho các chuyên gia IT trở nên thừa thải không?

Còn có nhiều vấn đề ảnh hưởng đến quyết định của người ra quyết định. Public Cloud đặt ở đâu? Công ty sở hữu nó đến từ quốc gia nào? Câu trả lời cho các câu hỏi trên có thể ảnh hưởng đến sự thống nhất vế quốc gia và những quy tắc của nền công nghiệp. Những sự hỗ trợ nào bạn có thể đạt được với công ty viễn thông? Bạn có nghĩ Public Cloud sẽ khác? Có thể Public Cloud có đội ngũ hỗ trợ tốt nhưng họ rất có thể sẽ đẩy bạn đi 5000 dặm để có được một cuộc đàm thoại qua email. Bạn có thể tùy biến bao nhiêu với đám mây công cộng này và nó kết hợp với dịch vụ nội bộ của doanh nghiệp bạn ra sao thì không biết được.

* + - 1. ***Hệ thống đám mây riêng (Private Cloud)***

Lợi ích của Private Cloud là doanh nghiệp có thể tự thiết kế nó rồi tùy biến theo thời gian cho phù hợp với mình. Họ có thể kiểm soát được chất lượng dịch vụ đã cung cấp. Với hệ thống chuẩn được lắp đặt, hoạt động theo nguyên tắc, đảm bảo tính bảo mật thì nhiệm vụ quản trị của IT sẽ được duy trì. Mặt bất lợi của coud này là mô hình triển khai của nó cần sự đầu tư nhiều về chuyên môn, tiền bạc và thời gian để tạo ra các giải pháp kinh doanh đúng đắn cho doanh nghiệp.



*Hình 1.12. Mô hình đám mây riêng*

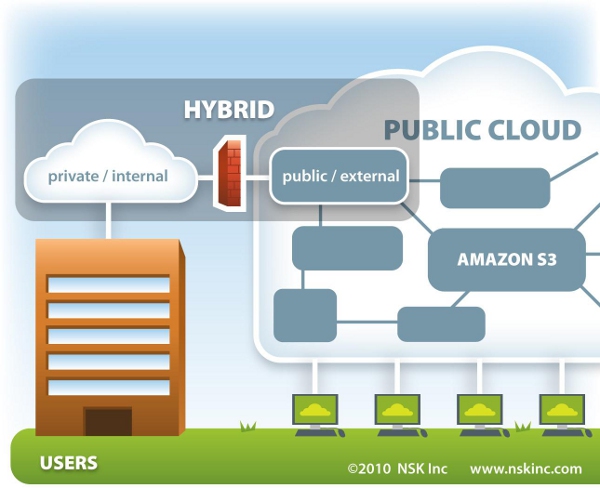
Private Cloud hoàn toàn thuộc về nhu cầu của một cá nhân doanh nghiệp nào đó. Nó có thể là trong hạ tầng cơ sở (on-premises) hoặc ngoài hạ tầng cơ sở (off-premises). Private Cloud chỉ thuộc về một doanh nghiệp sẽ thường trú trong phòng máy của chủ sở hữu hoặc data center và được quản lí bởi đội ngũ IT của doanh nghiệp. Với quyền sở hữu duy nhất của đám mây trong hạ tầng cơ sở, doanh nghiệp được toàn

quyền điều khiển data center, hệ thống hạ tầng và network. Về phía Private Cloud ngoài hạ tầng cơ sở, nó thừa hưởng cơ sở vật chất có sẵn và kiến thức chuyên môn từ các đơn vị outsourcing như là chức năng trung tâm dữ liệu máy chủ. Private Cloud ngoài cơ sở hạ tầng này khá lí tưởng cho các doanh nghiệp không muốn hoặc không có đủ khả năng xây dựng phòng máy hoặc trung tâm dữ liệu riêng.

Private Cloud đã thay đổi vai trò của những quản trị IT. Nếu không có Private Cloud, họ sẽ phải vướng bận rất nhiều ứng dụng được triển khai, bao gồm máy chủ ảo hoặc máy chủ vật lí, cấu hình mạng, network balancing, storage và cài đặt ứng dụng như SQL Server… Với Private Cloud, chức năng của họ trở thành việc quản lí tài nguyên chia sẻ tập trung và quản lí các cấp độ dịch vụ của hạ tầng. Quản trị viên IT tạo ra và quản lí các thành phần và hệ thống có thể tái sử dụng để nâng cấp và cho phép doanh nghiệp tự triển khai dịch vụ của mình. Điều này có nghĩa là họ cung cấp dịch vụ một cách thông minh hơn với số lượng dịch vụ nhiều hơn và có ích hơn cho doanh nghiệp.

* + - 1. ***Hệ thống đám mây lai (Hybrid Cloud)***

Các đám mây lai là sự kết hợp của các đám mây công cộng và đám mây riêng. Những đám mây này thường do công ty sử dụng tạo ra còn trách nhiệm quản lý sẽ được phân chia giữa công ty và nhà cung cấp đám mây công cộng. Đám mây lai sử dụng các dịch vụ có trong cả không gian công cộng và không gian riêng.



*Hình 1.13. Mô hình đám mây lai*

Các đám mây lai là câu trả lời khi công ty cần chế độ bảo mật, an toàn thông tin mà có thể tiết kiệm chi phí cũng như giảm trách nhiệm quản lý. Theo hướng này, công ty có thể phác thảo các mục tiêu và nhu cầu sử dụng các dịch vụ nào từ đám mây công cộng và cái nào đến từ đám mây riêng để có thể phục vụ một cách tốt nhất hoạt động của mình. Đám mây riêng cung cấp các dịch vụ cho các quy trình nhiệm vụ quan trọng tới hạn đòi hỏi an toàn, như nhận các khoản thanh toán của khách hàng, công nợ với nhà cung cấp. Còn đám mây công cộng phục vụ những công việc đơn giản hơn.

Nhớ lại về một công ty bán lẻ cần nhanh chóng mở rộng và cắt giảm dịch vụ điều khiển thông tin online theo nhu cầu thời vụ. Công ty này sử dụng Private Cloud để lưu trữ thông tin nhạy cảm của khách hàng. Private Cloud có thể hợp nhất với Public Cloud như là Windows Azure. Azure cung cấp các datacenter khổng lồ; nhà quản lí ứng dụng có thể mở rộng dung lượng đến cao nhất cho mùa kinh doanh và cắt giảm khi nhu cầu giảm. Công ty hưởng lợi từ hai thứ: quản lí được tính bảo mật và kết hợp được Private Cloud, rẻ-tiện-linh động và có khả năng mở rộng với Public Cloud, một dịch vụ đơn lẻ nhưng bao gồm cả hai loại đám mây.

Hạn chế chính của đám mây này là sự khó khăn trong việc xây dựng và quản lý có hiệu quả. Để có thể nhận và cung cấp các dịch vụ từ các nguồn khác nhau như thể chúng có chung nguồn gốc và sự tương tác giữa các thành phần riêng và chung khiến cho việc thực hiện trở nên phức tạp.

Đây là một trong những khái niệm kiến trúc tương đối mới trong điện toán đám mây nên việc áp dụng thực tiễn và các công cụ sử dụng luôn đặt ra những câu hỏi mới và việc chấp nhận mô hình này còn lúng túng cho đến khi hiểu rõ hơn.

* 1. **ĐÁNH GIÁ ĐIỆN TOÁN ĐÁM MÂY**
     1. **Ưu điểm**
* Tốc độ xử lý nhanh, cung cấp cho người dùng những dịch vụ nhanh chóng và giá thành rẻ dựa trên nền tảng cơ sở hạ tầng tập trung (đám mây).
* Chi phí đầu tư ban đầu về cơ sở hạ tầng, máy móc và nguồn nhân lực của người sử dụng điện toán đám mây được giảm đến mức thấp nhất.
* Chia sẻ tài nguyên và chi phí trên một địa bàn rộng lớn, mang lại các lợi ích cho người dùng.
* Không còn phụ thuộc vào thiết bị và vị trí địa lý, cho phép người dùng truy cập và sử dụng hệ thống thông qua trình duyệt web ở bất kỳ đâu và trên bất kỳ thiết bị nào mà họ sử dụng (chẳng hạn là PC hoặc là điện thoại di động…).
* Với độ tin cậy cao, không chỉ dành cho người dùng phổ thông, điện toán đám mây còn phù hợp với các yêu cầu cao và liên tục của các doanh nghiệp kinh doanh và các nghiên cứu khoa học.
* Khả năng mở rộng được, giúp cải thiện chất lượng các dịch vụ được cung cấp trên đám mây.
* Khả năng bảo mật được cải thiện do sự tập trung về dữ liệu.
* Các ứng dụng của điện toán đám mây dễ dàng để sửa chữa và cải thiện về tính năng bởi lẽ chúng không được cài đặt cố định trên một máy tính nào.
* Những ưu điểm và thế mạnh trên đây đã góp phần giúp "điện toán đám mây" trở thành mô hình điện toán được áp dụng rộng rãi trên toàn thế giới.

Hiện nay với những ưu điểm trên điện toán đám mây đang là xu hướng công nghệ chủ yếu hiện nay đang được nhiều nhà công nghệ thông tin đầu tư và đã được nhiều doanh nghiệp tổ chức và người dùng cá nhân sử dụng.

* + 1. **Nhược điểm**

Tuy nhiên, mô hình điện toán đám mây này vẫn còn gặp nhiều khó khăn trong thực tiễn và mắc phải một số nhược điểm sau:

* **Tính riêng tư**: “Các thông tin người dùng và dữ liệu được chứa trên điện toán đám mây có đảm bảo được riêng tư, và liệu các thông tin đó có bị sử dụng vì một mục đích nào khác hay không?” là câu hỏi rất được chú ý.
* **Tính sẵn sàng**: Liệu các dịch vụ đám mây có bị treo bất ngờ, khiến cho người dùng không thể truy cập các dịch vụ và dữ liệu của mình trong những khoảng thời gian nào đó khiến ảnh hưởng đến công việc.
* **Thất thoát dữ liệu**: Một vài dịch vụ lưu trữ dữ liệu trực tuyến trên đám mây bất ngờ ngừng hoạt động hoặc không tiếp tục cung cấp dịch vụ, khiến cho người dùng phải sao lưu dữ liệu của họ từ đám mây về máy tính cá nhân. Điều này sẽ mất nhiều thời gian. Thậm chí một vài trường hợp, vì một lý do nào đó, dữ liệu người dùng bị mất và không thể phục hồi được.
* **Tính di động của dữ liệu và quyền sở hữu**: Một câu hỏi đặt ra, liệu người dùng có thể chia sẻ dữ liệu từ dịch vụ đám mây này sang dịch vụ của đám mây khác? Hoặc trong trường hợp không muốn tiếp tục sử dụng dịch vụ cung cấp từ đám mây, liệu người dùng có thể sao lưu toàn bộ dữ liệu của họ từ đám mây? Và làm cách nào để người dùng có thể chắc chắn rằng các dịch vụ đám mây sẽ không hủy toàn bộ dữ liệu của họ trong trường hợp dịch vụ ngừng hoạt động.
* **Khả năng bảo mật**: Vấn đề tập trung dữ liệu trên các đám mây là cách thức hiệu quả để tăng cường bảo mật, nhưng mặt khác cũng lại chính là mối lo của người sử dụng dịch vụ của điện toán đám mây. Bởi lẽ một khi các đám mây bị tấn công hoặc đột nhập, toàn bộ dữ liệu sẽ bị chiếm dụng. Đây không chỉ là vấn đề của riêng hệ thống điện toán đám mây, bởi lẽ tấn công đánh cắp dữ liệu là vấn đề gặp phải trên bất kỳ môi trường nào, ngay cả trên các máy tính cá nhân.
* **Quá tải**: Một vài dịch vụ lớn của điện toán đám mây đôi khi rơi vào trạng thái quá tải, khiến hoạt động bị ngưng trệ. Khi rơi vào trạng thái này, người dùng không có khả năng để xử lý các sự cố mà phải nhờ vào các chuyên gia từ đám mây tiến hành xử lý.

Nhìn chung thì với mô hình điện toán đám mây nhược điểm lớn nhất chính là vấn đề về bảo mật và tính riêng tư của người dùng, bên cạnh đó tình trạng mất dữ liệu vấn có thể xảy ra khi lỗi hệ thống, việc bị các tổ chức hacker tấn công quấy rối gây tổn thất cho người dùng cũng nhưng nhà cung cấp cũng không thể tránh khỏi.

* 1. **LỰA CHỌN GIẢI PHÁP CHO MÔ HÌNH ĐIỆN TOÁN ĐÁM MÂY**
     1. **Lợi ích của việc triển khai điện toán đám mây bằng nguồn mở**

Theo Canonical, tập đoàn đã có những đóng góp to lớn trong các dự án nguồn mở nổi tiếng như Ubuntu platform, OpenStack, Ubuntu One, các đặc trưng nổi bật của việc triển khai Cloud computing bằng nguồn mở có thể kể đến như sau:

Sự phụ thuộc vào các phần mềm đóng kín và bản quyền (Avoiding vendor lock-in): các giải pháp thương mại thường là 1 bộ giải pháp với các tiêu chuẩn của nhà sản xuất chẳng hạn các APIs đặc trưng, các kiểu định dạng image và lưu trữ riêng, … sẽ làm cho “cloud” không tương thích, hoặc không tận dụng được những cơ sở hạ tầng sẵn có.

Sự hỗ trợ của cộng đồng: các dự án về “open source cloud computing” luôn luôn được hỗ trợ và giúp đỡ bởi cộng đồng toàn thế giới với hàng ngàn người tham gia phát triển các functions mới và sửa lỗi bugs (fix bugs). Lợi thế này của open source sẽ không thể có được ở bất kỳ một công ty đơn lẻ nào.

Khả năng mở rộng không hạn chế: chi phí là vấn đề nổi trội trong vấn đề mở rộng mạng “cloud” với giải pháp phần mềm bản quyền. Các chi phí cho licence được tính dựa trên số lượng các máy chủ hay thậm chí các máy ảo tùy thuộc vào từng cách tính khác nhau. Tuy nhiên với open source clouds, ví dụ mạng clouds sử dụng Ubuntu, hệ điều hành Ubuntu hỗ trợ “cloud computing” hoàn toàn miễn phí nên việc mở rộng rất dễ dàng.

Phát triển các modules: khi giải pháp thương mại thiếu một chức năng gì đó, sẽ rất khó để tìm ra phương thức thay thế trừ khi chờ một phiên bản mới hơn hỗ trợ. Nhưng với kỹ thuật open source có thể thay đổi code để thêm các chức năng phù hợp cho mục đích kinh doanh của hệ thống.

