

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



TRẦN THỊ DUNG

**TRIỂN KHAI VÀ CẢI TIẾN HỆ THỐNG CÂN
BẰNG TẢI GIỮA CÁC TRUNG TÂM DỮ LIỆU
DỰA TRÊN GIẢI PHÁP MÃ NGUỒN MỞ
POLARIS**

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Truyền thông và mạng máy tính

HÀ NỘI - 2018

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

TRẦN THỊ DUNG

**TRIỂN KHAI VÀ CẢI TIẾN HỆ THỐNG CÂN
BẰNG TẢI GIỮA CÁC TRUNG TÂM DỮ LIỆU DỰA
TRÊN GIẢI PHÁP MÃ NGUỒN MỞ POLARIS**

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Truyền thông và mạng máy tính

Cán bộ hướng dẫn: TS. Hoàng Xuân Tùng

HÀ NỘI - 2018

TÓM TẮT

Tóm tắt: Trong bối cảnh công nghệ thông tin ngày càng có những bước phát triển vượt bậc, mạng internet ngày càng trở nên gần gũi và trở thành một phần không thể thiếu của mỗi người sử dụng các thiết bị thông minh như máy tính, điện thoại hay tivi,... và lớn hơn là các tập đoàn, doanh nghiệp, công ty, trường học.

Số lượng truy cập của người dùng vào các dịch vụ ứng dụng ngày càng tăng. Nếu các doanh nghiệp đầu tư các hạ tầng hệ thống chứa dữ liệu tập trung để có thể chịu tải tốt, thì tốn rất nhiều chi phí. Hơn nữa, việc đầu tư như vậy vẫn không thể giải quyết được vấn đề độ trễ trên đường truyền.

Do vậy, các hệ thống cân bằng tải giữa các trung tâm dữ liệu ngày càng phát triển. Tuy nhiên các phần mềm triển khai hệ thống này đều tính phí. Hiểu được điều đó, báo cáo đưa ra giải pháp triển khai hệ thống cân bằng tải GSLB dựa vào phần mềm mã nguồn mở Polaris. Tuy nhiên do Polaris có nhiều nhược điểm, vì vậy báo cáo sẽ đưa phương pháp để cải tiến Polaris.

Báo cáo bao gồm 4 chương:

Chương 1: Trình bày tổng quan báo cáo đề án, lý do nên sử dụng cân bằng tải GSLB.

Chương 2: Trình bày kiến thức nền tảng được nghiên cứu, sử dụng trong hệ thống cân bằng tải cùng các công cụ hiện có trong việc triển khai và cải tiến hệ thống cân bằng tải giữa các trung tâm cơ sở dữ liệu.

Chương 3: Trình bày về một giải pháp cho hệ thống cân bằng tải toàn cầu, cải tiến mô hình mã nguồn mở Polaris.

Chương 4: Thử nghiệm và đánh giá về giải pháp cho hệ thống cân bằng tải dựa trên mã nguồn mở Polaris-GSLB. Cài đặt chương trình đã cải tiến dựa trên phần mềm mã nguồn mở Polaris-GSLB.

Cuối cùng, phần kết luận trình bày những kết quả đã đạt được của đề án và định hướng nghiên cứu và phát triển trong tương lai về việc bảo tính khả dụng của mỗi site cho người dùng trải nghiệm tốt nhất.

Từ khóa: DNS, IP, GSLB, Polaris-GSLB.

LỜI CẢM ƠN

Trước tiên, em xin bày tỏ lòng biết ơn chân thành và sâu sắc nhất tới Thầy giáo, Tiến sĩ Hoàng Xuân Tùng người đã tận tình chỉ bảo, hướng dẫn, động viên và giúp đỡ em trong suốt quá trình thực hiện đề tài.

Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý thầy cô giáo trong Khoa Công nghệ thông tin nói riêng và trong trường Đại học Công nghệ - Đại học Quốc Gia Hà Nội nói chung, đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Và đặc biệt, trong học kỳ này, Khoa đã tổ chức cho chúng em được tiếp cận với môn học mà theo em là rất hữu ích đối với sinh viên ngành truyền thông và mạng máy tính cũng như tất cả các sinh viên thuộc các ngành khác của trường.

Em xin gửi lời cảm ơn đến sự hướng dẫn, giúp đỡ tận tình của các anh trong phòng hệ thống của công ty cổ phần Bạch Minh (Vega corporation)

Cuối cùng, tôi xin gửi lời cảm ơn tới các anh chị và các bạn, đặc biệt là các thành viên lớp K59N đã ủng hộ, giúp đỡ trong suốt quá trình tôi học tập trên giảng đường đại học và thực hiện đề tài.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày 24 tháng 11 năm 2018

Sinh viên

Trần Thị Dung

LỜI CAM ĐOAN

Tôi xin cam đoan: đồ án “*Triển khai và cải tiến hệ thống cân bằng tải giữa các trung tâm dữ liệu dựa vào giải pháp mã nguồn mở polaris*” là công trình nghiên cứu khoa học của riêng tôi dưới sự giúp đỡ tận tình của Tiến sĩ Hoàng Xuân Tùng.

Tất cả những tham khảo từ các nghiên cứu liên quan đều được nêu nguồn gốc một cách rõ ràng từ danh mục tài liệu tham khảo trong đồ án. Trong đồ án, không có việc sao chép tài liệu, công trình nghiên cứu của người khác mà không chỉ rõ về tài liệu tham khảo.

Hà Nội, ngày 24 tháng 11 năm 2018

Tác giả

Trần Thị Dung

MỤC LỤC

Tóm tắt.....	I
Lời cảm ơn.....	II
Lời cam đoan	III
Danh sách thuật ngữ và từ viết tắt	i
Danh sách hình vẽ	ii
CHƯƠNG 1. Đặt vấn đề	1
CHƯƠNG 2. Kiến thức nền tảng.....	4
2.1. Cân bằng tải	4
2.1.1. Khái niệm.....	4
2.1.2. Mục tiêu.....	5
2.1.3. Ưu điểm của cân bằng tải	5
2.1.4. Nguyên lý hoạt động	5
2.1.5. Cơ chế cân bằng tải trong và giữa các trung tâm dữ liệu	6
2.2. Mô hình GSLB	8
2.2.1. Khái niệm gslb.....	8
2.2.2. Kiến trúc và cách thức hoạt động của GSLB	9
2.2.3. Giải pháp xây dựng GSLB	10
2.3. Các phần mềm cân bằng tải GSLB.....	14
2.3.1. Incapsula.....	14
2.3.2. Amazon Elastic Load Balancing	15
2.3.3. F5 Networks.....	15
2.4. Kết luận.....	16
CHƯƠNG 3. Phân tích hệ thống mã nguồn mở Polaris-gslb.....	17
3.1. Giới thiệu.....	17
3.2. Mô tả polaris-gslb.....	17

3.3. Cấu hình chương trình[20]	19
3.4. Nguyên lý hoạt động	21
3.5. Vấn đề của Polaris-GSLB	23
3.6. Kết luận.....	26
CHƯƠNG 4. Thử nghiệm và đánh giá hệ thống cân bằng tải Polaris-GSLB cải tiến.....	27
4.1. Cải tiến Polaris-GSLB	27
4.1.2. Ứng dụng phương pháp geoDNS	28
4.1.3. Sử dụng redis thay memcached.....	32
4.2. Mô hình hệ thống thử nghiệm.....	33
4.3. Cài đặt và cấu hình Polaris-gslb	34
Bước 1: Cập nhật hệ điều hành	34
Bước 2: Cài đặt powerDNS.....	34
4.4. Thử nghiệm hệ thống cân bằng tải Polaris-GSLB cải tiến khi người dùng truy cập đến ứng dụng	35
4.5. Kết luận.....	39
Kết luận	41
Tài liệu tham khảo	42

DANH SÁCH THUẬT NGỮ VÀ TỪ VIẾT TẮT

Từ viết tắt	Từ chuẩn
CDN	Content delivery network
CIDR	Classless Inter-Domain Routing
Computing	Những việc liên quan đến sử dụng máy tính
DNS	Domain name systems
EDNS	Extension of domain name system
GSLB	Global server load balancing
ISP	Internet service provider
RFC	Request for Comments
POP	Points of presence
CNTT	Công nghệ thông tin
PĐT	Phòng đào tạo
RRT	Round trip time
TT&MMT	Truyền thông và mạng máy tính
VPK	Văn phòng khoa

Danh sách hình vẽ

Hình 1.1. Mô hình cân bằng tải	6
Hình 1.2. Mô hình cân bằng tải theo cụm[18].....	7
Hình 1.3. Mô hình sử dụng Floating IPs[18]	8
Hình 1.4. Mô phỏng GSLB	10
Hình 1.5. Mô hình sử dụng anycast với BGP[19].....	11
Hình 1.6. Anycast với BGP[19]	12
Hình 1.7. Mô hình GSLB sử dụng GeoDNS[19]	12
Hình 1.8. Trang chủ dịch vụ INCAPSULA.....	14
Hình 1.9. Trang chủ dịch vụ Amazon ELB	15
Hình 2.1. Mô tả Polaris.....	18
Hình 2.2. Mô hình GSLB	21
Hình 2.3. Độ trễ truyền dữ liệu theo khoảng cách người dùng đến trung tâm dữ liệu[19]	24
Hình 2.4. TCP congestion window	25
Hình 3.1. Mô hình Polaris-GSLB sử dụng giải pháp geoIP	27
Hình 3.2. Phương pháp EDNS	29
Hình 3.3. Lưu đồ tìm kiếm sử dụng cuckoo hashing	30
Hình 3.4. Mô hình thử nghiệm hệ thống cân bằng tải Polaris-GSLB cải tiến	36
Hình 3.5. Kết quả truy vấn tên miền “vod.vpdns-dev.vws.vegacdn.vn”	38

CHƯƠNG 1. ĐẶT VẤN ĐỀ

Hiện nay với tốc độ phát triển chóng mặt của mạng internet và số lượng người dùng trên khắp thế giới truy cập vào các ứng dụng để mua sắm, tìm kiếm dữ liệu cho các trường học, công việc, chơi game hoặc trò chuyện với bạn bè trong qua các trang mạng xã hội ngày ngày tăng. Dẫn đến việc xử lý một lượng lớn yêu cầu của người dùng trở nên khó khăn hơn. Khi người dùng truy cập ứng dụng vào thời gian thông lượng đạt mức cao nhất, thì việc có một phần mềm theo dõi, phân phối các yêu cầu từ người dùng đến các máy chủ cho phép việc xử lý yêu cầu nhanh hơn và tin cậy hơn. Vì vậy, giải pháp cân bằng tải rất cần thiết cho các doanh nghiệp cả lớn và nhỏ để có thể duy trì kết nối, tốc độ tải và kiểm soát lưu lượng đến các máy chủ tốt hơn.

Cân bằng tải là chìa khóa quan trọng trong việc đảm bảo tính khả dụng của cơ sở hạ tầng mạng, thường được sử dụng để cải thiện hiệu suất và độ tin cậy của các trung tâm dữ liệu, các ứng dụng, cơ sở dữ liệu và các dịch vụ khác bằng cách phân phối tải giữa các máy chủ. Hệ thống cân bằng được xem là một chương trình quản lý lưu lượng.

Ban đầu, việc cân bằng tải được hiểu là phân tán lượng truy cập giữa các máy chủ trong cùng một vị trí, ví dụ: trung tâm dữ liệu. Tuy nhiên, việc mở rộng, tính khả dụng cao và hiệu năng lại là từ khóa để thành công cho bất kỳ một sự phát triển ứng dụng thương mại nào. Nhiều doanh nghiệp cố gắng mở rộng bằng cách triển khai thêm nhiều máy chủ và cơ sở hạ tầng trong cùng một vị trí duy nhất, đây là cách triển khai tập trung và cách triển khai này có nhiều hạn chế. Việc điều khiển tập trung một ứng dụng có nhiều dịch vụ được cho là một điểm thất bại trong việc phân phối dịch vụ ứng dụng. Ví dụ: Nếu trung tâm dữ liệu bị mất kết nối với toàn bộ hoặc một phần mạng internet public, thì người dùng sẽ không thể truy cập đến trung tâm dữ liệu này được và điều này có thể tác động xấu đến doanh nghiệp. Trải nghiệm dịch vụ của người dùng cũng bị ảnh hưởng do tắc nghẽn băng thông và tắc nghẽn các vùng xung quanh trung tâm dữ liệu bị mất kết nối với mạng internet public. Hơn nữa, khi người dùng truy cập đến trung tâm dữ liệu ở một vị trí địa lý cách xa với vị trí người dùng thì đường truyền có thể gặp sự chậm trễ lớn.

Các kiến trúc tập trung cũng không phù hợp với các công ty quốc tế phải phục vụ nội dung bản địa hóa cho người dùng ở khắp mọi nơi trên thế giới. Ngoài ra nếu một doanh nghiệp đặt các dịch vụ của họ ở các máy chủ thuộc các

trung tâm dữ liệu khác nhau thì việc chỉ sử dụng cân bằng tải giữa các máy chủ trong cùng một trung tâm dữ liệu không thể hoạt động như mong đợi.

Do vậy việc cân bằng tải không chỉ dừng lại ở một vị trí, mà được mở rộng ra với phạm vi toàn thế giới – giữa các trung tâm dữ liệu trên toàn cầu. Việc sử dụng cân bằng tải giữa các trung tâm dữ liệu trên toàn thế giới được gọi là cân bằng tải toàn cục (viết tắt GSLB)

GSLB được xây dựng dựa trên cùng nguyên tắc như cân bằng tải truyền thống. Tuy nhiên, việc phân phối tải không còn giới hạn trong một mạng cục bộ hoặc một trung tâm dữ liệu. GSLB sẽ giải quyết những hạn chế của hệ thống cân bằng tải truyền thống bằng cách phân phối lưu lượng giữa các cụm máy chủ hoặc các điểm truy cập dịch vụ được triển khai ở nhiều vị trí địa lý khác nhau. Bằng cách phục vụ nội dung từ nhiều điểm khác nhau trong internet, GSLB làm giảm bớt tác động của tắc nghẽn băng thông mạng và cung cấp cho người dùng máy chủ có trạng thái tốt nhất, tránh trường hợp máy chủ địa phương hoặc một trung tâm dữ liệu cụ thể bị lỗi. Người dùng có thể được tự động chuyển đến một trung tâm dữ liệu gần nhất hoặc ít tải nhất tại thời điểm yêu cầu, giảm thiểu khả năng độ trễ tải dài hoặc dịch vụ bị gián đoạn

Với sự phát triển về công nghệ và thiết bị điện tử như hiện nay, việc truy cập nhanh chóng và đáng tin cậy trong việc truy cập nội dung và ứng dụng là điều cần thiết cho các doanh nghiệp thành công. Ví dụ: ứng dụng skype để gọi video. Cuộc gọi phải đảm bảo truyền thoại với thời gian thực, nếu như cuộc gọi thoại bị trễ 7 giây có thể khiến một lượng lớn người dùng từ bỏ ứng dụng skype. Sự phát triển của internet di động và thiết bị cầm tay cá nhân thông minh khiến cho nhu cầu luôn luôn kết nối của các ứng dụng thương mại và kinh doanh tiếp tục phát triển một cách chóng mặt. Vì vậy giải pháp GSLB rất cần thiết cho các doanh nghiệp.

Hiện nay các phần mềm cung cấp giải pháp cân bằng tải GSLB thường có chi phí từ trung bình đến cao tùy thuộc vào nhà cung cấp. Ví dụ Kemp GEO Load Masters, amazon route 53,... Để giảm thiểu chi phí thuê phần mềm cung cấp tính năng GSLB, chúng ta có thể sử dụng phần mềm mã nguồn mở Polaris - GSLB.

