

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**HUỲNH NGUYỄN UYÊN NHI - NGUYỄN HUY HOÀNG - PHẠM NGỌC
THIỆN - HỒ CÔNG LONG**

BÁO CÁO ĐỒ ÁN MÔN HỌC

**ĐỀ TÀI
“TÌM HIỂU VÀ TRIỂN KHAI PHẦN MỀM HAPROXY ĐỂ CÂN BẰNG TẢI CHO
ỨNG DỤNG WEB VÀ DATABASE”**



TP. HỒ CHÍ MINH, 2023

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**HUỲNH NGUYỄN UYÊN NHI- 21522424
NGUYỄN HUY HOÀNG- 21522094
PHẠM NGỌC THIÊN - 21522627
HỒ CÔNG LONG - 21522297**

ĐỒ ÁN MÔN HỌC

**ĐỀ TÀI
“TÌM HIỂU VÀ TRIỂN KHAI PHẦN MỀM HAPROXY ĐỂ CÂN BẰNG TẢI CHO
ỨNG DỤNG WEB VÀ DATABASE”**

**MÔN HỌC: QUẢN TRỊ MẠNG VÀ HỆ THỐNG
LỚP: NT132.O11.ANTT**

**GIÁO VIÊN LÝ THUYẾT
ThS. Đỗ Hoàng Hiến**

TP HỒ CHÍ MINH, 2023

GIỚI THIỆU

Trong thời đại ngày nay, với sự bùng nổ của các dịch vụ trực tuyến và ứng dụng web, thách thức lớn đối với các hệ thống máy chủ là làm thế nào để duy trì hiệu suất cao và đảm bảo sự ổn định khi đối mặt với tải lưu lượng ngày càng tăng. Nhu cầu sử dụng Load Balancing đã trở thành một điều cần thiết, không chỉ để tối ưu hóa tài nguyên mà còn để đảm bảo trải nghiệm người dùng mượt mà và không gián đoạn.

Một trong những vấn đề chính của việc không triển khai cân bằng tải là tình trạng quá tải hoặc thiếu tải trên một số máy chủ, trong khi các máy chủ khác có thể còn dư thừa tài nguyên. Điều này không chỉ dẫn đến sự lãng phí nguồn lực mà còn làm giảm hiệu suất toàn bộ hệ thống. Ngoài ra, khi không áp dụng cân bằng tải, hệ thống cũng trở nên dễ bị tấn công và không ổn định hơn. Điều này mở ra cửa cho các hành động tấn công như tấn công từ chối dịch vụ (DDoS) hoặc thậm chí là việc lợi dụng các lỗ hổng an ninh để xâm nhập trái phép vào hệ thống. Áp dụng cân bằng tải chính là một giải pháp thông minh để đối mặt với những thách thức này và tối ưu hóa hoạt động của hệ thống.

Trong đồ án này, chúng tôi tập trung nghiên cứu đề tài *"Tìm hiểu và triển khai phần mềm HAProxy để cân bằng tải cho ứng dụng web và database"*. Được biết đây là một phần mềm cân bằng tải (load balancer) mã nguồn mở nổi tiếng và mạnh mẽ được sử dụng rộng rãi trong các môi trường máy chủ và hệ thống. HAProxy được thiết kế và cấu hình để phân phối tải lưu lượng truy cập đến các máy chủ một cách đều đặn, giúp tối ưu hóa hiệu suất và đảm bảo tính ổn định của hệ thống theo nhu cầu phù hợp. Bên cạnh đó, chúng tôi cũng áp dụng HAProxy vào một số mô hình hệ thống để chứng minh tính ứng dụng cao của phần mềm này.

Thông qua đồ án này, nhóm chúng tôi hy vọng sẽ cung cấp một cái nhìn tổng quan về load balancing và nhu cầu sử dụng cân bằng tải trong thực tế, cụ thể tìm hiểu sâu phần mềm cân bằng tải HAProxy để giải quyết các thách thức về tải cho hệ thống web server và database, tăng cường an ninh và sự tin cậy của hệ thống ngân hàng trong thời đại số hóa.

LỜI CẢM ƠN

Để hoàn thành đồ án môn học này, chúng tôi xin tỏ lòng biết ơn sâu sắc đến Thầy ThS. Đỗ Hoàng Hiến đã hướng dẫn, cung cấp kiến thức, kinh nghiệm quý báu trong suốt quá trình thực hiện đồ án. Sự tận tâm và sự đồng hành của thầy đã đóng góp không nhỏ vào việc hoàn thành đồ án môn học của chúng tôi.

Lời cảm ơn không thể đủ để bày tỏ sự biết ơn chân thành của nhóm chúng tôi. Đây là đồ án môn học về lĩnh vực cân bằng tải trong hệ thống đầu tay của nhóm nên không thể tránh được có nhiều thiếu sót, rất mong được sự góp ý và chỉ dẫn thêm. Và nhóm chúng tôi hy vọng rằng đồ án này sẽ đóng góp vào sự phát triển trong lĩnh vực công nghệ.

Sau cùng, nhóm chúng tôi xin gửi đến ThS. Đỗ Hoàng Hiến, gia đình, bạn bè thân thiết và mọi người lời kính chúc sức khỏe và lời tri ân chân thành nhất.

TP. Hồ Chí Minh, 2023

Trân trọng./.

Huỳnh Nguyễn Uyên Nhi, Nguyễn Huy Hoàng,
Phạm Ngọc Thiện, Hồ Công Long

MỤC LỤC

MỤC LỤC.....	i
DANH MỤC HÌNH ẢNH	ii
TÓM TẮT	iv
Chương I: Tìm hiểu về Load Balancing và HAProxy	1
1.1. Load Balancing là gì?.....	1
1.1.1 Khái niệm:.....	1
1.1.2. Lợi ích của Load Balancing:	1
1.2. Giới thiệu về HAProxy.....	2
1.2.1. HAProxy là gì?	2
1.2.2. Tính năng chính của HAProxy:.....	2
1.2.3. Các loại Load Balancing cơ bản:	3
1.2.4. Các thuật toán sử dụng cho HAProxy.....	5
1.2.5. Ứng dụng HAProxy trong thực tế.....	6
Chương II: Cài đặt và cấu hình HAProxy.....	7
2.1. Install.....	7
2.2. Configuration.....	7
2.2.1. Global	7
2.2.2. Default.....	8
2.2.3. Frontend	8
2.2.4. Backend.....	9
Chương III: Demo	11
3.1. Demo các thuật toán:	11
3.1.1. Tài nguyên và Setup hệ thống	11
3.1.1.1. Tài nguyên.....	11
3.1.1.2. Setup hệ thống.....	11
3.1.2. Phần demo chi tiết	14
3.2 Demo mô hình cân bằng tải với Webserver và Database	22
3.2.1 Tài nguyên và setup hệ thống.....	22
3.2.2. Cài đặt chi tiết	23
3.2.2.1. Thiết lập và đồng bộ database - mysql master to master replication	24
3.2.2.2. Tạo trang web cơ bản sử dụng mysql database.	29
3.2.2.3. Cấu hình HAProxy cho web server và database.....	31
3.2.2.4 Thử nghiệm hệ thống	32
REFERENCES.....	39