* + 1. **Các giải pháp mã nguồn mở cho mô hình điện toán đám mây**
       1. ***CloudStack***

CloudStack là một nền tảng phần mềm mã nguồn mở, nó tập hợp các tài nguyên máy tính để xây dựng hạ tầng mạng như một dịch vụ (Infrastructure as a Service). CloudStack quản lý network, storage, và compute của hạ tầng cloud. Sử dụng CloudStack để triển khai, quản lý, cấu hình môi trường điện toán đám mây một cách nhanh chóng và hoàn chỉnh nhất.

Có thể sử dụng CloudStack cho nhà cung cấp dịch vụ và doanh nghiệp lớn. Nhà cung cấp dịch vụ có thể bán các thực thể máy ảo theo yêu cầu của người dùng về network, storage, và compute.

CloudStack có ba cách thức quản lý: Giao diện web thân thiện, Command line và RESTful API đầy đủ.

* + - 1. ***Eucalyptus***

Eucalyptus là một phần mềm nguồn mở Linux-based để triển khai điện toán đám mây loại hình đám mây riêng và cả đám mây lai. Eucalyptus cung cấp IaaS thuận tiện cho việc chỉ định tài nguyên (phần cứng, dung lượng lưu trữ, và hạ tầng mạng) dựa trên yêu cầu sử dụng.

Điểm mạnh của Eucalyptus là triển khai Enterprise data centers mà không cần quá nhiều yêu cầu về cấu hình phần cứng. Hơn nữa, Eucalyptus hỗ trợ kết nối với dịch vụ đám mây nổi tiếng của Amazon – AWS (Amazone Web Services) thông qua một giao diện lập trình chung.

Kiến trúc của Eucalyptus đơn giản, linh hoạt (flexible), được module hoá (modular) và đạt được nhiều ưu điểm như chức năng snapshot, self-service.

* + - 1. ***OpenNebula***

OpenNebula là bộ công cụ nguồn mở sử dụng cho đám mây riêng, đám mây công cộng và đám mây lai.

OpenNebula hoạt động tương thích với các giải pháp của Xen, KVM, VMWare, và mới đây là Virtual Box.

* + - 1. ***OpenStack***

OpenStack là một dự án mở cộng đồng cho việc phát triển điện toán đám mây phù hợp với các nhà cung cấp (Cloud Providers) cũng như người dùng (Cloud Customers) được phát triển bởi Rackspace Hosting và Nasa.

OpenStack bao gồm ba dự án chính:

* **OpenStack Compute**:Triển khai việc quản lý và chỉ định tài nguyên cho các máy ảo.
* **OpenStack Object Storage**: Thực thi việc lưu trữ, backup.
* **OpenStack Image Service**: Đảm nhận việc phát hiện, đăng ký, truyền tải dịch vụ cho các images disk ảo.

OpenStack đang được đánh giá là phần mềm nguồn mở xây dựng điện toán đám mây mạnh nhất hiện nay với sự hỗ trợ của các hãng máy tính lớn trên thế giới như HP, Canonical, IBM, Cisco, Microsoft, …

* + 1. **So sánh các giải pháp**
       1. ***Một số thông tin chung và đặc điểm chính***
* ***Thông tin tổng quát***

*Bảng 1.3. Thông tin tổng quát về các giải pháp đám mây mã nguồn mở*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Software | Ngày phát hành | Giấy phép | Nền ngôn ngữ | Như một dịch vụ | Các cài đặt cục bộ |
| CloudStack | 2010-05-04 | [Apache license](http://en.wikipedia.org/wiki/Apache_license) | Java, C | Có | Có |
| Eucalyptus | 2008-05-29 | [Proprietary](http://en.wikipedia.org/wiki/Proprietary_software), [GPL v3](http://en.wikipedia.org/wiki/GPL_v3) | Java, C | Có | Có |
| OpenNebula | 2008-03-01 | [Apache License](http://en.wikipedia.org/wiki/Apache_License) | C++, C, Ruby, Java, Shell script, lex, yacc | Có | Có |
| OpenStack | 2010-10-21 | [Apache License](http://en.wikipedia.org/wiki/Apache_License) | Python | Có | Có |

* ***Tính chất nổi bật***

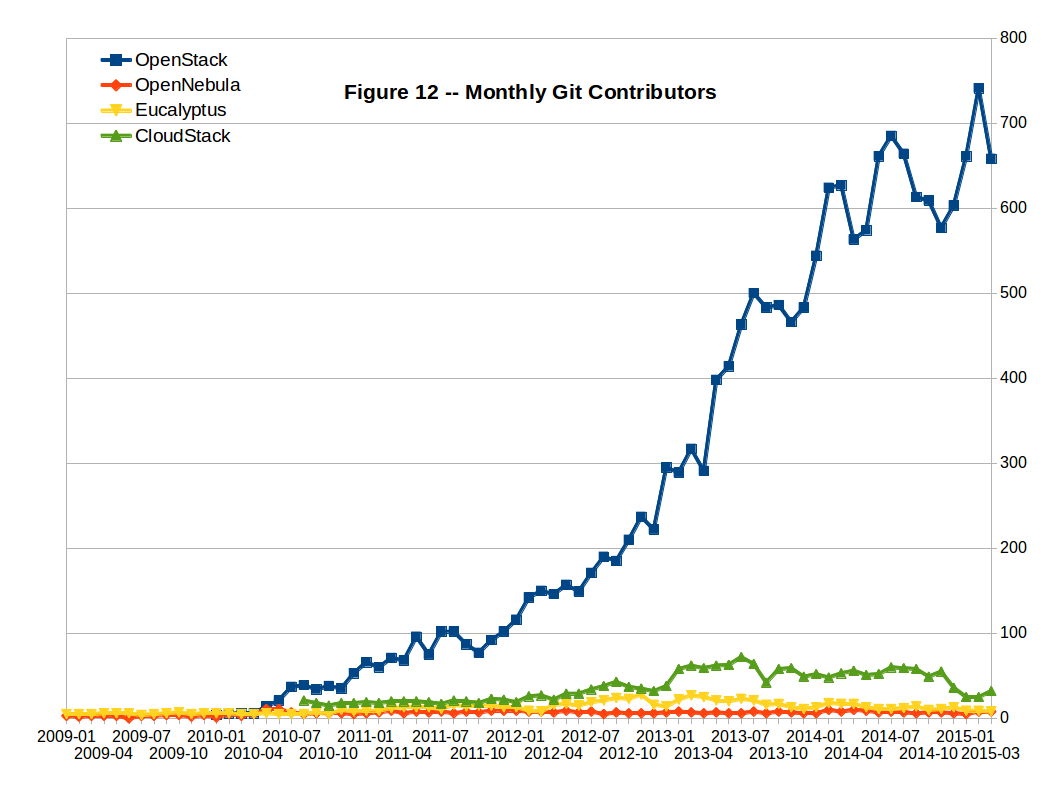
*Bảng 1.4. Tính chất nổi bật các giải pháp đám mây mã nguồn mở*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software | Failover | OCCI | vCloud | S3 |
| CloudStack | 1 phần | ? | ? | ? |
| [Eucalyptus](http://en.wikipedia.org/wiki/Eucalyptus_(computing)) | Không | ? | ? | Có |
| [OpenNebula](http://en.wikipedia.org/wiki/OpenNebula) | 1 phần | Có | Có | Không |
| [OpenStack](http://en.wikipedia.org/wiki/OpenStack) | Không | Có | Có | Có |

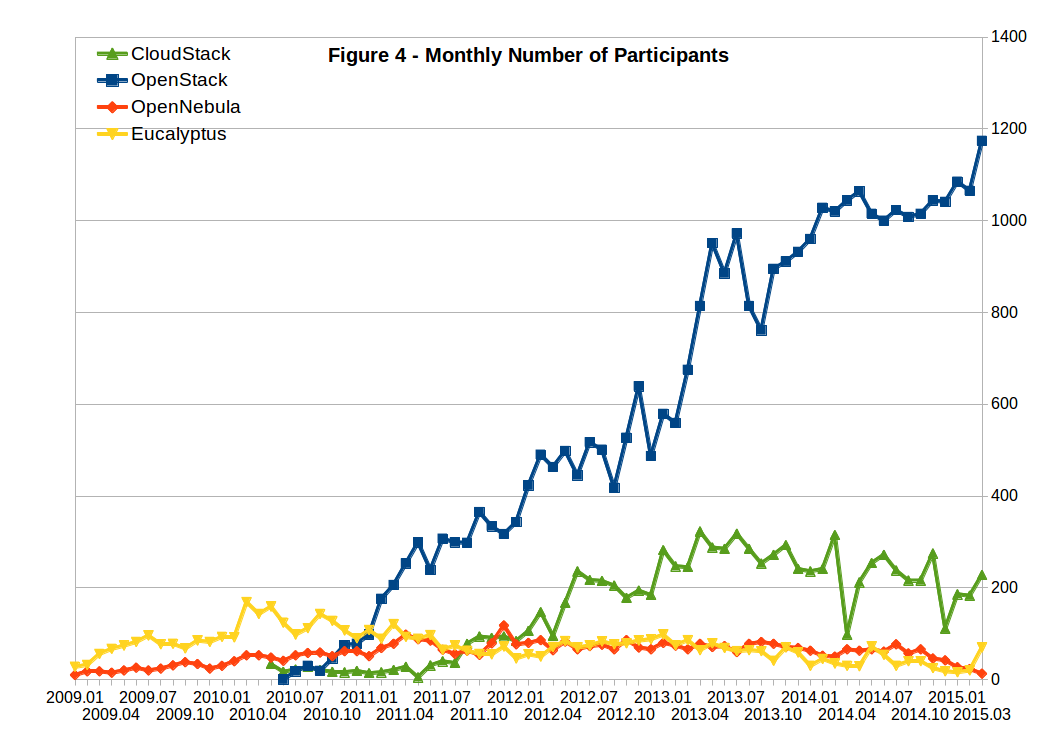
* Failover – Hỗ trợ điều chỉnh tự động các lỗi phần cứng
* OCCI – Hồ trợ Open Cloud Computing Interface
* vCloud – Hồ trợ di chuyển vCloud
* S3 – Hỗ trợ Amazon S3 volumes
  + - 1. ***Sự phát triển trong cộng đồng***

Nhắc đến tính cộng đồng thì dưới đây là so sánh tổng hợp về sự phát triển trong cộng đồng của các giải pháp đám mây trên cơ sở thống kê từ diễn đàn chính thức của mỗi giải pháp:

* CloudStack: http://cloudstack.org/forum/
* Eucalyptus: http://lists.eucalyptus.com/
* OpenNebula: http://lists.opennebula.org/
* OpenStack: <http://lists.openstack.org/>



*Hình 1.14. Số lượng chủ đề trao đổi hàng tháng*



*Hình 1.15. Số lượng người tích cực tham gia trao đổi hàng tháng*

Dựa trên những phân tích của hình 1.14 và 1.15, về các tính năng và đặc điểm cơ bản thì các hệ thống được so sánh ở trên đều là những hệ thống mạnh mẽ trong việc phát triển điện toán đám mây. Tuy nhiên có thể dễ dàng nhận thấy sự ưu ái vượt trội của cộng đồng mã nguồn mở dành cho OpenStack, số lượng người quan tâm nghiên cứu và phát triển về OpenStack ngày càng tăng và vượt xa các dự án còn lại.

Theo nguồn tin chính thức từ <https://www.openstack.org/> thì tính tới nay, tổng cộng thành viên đã tham gia vào dự án (các kỹ sư, nhà phát triển, người nghiên cứu, các đối tác và các chuyên gia) lên tới 38930 người, 586 công ty hỗ trợ, và 178 quốc gia.

Bên cạnh đó là sự hỗ trợ của các nhà cung cấp lớn mạnh đứng đằng sau như NASA, Rackspace, Intel, Microsoft, Citrixx, Cisco, cũng rất nhiều các tên tuổi hàng đầu trong lĩnh vực công nghệ thông tin.

Là một sản phẩm xuất hiện sau các dự án khác, OpenStack rút kinh nghiệm từ các thế hệ đàn anh, và có hướng phát triển tốt hơn, là một xu thế mới trong việc nghiên cứu và phát triển đám mây mã nguồn mở trên toàn thế giới.

# CHƯƠNG 2: KIẾN TRÚC OPENSTACK

1. 1. **TỔNG QUAN VỀ OPENSTACK**
      1. **Amazon Web Service - Nguồn cảm hứng cho sự ra đời của Openstack**

Phần này sẽ giới thiệu sơ lược về một trong những nhà cung cấp dịch vụ về CC hàng đầu hiện nay – Amazon. Amazon đã xây dựng được một hệ thống dịch vụ AWS cơ bản khá hoàn chỉnh và ổn định về IaaS và các dịch vụ đi kèm. Tiếp nữa AWS chính là nguồn cảm hứng để tạo ra những nền tảng về IaaS như Eucalyptus, Openstack ... sau này. Tại sao lại như vậy? Chúng ta sẽ lướt qua một số mốc thời gian, trở lại khoảng 10 năm trước tại thời điểm mà hầu như chưa có mấy công ty có khái niệm về CC, tuy nhiên đã có một số người có ý tưởng về việc cung cấp phần mềm, hạ tầng ... như là một dịch vụ.

Nhắc đến CC chúng ta thường nghĩ ngay đến những tên tuổi như Google, Microsoft, Apple... Tuy nhiên thực tế, họ không phải là những người đi đầu trong công nghệ cũng như ứng dụng về Cloud computing. Thực sự về tầm nhìn sớm và mức độ ứng dụng về CC thì phải nói đến Salesforce và tiếp đó là Amazon.

Saleforce đã bắt đầu từ rất sớm với CC, ngay từ năm 1999 hãng đã có định hướng phát triển về SaaS, từ việc cung cấp các dịch vụ quản lý khách hàng, kế toán, thống kê tài chính... Theo như báo cáo kinh doanh năm 2011, mảng dịch vụ về SaaS đã đem lại cho Saleforce hơn 3 tỉ USD đó là một con số đáng ngưỡng mộ. Ngay cả Google hay Microsoft những tên tuổi 'non trẻ' trong cùng mảng kinh doanh về CC cũng phải ghen tị với thành tích này.

Không dừng lại ở mức độ cung cấp về SaaS như Saleforce, Amazon từ một công ty bán lẻ các mặt hàng dân dụng, điện tử, sách...đã dần vươn lên và có thể nói là tên tuổi lớn nhất hiện nay về dịch vụ hạ tầng cho CC. Cách đây hơn 10 năm, sau khi tồn tại qua đợt khủng hoảng bong bóng dot com, Amazon đã dần chứng minh phương châm bán hàng qua mạng của họ là đúng đắn. Là công ty có tốc độ phát triển nhanh nhất sau 5 năm đầu tiên (từ năm 1995-2000 doanh thu là 2.8 tỉ USD) vượt xa Google (1998-2003 doanh thu 1.5 tỉ USD). Ban đầu tưởng chừng đối thủ cạnh tranh của Amazon chỉ là Wallmart hay BestBuy, eBay - những công ty bán lẻ. Giờ đây Amazon đã lấn sân và kinh doanh trong 16 lĩnh vực khác nhau trong đó mạnh nhất vẫn là lĩnh vực bán lẻ tiếp đến là các dịch vụ về CC.