Polaris-GSLB cho phép nhà quản trị hệ thống theo dõi được trạng thái hoạt động của các trung tâm dữ liệu thông qua phản hồi gói tin http và tcp. Tuy

nhien, Polaris-GSLB là chương trình cung cấp giải pháp GSLB nhưng lại không giải quyết được bài toán độ trễ đường truyền từ người dùng đến các trung tâm dữ liệu và không kiểm soát được hiệu năng xử lý yêu cầu của mỗi trung tâm dữ liệu đó.

Với ý nghĩa quan trọng của hệ thống cân bằng tải toàn cầu, đồ án sẽ trình bày về việc vận dụng mô hình cân bằng tải GSLB và đảm bảo tính khả dụng cho các dịch vụ ứng dụng bằng cách bổ sung tính năng geoIP và khả năng truy xuất địa chỉ IP từ danh sách chứa các subnet của các nhà cung cấp mạng cho phần mềm mã nguồn mở Polaris-GSLB.

CHƯƠNG 2. KIẾN THỨC NỀN TẢNG

Cùng với sự phát triển của internet, nhu cầu truy cập mạng bùng nổ, dẫn đến các máy chủ cung cấp dịch vụ trở nên quá tải. Việc lựa chọn quản lý, cài đặt dịch vụ tập trung trong một cụm máy chủ hay một vùng với cấu hình mạnh thì kéo theo chi phí phát sinh rất lớn. Vì vậy các doanh nghiệp cần phải nhân bản nội dung dịch vụ ra nhiều nơi. Tuy nhiên, nếu chỉ tập trung sử dụng cân bằng tải giữa các máy chủ trong cùng một vùng thì việc chuyển đổi giữa các trung tâm dữ liệu sẽ gây ra độ trễ lớn, và sẽ gây khó khăn cho người sử dụng dịch vụ khi chuyển vùng trong trường hợp doanh nghiệp sử dụng máy chủ tập bị sập. Vì vậy giải pháp cân bằng tải toàn cầu rất cần thiết.

2.1. Cân bằng tải

2.1.1. Khái niệm

Cân bằng tải, thông thường, có thể hiểu là phương pháp phân phối và phân bổ các nhiệm vụ nhất định giữa các tài nguyên sẵn có một cách hiệu quả. Trong điện toán, cân bằng tải được định nghĩa một phương thức mạng phân phối khối lượng công việc trên nhiều tài nguyên máy tính như máy tính và cụm của chúng, liên kết mạng, đơn vị xử lý trung tâm hoặc ổ đĩa. Mục đích của cân bằng tải vẫn là để tối ưu hóa việc sử dụng tài nguyên, giảm thiểu thời gian phản hồi, tối đa hóa thông lượng và tránh một nguồn tài nguyên duy nhất bị quá tải. Cân bằng tải thường được thực hiện trong phần mềm, mặc dù nó cũng có thể được thực hiện bằng phần cứng hoặc thậm chí là sự kết hợp của phần mềm và phần cứng. Chương trình cân bằng tải sẽ lắng nghe cổng nơi các máy khách bên ngoài kết nối để truy cập các dịch vụ. Các yêu cầu được chuyển tiếp bởi bộ cân bằng tải đến các máy chủ phía sau, và các máy chủ này sẽ trả phản hồi về cho người dùng thông qua bộ cân bằng tải.

Các website hiện đại có lưu lượng truy cập cao phải phục vụ hàng trăm nghìn, thậm chí hàng triệu yêu cầu đồng thời từ người dùng hoặc khách hàng, đồng thời phải phản hồi chính xác lại các văn bản, hình ảnh, video hoặc dữ liệu ứng dụng, tất cả đều được thực hiện rất nhanh chóng và đáng tin cậy. Để đảm bảo người dùng truy cập ứng dụng có độ trễ thấp và hiệu năng của các máy chủ thì việc phân phối các yêu cầu giữa các máy chủ sẽ rất cần thiết.

2.1.2. Mục tiêu

Theo [18], Mục tiêu của việc cân bằng tải là:

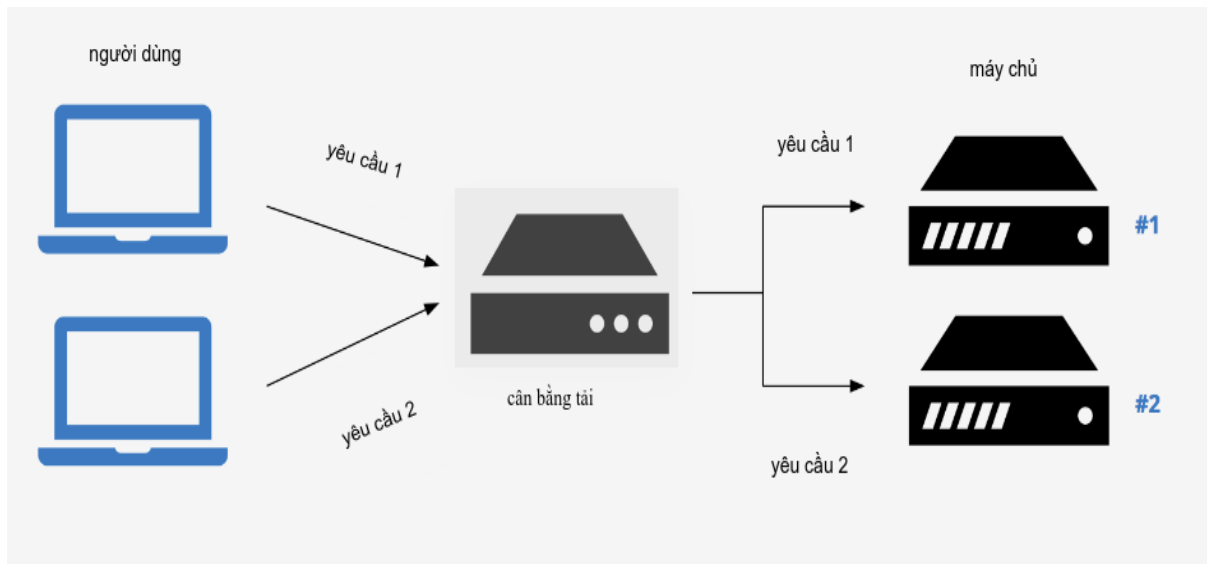
- Cải thiện hiệu suất của các máy chủ
- Dự phòng trong trường hợp hệ thống bị lỗi
- Duy trì sự ổn định của hệ thống

2.1.3. Ưu điểm của cân bằng tải

Khả năng của cân bằng tải thể hiện ở các mặt sau:

- Tính linh hoạt: Hệ thống cho phép bổ sung và loại bỏ các máy chủ bất kỳ khi nào cần và hiệu quả ngay tức thì. Thực tế, việc này không làm gián đoạn tới hoạt động của cả hệ thống, mà chỉ tại một điểm (node) trong hệ thống đó. Điều này cho phép duy trì, sửa chữa bất kỳ máy chủ nào trong hệ thống (thậm chí trong giờ cao điểm) mà ít tác động hoặc không có tác động nào tới hệ thống. Một cân bằng tải cũng có thể trực tiếp điều khiển lưu lượng mạng bằng cách sử dụng tập tin cookie, phân tích cú pháp URL, các thuật toán tĩnh/động... để tìm ra cách phân tải tối ưu cho hệ thống.
- Tính sẵn sàng cao: Hệ thống sẽ liên tục kiểm tra trạng thái của các máy chủ trong hệ thống và tự động “loại” bất kỳ máy chủ nào không “trả lời” trong một chu kỳ, cũng như tự động bổ sung máy chủ đó ngay khi nó hoạt động trở lại. Quá trình này là hoàn toàn tự động, thông qua cơ chế giao tiếp của cân bằng tải và các máy chủ, không cần có sự tham gia điều khiển trực tiếp của người quản trị. Do đó, một hệ thống cân bằng tải hướng đến tính dự phòng cho thiết bị chính trong trường hợp có thiết bị nào đó bị “hỏng”.
- Khả năng mở rộng: Cân bằng tải chịu trách nhiệm phân phối tải tới nhiều máy chủ trong một trung tâm dữ liệu, với mục đích là nâng cao hiệu quả, tăng sức mạnh phục vụ với số lượng lớn các máy chủ. Điều này mang lại lợi ích lớn về kinh tế, vì chỉ phải chi phí cho nhiều máy chủ nhỏ, thay vì đầu tư cho một máy chủ lớn, thiết bị chuyên dụng. Ngoài ra, trong quá trình hoạt động, số các máy chủ có thể thay đổi, thêm/bớt, loại bỏ, thay thế một cách dễ dàng mà không ảnh hưởng đến hoạt động của hệ thống, giữ cho hệ thống luôn có tính sẵn sàng cao.

2.1.4. Nguyên lý hoạt động



Hình 2.1. Mô hình cân bằng tải

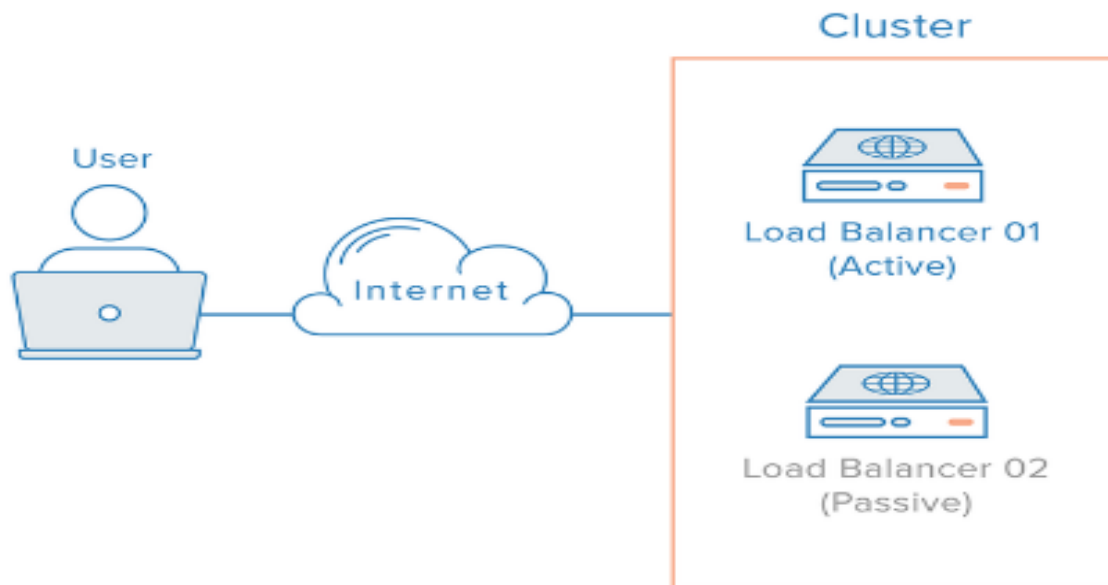
Hệ thống cân bằng tải thường được dùng chủ yếu là phần mềm, nó lắng nghe các cổng mà dùng để nhận yêu cầu từ người để xử lý các yêu cầu đó. Nó được đặt giữa các máy chủ xử lý các yêu cầu đó và internet. Khi một yêu cầu được nhận, bộ cân bằng tải đầu tiên sẽ xác định máy chủ nào trong một nhóm có đang ở trạng thái khả dụng, sau đó định tuyến yêu cầu đến máy chủ đó. Nếu một máy chủ bị hỏng, chương trình cân bằng tải sẽ chuyển hướng lưu lượng truy cập đến các máy chủ có trạng thái sức khỏe ổn định còn lại. Khi một máy chủ mới được thêm vào nhóm các máy chủ, bộ cân bằng tải tự động bắt đầu gửi yêu cầu đến nó.

Cân bằng tải lựa chọn máy chủ để chuyển tiếp yêu cầu dựa vào sự kết hợp của hai yếu tố đó là kiểm tra trạng thái hoạt động của máy chủ và thuật toán cân bằng tải. Trước tiên, họ sẽ đảm bảo rằng bất kỳ máy chủ nào họ có thể chọn thực sự phản hồi phù hợp với các yêu cầu và sau đó sử dụng thuật toán được định cấu hình trước để chọn máy chủ ở trạng thái tốt trong trung tâm dữ liệu

2.1.5. Cơ chế cân bằng tải trong và giữa các trung tâm dữ liệu

Để giải quyết vấn đề hạn chế của một bộ cân bằng trong một vùng, thì các doanh nghiệp, theo cách truyền thống, xây dựng hệ thống cân bằng tải gồm bộ cân bằng tải chính và bộ cân bằng tải dự phòng ở trong cùng một vùng

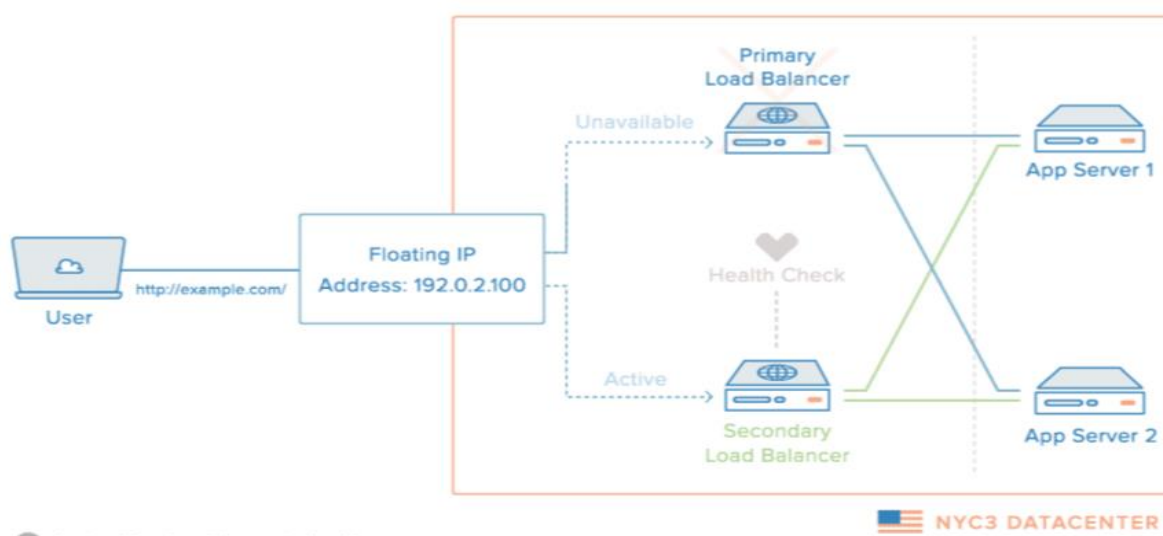
và các bộ cân bằng tải này liên kết với nhau tạo thành một cụm. Mỗi bộ cân bằng tải là đều có khả năng phát hiện lỗi và phục hồi.



Hình 2.2. Mô hình cân bằng tải theo cụm[11]

Trong trường hợp bộ cân bằng tải chính bị lỗi, DNS sẽ đưa người dùng đến với bộ cân bằng tải dự phòng. Để tiết kiệm thời gian khi quá trình thay đổi DNS có thể khá lâu mới thông báo được lên Internet, ngoài ra để việc chống lỗi này hoạt động tự động thì nhiều quản trị viên sử dụng hệ thống cho phép linh hoạt thay đổi IP, chẳng hạn như Floating IP.