DANH MỤC HÌNH ẢNH

Hình 1. Mô hình áp dụng load balancing	1
Hình 2. Layer 4 Load Balancing	4
Hình 3. Layer 7 Load Balancing	4
Hình 4. Ví dụ về phần global của file cấu hình haproxy.cfg	8
Hình 5. Ví dụ về phần default của file cấu hình haproxy.cfg	8
Hình 6. Ví dụ về phần frontend của file cấu hình haproxy.cfg	9
Hình 7. Ví dụ về phần default của file cấu hình haproxy.cfg	9
Hình 8. Mô hình đơn giản demo các thuật toán.	11
Hình 9. Các containers đã cài đặt.....	13
Hình 10. Trang quản trị HAProxy	13
Hình 11. Truy cập web1	13
Hình 12. Truy cập web2	14
Hình 13. Truy cập web3	14
Hình 14. Roundrobin-web1	15
Hình 15. Roundrobin-web2	15
Hình 16. Roundrobin-web3	15
Hình 17. Roundrobin- nhiều kết nối đến	16
Hình 18. Roundrobin- Config weight	16
Hình 19. Roundrobin- Test weight.	16
Hình 20. Roundrobin- Check weight khi đã thay đổi	17
Hình 21. Roundrobin- Kết quả sau khi connect.....	17
Hình 22. Static-rr: trang quản trị.....	17
Hình 23. Static-rr: test connection	17
Hình 24. Static-rr: Thay đổi weight.	18
Hình 25. Leastconn- result	18
Hình 26. Source- Thực hiện truy cập ở máy có IP: 192.168.15.131	19
Hình 27. Source- Thực hiện truy cập ở máy có IP: 192.168.15.132	19
Hình 28. Source-IP: 192.168.15.132- Thực hiện truy cập 3 lần.....	19
Hình 29. Uri- tải đến web1	20
Hình 30. Uri- tải đến web2.....	20
Hình 31. Uri- tải đến web3	21
Hình 32. url_param- id=1 – tải đến web3	22
Hình 33. url_param- id=2 – tải đến web2	22

Hình 34. url_param- id=5 – tải đến web1	22
Hình 35. Mô hình đơn giản HAProxy	23
Hình 36. Mô hình HAProxy triển khai cho webserver và database.....	24
Hình 37. Thư mục	24
Hình 38. my.cnf.....	25
Hình 39. Dockerfile.....	25
Hình 40. Thư mục db2	26
Hình 41. db2- my.cnf	26
Hình 42. Dockerfile- db2	26
Hình 43. docker IP check	27
Hình 44. Tạo DB trên 2 MYSQL Server.....	27
Hình 45. Check MYSQL	27
Hình 46. Page-html.....	29
Hình 47. File .php	30
Hình 48. haproxy.cfg- demo2	31
Hình 49. Test user	32
Hình 50. Truy cập trang quản trị HAProxy- demo2.....	32
Hình 51. HAProxy stat report 2 web- 2 server	33
Hình 52. Test web server	33
Hình 53. Kết quả- submit thông tin trên web.....	34
Hình 54. Check 2 database.....	34
Hình 55. Tắt 1 database server	34
Hình 56. Check status khi tắt 1 Database server.....	34
Hình 57. Truy cập web- nhập lại thông tin	35
Hình 58. Truy vấn lại dữ liệu ở database 2.....	35
Hình 59. Check lại database 1	35
Hình 60. Thực hiện tắt 1 web server	36
Hình 61. Check status sau khi tắt web server	36
Hình 62. Thực hiện submit dữ liệu lên web	37
Hình 63. Kiểm tra 2 database	37
Hình 64. Check status khi tắt 1 web và 1 server.....	37
Hình 65. Submit dữ liệu lên web	38
Hình 66. Check trên database server.....	38

TÓM TẮT

Trong khoảng thời gian hai tháng nghiên cứu về Load Balancing và ứng dụng HAProxy, nhóm 11 chúng tôi đã đạt được kiến thức nền tảng vững về cân bằng tải, hiểu rõ về ứng dụng thực tế của nó và nhận thức sâu sắc về những lợi ích mà nó mang lại cho các hệ thống đa máy chủ. Đồng thời, nhóm 11 đã dành thời gian tìm hiểu sâu về một giải pháp mã nguồn mở HAProxy, nắm vững các tính năng quan trọng giúp nó nổi bật và được ưa chuộng trong việc cân bằng tải.

Ngoài ra, nhóm 11 đã tập trung vào việc hiểu cách cấu hình HAProxy làm một load balancer trong hệ thống. Bằng cách áp dụng lý thuyết đã nghiên cứu, chúng tôi đã thực hiện hai kịch bản demo. Kịch bản đầu tiên được thiết kế để phân tích sâu về các thuật toán cân bằng tải của HAProxy. Kịch bản thứ hai tập trung vào triển khai cân bằng tải cho webserver và database, thể hiện khả năng ứng dụng của HAProxy trong môi trường thực tế.

Tóm lại, thông qua quá trình nghiên cứu và thực hành, nhóm 11 đã xây dựng được kiến thức chặt chẽ về cân bằng tải và ứng dụng thành công HAProxy để đáp ứng nhu cầu cụ thể của hệ thống.

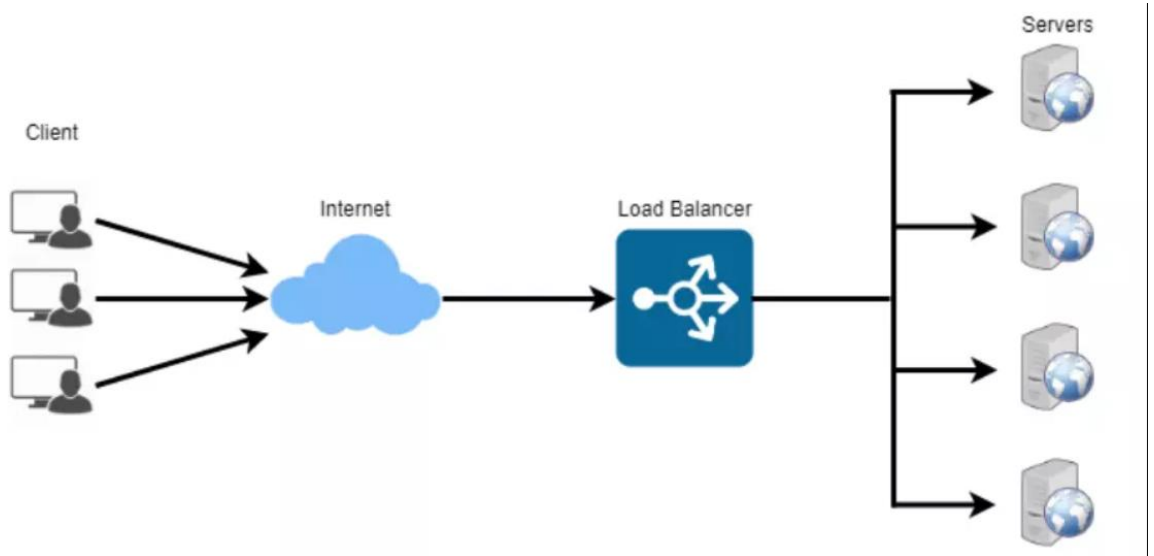
Chương I: Tìm hiểu về Load Balancing và HAProxy

1.1. Load Balancing là gì?

1.1.1 Khái niệm:

Load Balancing (cân bằng tải) là một kỹ thuật bảo đảm máy chủ không bị quá tải bởi lượng truy cập. Với các biện pháp cân bằng tải, khối lượng công việc, truy cập của người dùng được phân chia đều trên các máy chủ. Việc này cung cấp khả năng chịu lỗi cao, đảm bảo tính sẵn sàng phục vụ của hệ thống.