*Hình 2.1. Management console AWS*

Amazon thực sự đã xây dựng được một đế chế công nghệ hùng mạnh, cạnh tranh trực tiếp với các nhà cung cấp dịch vụ hosting truyền thống cũng như CC như Rackspace, GoDaddy, Google... Theo nhận định của giới chuyên môn Amazon đã tạo ra một kiến trúc về CC kinh điển AWS với đầy đủ các dịch vụ về tính toán, lưu trữ, cơ sở dữ liệu chuyên dụng... Thực tế cho thấy hầu hết các nền tảng khác như Eucalyptus, Openstack... đều được xây dựng theo một kiến trúc, các thành phần tựa như AWS. Tất nhiên chưa có một khẳng định nền tảng của ai tốt hơn một cách rõ ràng, nhưng với những đánh giá về tính ổn định, hiệu năng và quan trọng nhất là giá của dịch vụ. AWS vẫn đang là sản phẩm tốt nhất hiện nay.

AWS vẫn đang liên tục nghiên cứu cải thiện và bổ sung những tính năng mới cho tập các dịch vụ của họ. Do khuôn khổ của việc nghiên cứu thử nghiệm Openstack nên nhóm sẽ chỉ đưa ra một số giới thiệu cơ bản và nhận xét về các dịch vụ chính của Amazon. Từ đây sẽ có một cái nhìn trực quan hơn về Openstack và có một so sánh với 'đối thủ' lớn nhất của nó. Sau đây là một số mốc thời gian quan trọng của AWS:

Những dịch vụ chính của AWS phải kể đến là:

* Amazon Elastic Cloud Compute (EC2) cung cấp các instance (máy ảo) tùy theo nhu cầu, với khả năng tính toán, mở rộng vô cùng linh hoạt. Hiểu đơn giản, EC2 cung cấp cho người dùng khả năng tạo các máy ảo trên hạ tầng của Amazon, họ có thể cấp phát tài nguyên (CPU, RAM) theo yêu cầu, và từ đó Amazon sẽ tính toán các chi phí. Các instance có các mức cấu hình khác nhau: nhỏ nhất là mirco instance (1 CPU, 613 MB RAM) và lớn nhất tới hơn 64GB RAM và 88 EC2 CPU (tương đương 2 x Intel Xeon E5-2670).
* Amazon Elastic Block (EBS) cung cấp khả năng lưu trữ độc lập, kết hợp với EC2. Hiểu đơn giản giống như việc sử dụng thêm các ổ đĩa mở rộng trên các máy vật lý. Khi mà có sự cố tại instance thì dữ liệu lưu trên EBS vẫn có thể sử dụng độc lập, và có thể chia sẻ giữa những instace khác nhau.
* Amazon Simple Storage Service (S3) cung cấp khả năng lưu trữ không hạn chế, cũng giống như EBS, S3 giải quyết vấn đề về lưu trữ, tuy nhiên EBS được sử dụng bởi các instance thì S3 được sử dụng như một ổ đĩa mạng. Thông qua một giao diện (web hay một GUI) người dùng có thể lưu trữ dữ liệu của mình, backup dữ liệu từ các nguồn khác nhau (từ chính EBS, EC2...) S3 sử dụng cơ sở dữ liệu Dynamo để quản lý việc lưu trữ, chứ không sử dụng các CSDL quan hệ truyền thống vì đối với dịch vụ lưu trữ, người dùng chủ yếu đọc và ghi dữ liệu nên nếu lưu theo mô hình quan hệ sẽ không giải quyết hiệu quả.

Vì các thành phần trong AWS hoạt động độc lập với nhau, để chúng có thể kết hợp lại cần có một phần trung gian giúp truyền các thông điệp và đồng bộ thời gian giữa các dịch vụ. Amazon đã phát triển riêng một dịch vụ tên Simple Queue Service - đây chính là thành phần đầu tiên mà Amazon phát triển, và phải mất tới 2 năm (2002-2004) mới cơ bản hoàn thiện. Tuy có vẻ không mấy quan trọng nhưng đây lại chính là một điểm mấu chốt giúp tạo nên sức mạnh của hệ thống các dịch vụ AWS.

Ngoài ra thì AWS đang cung cấp rất nhiều dịch vụ khác nữa như SimpleDB (lưu trữ truy vấn theo kiểu quan hệ truyền thống), Elastic MapReduce Service (áp dụng trong việc tính toán hiệu năng cao, xử lý dữ liệu lớn, thông qua S3 và EC2)...

Tùy theo lưu lượng sử dụng, tài nguyên hệ thống bạn cần...Amazon sẽ tính toán chi phí và yêu cầu bạn thanh toán. Về cơ bản bạn chỉ phải trả cho những gì bạn sử dụng. Khi bạn không cần dùng đến tài nguyên nào đó, bạn có thể 'dừng' nó lại và không phải trả phí trong thời gian đó. Đây chính là một trong những điểm thú vị có thể thấy với CC.

Amazon hiện cho phép người sử dụng thử nghiệm các dịch vụ cơ bản (ở quy mô nhỏ nhất) miễn phí trong một năm đầu tiên. Để đăng ký rất đơn giản, bạn cần khai báo tài khoản ngân hàng của mình, sẽ không mất một khoản phí nào nếu chú ý đọc điều khoản từ Amazon. Ví dụ khi sử dụng EC2 nếu bạn 'lỡ tay' chọn instance không phải loại micro, vậy là bạn đã mất phí rồi đấy.

Người dùng có thể tương tác với AWS thông qua AWS Management Console bằng cách đăng nhập với username và mật khẩu, sau đó với một giao diện Web người dùng có thể sử dụng các chức năng của AWS. Với từng dịch vụ cụ thể như EC2, S3... AWS sẽ cung cấp cho người dùng các chứng chỉ, public/private key để chứng thực với hệ thống, sau đó người dùng có thể tương tác thông qua môi trường dòng lệnh (trong Linux sử dụng gói ec2tools).

AWS hỗ trợ một số ngôn ngữ lập trình cơ bản như Java, PHP, Ruby, .NET, Python... thông qua các API. Các lập trình viên có thể sử dụng những API này để tương tác, lập lịch, tự động khởi tạo mở rộng...với các dịch vụ của AWS. Theo đánh giá từ cộng đồng thì AWS API hoạt động rất tốt trên các nền tàng khác nhau. Ngôn ngữ được AWS khuyến cáo sử dụng là Python, Java.

* + 1. **Lịch sử về Openstack**

Trong phần giới thiệu về AWS ở trên, chúng ta cơ bản nắm được một số chức năng mà một sản phầm thương mại hiện tại đang cung cấp được cho khách hàng, từ đó ta có thể so sánh một cách tương đối giữa những chức năng mà gói công cụ nguồn mở này đã thực hiện được. Để làm rõ thêm lý do lấy AWS làm 'đối chiếu', xin được trích qua một số mốc quan trọng dẫn tới sự ra đời của Openstack.

Trở lại mốc 2005 khi mà Amazon ra mắt thử nghiệm EC2, đó là một thành công lớn gây bất ngờ cho cộng đồng. Với sự ổn định của nó, các công ty khác có thể đơn giản “thuê” C2 trong một vài giờ với một mức năng lực rất rất lớn để thực hiện các công việc tính toán cần tới hiệu năng cao của họ. Ví dụ mà Amazon thường đem ra so sánh là việc hợp tác giữa họ và NASDAQ - sàn chứng khoán cần xử lý một lượng dữ liệu tính toán cực lớn vào cuối tuần, thay vì đầu tư một hệ thống máy chủ phức tạp, họ chỉ thuê EC2 trong vài giờ và chi phí tiết kiệm rất rất nhiều hơn nữa hiệu quả công việc lại tốt hơn.

Một trong những công ty cần sử dụng khả năng tính toán hiệu năng cao kiểu như thế là NASA. Họ có kế hoạch tái cấu trúc lại trung tâm dữ liệu của họ, và họ cần một nền tảng IaaS để có thể sử dụng tốt hơn hạ tầng vật lý mà họ có. Amazon EC2 là một tấm gương tốt đáng ngưỡng mộ. Vào khoảng năm 2008 NASA bắt đầu sử dụng tham gia vào Eucalyptus một dự án nhằm cung cấp một IaaS giống như AWS (EC2 và S3). Tuy nhiên không như mong muốn của NASA, Eucalyptus không phải là một dự án mở hoàn toàn, công ty đỡ đầu cho nó không cho phép NASA xem một số thành phần đóng kín của Eucalyptus. Rạn nứt bắt đầu từ đây.

Sau đó NASA bắt đầu nghiên cứu dự án riêng của họ cũng với mục đích xây dựng một hạ tầng như Amazon EC2, và codename của dự án là Nebula. Với sự tác động từ nhiều phía khác nhau, cuối cùng vào năm 2010 NASA quyết định công bố mã nguồn của Nebula và phát triển nó dưới dạng nguồn mở với codename là Nova. Sau đó Rackspace tiếp tục đóng ghóp nền tảng lưu trữ của họ vào dự án với codename Swift. Dự án Openstack được thành lập với cam kết phát triển theo hướng mở. Nó nhanh chóng nhận được sự đồng thuận từ rất nhiều hãng công nghệ khác và cộng đồng. Hiện nay đã có hơn 160 công ty tham gia vào dự án này với hầu hết các tên tuổi lớn như: NASA, Rackspace, Cisco, Citrix, Microsoft, HP, Dell, Canonical...

Như đã nói AWS chính là nguồn cảm hứng tạo nên Openstack ngày nay, AWS là nền tảng đóng của Amazon và Openstack là một nền tảng mở dành cho tất cả các công ty và cộng đồng sử dụng. Mục đích của Openstack là cung cấp cho người dùng khả năng xây dựng một hạ tầng cho cả private cloud và public cloud. Đã có nhiều công ty sử dụng Openstack để xây dựng dịch vụ để phục vụ nhu cầu của chính họ và cho thuê như chính NASA và Rackspace.

* + 1. **Các phiên bản của Openstack**

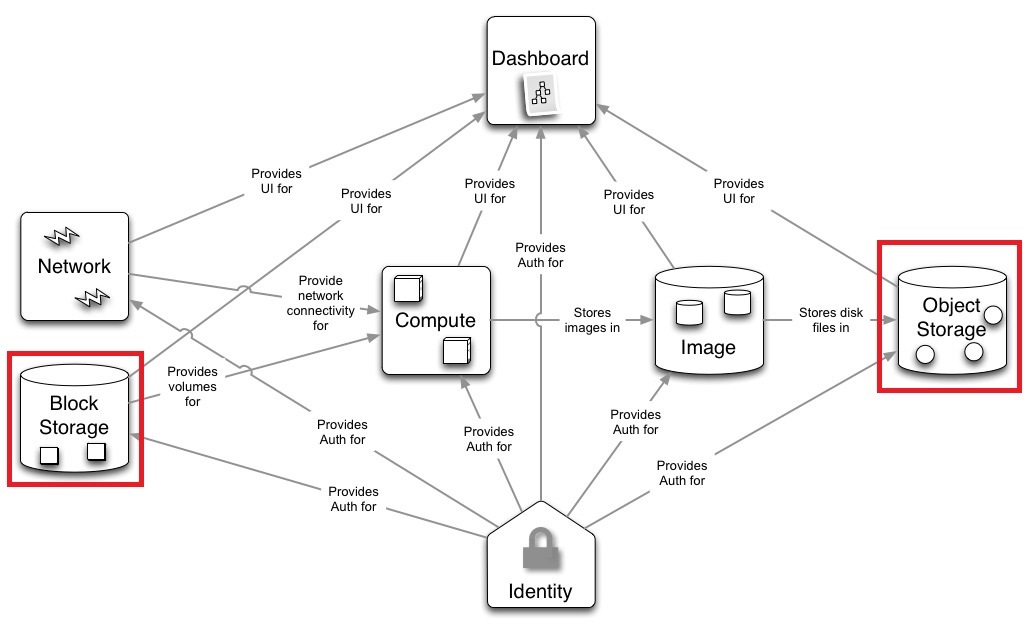
Openstack có chu kỳ phát triển 6 tháng, đi cùng với sự phát triển của CC, với mỗi phiên bản Openstack lại bổ sung thêm thành phần mới tương ứng với những chức năng mới. Openstack hoàn toàn là nguồn mở, các thành phần của nó được viết trên Python - ngôn ngữ đang được đánh giá rất cao những năm gần đây.

*Bảng 2.1. Các phiên bản của OpenStack*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phiên bản | Trạng thái | Mã nguồn | Ngày phát hành | Thành phần |
| Newton | Đang phát triển | 2016.2.0 | 07/10/2016 (dự kiến) |  |
| Mitaka | Phiên bản hiện tại, được hổ trợ | 2016.1.0 | 08/04/2016 | Aodh, Barbican, Ceilometer, Cinder, Cloudkitty, Congress, Designate, Freezer, Glance, Heat, Horizon, Ironic, Keystone, Magnum, Manila, Neutron, Nova, Sahara, Swift, Tacker, Trove, Zaqar |
| Liberty | Đã phát hành, được hỗ trợ | 2015.2.0 | 15/10/2015 | Swift, Neutron, Nova, Celiometer, Keystone, Cinder, Heat, Sahara, Designate, Zaqar, Horizon, DBaaS, Ironic, Barbican, Glance, Manila |
| Kilo | Đã phát hành, được hỗ trợ | 2015.1.0 | 30/04/2015 | Swift, Nova, Glance, Horizon, Keystone, Neutron, Cinder, Celiometer, Heat, Trove, Sahara, Ironic |
| 2015.1.1 | 30/07/2015 |
| 2015.1.2 | 13/10/2015 |
| Juno | Đã phát hành, được hỗ trợ | 2014.2 | 16/10/2014 | Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Ceilometer, Heat, Trove, Sahara |
| 2014.2.1 | 05/12/2014 |
| 2014.2.2 | 05/02/2015 |
| 2014.2.3 | 13/04/2015 |
| Icehouse | Đã phát hành, được hỗ trợ | 2014.1 | 17/04/2014 | Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Ceilometer, Heat, Trove |
| 2014.1.1 | 09/06/2014 |
| 2014.1.2 | 08/08/2014 |
| 2014.1.3 | 02/10/2014 |
| Havana | Không còn hỗ trợ | 2013.2 | 17/10/2013 | Nova, Swift, Glance, Keystone, Horizon, Neutron, Cinder, Ceilometer, Heat |
| 2013.2.1 | 16/12/2013 |
| 2013.2.2 | 13/2/2014 |
| 2013.2.3 | 03/04/2014 |
| 2013.2.4 | 22/09/2014 |
| Grizzly | Không còn hỗ trợ | 2013.1 | 04/04/2013 | Nova, Swift, Glance, Keystone, Horizon, Quantum, Cinder |
| 2013.1.1 | 09/05/2013 |
| 2013.1.2 | 06/06/2013 |
| 2013.1.3 | 08/08/2013 |
| 2013.1.4 | 17/09/2013 |
| Folsom | Không còn hỗ trợ | 2012.2 | 27/09/2012 | Nova, Swift, Glance, Keystone, Horizon, Quantum, Cinder |
| 2012.2.1 | 29/11/2012 |
| 2012.2.2 | 13/12/2012 |
| 2012.2.3 | 31/01/2013 |
| 2012.2.4 | 11/04/2013 |
| Essex | Không còn hỗ trợ | 2012.1 | 05/04/2012 | Nova, Swift, Glance, Keystone, Horizon |
| 2012.1.1 | 22/06/2012 |
| 2012.1.2 | 10/08/2012 |
| 2012.1.3 | 12/10/2012 |
| Diablo | Không còn hỗ trợ | 2011.3 | 22/09/2011 | Nova, Swift, Glance |
| 2011.3.1 | 19/01/2012 |
| Cactus | Không còn hỗ trợ | 2011.2 | 15/08/2011 | Nova, Swift, Glance |
| Bexar | Không còn hỗ trợ | 2011.1 | 03/02/2011 | Nova, Swift, Glance |
| Austin | Không còn hỗ trợ | 2011.1 | 21/11/2010 | Nova, Swift |

* 1. **CÁC THÀNH PHẦN CỦA OPENSTACK**

OpenStack cung cấp giải pháp IaaS thông qua tập hợp các dịch vụ tích hợp với nhau. Mỗi dịch vụ cung cấp một giao diện lập trình ứng dụng (Application Programming Interface – API) cho các thao tác xử lý cũng như tương tác với nhau.