Bằng cách cung cấp một địa chỉ IP tĩnh tùy chỉnh được khi cần thiết, các vấn đề về mất thời gian thông báo lên Internet và lưu bộ nhớ đệm khi thay đổi DNS có thể loại bỏ. Tên miền có thể duy trì liên kết với cùng một địa chỉ IP, trong khi địa chỉ IP này được di chuyển giữa các máy chủ.



Hình 2.3. Mô hình sử dụng Floating IPs[11]

Tuy nhiên khi các máy chủ của một trang cùng nằm ở một vị trí và không có hệ thống dự phòng, trong trường hợp trung tâm dữ liệu bị mất điện hay bị tấn công thì các dịch vụ cung cấp ở vị trí đó sẽ hoàn toàn tê liệt, điều này gây tổn thất cho doanh nghiệp. Điều này sẽ khiến cho các doanh nghiệp sẽ xây dựng thêm hệ thống dự phòng, các nhà quản trị sẽ sử dụng DNS để luân chuyển lượng truy cập người dùng giữa các trung tâm dữ liệu khác nhau.

Tuy nhiên máy chủ DNS không thể đo được RTT thông qua liên kết giữa người dùng và máy chủ ứng dụng. Vì vậy khi chọn máy chủ ứng dụng, thì DNS không thể chọn dựa trên trạng thái mạng của liên kết. Ngoài ra máy chủ DNS không chọn máy chủ dịch vụ dựa vào vị trí. Do đó nếu người dùng ở Việt Nam thì có thể sẽ phải liên kết đến máy chủ được đặt ở Mỹ.

2.2. Mô hình GSLB

2.2.1. Khái niệm gslb

Global server load balancing (viết tắt : GSLB) là công nghệ điều hướng lưu lượng mạng tới một nhóm các trung tâm dữ liệu ở các vị trí địa lý khác nhau. Mỗi trung tâm dữ liệu trong nhóm cùng cung cấp các dịch vụ ứng dụng giống nhau, và lưu lượng của người dùng sẽ được điều hướng đến trung tâm dữ liệu có chất lượng đường truyền với người dùng là tối ưu nhất. GSLB theo dõi trạng thái của mỗi site, và điều hướng lưu lượng tới site với thời gian phản hồi tối ưu nhất.

GSLB cung cấp một số chức năng sau:

- Dự phòng : khi dịch vụ được cung cấp bởi trung tâm dữ liệu không phục vụ được dịch vụ thì gslb sẽ điều hướng người dùng sang site khả dụng.
- Phục hồi trung tâm dữ liệu tự động: khi một trung tâm dữ liệu bị lỗi dịch vụ thì gslb sẽ được điều hướng tự động người dùng sang trung tâm dữ liệu khả dụng.
- Cân bằng tải: lưu lượng sẽ được tối ưu hóa bằng cách phân tải giữa các trung tâm dữ liệu khả dụng, điều này giúp cho giảm độ trễ và là cho dịch vụ được truyền đi nhanh hơn.
- Cải thiện độ trễ: lưu lượng ứng dụng của người dùng được điều hướng đến máy chủ thực, không cần phải chuyển sang các ứng dụng thông qua hệ thống cân bằng tải.

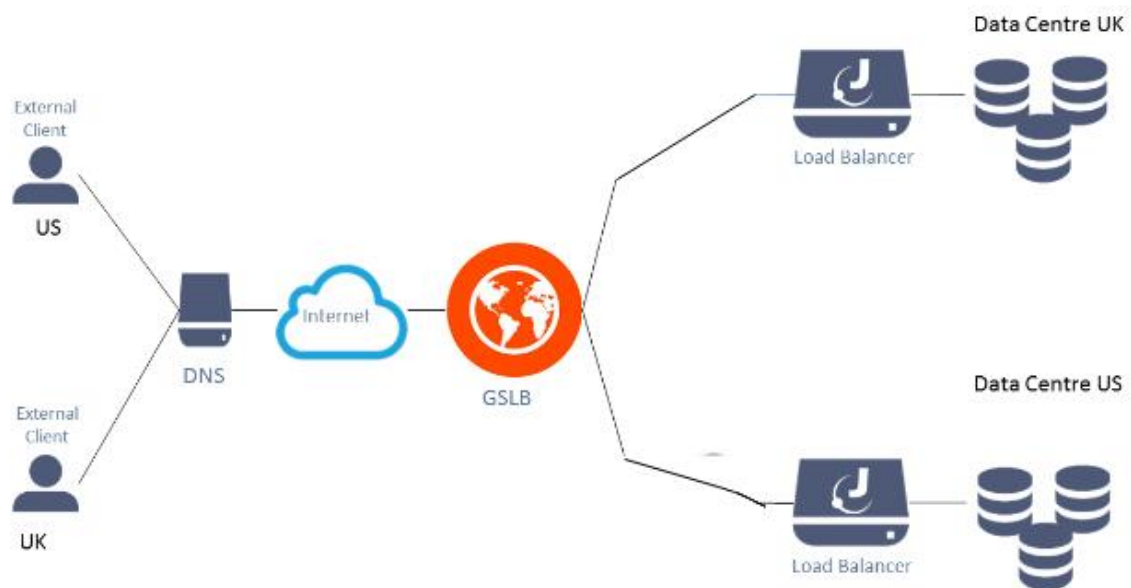
2.2.2. Kiến trúc và cách thức hoạt động của GSLB

GSLB là cơ chế cân bằng tải dựa trên giải pháp DNS, do đó khi sử dụng GSLB tốc độ đường truyền sẽ nhanh hơn và đáng tin cậy hơn vì GSLB sử dụng giao thức UDP và phản hồi người dùng gần như là bằng thời gian thực.

Vì lưu lượng client được định tuyến đến máy chủ dựa vào địa chỉ IP được nhận từ DNS, nên dịch vụ DNS có thể được mở rộng để tự động điều phối người dùng đến bất kỳ số lượng các trung tâm dữ liệu nào đang lưu trữ dịch vụ. Hệ thống GSLB hoạt động như authoritative DNS máy chủ và có thể điều hướng client traffic đến bất kỳ trung tâm dữ liệu nào mà lưu trữ dịch vụ ứng dụng.

Khi client truy vấn đến địa chỉ DNS, hệ dùng. Mỗi truy vấn DNS của người dùng được phản hồi bởi GSLB và những phản hồi được cung cấp cho client. Hệ thống GSLB có thể cung cấp chức năng của cân bằng tải máy chủ thông qua vị trí địa lý dữ liệu của các trung tâm dữ liệu ở bất kỳ đâu trên thế giới.

Ví dụ: Các máy chủ chứa các dịch vụ của *www.example.com* giống nhau được đặt ở các vị trí khác nhau. Máy chủ được đặt ở Anh và máy chủ còn lại được đặt ở Mỹ.



Hình 2.4. Mô phỏng GSLB

Bước 1: Người dùng gửi truy vấn đến local ISP DNS về địa chỉ IP của *www.example.com*. Nếu trong cache của local ISP DNS có chứa địa chỉ IP của site trên thì sẽ trả về cho người dùng. Nếu không có local ISP DNS sẽ chuyển tiếp truy vấn DNS của người dùng đến hệ thống GSLB. Lúc này hệ thống GSLB đóng vai trò như một hệ thống chứa các authoritative name máy chủ trong hệ thống DNS.

Bước 2: GSLB sẽ tính toán site đang hoạt động tối ưu nhất cùng vị trí với người dùng. Sau đó gửi trả lời truy vấn chứa địa chỉ IP của site tốt nhất cho local ISP DNS. Bằng cách, GSLB lưu lại địa chỉ địa lý của local ISP DNS, do local ISP DNS được đặt cùng một vị trí địa lý của người dùng. Từ đó GSLB sẽ suy ra vị trí địa lý của người dùng thông qua vị trí của local ISP DNS.

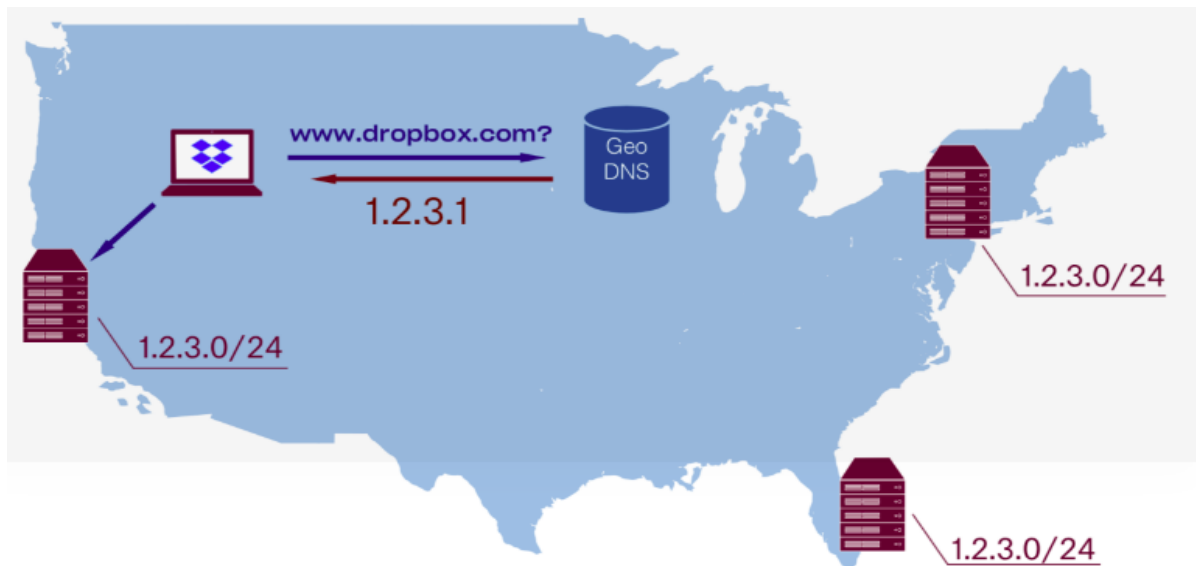
Bước 3: Bằng cách sử dụng thuật toán cân bằng tải, các chính sách về điều khiển lưu lượng và vị trí của người dùng, GSLB có thể điều hướng người dùng đến site chứa nội dung của www.example.com có trạng thái tối ưu nhất.

2.2.3. Giải pháp xây dựng GSLB

1. Giải pháp BGP anycast

Anycast là một cơ chế mạng cho phép nhiều máy chủ được triển khai tên toàn cầu cung cấp cùng một dịch vụ có thể chia sẻ cùng một địa chỉ IP. Dựa

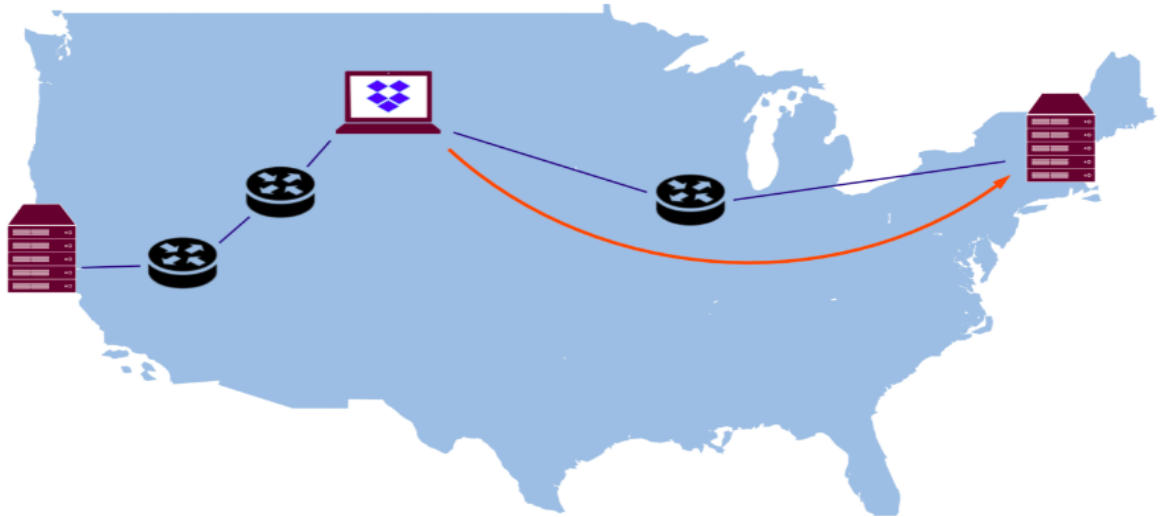
vào vị trí của yêu cầu người dùng, các router gửi gói tin yêu cầu đó đến một máy trong mạng mà ở gần người dùng nhất. Điều này giúp giảm độ trễ và tiết kiệm được chi phí băng thông, cải thiện thời gian tải cho người dùng và tăng tính sẵn sàng. Nếu một trung tâm chứa dữ liệu mà người dùng cần bị mất kết nối thì địa chỉ IP đã được anycast sẽ chọn tuyến đường tốt nhất cho người dùng và tự động chuyển hướng họ đến trung tâm dữ liệu gần nhất tiếp theo.



Hình 2.5. Mô hình sử dụng anycast với BGP[14]

Anycast liên kết với BGP, một giao thức định tuyến mạng tiêu chuẩn, để giúp người dùng có thể định tuyến gói tin đến địa chỉ trung tâm dữ liệu chứa nội dung dịch vụ gần nhất với họ mà đang hoạt động

Mặc dù BGP lựa chọn tuyến đường tối ưu nhất, nhưng BGP không phát hiện được độ trễ liên kết, thông lượng, gói tin mất,... Thông thường có rất nhiều tuyến đường có thể đến đích nhưng BGP sẽ chỉ chọn tuyến đường có ít hop nhất.

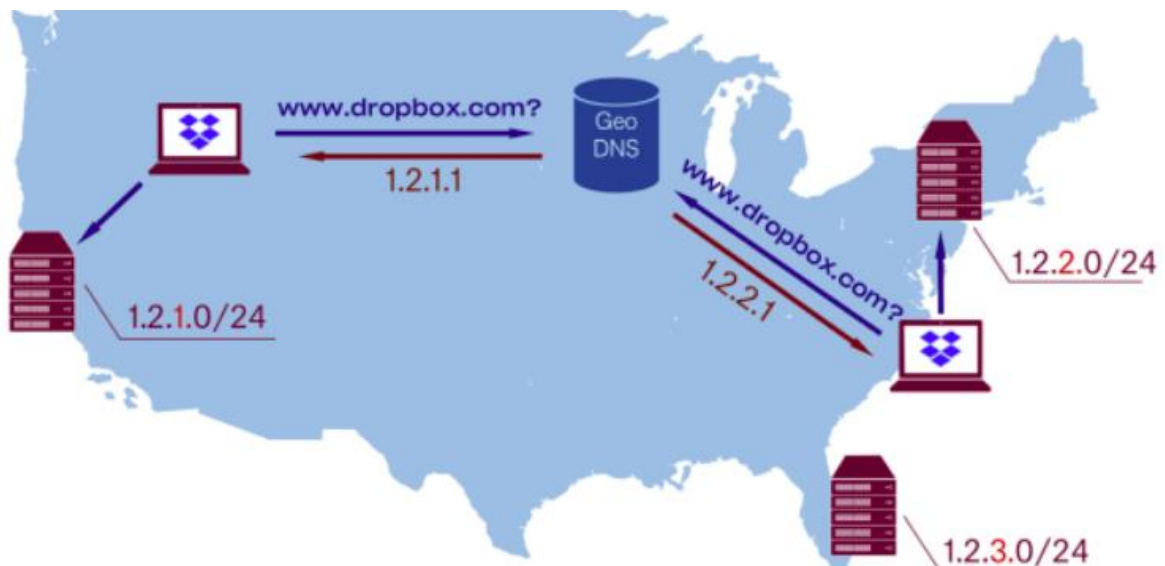


Hình 2.6. Anycast với BGP[14]

Cân bằng tải sử dụng anycast tối ưu nhất nhưng nó lại hoạt động kém nhất.