Trong giai đoạn đầu của thời đại công nghệ số, một máy chủ duy nhất khó có thể xử lý lưu lượng truy cập lớn. Các truy cập đồng loạt với khối lượng lớn khiến cho máy chủ thường xuyên bị quá tải. Điều này dẫn đến sự cần thiết của việc cân bằng tải – load balancing.



Hình 1. Mô hình áp dụng load balancing

1.1.2. Lợi ích của Load Balancing:

- Phân phối request của người dùng một cách hiệu quả đến nhiều backend, từ đó giúp giảm tải lượng công việc phải thực hiện trên mỗi backend.

- Đảm bảo tính sẵn sàng cao (high availability) và độ tin cậy (reliability) bằng việc chỉ gửi request tới các backend đang online. Từ đó giảm thiểu downtime khi một backend gặp sự cố.
- Tăng tính linh hoạt của toàn hệ thống khi có thể thêm hoặc bớt backend bất cứ lúc nào.
- Tăng tính bảo mật của hệ thống vì các backend lúc này thường sẽ nằm trong vùng DMZ (DeMilitary Zone – tạm hiểu là không được public), người dùng khi kết nối đến hệ thống là chỉ đang kết nối đến load balancer thay vì trực tiếp đến các backend.

1.2. Giới thiệu về HAProxy

1.2.1. HAProxy là gì?

HAProxy (High Availability Proxy) là một công cụ mã nguồn mở ứng dụng cho giải pháp cân bằng tải TCP và HTTP. Người dùng có thể sử dụng HAProxy để cải thiện suất hoàn thiện của các trang web và ứng dụng bằng cách phân tán khối lượng công việc của chúng trên nhiều máy chủ. Cải thiện hiệu suất bao gồm giảm phản hồi thời gian và tăng thông lượng.

HAProxy là một trong những phần mềm hàng đầu trong việc cân bằng tải và được sử dụng chạy trên Linux, Solaris và FreeBSD. Dù HAProxy là phần mềm mã nguồn mở được sử dụng miễn phí nhưng nó cũng có một phần mềm thương mại hóa dựa trên HAProxy Technologies được gọi là **HAProxy Enterprise**. HAProxy Enterprise bao gồm các tiện ích bổ sung, hỗ trợ chuyên sâu và nâng cao dịch vụ.

1.2.2. Tính năng chính của HAProxy:

- High Availability:
Đảm bảo các dịch vụ web services luôn có thể truy cập được, điều hướng các traffic trong mạng một cách hợp lý để các server thuộc server pool, giảm thiểu thời gian downtime của hệ thống.
- Load Balancing Methods:

HAProxy cung cấp hàng loạt các thuật toán cân bằng tải cho phép người quản trị hệ thống lựa chọn thuật toán phù hợp nhất ứng với từng trường hợp sử dụng cụ thể.

- **SSL Termination:**

HAProxy có thể xử lý các kết SSL encryption/decryption, các máy chủ backend không cần phải xử lý việc giải mã, điều này giảm tải cho chúng và làm tăng hiệu suất hệ thống.

- **Health Checks:**

HAProxy cung cấp cơ chế health check để tự động kiểm tra trạng thái của các máy chủ backend và loại bỏ các máy chủ không khỏe mạnh khỏi dịch vụ, giúp đảm bảo rằng chỉ có các máy chủ đang hoạt động được sử dụng để xử lý yêu cầu.

- **Session Persistence:**

Đây là một tính năng quan trọng giúp duy trì thông tin phiên (session) cho một client và chuyển hướng các yêu cầu từ cùng một client đó đến cùng một máy chủ backend. Điều này đảm bảo rằng các yêu cầu từ cùng một người dùng sẽ luôn được gửi đến cùng một máy chủ, giúp duy trì trạng thái của phiên làm việc.

- **Security Features:**

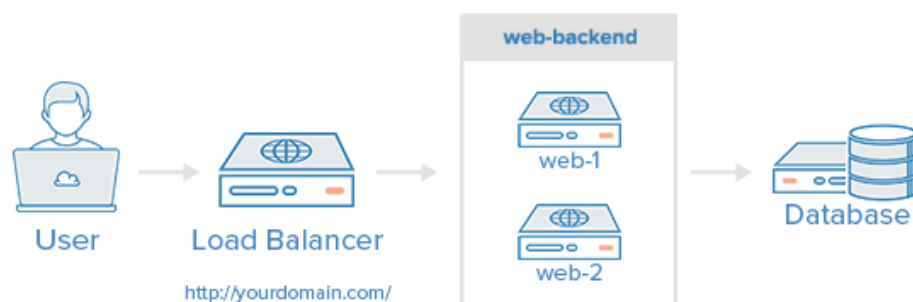
HAProxy bao gồm nhiều tính năng bảo mật như kiểm soát truy cập dựa trên ACL (Access Control List), bảo vệ chống tấn công DDoS, và làm sạch yêu cầu HTTP, tạo ra một môi trường an toàn cho các dịch vụ web.

1.2.3. Các loại Load Balancing cơ bản:

- **Load Balancing layer 4:** là một kỹ thuật được sử dụng để phân phối công việc và tải giữa các máy chủ dựa trên thông tin ở tầng transport layer của mô hình OSI. Tầng transport layer chịu trách nhiệm về việc truyền thông tin giữa các thiết bị trên mạng và cung cấp các dịch vụ như đảm bảo độ tin cậy và kiểm soát luồng dữ liệu. Các phương pháp cân bằng tải ở tầng 4 thường

sử dụng thông tin như địa chỉ IP và port để quyết định cách mà yêu cầu từ người dùng sẽ được chuyển tiếp đến máy chủ nào trong một nhóm máy chủ.