*Hình 2.2. Các thành phần chính của Openstack*

*Bảng 2.2. Các thành phần chính của Openstack*

|  |  |  |
| --- | --- | --- |
| Dịch vụ | Tên mã | Mô tả |
| Dashboard | Horizon | Cung cấp giao diện điều khiển dưới dạng web, cho phép người dùng tương tác với các dịch vụ của OpenStack để khởi tạo một máy ảo, gán địa chỉ IP, thiết lập kiểm soát truy cập… |
| Compute | Nova | Cung cấp việc xây dựng và quản lý các máy ảo |
| Networking | Neutron | Cung cấp kết nối mạng giữa các máy ảo, giữa máy ảo và Internet. |
| Storage | | |
| Object Storage | Swift | Cung cấp dịch vụ lưu trữ file. Không mount thư mục giống như một file server. |
| Block Storage | Cinder | Cung cấp thành phần lưu trữ dự phòng cho các máy ảo, tương tự USB, HDD di động,... |
| Shared services | | |
| Identity | Keystone | Cung cấp dịch vụ chứng thực và ủy quyền cho các dịch vụ khác trong OpenStack. |
| Image | Glance | Cung cấp các file chứa hệ điều hành dành cho các máy ảo. |
| Metering/Monitoring | Ceilometer | Quản lý và thống kê các thông số thay đổi trong quá trình hoạt động. |
| Higher-level services | | |
| Orchestration | Heat | Cung cấp khả năng xây dựng tự động các dịch vụ như web server, firewall,... cho máy ảo; Điều phối các hoạt động cho đám mây. |
| Database Service | Trove | Cung cấp khả năng mở rộng và tin cậy của chức năng Cloud Database as a Service cho cả hai công cụ cơ sở dữ liệu tin tưởng và không tin tưởng. |

* + 1. **OpenStack compute**

Đây là phần cơ bản nhất của OpenStack có chức năng điều khiển IaaS và phân phối lại tài nguyên hệ thống cho các instance (máy ảo) với khả năng tính toán lưu trữ độc lập – nó tương ứ ng với Amazon EC2.

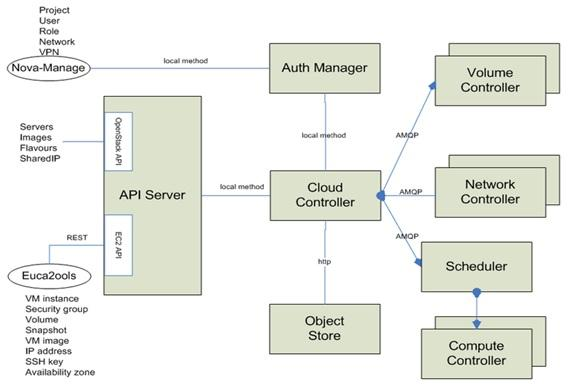
Về cơ bản Nova cung cấp cho người dùng khả năng chạy các instance và giao diện để quản lý các instance đó trên hạ tầng phần cứng. Tuy nhiên Nova không bao gồm bất cứ phần mềm ảo hóa nào. Dưới đây là một số hypervisor chính mà Nova hiện hỗ trợ:

* Hyper-V 2008
* KVM - Kernel-based Virtual Machine
* LXC - Linux Containers (through libvirt)
* QEMU - Quick EMUlator
* UML - User Mode Linux
* VMWare ESX/ESXi 4.1 update 1
* Xen - XenServer 5.5, Xen Cloud Platform (XCP)

**Các tính năng chính của OpenStack Compute:**

* Quản lý tài nguyên ảo hóa bao gồm CPU, memory, disks, network interfaces. Tất cả các tài nguyên được hợp nhất vào trong 1 “bể tài nguyên” -“pool of computing”. Việc này sẽ tăng tính tự động và tận dụng tài nguyên, đem lại lợi ích lớn về kinh tế.
* Quản lý mạng nội bộ (LAN) Flat, Flat DHCP, VLAN DHCP, IPv6. OpenStack được lập trình để chỉ định các địa chỉ IPs và VLAN (Virtual LAN). Chức năng này giúp cho việc cung cấp dịch vụ networking và nâng tính bảo mật khi các VLANs được tách rời nhau. Đồng thời tính linh hoạt trong mô hình mạng cũng phù hợp với mỗi ứng dụng cho mỗi user/group.
* API với nhiều tính năng và xác thực: Được thiết kế tự động và an toàn để quản lý việc users truy cập vào các tài nguyên và ngăn chặn truy cập trái phép qua lại giữa các users.
* Virtual Machine (VM) image management.
* Live VM management (Instance) khởi tạo, khởi động, đóng băng, hay xóa instances. Ngoài ra còn có tính năng lifecycle management.

**Nova có 7 thành phần chính:**



*Hình 2.3. Các thành phần của Nova*

* **Cloud Controller** - quản lý và tương tác với tất cả các thành phần của Nova.
* **API Server** - giống như một Web service đầu cuối của Cloud Controller.
* **Compute Controller** - cung cấp, quản lý tài nguyên từ các instance. Object Store - cung cấp khả năng lưu trữ, thành phần này đi cùng với Compute Controller.
* **Auth Manager** - dịch vụ authentication và authorization.
* **Volume Controller** - lưu trữ theo block-level - giống như Amazon EBS.
* **Network Controller** - tạo quản lý các kết nối trong virtual network để các server có thể tương tác với nhau và với public network.
* **Scheduler** - chọn ra compute controller thích hợp nhất để lưu instance.

Các thành phần của Nova hoạt động độc lập, kết nối với nhau bằng các thông điệp (message-based architecture). Các thành phần Compute Controller, Volume Controller, Network Controller và Object Store có thể cài đặt trên các server vật lý khác nhau. Như trong hình trên có thể thấy Cloud Controller giao tiếp với Object Store thông qua HTTP nhưng giao tiếp với Scheduler thông qua AMQP (Advanced Message Queue Protocol) Để tránh việc tắc nghẽn khi khi đợi các thành phần phản hồi, Nova sử dụng các hàm gọi không đồng bộ (asynchronous), với một call-back được gửi khi mà response đã được nhận.

Do được tạo thành từ nhiều thành phần khác nhau nên có một số chức năng đang được xây dựng lại, một số chức năng “bị lặp”. Điển hình như trong Nova, thành phần Object Store dùng để lưu các image (file ảnh của các hệ điều hành ảo khi chưa được chạy), đồng thời Glance cũng là nơi để lưu trữ các image đó. Tuy nhiên việc này không ảnh hưởng gì nhiều đến hệ thống. Người dùng có thể tùy chọn giữa các lựa chọn này. Theo khuyến cáo thì Glance vẫn được ưu tiên hơn.

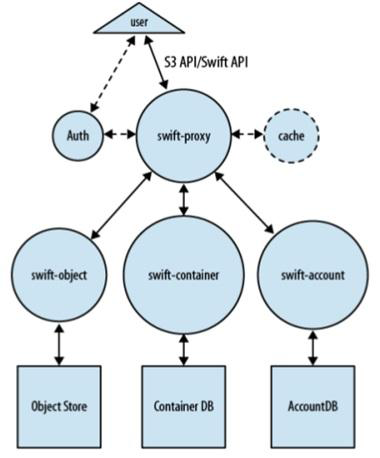
**User và Project:**

Nova được thiết kế để sử dụng cho nhiều đối tượng khác nhau, nó sử dụng các quy tắc phân quyền cơ bản thông qua Role-Based Access Control (RBAC) bao gồm 5 luật:

* **Cloud Aministrator (admin)**: Global role. User với quyền này có toàn quyền với cả hệ thống.
* **IT Security (itsec)**: Global role. Quyền này hạn chế hơn so với admin. Nó cho phép user giữ và cách ly các instance trong bất cứ project nào nếu có vấn đề.
* **Project Manager (projectmanager)**: Project role. Người sở hữu một project nào đó, người có quyền này có thể thêm user vào project, tương tác với các image, chạy và kết thúc (terminate) các instance trong vùng project quản lý.
* **Network Administrator (netadmin)**: Project role. Định vị (allocate) và gán public IP cho instance. Thay đổi các luật của firewall.
* **Developer (developer)**: Project role. Đây là quyền mặc định được gán cho người dùng.
  + 1. **OpenStack Object Storage**

OpenStack Object Storage hay còn gọi là Swift được Rackspace open-source từ năm 2010, nó chính là công nghệ được sử dụng đằng sau Rackspace's Cloud Files một trong những giải pháp lưu trữ thương mại rất tốt hiện nay cạnh tranh với Amazon S3.

Swift là phần mềm nguồn mở để tạo ra các phiên bản giống nhau cho việc lưu trữ dữ liệu, đồng thời với việc mở rộng lưu trữ rất linh hoạt và sử dụng cơ chế clusters, khả năng của swift có thể lưu trữ lên đến petabytes dữ liệu truy cập. Swift không chỉ là một hệ thống data thời gian thực, nó còn là một hệ thống lưu trữ lớn với tính chất “lâu dài – long term” với một lượng dữ liệu cực lớn mà vẫn đảm bảo việc truy xuất, phân cấp, và nâng cấp (retrieved, leveraged, and updated). Các đối tượng lưu trữ (Object Storage) sử dụng kiến trúc phân tán so với mô hình tập trung, nên sẽ không có điểm trung tâm. Việc này giúp cho nâng khả năng mở rộng, backup, và duy trì (scalability, redundancy and permanence). Các đối tượng được ghi lên nhiều thiết bị phần cứng khác nhau mà trong đó, OpenStack đóng vai trò chịu trách nhiệm đảm bảo việc tái tạo, sao nguyên, và toàn vẹn của dữ liệu qua các cluster. Mặt khác, các cụm lưu trữ dữ liệu có thể được mở rộng theo “chiều ngang” dễ dàng qua việc thêm các nodes lưu trữ mới. Nếu 1 node trục trặc, hoạt động của OpenStack ngay lập tức tái tạo lại nội dung của nó từ một nodes đã được active khác. Tất cả các công việc trên được OpenStack thực hiện về mặt logic mà không phụ thuộc vào bất kỳ thiết bị phần cứng nào, việc này đảm bảo chắc chắn hơn trong việc tái tạo, sao chép dữ liệu đồng thời tránh việc phụ thuộc vào thiết bị phần cứng, đặc biệt các thiết bị chuyên dụng giá thành cao.



*Hình 2.4. Kiến trúc Logic của Swift*

Các thành phần chính được miêu tả cụ thể như sau:

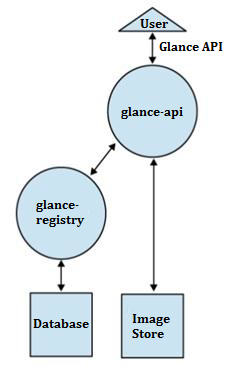
* **Proxy Server** - nhận các request và chứng thực user. Sau khi quá trình chứng thực hoàn tất, dữ liệu sẽ được chuyển trực tiếp từ (hoặc tới) user. Proxy server sẽ không kiểm tra chúng.
* **Object Server** - lưu trữ, quản lý các đối tượng được lưu. Các object sẽ được lưu theo dạng binary cùng với metadata miêu tả về dữ liệu đó.
* **Container Server** - lưu trữ thông tin và trả về danh sách các object đang được lưu bên Object Store.Nó không biết chính xác object được lưu ở đâu nhưng nó biết cụ thể object được lưu tại container nào. Dữ liệu được lưu mặc định trong một CSDL Sqlite, nếu Swift được cài đặt trên các cluster khác nhau thì CSDL này sẽ được tạo thêm các bản sao tương tự.
* **Account Server** - cũng giống như Container Server nhưng nhiệm vụ của nó là quản lý danh sách các Container chứ không phải là object.
* **The Ring** - Thành phần này sẽ tạo một ánh xạ giữa tên của các thực thể được lưu trên đĩa cứng và địa chỉ vật lý của nó. Có nhiều ring khác nhau cho account, container và ọbject. Khi mà các thành phần khác cần sử dụng bất cứ thao tác nào trê object, container hay account thì cần phải tương tác với ring tương ứng để tìm ra đúng địa chỉ lưu trữ trên cluster. Ring được sử dụng bởi proxy server và các tiến trình khác chạy trong background.
  + 1. **OpenStack Image Service**

OpenStack Image Service (còn gọi là Glance) cung cấp các tính năng về discovery, đăng ký (registration), và vận chuyển (delivery) các dịch vụ cho các đĩa images ảo. API của OpenStack Image Service cung cấp một giao diện tiêu chuẩn cho các thông tin truy vấn về các đĩa image ảo lưu trữ trong các back-end, bao gồm luôn cả OpenStack Object Storage. Clients có thể đăng ký một đĩa image ảo với các dịch vụ có sẵn, thực hiện việc truy vấn thông tin.