2. Giải pháp GeoDNS[7]

Một trong những giải pháp phổ biến nhất cho việc triển khai GSLB là GeoDNS. Với giải pháp này, mỗi trung tâm dữ liệu có riêng một không gian địa chỉ IP *unicast* và DNS sẽ chịu trách nhiệm xử lý các địa chỉ IP khác nhau đến những người dùng ở các vị trí địa lý khác nhau.



Hình 2.7. Mô hình GSLB sử dụng GeoDNS[14]

Phương pháp GeoDNS giúp nhà quản trị hệ thống có thể điều khiển lưu lượng hoạt động và cho phép giảm số lượng gói tin yêu cầu đến hệ thống lưu trữ dữ liệu mà không làm gián đoạn dịch vụ của người dùng. Phương pháp được thiết lập dựa vào unicast nên việc xử lý sự cố trở nên đơn giản hơn.

Ý tưởng của GeoDNS là lấy một gói tin yêu cầu DNS đến, thực hiện tra cứu vị trí địa lý tại thời điểm yêu cầu và trả về các kết quả khác nhau dựa trên địa chỉ IP đến. Theo [7], cách tiếp cận này đã được thực hiện bởi một số máy chủ DNS, như bind-geoDNS, powerDNS và tinyDNS. GeoDNS cho phép các trang web như Kernel.org, WikIPedia và nhiều ứng dụng khác có nội dung dịch vụ ở nhiều nơi có thể hướng người dùng đến một máy chủ tối ưu phù hợp. Điều này giúp phân phối tải yêu cầu của người dùng, giảm tải áp lực lên hệ thống trong một trung tâm dữ liệu.

Để cung cấp khả năng cân bằng tải giữa các trung tâm dữ liệu khác nhau một cách đơn giản mà không cần thay đổi phức tạp với các máy của người dùng hoặc các giao thức tùy chỉnh thì sử dụng GeoDNS là phương pháp tối ưu. DNS là một hệ thống cần có trong mạng internet, nó dùng để chuyển đổi địa chỉ tên miền sang địa chỉ IP. Mỗi máy người dùng đều có khả năng truy vấn DNS khi kết nối vào internet, để chuyển đổi chuỗi văn bản như loadbalance.com thành địa chỉ số, 208.77.188.166. Về cơ bản GeoDNS có các tính năng giống DNS, nhưng khác ở điểm là nó phản hồi dựa vào địa chỉ địa lý của người dùng.

GeoDNS kiểm tra địa chỉ IP của gói tin yêu cầu từ người trong cơ sở dữ liệu mà nó có, để xác định vị trí địa lý của IP gói tin yêu cầu đó. Sau đó, GeoDNS sẽ sinh ra gói tin phản hồi chứa địa chỉ IP của dịch vụ mà người dùng yêu cầu tương ứng với vị trí đã xác định trước đó, và gửi trả lại cho người dùng.

Tuy nhiên GeoDNS có một số giới hạn đó là cơ sở dữ liệu chứa địa chỉ IP tương ứng với vị trí địa lý không đảm bảo hoàn toàn chính xác. Để trở nên chính xác hơn, việc khai thác dựa trên nhiều dữ liệu là cần thiết, nhưng ngay cả những cơ sở dữ liệu sẵn có tốt nhất cũng chỉ có độ chính xác 99,8%, 11 có nghĩa là trong số 4,2 tỷ địa chỉ tiềm năng, khoảng 8,5 triệu địa chỉ không chính xác. [11]

2.3. Các phần mềm cân bằng tải GSLB

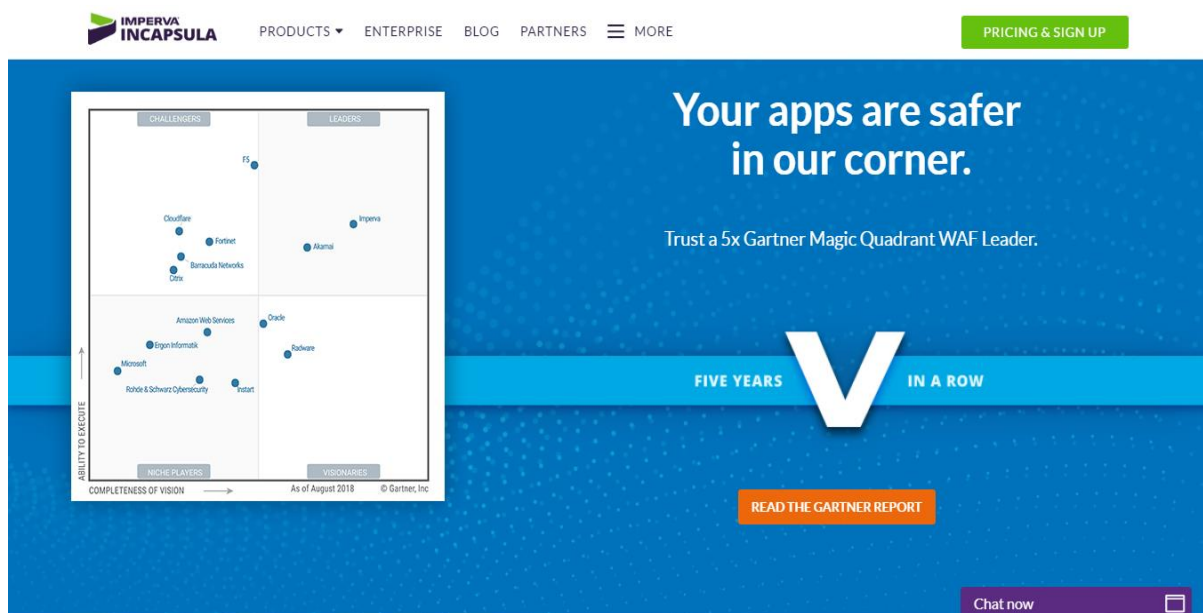
Hiện nay, có nhiều phần mềm có chức năng GSLB đã được triển khai. Phần này sẽ giới thiệu một số phần mềm giải pháp thông dụng hiện nay. Tuy nhiên các phần mềm này hiện nay có tính phí và chi phí để sử dụng nó khá cao.

2.3.1. Incapsula

Incapsula cung cấp dịch vụ cân bằng tải và chuyển đổi dựa trên đám mây cho phép khả năng mở rộng nhanh chóng và tiết kiệm chi phí mà không cần cài đặt phần cứng hoặc phần mềm. Dịch vụ này có thể được sử dụng trong triển khai vật lý, đám mây và máy chủ kết hợp.

Dịch vụ cân bằng tải của Incapsula cung cấp tính linh hoạt và chức năng tích hợp để hỗ trợ cân bằng tải máy chủ, cân bằng tải máy chủ toàn cầu (GSLB) và chuyển đổi giữa các trung tâm dữ liệu. Bằng cách cân bằng lưu lượng truy cập trực tiếp từ đám mây, Incapsula cho phép GSLB gần như ngay lập tức và khả năng chuyển đổi dự phòng mà không yêu cầu một thiết bị cục bộ hoặc ảo hóa.

Từ quan điểm của người dùng, Incapsula cung cấp chương trình theo dõi sức khỏe hệ thống máy chủ theo thời gian thực, như là bảng điều khiển, điều chỉnh độ nhạy và cảnh báo để giúp bạn chủ động xác định và khắc phục các vấn đề trước khi chúng có tác động trên trang dịch vụ web của người dùng.



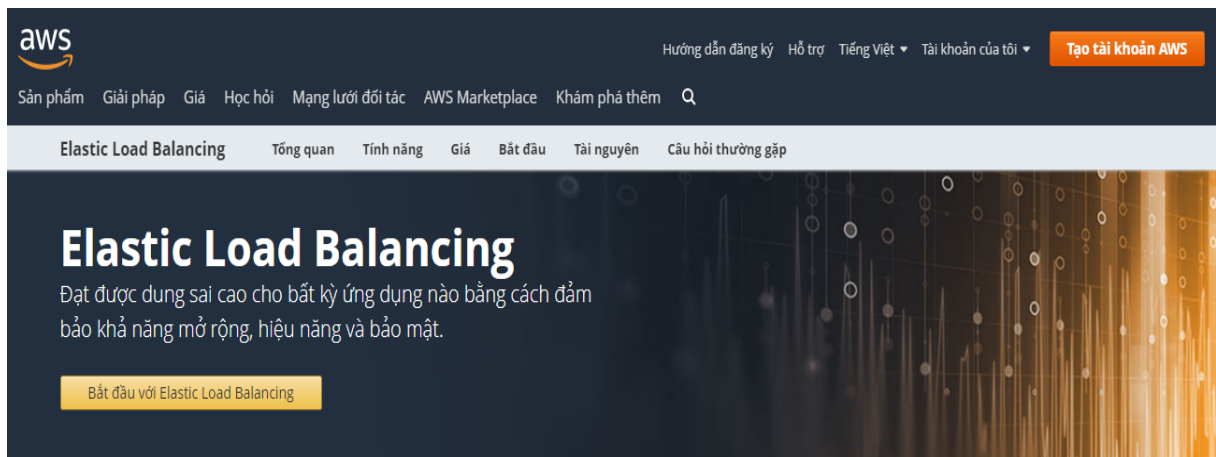
Hình 2.8. Trang chủ dịch vụ INCAPSULA

2.3.2. Amazon Elastic Load Balancing

Amazon Elastic Load Balancing (ELB) là một giải pháp rất tiện lợi cho các công ty có máy chủ ảo trên dịch vụ đám mây EC2 của Amazon. ELB là một cân bằng tải ảo phân phối lưu lượng truy cập ứng dụng đến trên nhiều cá thể Amazon EC2 trong một hoặc nhiều vùng sẵn có.

ELB phát hiện các trường hợp EC2 Amazon có trạng thái sức khỏe không tốt và định tuyến lại lưu lượng truy cập vào trung tâm dữ liệu đang ở trong trạng thái tốt. Một lợi thế quan trọng của ELB là tính mở rộng của nó. Khi doanh nghiệp bổ sung khả năng xử lý để đáp ứng nhu cầu lưu lượng ứng dụng, ELB sẽ cân bằng liên tục để hỗ trợ các máy chủ ảo mới.

Dịch vụ DNS Route 53 của Amazon là bắt buộc đối với GSLB hoặc các tình huống chuyển đổi dự phòng. Route 53 cung cấp các tính năng kiểm tra sức khỏe và chuyển đổi DNS, cho phép nhà quản trị hệ thống chạy các ứng dụng trong nhiều vùng AWS để tăng cường tính khả dụng.

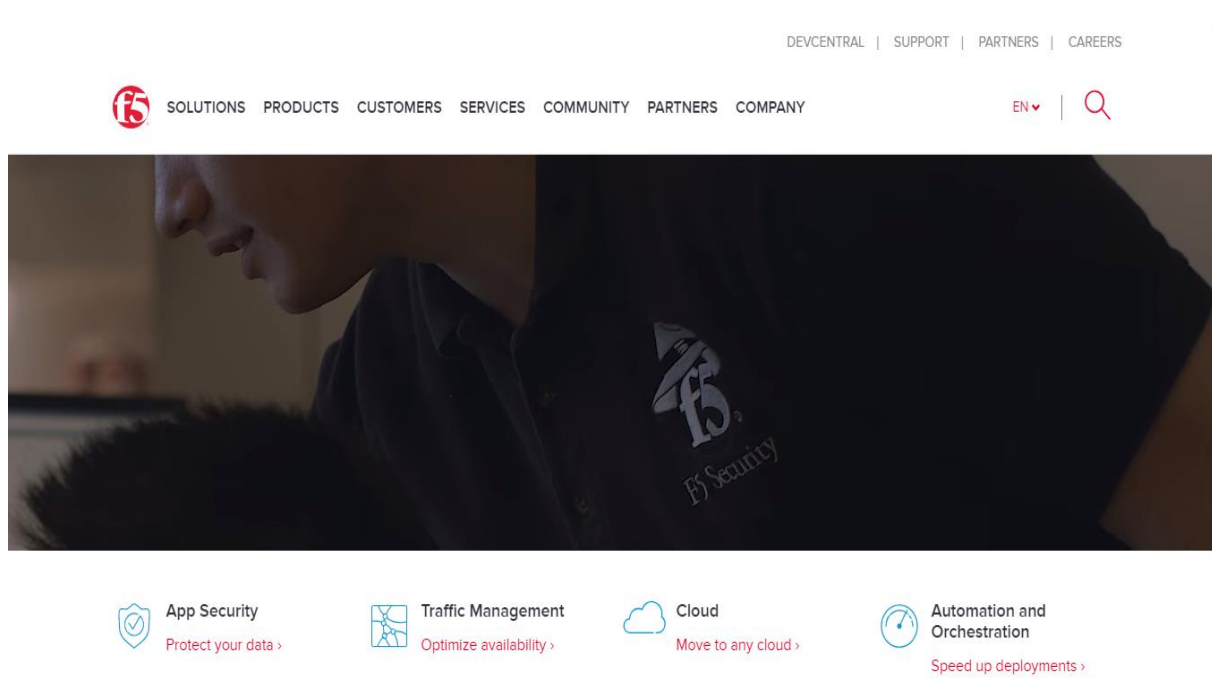


Elastic Load Balancing tự động phân phối lưu lượng truy cập đến của ứng dụng cho nhiều mục tiêu, chẳng hạn các máy ảo Amazon EC2, container và địa chỉ IP. Elastic Load Balancing có thể xử lý các tải lưu lượng truy cập khác nhau của ứng dụng của bạn trên một Vùng sẵn sàng hoặc trên nhiều Vùng sẵn sàng khác nhau. Elastic Load Balancing cung cấp ba loại bộ cân bằng tải, tất cả đều có độ khả dụng cao, tự động điều chỉnh quy mô và khả năng bảo mật mạnh mẽ cần thiết để giúp cho ứng dụng của bạn có được dung sai cao.

Hình 2.9. Trang chủ dịch vụ Amazon ELB

2.3.3. F5 Networks

F5 được sử dụng bởi nhiều doanh nghiệp CNTT lớn nhất thế giới. Thiết bị cân bằng tải dựa trên thiết bị của nó được thiết kế để tăng cường khả năng mở rộng ứng dụng và tính khả dụng cho các doanh nghiệp duy trì các máy chủ vật lý và trung tâm dữ liệu của riêng họ.



Hình 2.10. Trang chủ của F5 networks

F5 cung cấp các thiết bị riêng biệt cho cân bằng tải máy chủ cục bộ và toàn cầu. Trình quản lý lưu lượng truy cập cục bộ (cân bằng tải cục bộ) hỗ trợ cân bằng tải giữa các máy chủ, đồng thời theo dõi sức khỏe và hiệu suất của các máy chủ riêng lẻ trong thời gian thực.

Trình quản lý lưu lượng truy cập toàn cầu GSLB của nó là một máy chủ DNS hiệu năng cao với khả năng cân bằng tải toàn cầu. Sử dụng phương pháp GSLB, nó cho phép bạn định tuyến lưu lượng truy cập theo địa lý trên nhiều trung tâm dữ liệu. Nếu trung tâm dữ liệu bị hỏng, trình GSLB sẽ hướng người dùng của bạn đến trung tâm dữ liệu gần nhất hoặc hoạt động tốt nhất.