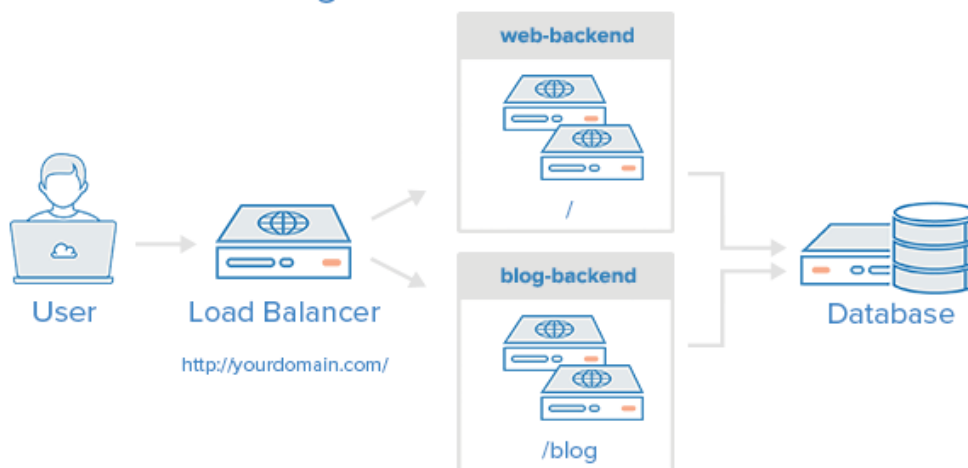
Layer 4 Load Balancing



Hình 2. Layer 4 Load Balancing

- **Load Balancing layer 7:** là một kỹ thuật được sử dụng để phân phối tải giữa các máy chủ dựa trên thông tin ở tầng ứng dụng trong mô hình OSI. Tầng 7 chịu trách nhiệm về các dịch vụ ứng dụng cụ thể và giao thức, như HTTP cho web servers, SMTP cho email servers, và nhiều dịch vụ khác. Cân bằng tải ở tầng 7 tập trung vào thông tin ứng dụng cụ thể như URL, tiêu đề HTTP, hoặc nội dung gói tin để quyết định cách mà yêu cầu từ client sẽ được chuyển tiếp đến máy chủ nào trong một nhóm máy chủ.

Layer 7 Load Balancing



Hình 3. Layer 7 Load Balancing

Ngữ cảnh ứng dụng của mỗi loại load balancing:

Layer 4 load balancing:

- **Web Servers:** Khi có nhiều máy chủ web cung cấp các trang web tĩnh hoặc các ứng dụng đơn giản, Layer 4 load balancing có thể được sử dụng để phân phối lưu lượng dựa trên địa chỉ IP và cổng.
- **DNS Load Balancing:** Đối với các dịch vụ DNS, Layer 4 có thể được sử dụng để phân phối yêu cầu DNS đến các máy chủ DNS khác nhau dựa trên thông tin địa chỉ IP và cổng.

Layer 7 load balancing:

- **Ứng dụng Web Phức Tạp:** Khi có nhiều dịch vụ web phức tạp như ứng dụng thương mại điện tử, Layer 7 load balancing có thể cân bằng tải dựa trên các thông tin ứng dụng như URL, thông tin đăng nhập, hoặc thông tin phiên làm việc.
- **Cân bằng tải Video Streaming:** Đối với các dịch vụ video streaming, Layer 7 có thể phân phối lưu lượng dựa trên loại nội dung (video, âm thanh), độ phân giải, hoặc yêu cầu đặc biệt của người dùng.
- **Ứng Dụng API:** Khi triển khai các dịch vụ API, Layer 7 có thể giúp cân bằng tải dựa trên các thông số như loại yêu cầu, phiên bản API, hoặc dữ liệu payload của yêu cầu API.

1.2.4. Các thuật toán sử dụng cho HAProxy

- **Round robin:** thuật toán này được thiết kế để phân phối requests đều đặn giữa các máy server nằm trong server pool theo thứ tự nhất định. Ngoài ra trong quá trình cân bằng tải thuật toán này còn cho phép người quản trị thiết lập hoặc thay đổi trọng số của các máy server để phân phối lượng requests.
- **Static Round robin:** tương tự Round robin, nhưng khác ở chỗ việc thay đổi trọng số của các server trong quá trình cân bằng tải không được cho phép.
- **Least Connection:** thuật toán này chuyển tiếp request đến máy chủ có ít kết nối nhất đến hiện tại, giảm áp lực cho các máy chủ đang xử lý nhiều kết nối.
- **Source:** là một phương pháp cân bằng tải dựa trên địa chỉ nguồn của các kết nối. Thuật toán này được thiết kế để đảm bảo rằng một địa chỉ nguồn cụ thể

sẽ luôn được chuyển tiếp đến cùng một máy chủ, giữ cho trạng thái phiên làm việc giữa nguồn và máy chủ.

- **URI:** sử dụng thông tin từ phần URI (Uniform Resource Identifier) của mỗi yêu cầu HTTP để xác định máy chủ nào sẽ xử lý yêu cầu đó. Điều này có thể hữu ích trong các tình huống nơi một số phần của ứng dụng có thể được tối ưu hóa để chạy trên một máy chủ cụ thể.
- **URL parameter:** tương tự như thuật toán "uri" nhưng thay vì xử lý toàn bộ URI, nó tập trung vào xử lý các tham số của URL. Điều này có thể hữu ích khi muốn phân phối tải dựa trên giá trị của một hoặc vài tham số cụ thể.

1.2.5. Ứng dụng HAProxy trong thực tế

Chính nhờ vào tính mạnh mẽ, linh hoạt và hiệu quả, HAProxy đã và đang được các công ty, doanh nghiệp lớn lựa chọn áp dụng:

- GitHub, nền tảng lưu trữ mã nguồn và quản lý dự án phần mềm lớn, sử dụng HAProxy để cân bằng tải và tối ưu hóa hiệu suất cho cơ sở dữ liệu MySQL.
- Twitter, Reddit: sử dụng HAProxy để quản lý tải và cung cấp trải nghiệm người dùng mượt mà.
- Stack Overflow, một nền tảng chia sẻ kiến thức và trao đổi với cộng đồng lập trình viên, sử dụng HAProxy để cân bằng tải và quản lý luồng truy cập đến trang web của họ.
- Airbnb, Booking.com một nền tảng đặt chỗ và chia sẻ nhà trực tuyến, cũng đã tích hợp HAProxy vào hạ tầng của họ để cải thiện khả năng mở rộng và hiệu suất.

Qua đây chúng ta thấy được sự phổ biến và độ linh hoạt của HAProxy trong việc cung cấp giải pháp cân bằng tải cho các hệ thống web và ứng dụng.

Chương II: Cài đặt và cấu hình HAProxy

2.1. Install

- Đối với các hệ thống dựa trên Ubuntu và Debian sử dụng trình quản lý gói APT, hãy làm như sau:
 - B1: Cập nhật danh sách gói: `sudo apt update`
 - B2: Cài đặt HAProxy bằng lệnh sau: `sudo apt install haproxy`
- Đối với các hệ thống dựa trên CentOS và RHEL sử dụng trình quản lý gói yum, để cài đặt HAProxy:
 - B1: Cập nhật danh sách gói yum: `sudo yum update`
 - B2: Cài đặt HAProxy bằng lệnh sau: `sudo yum install haproxy`.

2.2. Configuration

HAProxy cung cấp tệp cấu hình mẫu nằm trong `/etc/haproxy/haproxy.cfg`. Tệp chứa thiết lập tiêu chuẩn không có bất kỳ tùy chọn cân bằng tải nào. Sử dụng trình soạn thảo văn bản để xem tệp cấu hình và kiểm tra nội dung:

```
***sudo nano /etc/haproxy/haproxy.cfg
```

Cấu hình haproxy thường gồm 4 phần chính: global, defaults, frontend và backend. Các thành phần này sẽ định nghĩa cách máy chủ HAProxy tiếp nhận, xử lý yêu cầu, chọn máy chủ tiếp nhận yêu cầu và chuyển tiếp.