**Các tính năng hiện tại :**

* Image-as-a-service
* Multi-format/container support
* Image status
* Scalable API
* Metadata
* Image Checksum
* Extensive Logging
* Integrated testing
* Back-end store options
* Version control
* CLI access
* Built-in Mgmt. utilities
* Drive auditing
* VNC Proxy through web browser

**Glance gồm có ba phần:**



*Hình 2.5. Các thành phần của Glance*

* **Glance API server** - nhận các hàm gọi API, tương tự như nova-api, nó chờ các API request sau đó giao tiếp với các thành phần khác (glance-registry và image store) sau đó thực hiện các công việc được yêu cầu: truy vấn, upload, delete image...
* **Glance Registry server** - lưu và cung cấp các thông tin (metadata) về image (định dạng, ID, dung lượng...) Mặc định sử dụng Sqlite để lưu các metadata. Ngoài ra glance-registry luôn nghe cổng 9191.
* **Image Storage** - lưu trữ các file image.
  + 1. **OpenStack Dashboard**

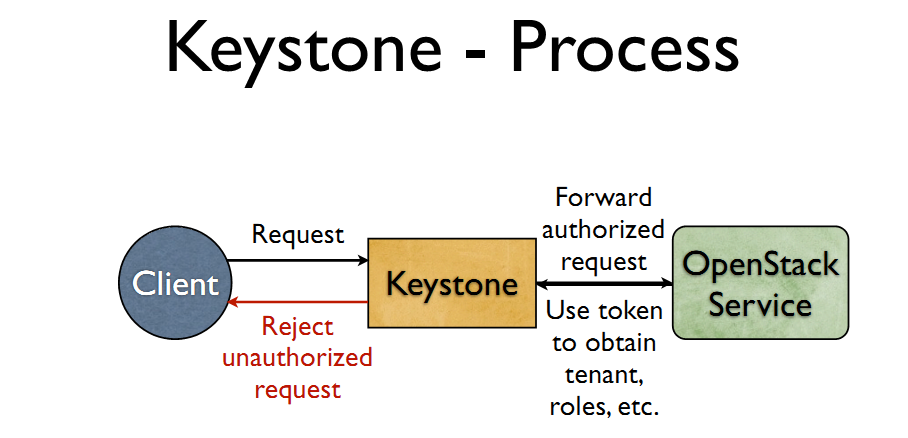
Cung cấp giao diện đồ họa cho người quản trị và người dùng giúp dự liệu và tự động các nguồn tài nguyên trên cloud. Có thiết kế giúp dễ dàng tích hợp dịch vụ và sản phẩm của bên thứ ba. Hỗ trợ VNC client dựa trên nền web, cung cấp khả năng truy cập đến máy ảo qua VNC consoles.

* + 1. **Identity Service**

Keystone cung cấp các chính sách, danh mục, token, và chứng thực cho toàn bộ hạ tầng cloud.

* Keystone xử lý các yêu cầu API để cung cấp cấu hình catalog, policy, token, và dịch vụ nhận dạng.
* Mỗi chức năng Keystone có một thành phần backend bên dưới cho phép có nhiều cách khác nhau sử dụng dịch vụ cụ thể. Hầu hết các backends tiêu chuẩn như LDAP hoặc SQL, hoặc Key Value Stores (KVS).

Thành phần Keystone có thể được triển khai riêng lẻ để cung cấp dịch vụ chứng thực tập trung cho các dịch vụ khác.



*Hình 2.6. Chứng thực Keystone*

Tất cả các thành phần trong OpenStack khi hoạt động đều phải gửi thông tin chứng thực về Keystone để chứng thực. Hoạt động của OpenStack dựa trên các hàm APIs gọi về các services. Khi cần thực hiện yêu cầu, Client sẽ gửi request về cho Keystone kèm theo thông tin chứng thực như: tenant’s ID, os\_name, auth\_tcp…. Sau đó Keystone sẽ chuyển tiếp các thông tin này cho service tương ứng để kiểm tra. Nếu hợp lệ, kết nối sẽ được tạo; còn không, thông tin yêu cầu chứng thực sẽ bị huỷ.

* + 1. **OpenStack Networking**

OpenStack Networking cung cấp một lượng API phong phú để quản lý kết nối và xác định kết nối trong hạ tầng cloud. Nó hỗ trợ nhiều công nghệ mạng khác nhau để kết nối vào hạ tầng cloud. Chúng ta sẽ tìm hiểu về mô hình triển khai phần Network, và một số dịch vụ cơ bản của network cung cấp cho hạ tầng cloud.

Nếu như trong Nova có VMs là tài nguyên trừu tượng, thì trong Neutron có Network, Subnet, Port là các tài nguyên trừu tượng của nó.

* **Network**: quản lý cô lập mạng ở Layer 2, giống như VLAN trong mạng vật lý.
* **Subnet**: nhóm địa chỉa IP v4/v6 được phân cấp và cấu hình cho hệ thống.
* **Port**: một port sẽ được gán với một thiết bị ảo trong hạ tầng.

Bạn có thể tạo ra mô hình mạng riêng của mình, bằng cách tạo ra Network và subnet và sau đó hướng dẫn cho các dịch vụ của khác, như Compute để nó có thể gán thiết bị ảo đến port trên network đó. Đặc biệt là Neutron cho phép tạo ra mạng riêng (Private network) và gán cho các tenant, một tenant có thể có nhiều private network và các private network có thể trùng nhau cho mỗi tenant. Các dịch vụ của Neutron cung cấp bao gồm:

* Tạo ra một topology riêng cho hệ thống của riêng mình. Ví dụ như tạo ra hệ thống webserver nhiều tầng, để backup hoặc cân bằng tải.
* Cung cấp sự linh hoạt trong quản trị hạ tầng mạng, cho phép tùy chỉnh hạ tầng cao.
* Cung cấp API mở rộng, cho phép phát triển và tích hợp Neutron vào nhiều hệ thống khác nhau.

Neutron hỗ trợ nhiều công nghệ mạng khác nhau để tạo ra nhiều chức năng hơn, ngoài việc chia VLAN, Subnet, còn có thể làm switch ảo qua công nghệ của vSwitch, các chức năng khác như Firewall, DHCP, VPN, Load balancing.

* + 1. **OpenStack Metering/Monitoring**

Ceilometer là một dự án được phát triển nhằm mục đích cung cấp một điểm liên lạc duy nhất cho hệ thống thanh toán để có được tất cả các phép đo cần thiết để thiết lập việc thanh toán cho khách hàng, module hoạt động trên tất cả các thành phần chính hiện nay của OpenStack và nó có thể hỗ trợ các thành phần mới của OpenStack trong tương lai.

**Mục đích của Ceilometer:**

* Cung cấp một công cụ hiệu quả để đo dữ liệu, chi phí CPU và mạng.
* Cho phép người phát triển tích hợp hệ thống đo lường một cách trực tiếp hoặc bằng cách thay thế các thành phần.
* Dữ liệu có thể được thu thập bới các thông báo theo dõi (monitoring notifications) được gửi từ các dịch vụ đang hoạt động hoặc bằng cách thăm dò (polling) cơ sở hạ tầng.
* Cho phép người phát triển có thể cấu hình các loại dự liệu thu thập để phục vụ cho mục đích hoạt động của hệ thống một cách phù hợp.

**Kiến trúc hệ thống:**

* Compute Agent chạy trên mỗi node Compute và tiến hành các cuộc khảo sát cho việc thống kê sử dụng tài nguyên.
* Central Agent chạy trên một máy chủ quản lý trung tâm (central management server) để tiến hành thăm dò cho việc thống kê sử dụng nguồn tài nguyên cho những tài nguyên không gắn với instances hoặc node compute.
* Collector chạy trên một hoặc nhiều máy chủ quản lý trung tâm để giám sát các massage queues (để thông báo và để tính toán các dữ liệu đến từ các agent).
* Data Store là một cơ sở dữ liệu có khả năng xử lý đồng thời ghi (từ một hoặc nhiều collector instances) và đọc (từ máy chủ API).
* Máy chủ API - API Server chạy trên một hoặc nhiều máy chủ quản lý trung tâm để cung cấp khả năng truy cập vào dữ liệu từ "data store".
  + 1. **OpenStack Orchestration**

Nhiệm vụ của OpenStack Orchestration là tạo ra một dịch vụ tiếp cận được cả người và máy để quản lý toàn bộ vòng đời của cơ sở hạ tầng cũng như ứng dụng trong đám mây OpenStack.

**Nguyên tắc làm việc:**

* Một mẫu Heat mô tả cơ sở hạ tầng cho một ứng dụng đám mây bằng một tập tin văn bản.
* Tài nguyên cơ sở hạ tầng mà có thể được mô tả bao gồm: máy chủ, nổi IP, khối lượng, nhóm bảo mật, người dùng,…
* Heat cũng cung cấp một dịch vụ tự động điều chỉnh tích hợp với Ceilometer.
* Mẫu Heat này cũng có thể xác định các mối quan hệ giữa các nguồn tài nguyên (ví dụ như một khối lượng công việc này được kết nối với một hệ phục vụ). Điều này cho phép Heat gọi các OpenStack API để tạo ra hệ thống cơ sở hạ tầng theo đúng thứ tự để tạo ra ứng dụng.
* Heat quản lý toàn bộ vòng đời của các ứng dụng – k hi cần thay đổi cơ sở hạ tầng của bạn, chỉ cần sửa đổi thông tin và cập nhật lại.
* Heat chủ yếu là quản lý cơ sở hạ tầng, nhưng các mẫu tích hợp tốt với các công cụ quản lý cấu hình phần mềm chẳng hạn như đầu bếp và múa rối. Nhóm heat làm việc trên cung cấp tích hợp tốt hơn giữa các cơ sở hạ tầng và phần mềm.

**Thành phần chính (Heat API và Heat Engine):**

* Heat-api cung cấp một API là OpenStack-native ReST API đảm nhiệm gửi các yêu cầu tới thành phần heat-engine.
* Heat-api-cfn cung cấp một API là AWS-style Query API có thể tích hợp được với AWS CloudFormation và nó gửi yêu cầu tới heat-engine.
* Heat-engine thực hiện các công việc chính trong orchestrating và gửi trả lời.
  1. **TRIỂN KHAI MẠNG TRÊN OPENSTACK**
     1. **Nova-network**

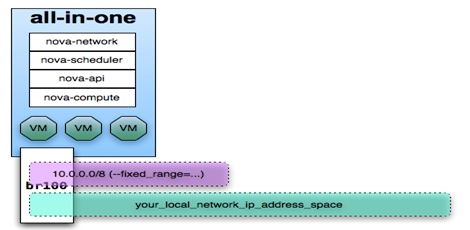
Cho đến phiên bản Folsom thì OpenStack chưa có thành phần chuyên biệt để quản lý network. Thời kì này hệ thống sử dụng một phần của nova để gán địa chỉ IP cho từng máy ảo, cơ chế hoạt động tương tự Vmware network adapter.

Thành phần này tương tác với nova-compute, có nhiệm vụ kết nối giữa các máy ảo với nhau và các máy ảo với public network. Cũng giống như AWS hay Eucalyptus một máy ảo trong Openstack có thể có 2 IP. Một private IP được dùng để kết nối giữa các máy ảo và public IP được dùng để kết nối máy ảo với Internet (public network).

Nova-network có hai cách quản lý khác nhau: Flat Network và VLAN Network.

* + - 1. ***Flat Network***

Flat Network tạo một giao diện bridge dựa trên ethernet adapter để giao tiếp giữa các node. Cấu hình này có thể sử dụng với một adapter duy nhất trên máy vật lý, hoặc có thể dùng nhiều hơn.

****

*Hình 2.7. Cấu hình all-in-one, sử dụng một network adapter*

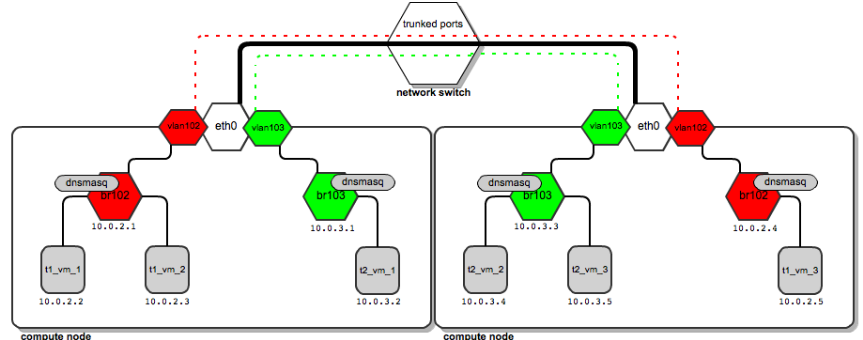
Theo hình trên ta có thể thấy tất cả các thành phần đều được cài đặt trên một máy vật lý duy nhất. Nova-network tạo một pool IP để cấp phát địa chỉ private cho các máy ảo, và một bridge ảo (br100) hoạt động như một gateway để kết nối các máy ảo với mạng ngoài.

Với mô hình nhiều nodes, yêu cầu phải có node controller làm nhiệm vụ nhận yêu cầu và điều khiển việc phân phối địa chỉ IP cho các máy ảo. Lúc này, trên mỗi node compute và node controller đều cấu hình card br100 để đồng bộ hóa việc kết nối ra mạng ngoài của các máy ảo. Khi một máy ảo gửi dữ liệu ra mạng ngoài, nó sẽ gửi đến default gateway (ở đây là controller node, chứa nova-network). Tiếp theo controller node đã được cấu hình như một router sẽ chuyển tiếp dữ liệu này ra mạng ngoài. Một điều cần chú ý là các card mạng của các máy chủ cần phải được cấu hình ở chế độ promiscous để việc chuyển tiếp dữ liệu được chính xác.

Trường hợp sử dụng nhiều network adapter, các node phải được kết nối với nhau thông qua một switch và tất cả các adapter kết nối vào mạng này phải được cấu hình như một bridge ảo.

Ưu điểm của Flat Network là dễ cấu hình. Tuy nhiên nhược điểm là chỉ có một IP pool duy nhất sử dụng chung cho tất cả các máy ảo, nên chỉ có một miền L2 và vùng ARP duy nhất, không thể phân biệt các nhóm máy ảo với nhau.

* + - 1. ***Vlan Network***



*Hình 2.8. VLAN network*

Là cấu hình mặc định của Nova. Nó cho phép admin gán các vùng private network cho mỗi project. Và máy ảo có thể được truy cập thông qua VPN từ ngoài Internet. Trong kiểu cấu hình này, mỗi project sẽ có một VLAN riêng, một Linux networking bridge và subnet. Subnet được chỉ định bởi admin và được gán động cho project khi được yêu cầu. Một DHCP server được chạy quản lý cho mỗi VLAN để gán IP cho mỗi máy ảo trong vùng subnet được gán cho project. Tất cả các máy ảo thuộc cùng project được đặt trong một VLAN riêng.