2.4. Kết luận

Kết thúc chương, báo cáo đã giới thiệu khái quát về bộ cân bằng tải, chỉ ra việc cần thiết đối với cân bằng tải giữa các trung tâm dữ liệu. Từ đó tìm hiểu một số giải pháp cân bằng GSLB phổ biến thông dụng hiện tại. Trên cơ sở đó, dựa vào yêu cầu bài toán cụ thể, báo cáo sẽ tìm kiếm, lựa chọn giải pháp phù hợp để triển khai, cải tiến hệ thống cân bằng tải Polaris-GSLB.

CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG MÃ NGUỒN MỞ POLARIS-GSLB

Chương trước đã trình bày về kiến thức trong hệ thống cân bằng tải, chương này sẽ cài đặt và thực nghiệm cụ thể vào thực tế sử dụng công cụ Polaris-GSLB. Tìm ra các vấn đề mà Polaris-GSLB chưa giải quyết được.

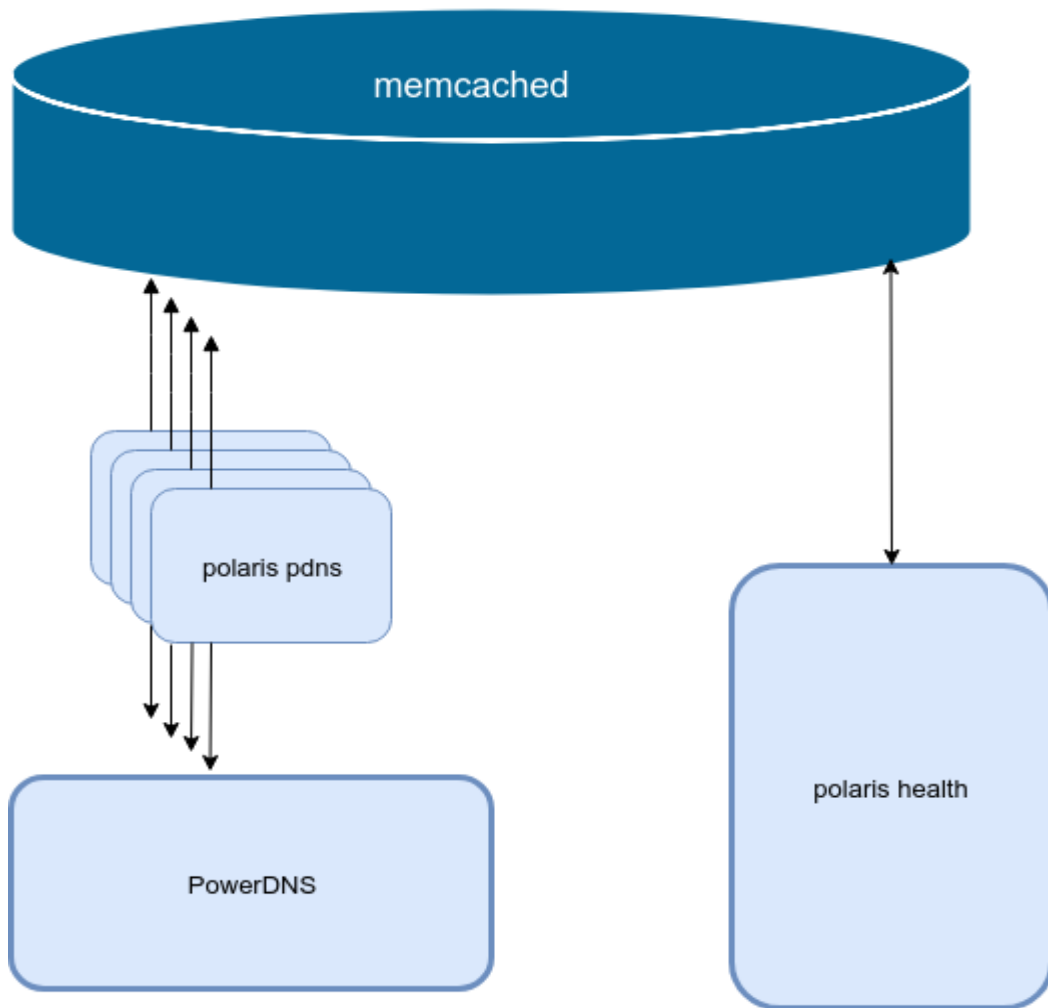
3.1. Giới thiệu

Sau khi định nghĩa được mô hình GSLB và giải pháp xây dựng nó, ta cần áp dụng mô hình cân bằng tải đó một cách hiệu quả. Để làm được điều đó, chúng ta sẽ nghiên cứu và sử dụng open source Polaris-GSLB và tiến hành phát triển công cụ trên Polaris-GSLB để cho thể giảm tải và đảm bảo chất lượng đường truyền tốt nhất.

Chương này sẽ mô tả cụ thể về kiến trúc công cụ được phát triển, giới thiệu về Polaris-GSLB và các chức năng quan trọng mà Polaris-GSLB đem lại trong việc đảm bảo chất lượng dịch vụ của các trung tâm dữ liệu, đảm bảo lưu lượng trạng thái của site được hoạt động ở mức tối ưu nhất. Nhằm đem lại những trải nghiệm tốt cho người dùng.

Ngoài ra Polaris-GSLB còn giúp các nhà cung cấp dịch vụ giải quyết được nỗi lo về tính khả dụng của dữ liệu, độ trễ đường truyền khi các nhà cung cấp có nguyện vọng mở rộng hệ thống ra tầm cỡ quốc tế.

3.2. Mô tả polaris-gslb



Hình 3.1. Mô tả Polaris

Polaris là chương trình mã nguồn mở, được xây dựng mở rộng cho giải pháp GSLB, quản lý lưu lượng dựa vào các dịch vụ của hệ thống DNS. Về cơ bản Polaris-GSLB đóng vai trò là thành phần máy chủ xác thực tên miền trong hệ thống DNS, và bao gồm:

1. Polaris-health: làm nhiệm vụ lấy từ điển cấu hình được xác định trong các vùng và kết hợp với các thông số kiểm tra tính sẵn sàng của các trung tâm dữ liệu, để xây dựng lên bảng trạng thái sức khỏe. Chương trình Polaris-health kiểm tra theo dõi tình trạng của trung tâm dữ liệu và cập nhật bảng được lặp lại theo định kỳ. Thông tin trạng thái được truyền vào bộ nhớ được chia sẻ giữa các tiến trình (memcached). Chạy như một daemon.
2. Polaris-PDNS - PowerDNS remote backend json-api plugin: thực hiện phân phối DNS truy vấn dựa vào tính sẵn sàng của các trung tâm dữ liệu (được đồng bộ định kỳ từ memcache) và thuật toán cân bằng tải được chọn.

3. PowerDNS (PDNS): là phần mềm mã nguồn mở DNS máy chủ xác thực. PDNS cung cấp hiệu năng tốt và giảm yêu cầu bộ nhớ tối thiểu. PDNS sẽ chứa tệp thông tin DNS cho tên miền website và sẽ trả lời truy vấn cho người dùng với thông tin trực tiếp từ tệp thông tin của nó.

4. Memcached: là một hệ thống lưu trữ bản sao các đối tượng và dữ liệu được truy cập nhiều lần để tăng tốc độ truy xuất.

3.3. Cấu hình chương trình[1]

File cấu hình cân bằng tải **polaris-lb.yaml** bao gồm 2 phần:

- pools: Định nghĩa các trung tâm dữ liệu, dùng để cấu hình vùng của các nó hoặc trả lời và thiết lập các thông số kiểm tra tính sẵn sàng và một số tùy chọn khác.
- globalnames: Định nghĩa tên miền.

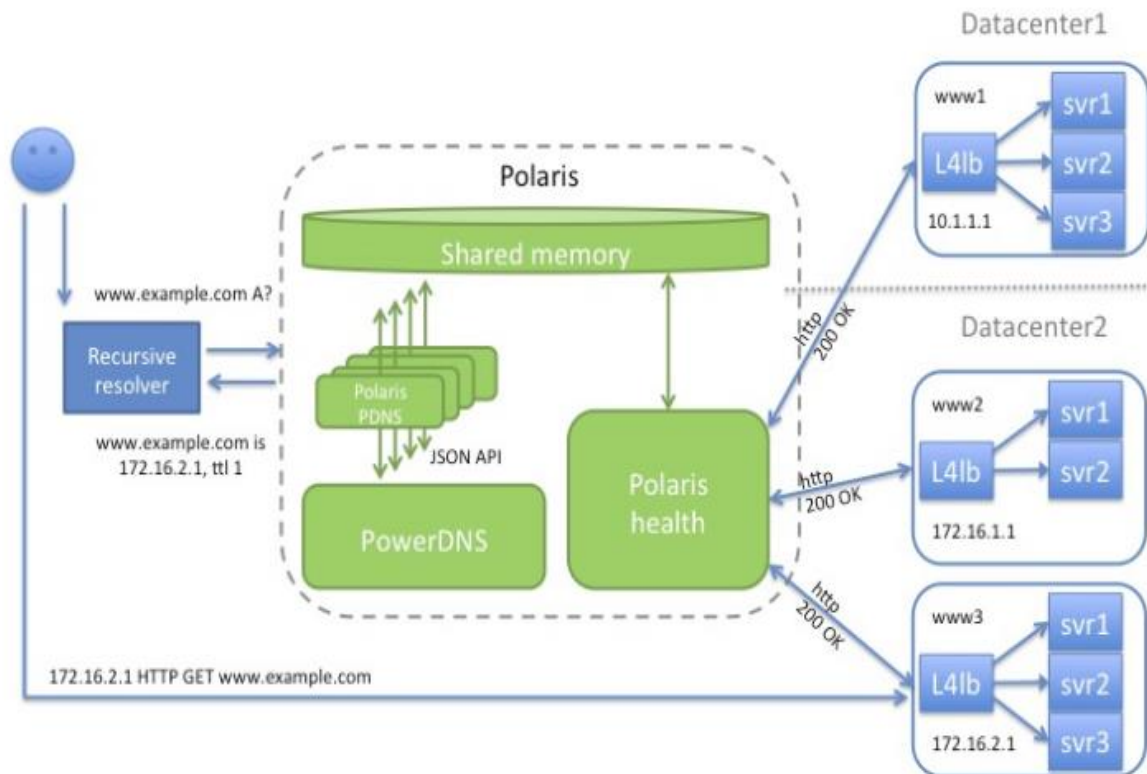
```
globalnames:
  www.loadbalance.com:
    pool: www-loadbalance
    ttl: 1
pools:
  www-loadbalance:
    monitor: http
    monitor_params:
      hostname: www.loadbalance.com
    lb_method: twrr
    fallback: any
    max_addrs_returned: 2
    members:
      - IP: 1.1.2.11
        monitor_IP: 1.1.2.15
        name: www1-dc1
```

```
weight: 1
- IP: 1.1.2.12
name: www2-dc1
monitor_IP: 1.1.2.15
weight: 1
- IP: 2.2.1.11
name: www1-dc2
weight: 1
```

- Khi tất cả máy chủ được bật được định nghĩa trong file cấu hình polaris-lb.yaml phải được khớp với cấu hình vị trí subnet ở trong cấu hình topology sử dụng file polaris-topology.yaml.

```
location1:
- 1.1.2.0/24
- 3.3.3.0/24
- 10.0.0.0/8
location2:
- 4.4.1.0/16
- 2.2.1.11/32
- 172.16.254.0/16
- 192.168.0.0/24
```

3.4. Nguyên lý hoạt động



Hình 3.2. Mô hình GSLB

Polaris-GSLB xây dựng giải pháp cân bằng tải toàn cầu dựa vào DNS, vì vậy cách hoạt động cũng giống như mô hình DNS.

Người dùng gửi gói tin truy vấn DNS đến là recursive-resolver, nó có thể được thiết lập hoạt động trong cùng một vùng với người dùng. Recursive-resolver sẽ tìm trong bộ nhớ đệm của nó địa chỉ IP của `www.loadbalance.com`. Nếu không có thì recursive-resolver sẽ chuyển tiếp gói tin đến PowerDNS đóng vai trò là máy chủ xác thực tên.

PowerDNS chuyển tiếp gói DNS dạng tài nguyên SOA hoặc A truy vấn qua pipe đến các backend phía sau nhờ sự hỗ trợ của Polaris-PDNS. Polaris-PDNS nhận lấy gói tin và bóc tách yêu cầu của người dùng về dạng json.

Ví dụ: người dùng yêu cầu địa chỉ IP của “`www.loadbalance.com`” loại A. Tập thông tin DNS chứa dữ liệu được yêu cầu bởi người dùng theo dạng

“www.loadbalance.com IN A” thì sẽ được Polaris-PDNS đưa về dạng json như sau:

```
{
  "method": "lookup",
  "parameters": {
    "local": "0.0.0.0",
    "qname": "www.loadbalance.com",
    "real-remote": "1.1.2.11/32",
    "qtype": "SOA",
    "remote": "1.1.2.11",
    "zone-id": -1
  }
}
```

Trong đó, các thông số trong 'parameters' là:

- local: địa chỉ IP của người dùng
- Real remote: địa chỉ của www.loadbalance.com được lưu trữ trong máy chủ xác thực tên.
- qtype: dạng thông tin yêu cầu đến DNS
- remote: địa chỉ IP của www.loadbalance.com

Sau khi Polaris-PDNS phân tích xong gói tin truy vấn DNS, thì lấy dữ liệu từ memcached để đóng gói gói tin DNS. Dữ liệu luôn được cập nhật bởi Polaris-health.

Polaris-health luôn chạy ngầm để kiểm tra tính sẵn sàng của các trung tâm dữ liệu, nhằm mục đích xác định trung tâm nào sẵn sàng có thể xử lý yêu cầu bằng cách gửi một gói tin yêu cầu đến các trung tâm cần kiểm tra. Với phản hồi mà mỗi trung tâm dữ liệu trả về được đánh giá để xác định trạng thái sức khỏe của mỗi trung tâm đó.

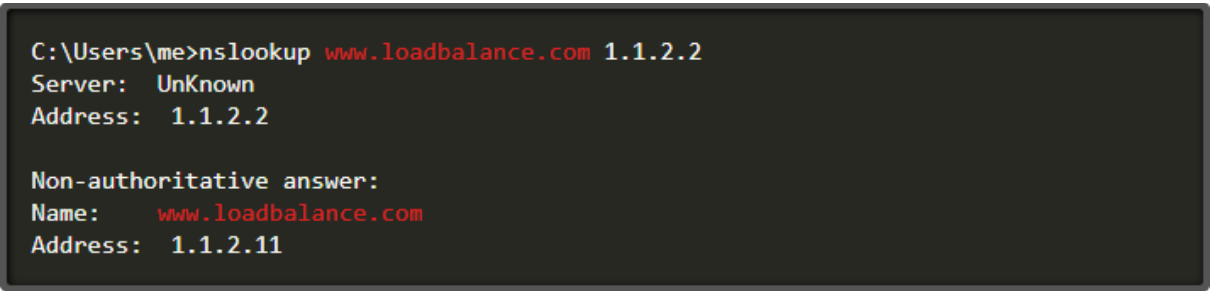
Nếu các trung tâm dữ liệu được kiểm tra sức khỏe vượt qua các thông số đo kết nối HTTP và TCP thì quá trình kiểm tra hoàn thành. Phản hồi trả về từ các

theo dõi cho việc kiểm tra sức khỏe sử dụng giao thức HTTP thành công nếu polaris-health nhận được phản hồi *HTTP 200 (OK)* và phản hồi được trả về trước thời gian theo dõi kết thúc.