2.2.1. Global

Các thiết lập global nằm ở phần đầu của file cấu hình haproxy.cfg, được định danh bằng từ khóa global và chỉ được định nghĩa riêng một mình, quy định các thiết lập bảo mật, điều chỉnh hiệu năng... cho toàn hệ thống HAProxy. Một số tham số quan trọng được đặt trong global như số lượng kết nối tối đa (maxconn), đường dẫn file log, số tiến trình... Chỉ có một global trong tệp cấu hình và không được thay đổi các giá trị.

```

global
    log /dev/log      local0
    log /dev/log      local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

    # Default SSL material locations
    ca-base /etc/ssl/certs
    crt-base /etc/ssl/private

    # See: https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&confi
    ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA2
    ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_
    ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

```

Hình 4. Ví dụ về phần global của file cấu hình haproxy.cfg

2.2.2. Default

Defaults chứa các cấu hình được thiết lập áp dụng chung cho cả phần frontend và backend trong file cấu hình. Các thiết lập defaults có thể được thiết lập nhiều lần trong file cấu hình và chúng được ghi đè lên nhau (các mục defaults sau sẽ ghi đè lên các mục defaults trước nó), ngược lại các thiết lập trong frontend và backend sẽ ghi đè lên defaults.

```

defaults
    log      global
    mode     http
    option   httplog
    option   dontlognull
    timeout  connect 5000
    timeout  client  50000
    timeout  server  50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http

```

Hình 5. Ví dụ về phần default của file cấu hình haproxy.cfg

- *Mode*: định nghĩa HAproxy sẽ sử dụng TCP proxy hay HTTP proxy.
- *Timeout*: là giá trị thời gian chờ được chỉ định theo mili giây theo mặc định, nhưng có thể ở bất kỳ đơn vị nào khác nếu số đó được đơn vị thêm vào.
- *Errorfile*: Liên kết nội dung tệp với mã lỗi HTTP.

2.2.3. Frontend

- Các thiết lập trong phần frontend định nghĩa địa chỉ IP và port mà client có thể kết nối tới, có thể có nhiều mục frontend tùy ý, chỉ cần đặt label của chúng khác nhau : frontend <tên>


```
defaults
mode http
timeout client 10s
timeout connect 5s
timeout server 10s
timeout http-request 10s

frontend my_frontend
bind 127.0.0.1:80
```

Hình 6. Ví dụ về phần frontend của file cấu hình haproxy.cfg

Một số thiết lập trong Haproxy được định nghĩa trong phần frontend như sau:

- bind: IP và Port HAProxy sẽ lắng nghe để mở kết nối. IP có thể bind tất cả địa chỉ sẵn có hoặc chỉ 1 địa chỉ duy nhất, port có thể là một port hoặc nhiều port (1 khoảng hoặc 1 list).
- http-request redirect: Phản hồi tới client với đường dẫn khác. Ứng dụng khi client sử dụng http và phản hồi từ HAProxy là https, điều hướng người dùng sang giao thức https.
- userbackend: Chỉ định backend sẽ xử lý request nếu thỏa mãn điều kiện (Khi sử dụng ACL).
- defaultbackend: Backend mặc định sẽ xử lý request (Nếu request không thỏa mãn bất kỳ điều hướng nào).

2.2.4. Backend

Các thiết lập trong phần backend định nghĩa tập server sẽ được cân bằng tải khi có các kết nối tới (ví dụ tập các server chạy dịch vụ web giống nhau)

```
defaults
mode http
timeout client 10s
timeout connect 5s
timeout server 10s
timeout http-request 10s

frontend my_frontend
bind 127.0.0.1:80
default_backend my_backend

backend my_backend
balance leastconn
server server1 127.0.0.1:8001
server server2 127.0.0.1:8002
```

Hình 7. Ví dụ về phần default của file cấu hình haproxy.cfg

Một số thiết lập trong Haproxy được định nghĩa trong phần backend như sau:

- balance: Kiểm soát cách HAProxy nhận, điều phối request tới các backend server. Đây chính là các thuật toán cân bằng tải.

- cookie: Sử dụng cookie-based. Cấu hình sẽ khiến HAProxy gửi cookie tên SERVERUSER tới client, liên kết backend server với client. Từ đó các request xuất phát từ client sẽ tiếp tục nói chuyện với server chỉ định. Cần bổ sung thêm tùy chọn cookie trên server line.
- server: Tùy chọn quan trọng nhất trong backend section. Tùy chọn đi kèm bao gồm tên, id, port. Có thể dùng domain thay cho IP.
- default server: Bổ sung tùy chọn cho bất kỳ backend server thuộc backend section (VD: health checks, max connections, v.v). Điều này khiến cấu hình dễ dàng hơn khi đọc.

Chương III: Demo

3.1. Demo các thuật toán:

3.1.1. Tài nguyên và Setup hệ thống

3.1.1.1. Tài nguyên

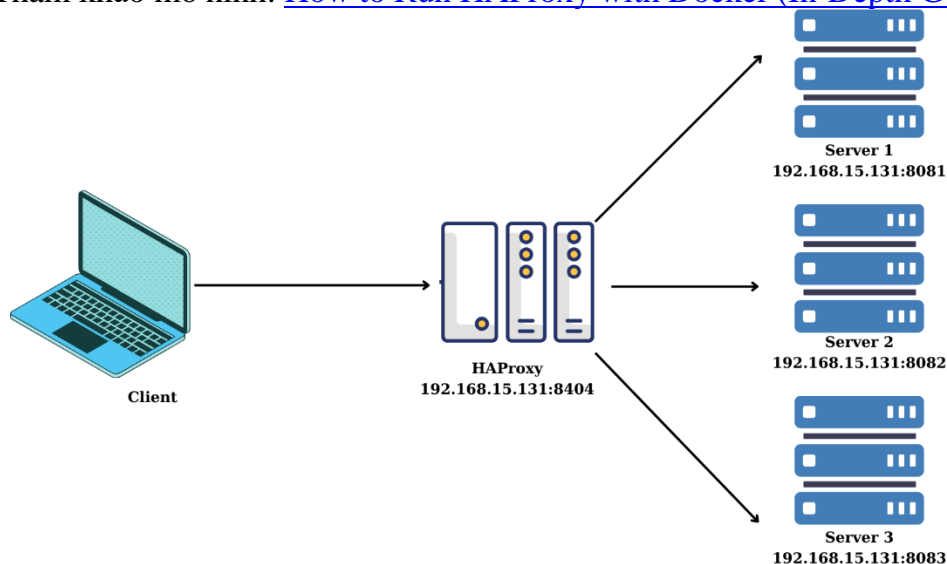
02 mages: haproxytech/haproxy-alpine và jmalloc/echo-server

```
$ docker pull haproxytech/haproxy-alpine
$ docker pull jmalloc/echo-server
```

02 máy ảo Ubuntu: 192.168.15.131 (Host web), 1 máy có địa chỉ IP cùng lớp mạng với máy host (192.168.15.132).

3.1.1.2. Setup hệ thống

Tham khảo mô hình: [How to Run HAProxy with Docker \(In-Depth Guide\)](#)



Hình 8. Mô hình đơn giản demo các thuật toán.