Nhược điểm của chế độ này là chỉ sử dụng được 4000 vlan tags, tương ứng với 4000 mạng. Các vlan phải được cấu hình trên từng switch để cho phép mạng thông suốt, và có thể gặp vấn đề với trường hợp chồng chéo địa chỉ MAC. Việc sử dụng Nova-network đặt ra hai vấn đề:

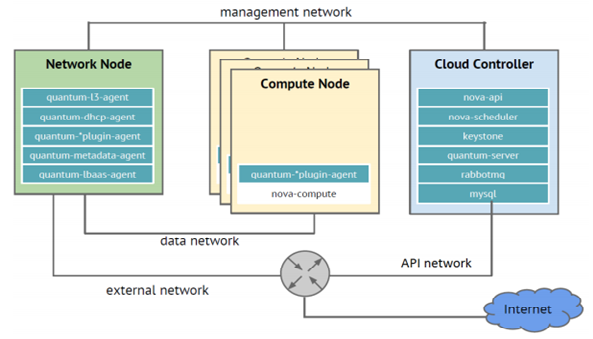
Giới hạn công nghệ: VLAN network là lựa chọn duy nhất khi muốn xây dựng hệ thống có thể phân chia các nhóm máy ảo theo nhu cầu. Cơ chế này chỉ hỗ trợ Linux bridge mà ko có ACLs, QoS và khả năng quản lý.

Network controller node sẽ là điểm chịu lỗi duy nhất, gây nên tình trạng nghẽn lưu lượng mạng và nguy cơ sụp đổ hệ thống.

Giới hạn người dùng cũng như tenant kiểm soát hệ thống mạng: Tenant không thể tạo mô hình mạng theo ý riêng của mình, cũng như sử dụng các dịch vụ mở rộng khác (firewall, IPS..).

* + 1. **Neutron – Thành phần và kiến trúc**

Để giải quyết các vấn đề của Nova-network, đến phiên bản Folsom thì thành phần network đã ra đời, với tên gọi là Quantum, sau đó được chính thức đổi lại là Neutron. Neutron sử dụng một API để thiết lập và cung cấp cho các mạng ảo (vNIC) liên kết với các dịch vụ khác trong OpenStack.

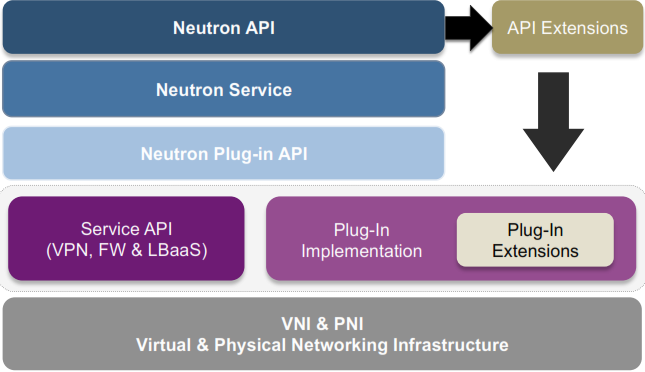


*Hình 2.9. Mô hình triển khai Quantum (Neutron)*

Theo hình trên, Management network sẽ cung cấp các kết nối tới các thành phần trong OpenStack . Các máy ảo sẽ giao tiếp với nhau thông qua Data network.

* + - 1. ***Neutron API***

Neutron là một dịch vụ mạng ảo cung cấp một API mạnh mẽ để xác định kết nối mạng và địa chỉ được sử dụng bởi các thiết bị từ các dịch vụ khác, chẳng hạn như OpenStack Compute. Nó có khả năng mở rộng thông qua các plugin, có thể hỗ trợ việc ảo hóa mạng, phần cứng. Ngoài ra Neutron còn cung cấp thêm các tính năng mới như QoS hay security group, làm nền tảng cho các dịch vụ khác như VPN, cân bằng tải (Load Balancing as a Service), tường lửa (Firewall as a Service)..

****

*Hình 2.10. Neutron API*

OpenStack Compute API có một máy chủ ảo trừu tượng để mô tả tài nguyên máy tính. Tương tự như vậy, Neutron API có mạng ảo, mạng con, và cổng trừu tượng để mô tả tài nguyên mạng.

**Network**: Một phân khúc lớp 2 biệt lập, tương tự như VLAN.

**Subnet**: Một khối địa chỉ Ipv4/IPv6 và cấu hình liên quan.

**Floating IP**: Liên kết tĩnh giữa địa chỉ IP public từ mạng external đến IP private ở mạng internal.

**Router**: Một gateway kết nối các mạng với nhau.

**Port**: Một điểm kết nối được gắn một thiết bị đơn, chẳng hạn từ card NIC của một máy ảo đến một mạng ảo. Nó cũng có thể mô tả các cấu hình mạng liên quan khác như địa chỉ MAC và địa chỉ IP được sử dụng trên cổng đó.

Ta có thể cấu hình hình trạng mạng bằng cách tạo ra và cấu hình mạng, mạng con, sau đó chỉ định các dịch vụ khác như OpenStack Compute để kết nối các thiết bị ảo đến các cổng trên các mạng này. Đặc biệt, Neutron hỗ trợ mỗi tenant có nhiều mạng riêng, và cho phép tenant được lựa chọn địa chỉ IP của mình (ngay cả khi những địa chỉ IP này trùng với những tenant khác).

Neutron cho phép các đám mây sử dụng các tiện ích tiên tiến, chẳng hạn như xây dựng các ứng dụng web nhiều tầng và cho phép các ứng dụng được di chuyển đến các đám mây mà không thay đổi địa chỉ IP. Neutron cung cấp sự linh hoạt cho người quản trị để tùy chỉnh các dịch vụ mạng, cho phép các nhà phát triển mở rộng các API. Theo thời gian, các chức năng mở rộng trở thành một phần của API lõi.

* + - 1. ***Neutron plugin***

Neutron sử dụng công nghệ Agnostic, một framework được phát triển dựa trên cơ chế plug-in. Một plugin có thể sử dụng một loạt các công nghệ để thực hiện các yêu cầu API. Một số plugin mạng có thể sử dụng Linux VLAN và bảng IP, trong khi những plugin khác có thể sử dụng công nghệ tiên tiến hơn, chẳng hạn như L2-in-L3 tunneling hoặc OpenFlow, để cung cấp những tiện ích tương tự.

Các plugin có thể có tính chất khác nhau cho các yêu cầu phần cứng, tính năng, hiệu suất, quy mô, hoặc các công cụ điều hành. Bởi vì Neutron hỗ trợ một số lượng lớn các plugin, người quản trị có thể cân nhắc lựa chọn để quyết định công nghệ thích hợp cho việc triển khai. Một số plugin có thể sử dụng: Big Switch Plug-in (Floodlight REST Proxy), Brocade Plug-in, Cisco, Cloudbase Hyper-V Plug-in, Linux Bridge Plug-in, Mellanox Plug-in, Nicira NVP Plug-in, Open vSwitch Plug-in, Ryu Plug-in..

Open vSwitch (OvS) là một lựa chọn phổ biến cho mô hình triển khai thông thường. Đó là một chương trình ảo hóa switch mã nguồn mở, hỗ trợ OpenFlow, 802.1Q VLAN, LACP và STP, được tích hợp trong các nền tảng ảo hóa như KVM hay

Xen. OVS xây dựng cơ sở kết nối cho các nền tảng ảo hóa mạng khác nhau (ở đây là Neutron), cung cấp tính năng mở rộng, điều khiển linh hoạt và xử lý luồng dữ liệu nhanh chóng.

* + - 1. ***Kiến trúc Neutron***

Neutron là một dịch vụ độc lập, giống như các dịch vụ OpenStack khác như OpenStack Compute, Glance, Keystone hay Dashboard nên việc triển khai của Neutron thường liên quan đến việc triển khai một số thành phần trên nhiều máy chủ tương ứng (các nodes trong mô hình triển khai).

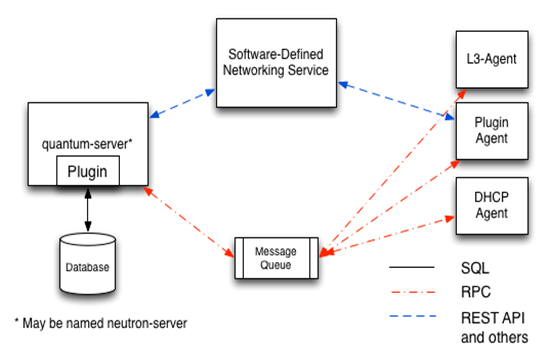
Thành phần chính của Neutron là Neutron-server, đây là một tiến trình nền viết bằng Python cung cấp các hàm API và chuyển tiếp các yêu cầu từ người dùng để cấu hình các plugin nhằm xử lý bổ sung. Thông thường, các plugin sẽ yêu cầu truy cập vào một cơ sở dữ liệu để lưu trữ liên tục (thông thường sử dụng MySQL Server).

Neutron dựa trên dịch vụ Keystone để chứng thực và uỷ quyền tất cả các yêu cầu API.

OpenStack Compute (Nova) tương tác với Neutron thông qua các lời triệu gọi đến các hàm API tiêu chuẩn của nó. Trong quá trình tạo máy ảo, dịch vụ nova-compute giao tiếp với hàm API của Neutron để liên kết NIC ảo trên các máy này vào một mạng cụ thể.

OpenStack Dashboard (Horizon) tích hợp với các hàm API của Neutron, cho phép quản trị viên và người dùng của các tenant tạo và quản lý dịch vụ mạng thông qua giao diện đồ hoạ.

Nếu yêu cầu triển khai sử dụng một máy chủ controller để chạy tập trung các thành phần của OpenStack Compute, ta có thể triển khai Neutron server trên cùng host này. Tuy nhiên, Neutron hoạt động hoàn toàn độc lập và tốt nhất phải được triển khai trên máy chủ riêng. Neutron cũng bao gồm các agents bổ sung khi được yêu cầu, tuỳ theo mục đích triển khai.



*Hình 2.11. Neutron agents*

Network provider services (SDN server/services): Cung cấp dịch vụ mở rộng cho các tenant (Firewall, VPN..). Các dịch vụ SDN có thể tương tác với Neutron-server, Neutron-plugin và các plugin khác thông qua các REST API hoặc kênh liên lạc khác.

**L3 agent** (neutron-l3-agent): Cung cấp khả năng chuyển tiếp ở layer 3 hay NAT để cung cấp truy cập mạng ngoài cho các máy ảo. Agent này hoạt động giống nhau trên tất cả các plugins.

**Plugin agent** (neutron-\*-agent): Chạy trên mỗi Nova-compute node để quản lý cấu hình vswitch. Agent này để chạy được phải phụ thuộc vào plugin mà hệ thống sử dụng, bởi vì một số plugins không thực sự yêu cầu một agent.

**DHCP agent** (neutron-dhcp-agent): Cung cấp dịch vụ DHCP cho các mạng. Agent này hoạt động giống nhau trên tất cả các plugins.

Các agent trên tương tác với tiến trình Neutron chính thông qua hàm RPC (RabbitMQ hay QpiD) hoặc thông qua hàm API tiêu chuẩn.

* 1. **HIỆU SUẤT MẠNG**

Hiệu suất mạng là một quá trình và định lượng để đo và xác định hiệu suật của mạng nào đó. Nó giúp cho người quản trị dễ dàng trong việc xem xét, đo lường và cải thiện các dịch vụ mạng. Hiệu suất mạng chủ yếu được đo từ người dùng đầu cuối (tức là chất lượng của dịch vụ mạng cung cấp cho người sử dụng).

Trong điện toán đám mây, cụ thể là OpenStack, việc cấu hình và sử dụng công nghệ hypervisor khác nhau, sử dụng công nghệ chuyển mạch mạng khác nhau như VLAN hoặc GRE tuning,… nó đóng vai trò quan trọng trong việc xác định hiệu năng của hệ thống. Trong bài nghiên cứu này, chủ yếu phân tích hiệu suất của các gói tin UDP.

UDP (User Datagram Protocol) là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.

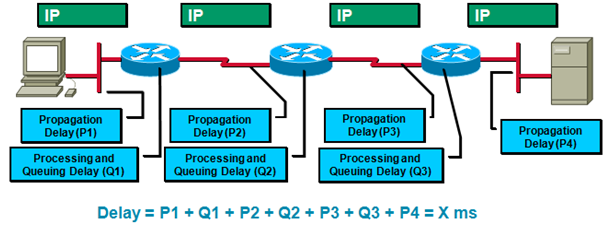
* + 1. **Truyền băng thông**

Bandwidth có nghĩa là "dải tần (dải tần số)", còn gọi là "băng thông", đại diện cho tốc độ truyền dữ liệu của một đường truyền, hay là độ rộng (width) của một dải tần số mà các tín hiệu điện tử chiếm giữ trên một phương tiện truyền dẫn.

Bandwidth đồng nghĩa với số lượng dữ liệu được truyền trên một đơn vị thời gian. Bandwidth cũng đồng nghĩa với độ phức tạp của dữ liệu đối với khả năng của hệ thống. Ví dụ, trong một đơn vị thời gian, download 1 bức ảnh sẽ tốn nhiều bandwidth hơn là download 1 trang văn bản thô (chỉ có text).

Có một sự khác biệt lớn giữa các băng thông thực tế và lý thuyết. Về mặt lý thuyết một mạng có thể hỗ trợ băng thông rất cao tuy nhiên trên thực tế do ảnh hưởng của phần cứng và hệ điều hành làm cho băng thông thấp hơn nhiều.

* + 1. **End-to-end delay**

******

*Hình 2.12. Cách tính End-to-end delay*

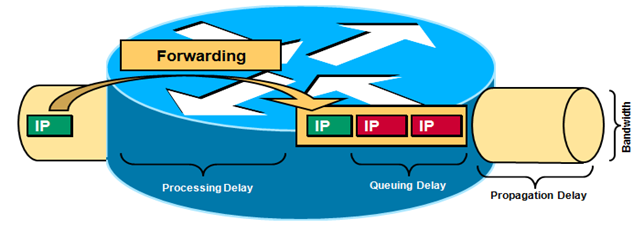
End-to-end delay bằng tổng tất cả propagation, processing và queuing delay có trên đường dẫn.

Propagation (serialization): Sự chậm trễ truyền tải dữ liệu trên phương tiện truyền dẫn, hầu hết xảy ra ở các phần, chỉ phụ thuộc vào băng thông.

Processing, queuing delay: Xảy ra trong Router.

Ping ( ICMP ) có thể sử dụng để đánh giá thời gian đi của gói tin IP trong mạng. Một vài công cụ khác cũng có thể đánh giá thời gian đáp ứng của mạng.