Polaris-health kiểm tra tính khả dụng của kết nối TCP được thực hiện bằng cách gửi yêu cầu kết nối TCP đến cổng trung tâm dữ liệu cần kiểm tra sức khỏe. Nếu phiên bắt tay ba bước được hoàn thành trước khi thời gian theo dõi hết hạn thì trạng thái kết nối TCP của trung tâm dữ liệu đó tốt. Polaris-health sẽ nhận được thông tin gửi về giống với thông tin phản hồi mà nó mong muốn.

Sau khi xác định được trạng thái của các trung tâm dữ liệu. Polaris-health sẽ cập nhật thông tin trạng thái vào memcached. Polaris-PDNS sẽ lấy thông tin từ memcached, xác định thuật toán cân bằng tải đã được cấu hình trong file polaris-lb.yaml để tạo ra gói tin DNS phản hồi và gửi phản hồi cho người dùng dưới dạng thông tin phản hồi DNS thông qua PowerDNS và Recursive-resolver.

Theo [1], ví dụ:



```
C:\Users\me>nslookup www.loadbalance.com 1.1.2.2
Server:      UnKnown
Address:     1.1.2.2

Non-authoritative answer:
Name:        www.loadbalance.com
Address:     1.1.2.11
```

Hình 3.3. Kết quả truy vấn “www.loadbalance.com” của Polaris-GSLB

3.5. Vấn đề của Polaris-GSLB

Theo nguyên lý hoạt động của Polaris-GSLB, thì kết quả phản hồi DNS trả về cho người dùng là địa chỉ của các trung tâm dữ liệu mà có phản hồi HTTP và TCP mong muốn đến polaris-health trước thời hạn theo dõi kết thúc.

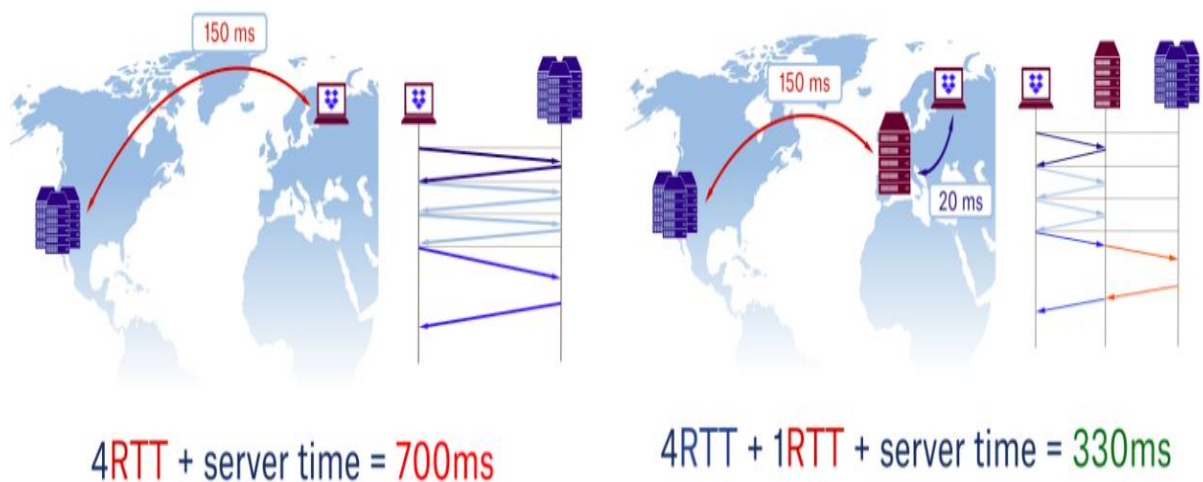
Qua đó, cho thấy polaris-PDNS có hai vấn đề chưa giải quyết được trong việc kiểm tra tính sẵn sàng của hệ thống cung cấp dữ liệu đó là :

1. Độ trễ phản hồi từ người dùng đến trung tâm dữ liệu:

Polaris-GSLB trả về cho người dùng địa chỉ của trung tâm dữ liệu đang hoạt động theo cơ chế *weighted round robin*, nhưng không tính toán khoảng cách giữa người dùng và trung tâm dữ liệu đó. Một nghiên cứu của Dropbox

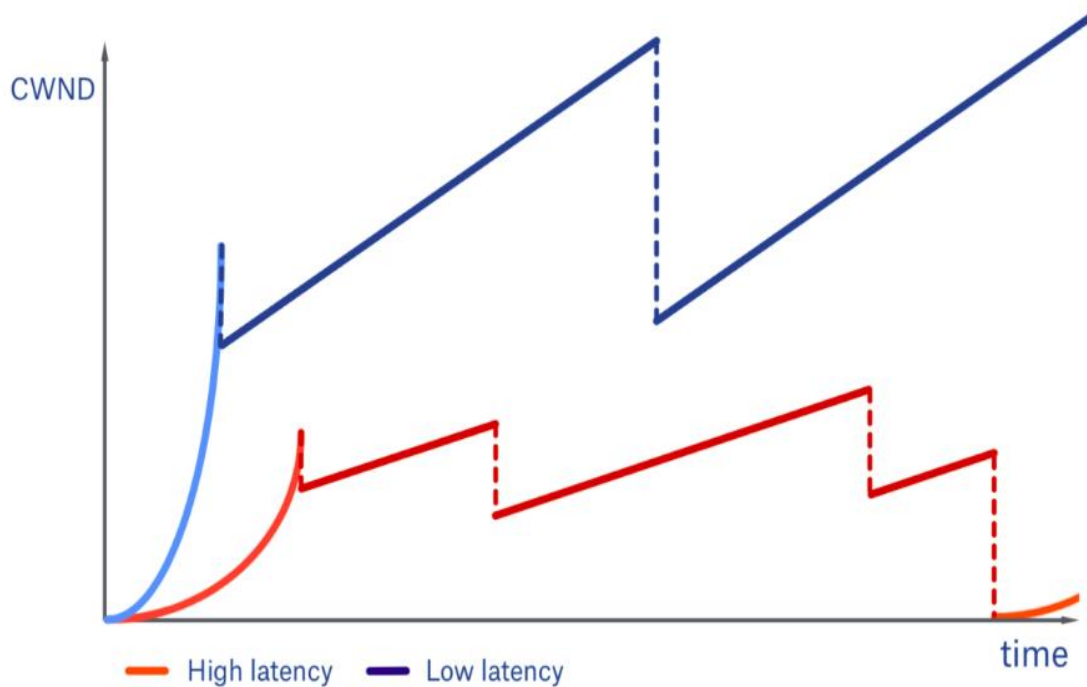
[14] cho thấy việc điều hướng người dùng đến trung tâm dữ liệu gần với người dùng thì sẽ cải thiện được độ trễ. Đây là một ví dụ so sánh độ trễ của một gói tin yêu cầu HTTP được điều hướng đến thẳng trung tâm dữ liệu và một gói tin tương tự đến trung tâm khác được đặt gần với người dùng. Số liệu thu được dựa trên:

- Từ trung tâm dữ liệu đến người dùng và ngược lại có độ trễ 20ms.
- Từ trung tâm dữ liệu được nhân bản đến trung tâm dữ liệu chính có độ trễ là 150ms
- 100ms thời gian máy chủ thực hiện.



Hình 3.4. Độ trễ truyền dữ liệu theo khoảng cách người dùng đến trung tâm dữ liệu[14]

Qua việc so sánh trên, có thể thấy việc điều hướng người dùng đến trung tâm dữ liệu gần với người dùng hơn sẽ giúp cải thiện độ trễ hơn. Ngoài ra, người dùng sẽ được hưởng lợi nhiều hơn từ việc tải xuống và tải lên nhanh hơn. Một phép so sánh khác để chứng minh độ trễ ảnh hưởng đến việc tăng TCP congestion window trong thời gian tải



Hình 3.5. TCP congestion window

Hình 3.5 là hình ảnh thể hiện trạng thái kết nối TCP, kết nối có độ trễ thấp tương ứng với người dùng kết nối đến trung tâm dữ liệu nhân bản được đặt gần với người dùng và độ trễ cao tương ứng với trung tâm dữ liệu chính. Qua đó cho thấy:

- Máy người dùng có độ trễ thấp hơn đang di chuyển nhanh hơn do phiên *slow start* nhanh hơn.
- Người dùng này cũng đang ở ngưỡng cao hơn để tránh tắc nghẽn, vì khả năng mất gói tin thấp hơn. Điều này xảy ra bởi vì các gói tin đó không cần phải mất quá nhiều thời gian hơn ở mạng *public*, do có thể bị tắc nghẽn trên đường truyền.

2. Hiệu năng xử lý yêu cầu của các trung tâm dữ liệu.

Polaris không kiểm tra được tính khả dụng của các trung tâm dữ liệu. Việc kiểm tra khả năng xử lý yêu cầu của các trung tâm dữ liệu rất quan trọng. Nếu như có hàng ngàn yêu cầu của nhiều người dùng đến cùng một trung tâm dữ liệu tại cùng một thời điểm, khả năng máy chủ bị lỗi có thể xảy ra làm gián đoạn dịch vụ của người. Điều đó sẽ làm trải nghiệm người dùng với dịch vụ không tốt.

3.6. Kết luận

Nội dung chương đã mô tả và nêu ra những yêu cầu, nguyên lý hoạt động của hệ thống chương trình mã nguồn mở Polaris-GSLB. Trên cơ sở đó, nêu ra vấn đề mà Polaris-GSLB chưa giải quyết được trong bài toán thực tế.

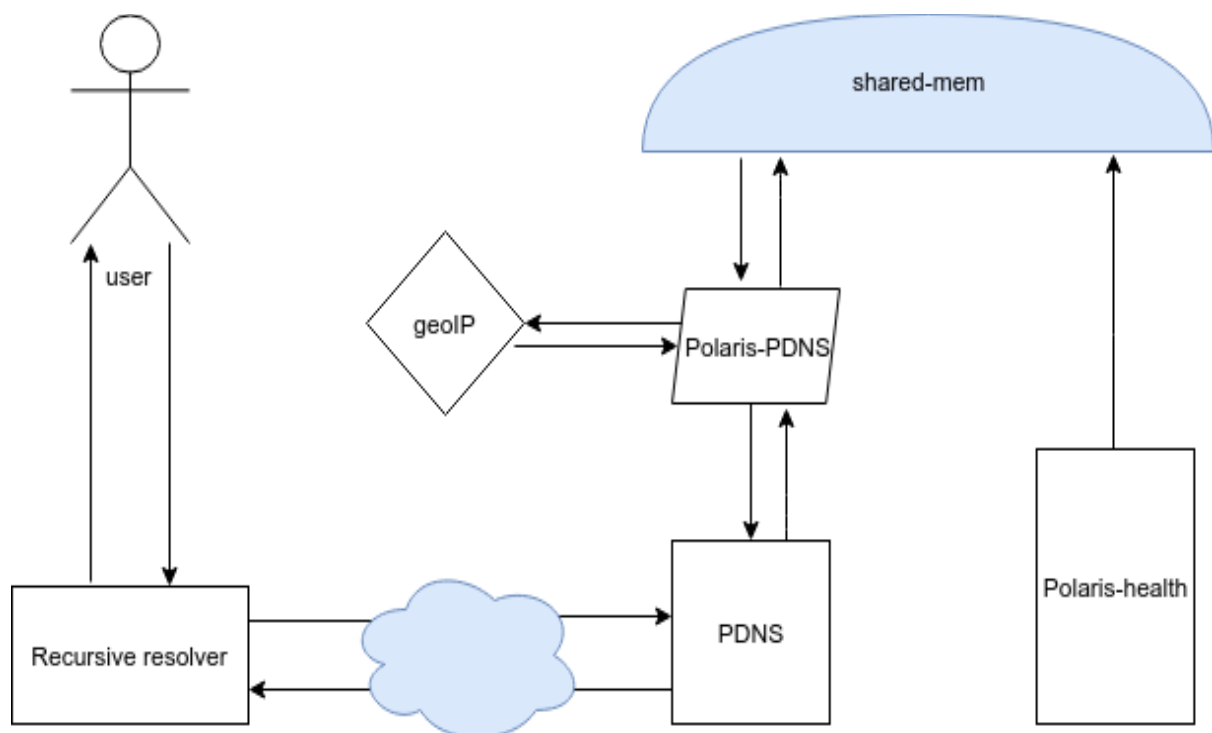
Kết quả trả về của Polaris-GSLB là địa chỉ IP gắn với máy chủ xác thực tên miền, chứa Polaris-GSLB, nhưng điều này không giải quyết vấn đề được nêu ở [3.5].

CHƯƠNG 4. THỬ NGHIỆM VÀ ĐÁNH GIÁ HỆ THỐNG CÂN BẰNG TẢI POLARIS-GSLB CẢI TIẾN

Chương này sẽ trình bày các bước phân tích yêu cầu phía người dùng để xây dựng hệ thống với những yêu cầu cấu hình phù hợp nhất. hực nghiệm hệ thống Polaris-GLSB đã cải tiến nhằm áp dụng các phương pháp kiểm tra vào thực tế thông qua công cụ Polaris-GLSB để đem lại hiệu quả trong việc thiết kế kiểm tra và áp dụng các hệ thống cân bằng tải vào thực tế.

4.1. Cải tiến Polaris-GSLB

Như đã nêu ở mục 2.4.2, Polaris-GSLB có nhiều hạn chế để có thể ứng dụng cân bằng tải GSLB cho các hệ thống trung tâm dữ liệu được phân phối khắp mọi nơi. Giải pháp mà tôi đề xuất cải tiến là bổ sung tính năng GeoIP và cải thiện khả năng truy xuất địa chỉ subnet mà người dùng và trung tâm dữ liệu cùng thuộc một nhà cung cấp mạng ở vị trí gần nhất.



Hình 4.1. Mô hình Polaris-GSLB sử dụng giải pháp geoIP

- Polaris-GSLB lấy địa chỉ IP của người dùng trong gói tin truy vấn DNS

mở rộng được gửi từ *recursive resolver*, sau đó sử dụng module Polaris-GeoIP để tìm vị trí người dùng dựa vào địa chỉ IP.

- Polaris-GeoIP sử dụng thuật toán *cuckoo hashing* để tìm vùng mạng mà người dùng thuộc, được lưu trong tệp chứa danh sách các subnet của các nhà cung cấp mạng đã tổng hợp. Sau đó trả tên vùng mà người dùng thuộc cho Polaris-PDNS.

- Polaris-PDNS sẽ dựa vào tên vùng mạng của người dùng, tìm kiếm thông tin các nốt ở trong trung tâm dữ liệu thuộc vùng mạng đó. Sau đó ngẫu nhiên chọn địa chỉ của một nốt đang hoạt động trả về cho người dùng trong gói tin DNS phản hồi.

4.1.2. Ứng dụng phương pháp geoDNS

4.1.2.1. Sử dụng phương pháp EDNS

Năm 2011, Google đã viết một bản nháp IETF để gửi thông tin IP của người dùng bằng cách sử dụng EDNS0 và đây được gọi là eDNS-client-subnet. Máy chủ DNS sẽ sử dụng địa chỉ IP của người dùng để đưa ra quyết định về việc phản hồi vì vậy người dùng có thể được kết nối đến máy chủ tối ưu hơn. Tiêu chuẩn này được tán thành do việc này khiến internet nhanh hơn và được một số nhà cung cấp hàng đầu áp dụng.