Cài đặt bridge network cho hệ thống:

```
$ sudo docker network create --driver=bridge mynetwork
```

Cài đặt web server:

```
$ sudo docker run -d W
    --name webX --net mynetwork -p YYYY:8080 jmalloc/echo-
server:latest
```

Với webX là tên cài đặt, YYYY: là port của web đang lắng nghe.
File haproxy.cfg (Với thuật toán cân bằng là mặc định: roundrobin)

```
global
    stats socket /var/run/api.sock user haproxy group haproxy mode 660 level admin expose-fd listeners
    log stdout format raw local0 info

defaults
    mode http
    timeout client 30s
    timeout connect 10s
    timeout server 30s
    timeout http-request 10s
    log global
    maxconn 100

frontend stats
    bind *:8404
    stats enable
    stats uri /
    stats refresh 10s
    stats admin if LOCALHOST

frontend myfrontend
    bind :80
    default_backend webservers

backend webservers

    server s1 192.168.15.131:8081 check
    server s2 192.168.15.131:8082 check
    server s3 192.168.15.131:8083 check
```

Cài đặt HAProxy server:

```
$ sudo docker run -d W
--name haproxy W
--net mynetwork W
-v $(pwd):/usr/local/etc/haproxy:ro W
-p 80:80 W
-p 8404:8404 W haproxytech/haproxy-alpine:2.4
```

- Vào trang quản trị Haproxy: <http://localhost:8404>
- Truy cập web : <ip>:80 // ở đây máy host có địa chỉ IP là 192.168.15.131
- Khi có thay đổi nào ở file .cfg dùng lệnh:

```
$ sudo docker exec -it haproxy kill -HUP 1
```

- Kiểm tra các containers:

```

ngghoang@hoangvirtualmachine:~/qt5_demo$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
NAMES
121ab7ee7b15   haproxytech/haproxy-alpine:2.4     "/docker-entrypoint..." 5 seconds ago   Up 4 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:8404->8404/tcp, :::8404->8404/tcp
haproxy
561de7f8ef3e   jnalloc/echo-server:latest         "/bin/echo-server"       43 seconds ago   Up 42 seconds   0.0.0.0:8083->8080/tcp, :::8083->8080/tcp
web3
f07676cf63ae   jnalloc/echo-server:latest         "/bin/echo-server"       53 seconds ago   Up 52 seconds   0.0.0.0:8082->8080/tcp, :::8082->8080/tcp
web2
78d33973f19e   jnalloc/echo-server:latest         "/bin/echo-server"       About a minute ago   Up About a minute   0.0.0.0:8081->8080/tcp, :::8081->8080/tcp
web1

```

Hình 9. Các containers đã cài đặt

Trang quản trị HAProxy:

Statistics Report for HAP: X +

localhost:8404

HAProxy version 2.4.24-d175670, released 2023/08/19

Statistics Report for pid 8

> General process information

pid = 8 (process #1, nbproc = 1, nbthread = 4)
 uptime = 0s (0h0m0s)
 system limits: memmax = unlimited, ulimit = 524287
 maxsock = 524287, maxconn = 32768, maxpipes = 0
 current conn = 1, current pipes = 0, conn rate = 1/sec, bit rate = 0.000 kbps
 Running tasks: 0/20, idle = 100 %

Display options:
 • Scope:
 • Hide DOWN servers
 • Disable refresh
 • Refresh now
 • CSV export
 • JSON export (schema)

External resources:
 • Primary file
 • Updates (v2.4)
 • Online manual

Stats		Queue		Session rate		Sessions		Bytes		Denied		Errors		Warnings		Status		Server	
Frontend	Backend	Cur	Max	Limit	Cur	Max	Limit	Total	LiTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis
Frontend		1	1	-	1	1	100	1			0	0	0	0	0	0	0	0	0
myfrontend																			
Frontend		0	0	-	0	0	100	0			0	0	0	0	0	0	0	0	0
webserver																			
Frontend		0	0	-	0	0	100	0			0	0	0	0	0	0	0	0	0
Backend		0	0	-	0	0	10	0			0	0	0	0	0	0	0	0	0

Hình 10. Trang quản trị HAProxy

Thực hiện kiểm tra tình trạng các webserver:

Statistics Report for HAP: X +

192.168.15.131:8081/ X +

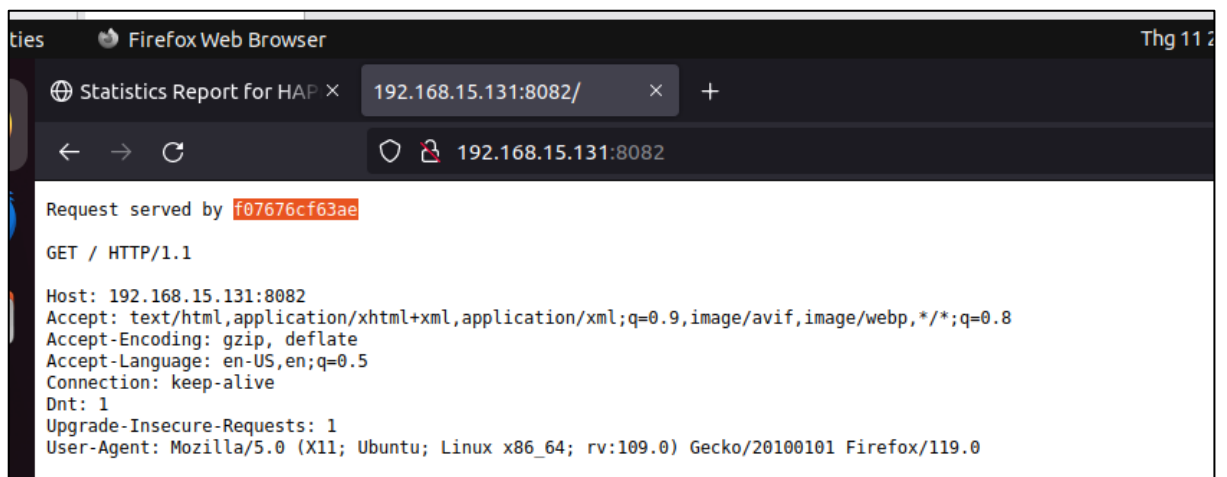
192.168.15.131:8081

Request served by 78d33973f19e

GET / HTTP/1.1

Host: 192.168.15.131:8081
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
 Accept-Encoding: gzip, deflate
 Accept-Language: en-US,en;q=0.5
 Connection: keep-alive
 Dnt: 1
 Upgrade-Insecure-Requests: 1
 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0

Hình 11. Truy cập web1



Hình 12. Truy cập web2



Hình 13. Truy cập web3

Dòng **Request served by: <any>** ứng với Container id. Ta sử dụng giá trị này để tìm hiểu và chứng minh các thuật toán.

3.1.2. Phần demo chi tiết

Với phần demo các thuật toán trong đề án này, nhóm sẽ demo 06 loại thuật toán của HAProxy tiêu biểu: roundrobin, static-rr, leastconn, source, uri, url_param.

Với việc thay đổi các thuật toán ta chỉ cần thay đổi tham số **balance <alg>** trong phần backend của file .cfg.