* + 1. **Processing, Queuing và Propagation delay**

******

*Hình 2.13. Processing Delay*

Processing Delay là thời gian để một router đặt một gói tin vào cổng vào và đẩy gói tin đó ra hàng đợi của cổng ra router. Thời gian xử lý phụ thuộc vào nhiều vấn đề như:

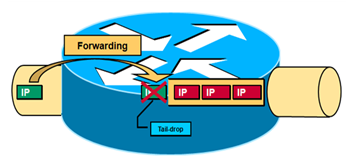
* Tốc độ xử lý CPU.
* Tài nguyên CPU
* Chế độ chuyển đổi IP.
* Cấu trúc router.

Cấu hình các tính năng ở cả hai cổng vào và ra của router.

Queuing Delay là thời gian gói tin nằm trong hàng đợi cổng ra của một router. Nó phụ thuộc vào số lượng và kích thước của các gói tin đã có trong hàng đợi, và băng thông của cổng. Nó cũng phụ thuộc vào cơ chế hàng đợi.

Propagation Delay ( Serialization Delay) là thời gian để vận chuyển gói tin. Nó thường chỉ phụ thuộc vào băng thông của cổng. Công nghệ CSMA/CD có thể làm chậm trễ thêm một chút vì khả năng xung đột các gói tin khi một cổng sắp bị nghẽn.

* + 1. **Mất gói tin**

******

*Hình 2.14. Đánh rớt ở đuôi của hàng đợi*

Mất gói tin thông thường xảy ra khi các router hết bộ nhớ đệm ở hàng đợi cổng ra. Hình trên cho thấy ở hàng đợi cổng ra đã đầy, dẫn tới việc gói tin mới tới sẽ bị đánh rớt. Được tạm gọi đơn giản là “ đánh rớt gói tin cổng ra (output drop)” hoặc “đánh rớt gói tin ở đuôi (tail-drop)” ( tức là gói tin bị đánh rớt ở đuôi của hàng đợi).

Router cũng đánh rớt các gói tin với nhiều lý do khác, ví dụ như:

* Hàng đợi vào – CPU bị tắc nghẽn và không thể xử lý các gói tin ( hàng đợi cổng vào đầy).
* Từ chối – bộ nhớ đệm của router đầy.
* Tràn – CPU bị tắc nghẽn và không thể chỉ định bộ nhớ đệm cho gói tin mới.
* Lỗi khung – lỗi phần cứng trong khung.

# CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM VÀ ĐÁNH GIÁ HIỆU NĂNG MẠNG NỘI BỘ CỦA OPENSTACK

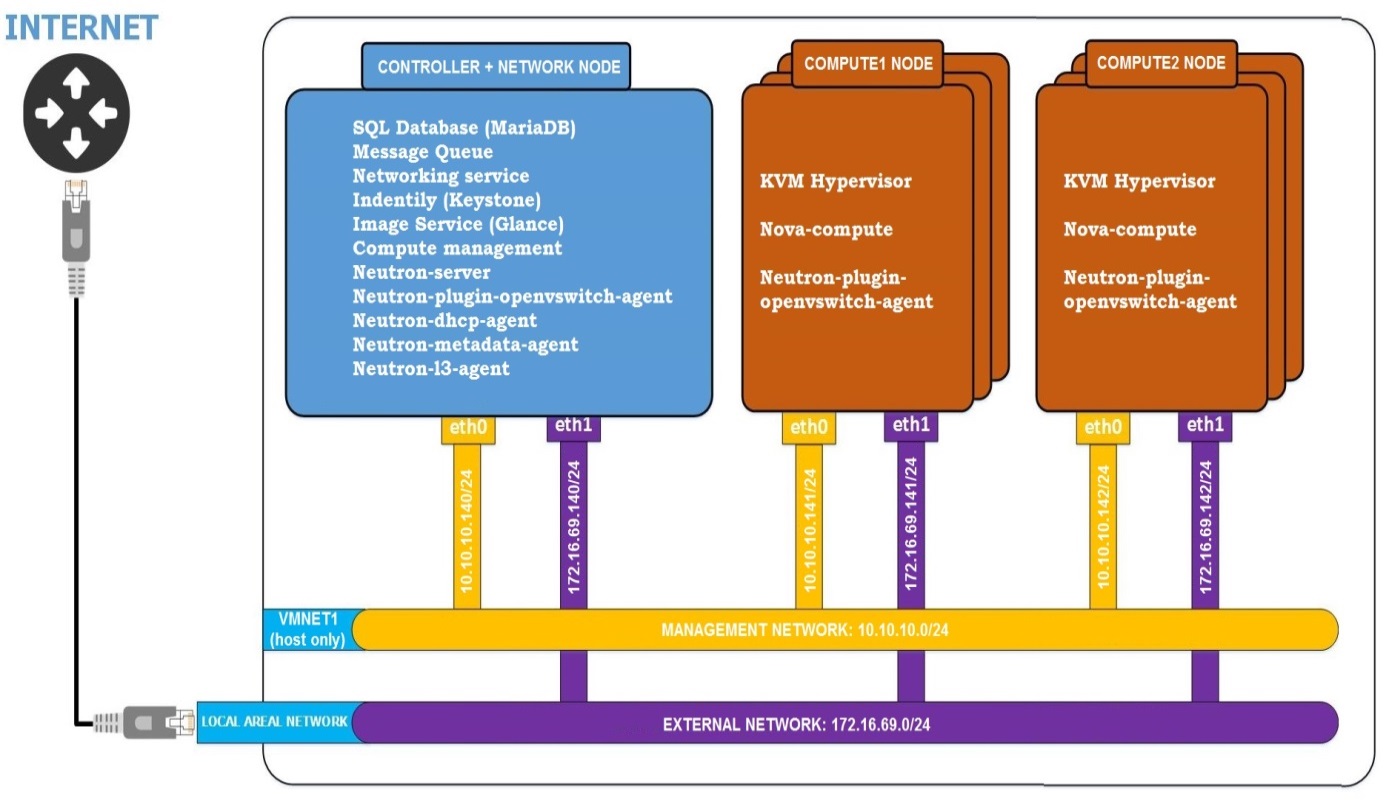
1. 1. **MÔ HÌNH TRIỂN THỰC NGHIỆM**

Dịch vụ điện toán đám mây sẽ cung cấp các máy ảo với VCPU, RAM, khả năng lưu trữ khác nhau. Trong OpenStack, các thành phần liên quan đến mạng và chuyển mạch cũng được ảo hóa thông qua Neutron. Neutron – phần mềm để định nghĩa các chức năng của một hệ thống mạng, từ lớp 2 cho đến lớp 3, từ switch cho đến router và gán vào hoạt động của các instances. Việc truyền thông của các máy ảo trong OpenStack sẽ không giống như việc truyền dẫn trên các hệ thống mạng thông thường vì các thành phần của mạng được định hướng theo mô hình SDN (Software Define Network).

* + 1. **Mô hình triển khai**

Mô hình được triển khai trên 1 Laptop Lenovo Core i3–2348M CPU, 8G Ram. Sử dụng công cụ hổ trợ VMware Workstation 10 để giả lập các máy chủ, bao gồm 3 máy chủ:

* 01 máy chủ đóng vai trò là Controller và Network Node:
  + Hệ điều hành: Ubuntu server 14.04.4 LTS
  + RAM: 2GB
  + CPU: 1x1 có hỗ trợ VT
  + NIC1: eth0: 10.10.10.0/24 (Hostonly)
  + NIC2: eth1: 172.16.69.0/24, gateway 172.16.69.1 (Bridge)
  + HDD: 70GB
* 02 máy chủ có cùng cấu hình đóng vai trò Compute Node:
  + Hệ điều hành: Ubuntu server 14.04.4 LTS
  + RAM: 2GB
  + CPU: 2x2 có hỗ trợ VT
  + NIC1: eth0: 10.10.10.0/24 (Hostonly)
  + NIC2: eth1: 172.16.69.0/24, gateway 172.16.69.1 (Bridge)
  + HDD: 80GB
* Các máy ảo VMs trong OpenStack (Instance):
  + VCPU: 1
  + RAM: 512MB
  + Disk Size: 10GB



*Hình 3.1. Mô hình triển khai*

Các dịch vụ cài đặt trên 3 máy chủ, bao gồm:

* 01 máy chủ làm Controller và Network, chạy các dịch vụ quản trị Controller của OpenStack như Keystone, MySQL, Neutron, và các dịch vụ liên quan khác.
* 02 máy chủ làm Conpute nodes cài đặt Neutron L3 Agent và nền tảng ảo hóa Hypervisor KVM cho việc cung cấp năng lực tính toán dành cho các máy chủ ảo.

Hệ thống mạng của mô hình triển khai bao gồm các mạng như sau:

* External và Floating IP: 172.16.69.0/24.
* Management Network: 10.10.10.0/24.
* Internal Network – network dành riêng cho các tenants: 192.168.2.0/24 và 192.168.3.0/24.
  + 1. **Kịch bản thử nghiệm**

Trong OpenStack, việc triển khai Neutron được hướng đến theo 2 hình thức: sử dụng GRE tunnel hoặc theo mô hình VLAN. GRE tunnel với ưu thế linh hoạt và dễ cấu hình tuy nhiên hiệu suất thấp, mô hình VLAN có nhiều ưu điểm trong việc triển khai các hệ thống lớn tuy nhiên cần đòi hỏi số lượng thiết bị, các thiết kế và cấu hình cao hơn. Để phục vụ cho mục đích thử nghiệm, mô hình GRE tunnel được sử dụng để hạn chế việc triển khai phức tạp đối với hạ tầng vật lý.

Do đó, mô hình thử nghiệm sử dụng GRE tunnel sẽ được thực hiện trong 4 kịch bản nhằm mục đích nghiên cứu hiệu suất mạng về thông lượng, thời gian trễ và vấn đề mất gói tin UDP giữa các máy ảo.

Trong bài báo này sẽ đưa ra 4 kịch bản thử nghiệm như sau:

* Trường hợp 1: Cùng mạng cùng compute node.
* Trường hợp 2: Khác mạng cùng compute node.
* Trường hợp 3: Cùng mạng khác compute node.
* Trường hợp 4: Khác mạng khác compute node.

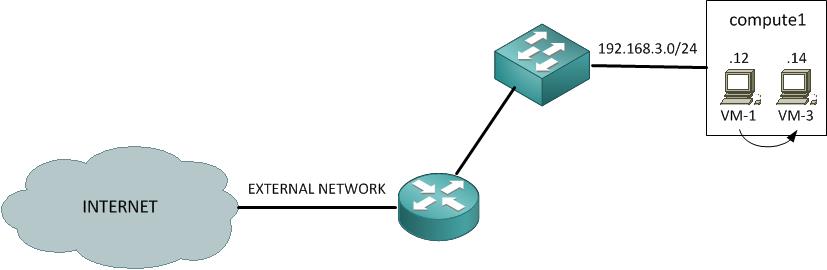
Bởi vì Network trong OpenStack được xem như là một tài nguyên và được ảo hóa toàn bộ để hướng đến việc thao tác và quản trị đơn giản. Về lý thuyết, hoạt động của hệ thống network “ảo hóa” này không khác gì so với hệ thống network vật lý. Trong OpenStack cũng hình thành nên các port ảo tương đương với các kết nối vật lý giữa các thiết bị, các router ảo tương đương với việc phân chia các dải mạng với nhau, các khái niệm như VLAN hoặc access control list, … cũng được thể hiện tương tự. Vì vậy để đánh giá hoạt động của lớp ảo hóa Network này thì cần đánh giá dựa trên các tiêu chí sau:

* Xem xét việc kết nối mạng trong hệ thống network ảo tại các máy compute node
* Xem xét việc kết nối mạng trong hệ thống network ảo giữa các máy compute node với nhau.

Từ kết quả đánh giá đó (chính là 4 trường hợp được thực hiện) mới xem xét đến việc sử dụng OpenStack Network trong thực tế có bảo đảm được như các mô hình network vật lý thông thường.

* + - 1. ***Trường hợp 1 - Cùng mạng cùng compute node.***

Mô hình thử nghiệm này sẽ thực hiện nhằm nghiên cứu thông lượng, độ trễ, tỉ lệ mất gói tin UDP giữa hai máy ảo cùng node compute và cùng mạng.

******

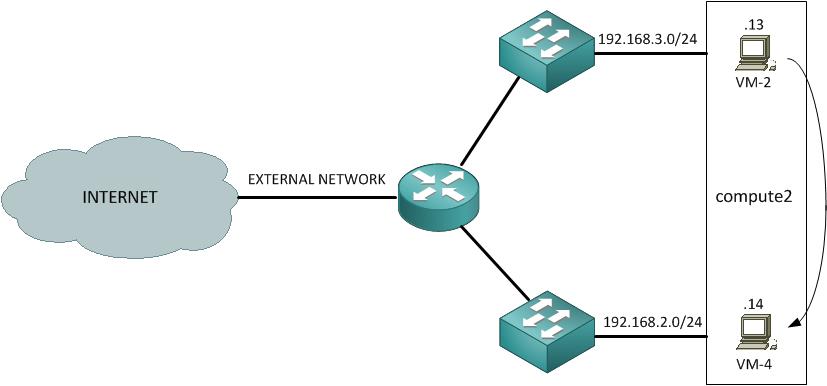
*Hình 3.2. Mô hình triển khai với 2 máy ảo cùng mạng, cùng compute node*

VM-1 thuộc node compute1 có địa chỉ mạng 192.168.3.12/24.

VM-3 thuộc node compute1 có địa chỉ mạng 192.168.3.14/24.

* + - 1. ***Trường hợp 2 - Khác mạng cùng compute node.***

Mô hình thử nghiệm này sẽ thực hiện nhằm nghiên cứu thông lượng, thời gian trễ và tỉ lệ mất gói tin UDP giữa hai máy ảo có cùng node compute nhưng nằm ở hai mạng khác nhau. Hai mạng khác nhau có thể nằm trên cùng một router hoặc hai router khác nhau.

******

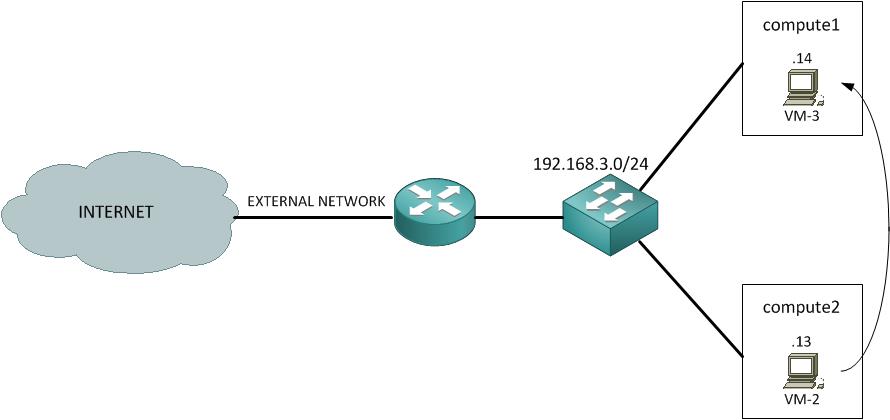
*Hình 3.3. Mô hình triển khai với 2 máy ảo khác mạng, cùng compute node*

VM-2 thuộc node compute2 có địa chỉ mạng 192.168.3.13/24

VM-4 thuộc node compute2 có địa chỉ mạng 192.168.2.14/24.