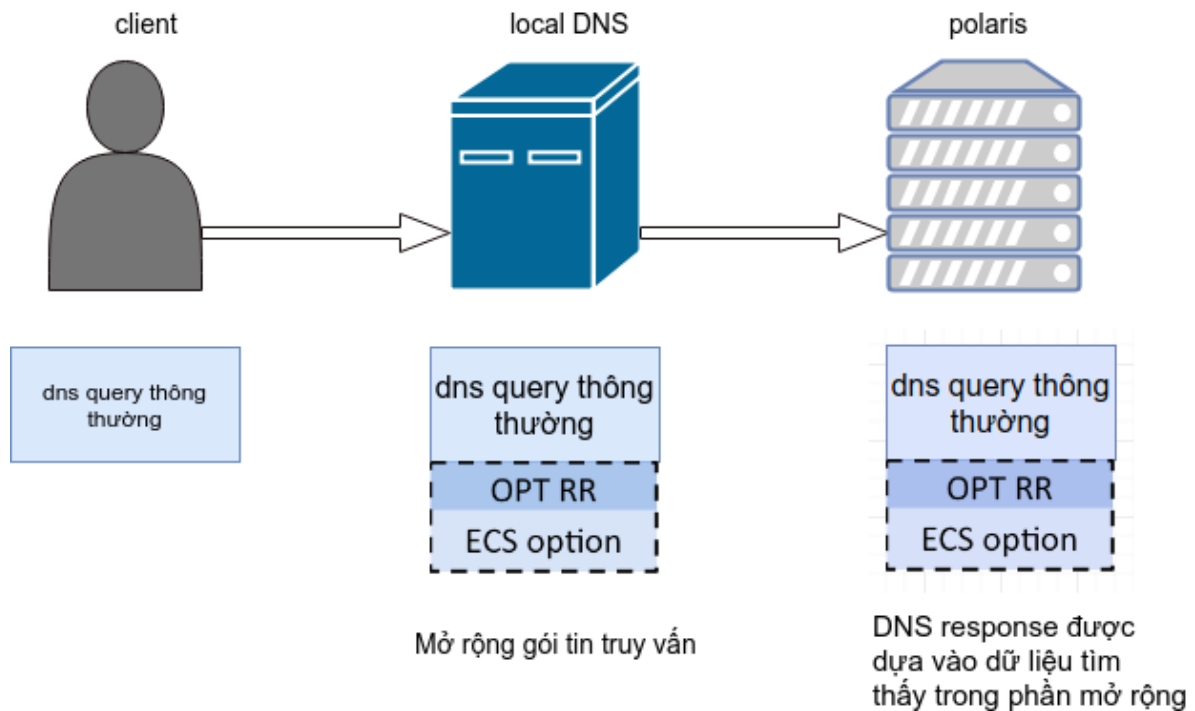
Thiết kế ban đầu của DNS hạn chế gói tin có kích thước tổng cộng là 512 byte, điều này giúp bảo vệ gói tin DNS tránh khỏi bị tấn công mạng. EDNS (extend DNS) sẽ giải quyết việc mở rộng gói tin mà vẫn đảm bảo tính an toàn dữ liệu cho hệ thống và người dùng bằng cách: nếu cả 2 máy chủ DNS hỗ trợ EDNS, chúng có thể trao đổi các gói tin lớn hơn 512 byte, nếu không, chúng sẽ quay trở lại DNS thông thường.

Vì nó được thiết kế để giữ quyền riêng tư, người dùng có quyền tự do giới hạn thông tin IP của client. Thay vì gửi địa chỉ IP đầy đủ, DNS máy chủ sẽ gửi một phần thông tin - subnet.

Ví dụ: Nếu địa chỉ IP client là 66.214.81.22, DNS máy chủ sẽ chỉ hiển thị 3 octet đầu tiên là 66-214-81. Do được trang bị địa chỉ IP thực của thiết bị truy cập, máy chủ DNS có thể đưa ra phản hồi chính xác hơn.

Áp dụng vào chương trình Polaris-GSLB: Sau khi nhận được gói tin truy vấn DNS từ PDNS thì Polaris-PDNS sẽ phân tích gói tin truy vấn dưới dạng json và lấy thông tin địa chỉ IP của người dùng trong phần mở rộng của gói tin để tìm kiếm nhà cung cấp mạng của người dùng. Sau đó lấy thông tin của trung

tâm dữ liệu trong memcached để tạo gói tin phản hồi DNS trả về cho người dùng.



Hình 4.2. Phương pháp EDNS

4.1.2.2. Ứng dụng thuật toán cuckoo hashing cho việc tìm kiếm

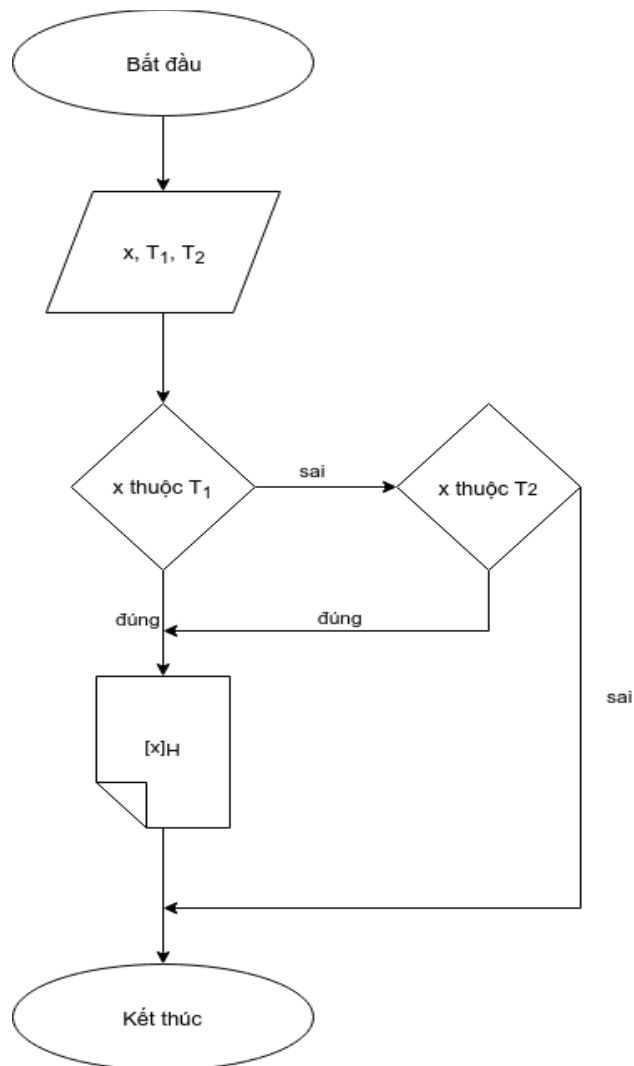
Cuckoo hashing được Pagh and Rodler giới thiệu vào năm 2001. Cuckoo hashing là phương pháp băm địa chỉ mở để giải quyết xung đột khóa trong bảng băm, cho phép tra cứu và tính năng xóa trong trường hợp xấu nhất là $O(1)$, và thời gian thêm giá trị trong trường hợp xấu là $O(n)$.

Cuckoo hashing sử dụng hai bảng băm T_1 và T_2 , mỗi bảng có r buckets với hai hàm băm độc lập h_1 và h_2 dùng để ánh xạ dữ liệu đến mảng các bucket $\{0, \dots, r-1\}$. Khóa x có thể được lưu trong một trong hai vị trí $T_1[h_1(x)]$ và $T_2[h_2(x)]$. Trong cuckoo hashing, việc tra cứu sẽ thực hiện tìm kiếm trong cả hai vị trí $T_1[h_1(x)]$ và $T_2[h_2(x)]$ và tìm kiếm thành công nếu khóa x được lưu trong một trong hai vị trí được tìm kiếm trong hai bảng băm T_1 và T_2 . Tra cứu trong cuckoo hashing mất thời gian $O(1)$ vì chỉ có hai vị trí được kiểm tra. Thuật toán sau mô tả cuckoo hashing thực hiện tra cứu:

```
function lookup(x)
    return  $T_1[h_1(x)] = x \vee T_2[h_2(x)] = x$ 
```

end

Ví dụ: Cuckoo hashing thực hiện việc tra cứu $x = 32$. h_1 và h_2 ánh xạ giá trị x đến các bucket trong bảng T_1 và T_2 . Đầu tiên sẽ tìm kiếm vị trí $h_1(x)$ trong bảng T_1 . Nếu x được tìm thấy trong bảng T_1 thì việc tìm kiếm sẽ được dừng, còn nếu không sẽ chuyển sang tìm kiếm trong bảng T_2 . Nếu tìm kiếm trên cả hai bảng đều không có x thì kết luận không tìm thấy x .



Hình 4.3. Lưu đồ tìm kiếm sử dụng cuckoo hashing

Với chức năng chèn, cuckoo hashing sẽ đẩy khóa khác đã có chỗ sẵn ra và bấm lại cho đến khi tất cả các khóa có chỗ của mình. Nếu x được thêm vào, đầu tiên cuckoo hash sẽ kiểm tra xem ô $h_1(x)$ của bảng T_1 đã có giá trị chưa. Nếu không thì x chèn vào $h_1(x)$. Ngược lại, x sẽ được gán vào $T_1[h_1(x)]$, và khóa y trước sẽ được thêm vào T_2 . Điều này có thể xảy ra tiến trình lặp, số lần

lặp lại được giới hạn bởi giá trị “MaxLoop”. Nếu số lần lặp đạt mức tối đa, tất cả khóa sẽ được băm lại với hàm băm mới.

Thiết kế của cuckoo hashing là sử dụng hai hàm băm thay vì chỉ một. Điều này giúp mỗi khóa có thể có hai vị trí để lưu trữ. Để sử dụng khả năng của bảng băm tối ưu và tốc độ chèn của cuckoo hashing sẽ sử dụng nhiều hơn hai hàm băm thay thế có thể được mong đợi.

Cuckoo hashing có một số tính chất:

- Dễ triển khai.
- Tra cứu sử dụng hai thăm dò.
- Hiệu quả trong trường hợp trung bình.

4.1.2.3. Tìm IP của trung tâm dữ liệu gần với người dùng

- Thêm toàn bộ dữ liệu subnet gồm địa chỉ IP và tên vùng của các nhà cung cấp mạng vào bảng băm sử dụng thuật toán cuckoo hashing.
- Đổi địa chỉ IP dạng CIDR x.x.x.x/y về dạng số nguyên.
- Tính khoảng cách giữa địa chỉ IP người dùng với các IP trong bảng băm. Trả về tên vùng có địa chỉ khoảng cách nhỏ nhất với địa chỉ người dùng đó.
- Tìm địa chỉ IP của trung tâm dữ liệu dựa vào tên vùng của người dùng.
- Trả về địa chỉ IP của máy chủ thuộc trung tâm dữ liệu đang hoạt động

Ví dụ: Người dùng có địa chỉ 42.112.192.10/24. Và danh sách dữ liệu bao gồm các subnet của nhà cung cấp mạng như sau:

42.112.128.0/17	fpt-02
183.91.0.0/19	cmc-01
203.196.24.0/22	vt-02

Bước 1: Đầu tiên đổi địa chỉ IP sang dạng số nguyên:

IP người dùng	42.112.192.10/24	712032266
fpt-02	42.112.128.0/17	712015872

cmc-01	183.91.0.0/19	3076194304
vt-02	203.196.24.0/22	3418626048

Bước 2: Tính khoảng cách giữa IP người dùng và subnet của các trung tâm dữ liệu trong danh sách:

Trung tâm dữ liệu	712015872	3076194304	3418626048
Người dùng	fpt-02	cmc-01	vt-02
712032266	16394	2364162038	2706593782

Vì khoảng cách giữa IP người dùng và subnet của fpt-02 nhỏ nhất. Nên người dùng thuộc vùng mạng với fpt-02.

Bước 3: Tìm kiếm IP chứa nội dung ứng dụng mà người dùng yêu cầu dựa vào tên vùng của người dùng fpt-02. Và trả về địa chỉ IP máy chủ thuộc trung tâm dữ liệu fpt-02.

Trong trường hợp không có máy chủ nào phục vụ ứng dụng được yêu cầu hoặc máy chủ trong trung tâm dữ liệu cùng ISP với người dùng bị lỗi thì geoIP sẽ chọn ngẫu nhiên máy chủ đang hoạt động chứa ứng dụng đó.

4.1.3. Sử dụng redis thay memcached

Redis là một phần mềm mã nguồn mở được dùng để lưu trữ một cách tạm thời trên bộ nhớ (hay còn gọi là cache data) và giúp truy xuất dữ liệu một cách nhanh chóng. Do tốc độ truy xuất dữ liệu vượt trội so với các cơ sở dữ liệu thông thường như MySQL nên redis được sử dụng rất nhiều trong kỹ thuật Caching, nhân bản để mở rộng hiệu năng đọc, và phân mảnh dữ liệu phía khách hàng để tăng hiệu năng viết.

Theo [8], redis là hệ thống lưu trữ khóa - giá trị với rất nhiều tính năng. Redis nổi bật bởi việc hỗ trợ nhiều cấu trúc dữ liệu cơ bản (hash, list, set, sorted set, string), đồng thời cho phép scripting bằng ngôn ngữ lua. Bên cạnh lưu trữ khóa - giá trị trên RAM với hiệu năng cao, redis hỗ trợ lưu lâu dài trong bộ nhớ, và còn hỗ trợ lưu trữ dữ liệu trên đĩa cứng (persistent redis) cho phép phục hồi dữ liệu khi gặp sự cố. Ngoài tính năng nhân bản (sao chép giữa master-slave), tính năng cluster (sao lưu master-master) cũng đang được phát

triển. Bên cạnh đó, redis hỗ trợ ghi data vào đĩa tự động bằng 2 cách khác nhau, và có thể lưu trữ data trong 4 cấu trúc ngoài các chuỗi khóa đơn giản như memcached lưu trữ. Đây cũng là điểm nổi bật của redis.

Redis lưu trữ dạng cơ sở dữ liệu như NoSQL. Do đó trong redis, không có bảng, ko có định nghĩa dữ liệu hoặc cách ràng buộc của dữ liệu này liên quan đến dữ liệu khác trong redis. Giống như memcached ("dạng lưu trữ khóa - giá trị trong bộ nhớ"), redis cũng lưu trữ ánh xạ khóa - giá trị và có thể đạt được mức độ hiệu suất như memcached.

Khác với memcached, vì memcached đơn giản nhất, chỉ có dạng khóa-giá trị, tất cả dữ liệu lưu trong RAM. Và memcached chỉ là tầng cache, ko có khả năng dự phòng, tức là khó backup dữ liệu, và dữ liệu có thể mất.

Redis giải quyết nhiều vấn đề và cho phép redis có thể sử dụng cả cơ sở dữ liệu chính hoặc phụ với các hệ thống lưu trữ khác nhau.

Redis sử dụng cả lưu trung gian lưu trữ bản chính và bản phụ cho dữ liệu, hỗ trợ nhiều trường hợp và nhiều loại kiểu truy vấn.

Do đó Redis mạnh hơn, phổ biến và được hỗ trợ tốt hơn memcached. Memcached chỉ có thể làm một phần nhỏ những thứ Redis có thể làm, ngay cả khi có các tính năng của chúng giống nhau.

Mặc dù redis chỉ xử lý đơn luồng, nhưng rất khó để CPU bị nghẽn cổ chai. Thiết kế đơn luồng làm cho nó rất ổn định và hiệu quả.

4.2. Mô hình hệ thống thử nghiệm

Mô hình thử nghiệm gồm 1 máy chủ chạy hệ thống cân bằng tải Polaris-GSLB. Cấu hình cụ thể được ghi ở bảng sau:

Cấu hình	Polaris-GSLB
Hệ điều hành	Centos 7.5.1804
CPU	1 core
RAM	2GB
Disk	20GB

IP	171.244.36.155
Ports	PDNS_máy chủ : 53

Các chương trình trong hệ thống chiếm ít tài nguyên của máy chủ, nên ta chọn máy chủ có cấu hình như trên.