Thuật toán Roundrobin

Config:

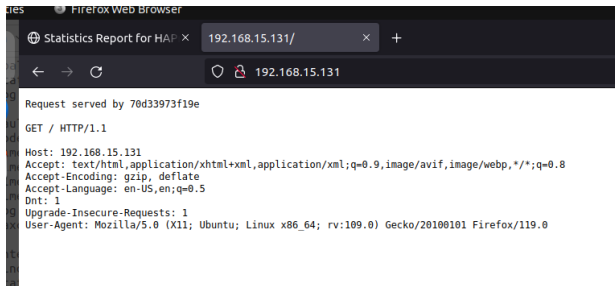
```
# các phần trên không thay đổi.
backend webservers
  balance roundrobin
  server s1 192.168.15.131:8081 check
  server s2 192.168.15.131:8082 check
  server s3 192.168.15.131:8083 check
```

Mặc định vẫn sẽ là roundrobin nếu ta không ghi dòng **balance roundrobin**

Thuật toán roundrobin dùng để phân phối tải đều đặn cho các server. Khi một yêu cầu đến, HAProxy chọn server trong danh sách và chuyển hướng yêu cầu đến đó. Quá trình này được lặp lại, bắt đầu từ đầu danh sách các máy chủ và duyệt đến hết.

Thực hiện demo:

Với lần request đầu tiên: cân bằng tải sẽ chuyển hướng đến web1



Request served by 70d3973f19e

GET / HTTP/1.1

Host: 192.168.15.131

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.5

Dnt: 1

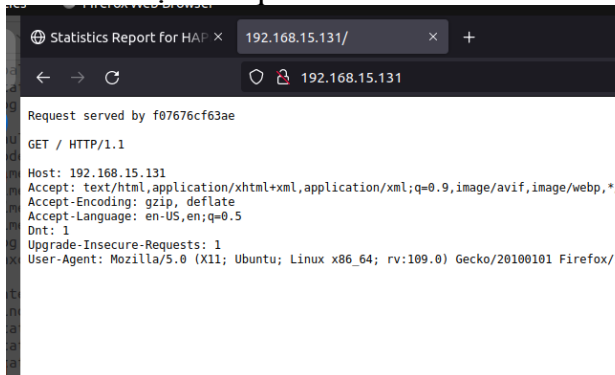
Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0

webservers											
	Queue			Session rate			Sessions				
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTo
s1	0	0	-	0	1		0	1	-	1	
s2	0	0	-	0	0		0	0	-	0	
s3	0	0	-	0	0		0	0	-	0	
Backend	0	0		0	1		0	1	10	1	

Hình 14. Roundrobin-web1

Lần lượt ta request lần 2 và lần 3:



Request served by f07676cf63ae

GET / HTTP/1.1

Host: 192.168.15.131

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.5

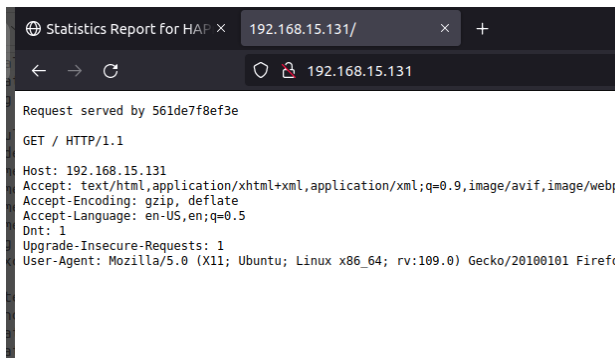
Dnt: 1

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0

webservers											
	Queue			Session rate			Sessions				
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTo
s1	0	0	-	0	1		0	1	-	1	
s2	0	0	-	0	1		0	1	-	1	
s3	0	0	-	0	0		0	0	-	0	
Backend	0	0		0	1		0	1	10	1	

Hình 15. Roundrobin-web2



Request served by 561de7f8ef3e

GET / HTTP/1.1

Host: 192.168.15.131

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.5

Dnt: 1

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0

webservers											
	Queue			Session rate			Sessions				
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTo
s1	0	0	-	0	1		0	1	-	1	
s2	0	0	-	0	1		0	1	-	1	
s3	0	0	-	0	1		0	1	-	1	
Backend	0	0		0	1		0	1	10	1	

Hình 16. Roundrobin-web3

Với 3 lần request ta thấy thuật toán này phân bố đều đến các web server. Test với nhiều lần request để kiểm chứng:

\$ ab -n 300 -c 30 <http://192.168.15.131:80/>

Với 300 kết nối – 30 kết nối đến server đồng thời.
Sau khi đó xem xét kết quả ở trang quản trị:

webservers													
	Queue			Session rate			Sessions						Bytes
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	
s1	0	0	-	0	100		0	23	-	101	101	5s	8 817
s2	0	0	-	0	100		0	18	-	101	101	5s	8 817
s3	0	0	-	0	100		0	17	-	101	101	5s	8 817
Backend	0	0		0	300		0	31	10	303	303	5s	26 451

Hình 17. Roundrobin- nhiều kết nối đến

Ta nhận thấy, thuật toán vẫn tải đều đến các server.

Roundrobin với trọng số:

Nếu ta sử dụng thuật toán như trên, trọng số sẽ mặc định là 1-1-1 đối với cả 3 server. Nếu ta set trọng số khác cho 3 server thì HAProxy sẽ phân phối tải theo tỉ lệ trọng số đã cài đặt trong file config.

Thực hiện chỉnh sửa trọng số:

```

24
25 backend webservers
26
27 server s1 192.168.15.131:8081 weight 2 check
28 server s2 192.168.15.131:8082 weight 3 check
29 server s3 192.168.15.131:8083 weight 5 check

```

Server					
Status	LastChk	Wght	Act	Bck	C
2s UP	L4OK in 0ms	2/2	Y	-	
2s UP	L4OK in 0ms	3/3	Y	-	
2s UP	L4OK in 0ms	5/5	Y	-	
2s UP		10/10	3	0	

Hình 18. Roundrobin- Config weight

Thực hiện kiểm tra với 1000 kết nối:

webservers													
	Queue			Session rate			Sessions						Bytes
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	
s1	0	0	-	0	200		0	5	-	200	200	3s	17 000
s2	0	0	-	0	300		0	6	-	300	300	3s	25 500
s3	0	0	-	0	500		0	9	-	500	500	3s	42 500
Backend	0	0		0	1 000		0	11	10	1 000	1 000	3s	85 000

Hình 19. Roundrobin- Test weight.

Ta nhận thấy tải đã được phân phối đến các web server đúng với tỉ lệ đã config.

Thuật toán static- roundrobin

Tương tự thuật toán roundrobin, tuy nhiên static-rr là một biến thể thuật toán tĩnh, không hỗ trợ việc điều chỉnh trọng số trong khi hệ thống đang chạy.

Trước hết, thử nghiệm với việc thay đổi trọng số khi server đang hoạt động với thuật toán roundrobin.

```
$ echo "set weight webservers/s1 10" | socat stdio unix-connect:/var/run/api.sock
```

Ví dụ chuyển trọng số của s1 thành 10.