* + - 1. ***Trường hợp 3 - Cùng mạng khác compute node.***

Mô hình thử nghiệm này sẽ thực hiện nhằm nghiên cứu thông lượng, thời gian trễ và tỉ lệ mất gói tin UDP giữa hai máy ảo có node compute khác nhau nhưng cùng một mạng với nhau.

******

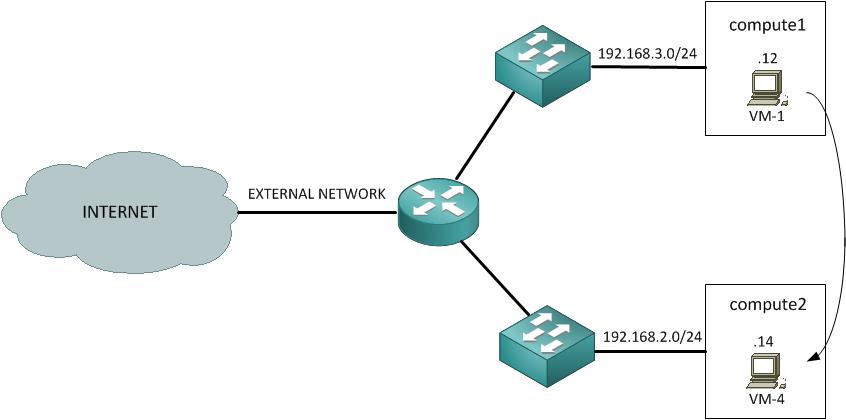
*Hình 3.4. Mô hình triển khai với 2 máy ảo cùng mạng, khác compute node*

VM-2 thuộc node compute2 có địa chỉ mạng 192.168.3.13/24

VM-3 thuộc node compute1 có địa chỉ mạng 192.168.3.14/24.

* + - 1. ***Trường hợp 4 - Khác mạng khác compute node.***

Mô hình thử nghiệm này sẽ thực hiện nhằm nghiên cứu thông lượng, thời gian trễ, tỉ lệ mất gói tin UDP giữa hai máy ảo có node compute khác nhau và khác mạng với nhau.

******

*Hình 3.5. Mô hình triển khai với 2 máy ảo khác mạng, khác compute node*

VM-1 thuộc node compute1 có địa chỉ mạng 192.168.3.12/24

VM-4 thuộc node compute2 có địa chỉ mạng 192.168.2.14/24.

* 1. **PHÂN TÍCH VÀ ĐÁNH GIÁ**

Dịch vụ Network trong OpenStack chính là Neutron, và Neutron tạo ra các môi trường chuyển mạch (switch), định tuyến (router) thông qua các dịch vụ Neutron L2 Agent, Neutron L3 Agent. Các mô hình thử nghiệm về mạng được mô tả ở trên do module Neutron trong OpenStack tạo ra.

Kết quả thử nghiệm tập trung vào việc phân tích các tham số liên quan đến việc truyền nhận gói tin trong hệ thống các máy ảo của OpenStack, các thông số được phân tích bao gồm thông lượng, tỉ lệ mất gói tin trên đường truyền, và thời gian trễ của gói tin.

* + 1. **Thông lượng và tỉ lệ mất gói tin UDP**

Để đo lường thông lượng gói tin UDP. IPERF được thực thi trong vòng 500 giây và dữ liệu sẽ được lấy định kỳ mỗi 5 giây 1 lần. Trong khi dữ liệu được lấy ở một mô hình thì các mô hình thử nghiêm khác sẽ bị dừng lại.

Sử dụng IPERF: iperf -u -c server\_ip -i 5 -t 500 -b 10G

Lựa chọn -u để tiến hành đo thông lượng gói tin UDP, -c xác định địa chỉ IP server, -i chu kỳ cập nhật dữ liệu, -t tổng thời gian tiến hành đo, -b sử dụng để xác định tổng băng thông có sẵn do mặc định công cụ IPERF sử dụng 1.05 Gbps cho UDP.

*Hình 3.6. Thông lượng UDP trong 4 trường hợp*

Hình 3.6 mô tả kết quả đo thông lượng gói tin UDP của 4 trường hợp với dữ liệu lấy theo chu kỳ là 5 giây, thời gian thực hiện là 500 giây.

*Hình 3.7. Thông lượng UDP trung bình trong 4 trường hợp*

Hình 3.7 mô tả kết quả đo thông lượng UDP trung bình 4 trường hợp với dữ liệu lấy theo chu kỳ là 5 giây, thời gian thực hiện là 500 giây.

*Hình 3.8. Tỉ lệ mất gói tin UDP trong 4 trường hợp*

Hình 3.8 mô tả kết quả đo tỉ lệ mất gói tin UDP của 4 trường hợp với dữ liệu lấy theo chu kỳ là 5 giây, thời gian thực hiện là 500 giây.

Từ những kết quả trên, có thể nhận thấy rằng khi 2 máy ảo được đặt trên cùng node và cùng mạng, chúng thực hiện tốt hơn so với khi chúng ở các node và mạng trong các trường hợp khác. Việc đo lường cho thấy các máy ảo trên cùng node, cùng mạng thì đạt được thông lượng trung bình xấp xỉ 4.2% so với các trường hợp còn lại.

Tương tự trong trường hợp đo lường tỉ lệ mất gói tin thì các máy ảo ở cùng node nhưng khác mạng hoặc khác node và khác mạng thì có thông số cao hơn so với các máy ảo cùng node và cùng mạng. Đặc biệt trong trường hợp các máy ảo cùng node nhưng khác mạng cho chúng ta thấy tỉ lệ mất gói tin xấp xỉ 40% so với trường hợp đặt các máy ảo cùng node và cùng mạng.

* + 1. **Thời gian trễ**

Để đo thời gian trễ gói tin ta sử dụng lệnh ping với tham số -c để thiết lập số lần gửi gói tin. Trong thử nghiệm này giá trị tham số -c truyền vào là 100.

*Hình 3.9. Thời gian trễ gói tin*

Hình 3.9 mô tả kết quả đo thời gian trể gói tin trung bình của 4 trường hợp. Từ kết quả này, ta nhận thấy giá trị ở trường hợp 1 thấp hơn giá trị ở ở trường hợp 2 và giá trị ở ở trường hợp 3 thấp hơn giá trị ở ở trường hợp 4. Điều này cho chúng ta kết luận các máy ảo trên khác địa chỉ mạng có giá trị cao hơn so với các máy ảo có cùng địa chỉ mạng. Kết quả này là bình thường vì khi máy ảo khác mạng nhau thì lưu lượng đi từ nguồn tới đích phải thông qua nhiều hop hơn khi chúng cùng mạng với nhau.

* + 1. **Đánh giá** 
       1. ***Giữa các trường hợp cùng node compute***
* ***Thông lượng UDP giữa trường hợp 1 và trường hợp 2***

*Hình 3.10. Thông lượng UDP giữa trường hợp 1 và trường hợp 2*

*Hình 3.11. Thông lượng UDP trung bình giữa trường hợp 1 và trường hợp 2*

Từ hình 3.10 và 3.11 ta thấy thông lượng UDP của trường hợp 1 cao hơn thông lượng của trường hợp 2 (thông lượng UDP trung bình trường hợp 1 là 349,38 và trường hợp 2 là 162,55). Điều đó chứng tỏ những mấy ảo cùng node compute và cùng mạng sẽ có thông lượng cao hơn những máy cùng node compute nhưng khác mạng.

* ***Tỉ lệ mất gói tin UDP giữa trường hợp 1 và trường hợp 2***

*Hình 3.12. Tỉ lệ mất gói tin UDP giữa trường hợp 1 và trường hợp 2*

Từ hình 3.12 ta nhận thấy tỉ lệ mất gói tin UDP của trường hợp 2 cao hơn trường hợp 1 (trường hợp 1 là 28,25% còn trường hợp 2 là 53,45%). Do đó ta có thể kết luận những máy ảo có cùng node compute và nằm cùng mạng sẽ có tỉ lệ mất gói tin UDP thấp hơn so với những máy ảo cùng node compute nhưng khác mạng.

* ***Thời gian trễ gói giữa trường hợp 1 và trường hợp 2***

*Hình 3.13. Thời gian trễ gói giữa trường hợp 1 và trường hợp 2*

Từ hình 3.13 cho thấy thời gian trễ gói trung bình thu được cho các máy ảo được đặt trên cùng node compute và nằm cùng mạng thấp hơn so với các máy ảo được đặt cùng node compute nhưng nằm khác mạng.

* + - 1. ***Giữa các trường hợp khác node compute***
* ***Thông lượng UDP giữa trường hợp 3 và trường hợp 4***

*Hình 3.14. Thông lượng UDP giữa trường hợp 3 và trường hợp 4*

*Hình 3.15. Thông lượng UDP trung bình giữa trường hợp 3 và trường hợp 4*

Từ hình 3.14 và 3.15 ta thấy thông lượng UDP của trường hợp3 cao hơn thông lượng của trường hợp4 (thông lượng UDP trung bình trường hợp3 là 167,45 và trường hợp 4 là 136,47). Điều đó chứng tỏ những mấy ảo nằm khác node compute và cùng mạng sẽ có thông lượng cao hơn những máy nằm khác node compute và khác mạng.

* 1. **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Điện toán đám mây mã nguồn mở và đặc biệt là OpenStack hiện nay là một trong những công nghệ được lựa chọn để triển khai cho các công ty trên toàn thế giới. Trong nghiên cứu này, em chỉ phân tích và đánh giá hiệu năng mạng dựa trên các thông số thông lượng UDP, thời gian trễ, tỉ lệ mất gói tin trên OpenStack. Từ đó đưa ra kết luận OpenStack Neutron đảm bảo một hiệu suất cao, băng thông hầu như không bị hiện tượng thắt cổ chai. Ngoài ra, kết quả cũng cho thấy rằng, vị trí của máy ảo cũng như địa chỉ mạng ảnh hưởng đến hiệu suất, cụ thể các máy ảo cùng node và cùng mạng sẽ cho hiệu năng cao hơn tất cả các trường hợp còn lại bởi vì đường truyền của các máy ảo ngắn hơn.

Bài nghiên cứu chỉ dừng lại trong môi trường LAB với 03 máy chủ được cài đặt trong cùng 1 máy tính với cấu hình thấp nên việc cung cấp tài nguyên phần cứng cho 03 máy chủ có phần hạn chế hơn rất nhiều so với thực tế. Ngoài ra, trong thực tế các nút mạng cũng như các máy chủ sẽ chịu một lượng tải nhất định của môi trường xung quanh, vì vậy nên giá trị thu được trong môi trường LAB sẽ khác biệt so với thực tế và cần được kiểm chứng đối với từng trường hợp cụ thể, ví dụ trong môi trường data center của FTP, hoặc data center của VNPT,…

Nghiên cứu chỉ dừng lại ở việc kiểm tra thông lượng, tỉ lệ mất gói tin UDP và thời gian trể gói tin chưa đi vào ngiên cứu gói tin TCP. Việc nghiên cứu gói tin UDP cũng chỉ dừng lại trong mô hình nhỏ cần được mở rộng để cho kết quả và kết luận tốt hơn. Mục tiêu tương lại của đề tài sẽ tìm hiểu thuật toán phân phối máy ảo trong OpenStack để đưa ra mô hình tối ưu hơn.

# KẾT LUẬN

Nghiên cứu đã thực hiện được việc triển khai mô hình OpenStack Liberty sử dụng OpenvSwitch cho việc chuyển mạch mạng ảo. Với 03 máy chủ: 01 máy chủ đóng vai trò là Controller và Network Node, 02 máy chủ đóng vai trò là Compute Node. Tiến hành khởi tạo thành công các máy ảo trên các Node Compute theo mô hình mạng như mong đợi. Trên các máy ảo đã tiến hành cài đặt thành công IPERF.

Nghiên cứu đã sử dụng công cụ IPERF để tiến hành đo các thông số mạng như thông lượng, tỉ lệ mất gói tin UDP và thời gian trể gói. Tiến hành phân tích các số liệu đo được thông qua 4 trường hợp. Từ đó đưa ra và tiến hành so sánh các kết quả thu được trong từng trường hợp.

Do điều kiện về cơ sở vật chất không cho phép nên nghiên cứu chỉ dừng lại ở mô hình LAB nhỏ nên chưa có thể đưa ra kết luận chính xác nhất cho vấn đề liên quan đến hiệu năng mạng nội bộ trong OpenSatck.

Đề tài nghiên cứu được thực hiện trong khoản thời gian ngắn nên việc thiết xót là việc không thể tránh khỏi, em rất mong nhận được những phản hồi và đóng góp của các thầy cô để đề tài có thể được hoàn thiện hơn.

# DANH MỤC TÀI LIỆU THAM KHẢO

**TIẾNG VIỆT**

1. Bùi Trung Hiếu, Giới thiệu về OpenStack. 2014; Available from: <http://www.slideshare.net/lanhuonga3/tm-hiu-v-openstack>
2. Ngô Bá Hùng, Phạm Thế Phi, Quản lý hiệu suất mạng (Performance management); Available from: http://voer.edu.vn/m/quan-ly-hieu-suat-mang-performance-management/efdffd10

**TIẾNG ANH**

1. Canonical. Ubuntu Cloud: Technologies for future-thinking companies. 2012 [cited 2012 04/06/2012]; Available from: http://www.canonical.com/about-canonical/resources/white-papers/ubuntu-cloud-technologies-future-thinking-companies
2. How to compare VMware and OpenStack. 2012; Available from: http://www.slideshare.net/mirantis/how-to-compare-vmware-and-openstack
3. Infrastructure as a Service with Comprehensive Management and Security. 2012 [cited 2012; Available from: http://www.vmware.com/products/datacenter-virtualization/vcloud-suite/overview.html
4. OpenStack Compute. 2012; Available from: http://openstack.org/projects/compute/
5. OpenStack Image Service. 2012; Available from: http://openstack.org/projects/image-service/
6. OpenStack Object Storage. 2012; Available from: http://openstack.org/projects/storage/
7. Renski, B. VMWare and OpenStack: Comparing Technology and Philosophy. 2012; Available from: http://www.mirantis.com/why-mirantis/openstack-technology/comparing-vmware-and-openstack/