4.3. Cài đặt và cấu hình Polaris-gslb

Bước 1: Cập nhật hệ điều hành

```
# yum install -y wget
# wget
https://raw.githubusercontent.com/jokerbui/tools/master/centos7\_update\_os.sh
# sh centos7_update_os.sh
# reboot
```

Bước 2: Cài đặt powerDNS

```
// PowerDNS install
# yum install pdns pdns-backend-pipe
```

```
# systemctl enable pdns
```

Bước 3: cài đặt redis

```
# yum install redis python-redis
# systemctl enable redis
# systemctl start redis
# systemctl status redis
```

Bước 4: Cấu hình powerDNS

Chỉnh sửa file /etc/pdns/pdns.conf

```
setuid=pdns
```

```
setgid=pdns
launch=pipe
pipe-command=/etc/pdns/polaris/bin/pdns
cache-ttl=0
negquery-cache-ttl=0
pipe-backend-abi-version=3
edns-subnet-processing=yes
query-cache-ttl=
local-port=53
```

Bước 5: đẩy source code vào thư mục /etc/pdns/

```
# chown pdns. /var/log/polaris.log
# chmod +x /etc/pdns/polaris/bin/pdns
```

Bước 6: khởi động dịch vụ pdns

```
# systemctl start pdns
# systemctl status pdns
```

Bước 7: Kiểm tra pdns

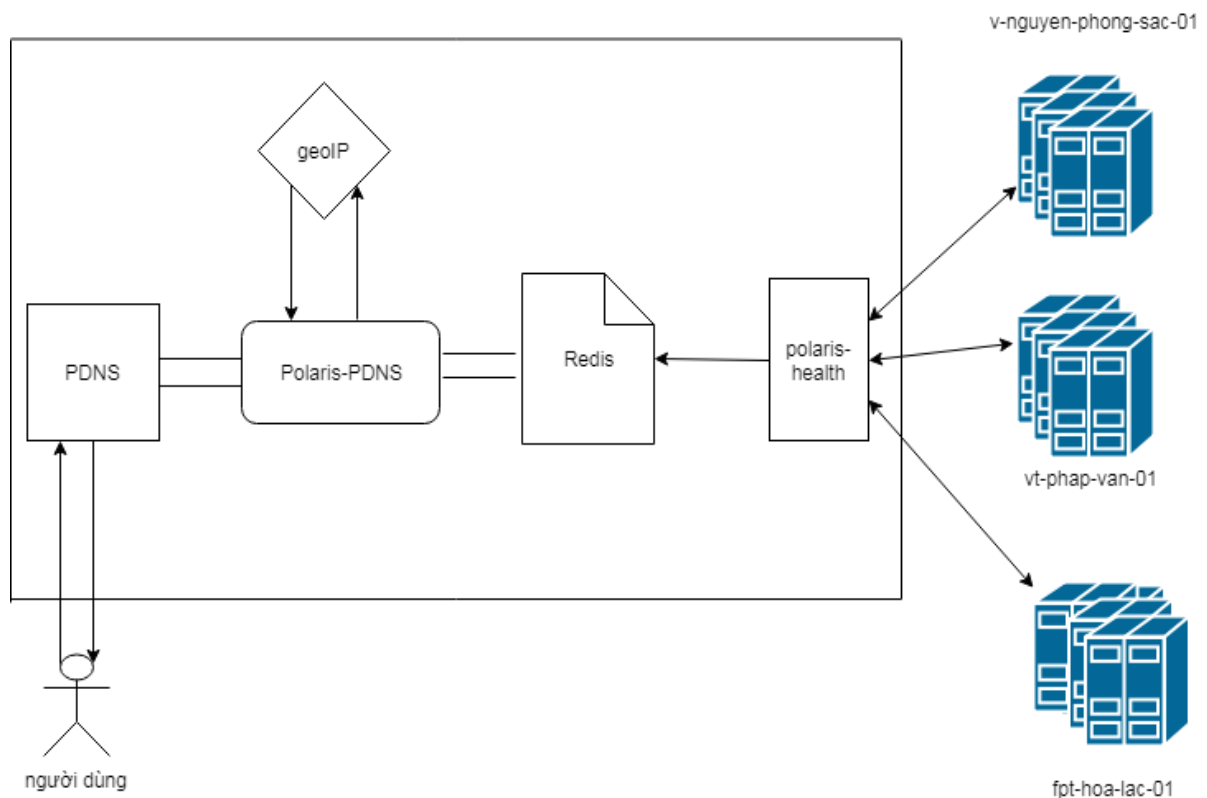
Cài đặt gói bind-utils

```
# yum install bind-utils
```

Kiểm tra truy vấn đến domain

```
# nslookup vod.vpdns-dev.vws.vegacdn.vn
```

4.4. Thử nghiệm hệ thống cân bằng tải Polaris-GSLB cải tiến khi người dùng truy cập đến ứng dụng



Hình 4.4. Mô hình thử nghiệm hệ thống cân bằng tải Polaris-GSLB cải tiến

Tình huống:

Người dùng có địa chỉ IP:

DB-IP (04.12.2018)	
IP address	58.187.29.158
Host name	adsl-dynamic-pool-xxx.fpt.vn
IP range	58.187.27.0-58.187.63.255 CIDR
ISP	FPTDYNAMICIP
Organization	FPT Telecom Company

muốn truy cập vào website có tên miền là "vod.vpdns-dev.vws.vegacdn.vn" và nội dung của trang này được nhân bản ra các trung tâm dữ liệu khác nhau.

Thực hiện:

Trong máy chủ Polaris-GSLB, tiến hành cài đặt powerDNS và cấu hình DNS zone file về các thông tin của tên miền

"vpdns-dev.vws.vegacdn.vn"

\$TTL 180

```
@      IN      SOA      vpdns-dev.vws.vegacdn.vn. (
                2014080704      ; Serial Number (date YYYYMMDD++)
                86400      ; Refresh (24 hours)
                1800      ; Retry (1/2 hour)
                3600000      ; Expire (42 days)
                21600)      ; Minimum (6 hours)
      IN      NS      171.244.36.155.
```

Cấu hình DNS zone ở trên cho thấy, miền "vpdns-dev.vws.vegacdn.vn" có máy chủ xác thực tên ở địa chỉ "171.244.36.155". Ví dụ: Mỗi khi người dùng truy cập đến subdomain "vod.vpdns-dev.vws.vegacdn.vn" thì theo cơ chế DNS, nó sẽ được gửi truy vấn đến "ns1.vpdns-dev.vws.vegacdn.vn" có địa chỉ "171.244.36.155".

Khi nhận được gói tin truy vấn từ người dùng, Polaris-pDNS sẽ phân tích gói tin này để lấy thông tin địa chỉ của người dùng và thông tin domain người dùng yêu cầu.

```
Q      vod.vpdns-dev.vws.vegacdn.vn      IN      ANY      -1
      113.190.252.218      0.0.0.0      113.190.252.218/32
```

113.190.252.218/32 : IP người dùng

vod.vpdns-dev.vws.vegacdn.vn: domain yêu cầu

Module "geoIP" sẽ tìm kiếm tên ISP mà IP người dùng thuộc trong cơ sở dữ liệu sử dụng thuật toán cuckoo hashing và thuật toán geoIP đã được đề xuất. Sau khi tìm thấy ISP, module này sẽ tìm trong dữ liệu trong "topology.yaml", file chứa thông tin của các subdomain. Ví dụ như sau:

```
pools:
  vod:
    uplinks:
      - name: vdc-nguyen-phong-sac-01
      - name: vt-phap-van-01
      - name: fpt-hoa-lac-01
    members:
```

```
- IP: 103.216.123.119
  name: vdc-nguyen-phong-sac-01
  az: vdc-01
- IP: 171.244.36.151
  name: vt-phap-van-01
  az: vt-01
- IP: 103.216.120.212
  name: fpt-hoa-lac-01
  az: vt-01
```

dns:

```
vod.vpdns-dev.vws.vegacdn.vn:
  pool: vod
  ttl: 1
```

Đồng thời kiểm tra trạng thái của các "uplink", lấy thông tin từ redis cache do Polaris-health up lên. Nếu các phản hồi "uplink" tương ứng với ISP của người dùng trả về đang hoạt động, thì phản hồi người dùng là địa chỉ IP đó. Nếu không thì sẽ ngẫu nhiên chọn một địa chỉ IP của những "uplink" đang hoạt động để trả về cho người dùng.

Kết quả:

IP người dùng thuộc ISP "fpt-01" và ở "fpt-01", địa chỉ IP có trạng thái "uplink" đang hoạt động, vì vậy IP tương ứng với domain là

```
laptop@laptop-Vostro-5470:~$ nslookup vod.vpdns-dev.vws.vegacdn.vn
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
Name:   vod.vpdns-dev.vws.vegacdn.vn
Address: 103.216.120.212
```

Hình 4.5. Kết quả truy vấn tên miền “vod.vpdns-dev.vws.vegacdn.vn”
Thực nghiệm sử dụng IP máy chủ proxy, kết quả được tổng hợp lại như sau:

IP	ISP	• DC
113.161.173.10:3128	VNPT	VDC
117.2.121.203:8080	VT	VT
115.78.160.247:8080	VT	VT
118.70.81.253:8080	FPT	FPT
123.31.20.75:3128	VNPT	FPT
117.2.121.203:8080	VT	VT
123.30.172.60:3128	VNPT	VDC
113.190.234.84:8080	VNPT	VDC
117.2.125.41:9090	VT	VT
14.161.25.161:8080	VNPT	VDC
203.205.29.106:52822	CMC	FPT

Nhận xét:

Kết quả truy vấn tên miền “vod.vpdns-dev.vws.vegacdn.vn” sau khi chương trình Polaris-GSLB sau khi đã được cải tiến bổ sung tính năng GeoIP là địa chỉ IP có cùng ISP với người dùng. Với Polaris-GSLB nguyên bản, khi truy vấn kết quả trả về là địa chỉ của máy chủ có trạng thái sức khỏe tốt theo cơ chế “weighted round-robin”, trả về về mã *HTTP 200*, gần với máy chủ xác thực tên miền chứa Polaris-GSLB, [3.4].

4.5. Kết luận

Trên những phân tích về nghiệp vụ và vấn đề phân luồng tải của hệ thống trong công ty cổ phần Bạch Minh (Vega corporation), nghiên cứu đã đưa ra giải pháp xây dựng và cải tiến mô hình cân bằng tải GSLB dựa vào mã nguồn mở Polaris-GSLB

Tất cả các thành phần của hệ thống đều là phần mềm mã nguồn mở, cho phép người dùng có khả năng tùy biến cao. Đây là vừa là thuận lợi vừa là thách thức đối với người thiết kế xây dựng hệ thống. Thuận lợi ở chỗ các công cụ mã nguồn mở đều cho phép tùy chỉnh phù hợp với yêu cầu hệ thống. Khó cũng chính là vì các công cụ này phụ thuộc rất nhiều vào cách sử dụng của người xây dựng hệ thống nên nếu bước phân tích yêu cầu chưa tốt, sẽ dẫn đến những cấu hình, chỉnh sửa sai sót, dẫn đến tình trạng sử dụng không hiệu quả tài nguyên hệ thống để cân bằng tải cũng như không đáp ứng được yêu cầu người dùng.

Báo cáo cũng đưa ra một số yêu cầu đối với các thành phần của hệ thống mà người dựng hệ thống phải lưu ý trước khi tiến hành cài đặt và vận hành.

Kết luận

Với sự bùng nổ của mạng internet hiện nay, dịch vụ ứng dụng được đặt khắp nơi, để giúp đảm bảo dịch vụ luôn hoạt động ở trạng thái tốt nhất, tránh gói tin người dùng yêu cầu bị trễ trên đường truyền, giảm áp lực lên hệ thống có mô hình tập trung thì việc các cơ quan, doanh nghiệp đang phát triển sản phẩm trên mô hình có khả năng mở rộng lớn, lượng truy cập cao thì hệ thống cân bằng tải là rất cần thiết.

Hệ thống cân bằng GSLB được xây dựng dựa trên bộ công cụ mã nguồn mở Polaris-GSLB, tiết kiệm chi phí, dễ dàng cài đặt và sử dụng là một giải pháp đầy hứa hẹn. Quá trình thử nghiệm với cấu hình máy chủ ở mức trung bình cho hiệu năng hoạt động tương đối tốt.

Những kết quả chính đã đạt được:

1. Nghiên cứu hệ thống cân bằng tải, cân bằng tải giữa các trung tâm dữ liệu cho các hệ thống dịch vụ lớn,
2. Tìm hiểu cơ chế áp dụng cho giải pháp cân bằng tải giữa các trung tâm dữ liệu
3. Dựa vào các yêu cầu thực tế, đề xuất giải pháp cải tiến hệ thống Polaris-GSLB đáp ứng được các yêu cầu về kiểm tra hiệu năng của hệ thống trung tâm dữ liệu và giảm độ trễ yêu cầu người dùng.

Hướng phát triển tiếp theo:

1. Phát triển các công cụ để tính toán thông lượng trên mỗi uplink, giúp đưa ra chính xác trạng thái theo từng thời điểm của các trung tâm dữ liệu.

Tài liệu tham khảo

Tiếng Anh

- [1] Aaron West, *I've finally found a decent reason to use GSLB!*, December 19, 2017.
- [2] Ali M. Alakeel, *A Guide to Dynamic Load Balancing in Distributed Computer Systems*, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.
- [3] Contavalli, Carlo. "RFC 7871: Client Subnet in DNS Queries" 2018-02-02.
- [4] Charles Chen, *An Overview of Cuckoo Hashing*.
- [5] F. Baker, *Requirements for IP Version 4 Routers*, RFC 1812, June 1995, sec. 4.2.3.1.
- [6] *Global server Load Balancing Scalability, High Availability and Performance for Distributed Networks*, array networks, 2017
- [7] John Hawley, *GeoDNS—Geographically-aware, protocol-agnostic load balancing at the DNS level*, Linux Foundation.
- [8] Josiah Carlson, *Redis in Action*, Foreword by Salvatore Sanfilippo.
- [9] Karl Seguin, *The Little Redis Book*, Free Book, 2012.
- [10] Li, *An Architecture for IP Address Allocation with CIDR*, RFC 1518, September 1993.
- [11] Melissa Anderson, *What is load balancing*, February 14, 2017.
- [12] Michael Mitzenmacher, *Cuckoo Hashing, Theory and Practice* (Part 1, Part 2 and Part 3), 2007.
- [13] Monika Kushwaha Pranveer, Saurabh Gupta, *Various Schemes of Load Balancing in Distributed Systems- A Review*, Pranveer Singh Institute of Technology Kanpur.
- [14] Oleg Guba and Alexey Ivanov, *Dropbox traffic infrastructure: Edge network*, October 10.
- [15] P. Mockapetris. "RFC 1035, Domain Names - Implementation and Specification", November 1987.

- [16] P. Vixie. “RFC 2671: Extension Mechanisms for DNS (EDNS0)”, The Internet Society, August 1999.
- [17] P.Vixie, M.Graff, J.Damas. “RFC 6891, Extension Mechanisms for DNS (EDNS(0))”, April 2013.
- [18] Rasmus Pagh, Flemming Friche Rodler. *Cuckoo Hashing. In Journal of Algorithms*, 2004, pp 122–144.
- [19] Singh, M. *Choosing Best Hashing Strategies and Hash Functions*, Thapar university-India, june 2009.
- [20] Sven Ingebrigt Ulland, *High-Level Load Balancing for Web Services*, university of oslo, Department of Informatics, 20th May 2006.