Hình bên dưới cho thấy, chúng ta đã thay đổi thành công weight của s1.

webservers		Queue		Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server			
							Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act
s1	0	0	-	0	200		0	5	-	200	200	5m8s	17 000	47 600		0	0	0	0	0	0	6m6s UP	L4OK in 0ms	10/10	Y
s2	0	0	-	0	300		0	6	-	300	300	5m8s	25 500	71 400		0	0	0	0	0	0	6m6s UP	L4OK in 0ms	3/3	Y
s3	0	0	-	0	500		0	9	-	500	500	5m8s	42 500	119 000		0	0	0	0	0	0	6m6s UP	L4OK in 0ms	5/5	Y
Backend	0	0		0	1 000		0	11	10	1 000	1 000	5m8s	85 000	238 000	0	0		0	0	0	0	6m6s UP		18/18	3

Hình 20. Roundrobin- Check weight khi đã thay đổi

Ta tiến hành kết thêm 1800 kết nối đến server:

Full Disk Read Return 100% Not Return Server																															
webmasters			Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server										
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thr	
s1	0	0	-	39	556		0	10	-	1 200	1 200	2s	102 000	285 600	0	0	0	0	0	0	0	10m10s UP	L4OK in 0ms	10/10	Y	-	0	0	0	0s	-
s2	0	0	-	12	300		0	6	-	600	600	2s	51 000	142 800	0	0	0	0	0	0	0	10m10s UP	L4OK in 0ms	3/3	Y	-	0	0	0	0s	-
s3	0	0	-	20	500		0	9	-	1 000	1 000	2s	85 000	238 000	0	0	0	0	0	0	0	10m10s UP	L4OK in 0ms	5/5	Y	-	0	0	0	0s	-
Backend	0	0		71	1 000		0	11	10	2 800	2 800	2s	238 000	666 400	0	0	0	0	0	0	0	10m10s UP		18/18	3	0	0	0	0	0s	

Hình 21. Roundrobin- Kết quả sau khi connect

Số lần kết nối cũ là 200, 300, và 500 (ứng với s1, s2 và s3) trước khi thay đổi trọng số. Tiến hành vài phép tính trừ ta thấy số lần mà server đã cân bằng tải đến các web s1, s2 và s3 lần lượt là 1000, 300 và 500 (Tương ứng với tỉ lệ weight đã thay đổi 10:3:5)

Tương tự như trên, ta thử với việc config (sử dụng thuật toán static-rr) và tiến hành thay đổi trọng số khi server đang chạy.

các phần trên không thay đổi.

backend webmasters

balance static-rr

server s1 192.168.15.131:8081 weight 2 check

server s2 192.168.15.131:8082 weight 3 check

server s3 192.168.15.131:8083 weight 5 check

webmasters																												
	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Status	LastChk	Wght	Ser				
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr					Redis			
s1	0	0	-	0	0		0	0	-	0	0	0	?	0	0		0	0	0	0	0	3s UP	L4OK in 0ms	2/2	Y			
s2	0	0	-	0	0		0	0	-	0	0	0	?	0	0		0	0	0	0	0	3s UP	L4OK in 0ms	3/3	Y			
s3	0	0	-	0	0		0	0	-	0	0	0	?	0	0		0	0	0	0	0	3s UP	L4OK in 0ms	5/5	Y			
Backend	0	0		0	0		0	0	10	0	0	?	0	0	0	0		0	0	0	0	3s UP		10/10	3			

Hình 22. Static-rr: trạng quản trị

Tiến hành với 1000 kết nối đến server:

Webmasters																													
webservers			Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwnt
s1	0	0	-	0	200		0	6	-	200	200	2s	17 000	47 600	0	0	0	0	0	0	0	1m18s UP	L4OK in 0ms	2/2	Y	-	0	0	0
s2	0	0	-	0	300		0	8	-	300	300	2s	25 500	71 400	0	0	0	0	0	0	0	1m18s UP	L4OK in 0ms	3/3	Y	-	0	0	0
s3	0	0	-	0	500		0	9	-	500	500	2s	42 500	119 000	0	0	0	0	0	0	0	1m18s UP	L4OK in 0ms	5/5	Y	-	0	0	0
Backend	0	0		0	1 000		0	11	10	1 000	1 000	2s	85 000	238 000	0	0	0	0	0	0	0	1m18s UP		10/10	3	0	0	0	0

Hình 23. Static-rr: test connection

Với lần thử nghiệm trên, ta đưa ra kết luận thuật toán static-rr giống với thuật toán roundrobin.

Ta tiến hành thử nghiệm với việc thay đổi trọng số khi server đang hoạt động (Thuật toán static-rr).

```
$ echo "set weight webservers/s1 10" | socat stdio unix-connect:/var/run/api.sock
```

```
/ # echo "set weight webservers/s1 10" | socat stdio unix-connect:/var/run/api.sock
Backend is using a static LB algorithm and only accepts weights '0%' and '100%'.

/ #
```

Hình 24. Static-rr: Thay đổi weight.

Terminal xuất ra thông báo không thể thay đổi được trọng số như ta mong muốn.

Thuật toán leastconn

Khi một yêu cầu đến, HAProxy sẽ chọn server có số kết nối ít nhất từ danh sách các server khả dụng. Điều này giúp đảm bảo rằng yêu cầu mới sẽ được chuyển hướng đến máy chủ có khả năng xử lý tốt nhất, dựa trên số lượng kết nối hiện tại.

Config:

```
# các phần trên không thay đổi.
backend webservers
    balance leastconn
    server s1 192.168.15.131:8081 maxconn 1 check
    server s2 192.168.15.131:8082 maxconn 10 check
    server s3 192.168.15.131:8083 maxconn 20 check
```

Thực nghiệm cho thấy giao thức HTTP là short session nên khi ta truy cập thủ công đến webserver thì lưu lượng kết nối ít và session rất ngắn nên khi chạy thuật toán leastconn cũng không khác với thuật toán roundrobin(với trọng số đã set).

Do đó ta sẽ test với câu lệnh:

```
$ ab -n 1000 -c 10 http://192.168.15.131:80/
```

Với 1000 kết nối và 10 kết nối cùng lúc và ta sẽ xem xét kết quả.

websockets		Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		LastChk		Server		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis			Wght	Act	Bck
s1	0	0	-	142	142	0	0	1	1	142	142	1s	12 070	33 796	0	0	0	0	0	0	0	4m1s UP	L4OK in 0ms	1/1	Y	-
s2	0	0	-	448	448	0	5	10	448	448	1s	38 080	106 624	0	0	0	0	0	0	0	4m1s UP	L4OK in 0ms	1/1	Y	-	
s3	0	0	-	410	410	0	5	20	410	410	1s	34 850	97 580	0	0	0	0	0	0	0	4m1s UP	L4OK in 0ms	1/1	Y	-	
Backend	0	0	-	1 000	1 000	0	10	10	1 000	1 000	1s	85 000	238 000	0	0	0	0	0	0	0	4m1s UP		3/3	3	0	

Hình 25. Leastconn- result

Kết quả cho thấy tỉ lệ tải đến các server không bằng với tỉ lệ 1:1:1 (mặc định). Và số lượng kết nối đến s1 là ít nhất chứng tỏ thuật toán này hoạt động: vì s1 được set maxconn là 1 nên với 10 kết nối cùng lúc, cân bằng tải sẽ điều hướng đến server s2 hoặc s3 có lượng kết nối đến ít nhất.

Thuật toán source

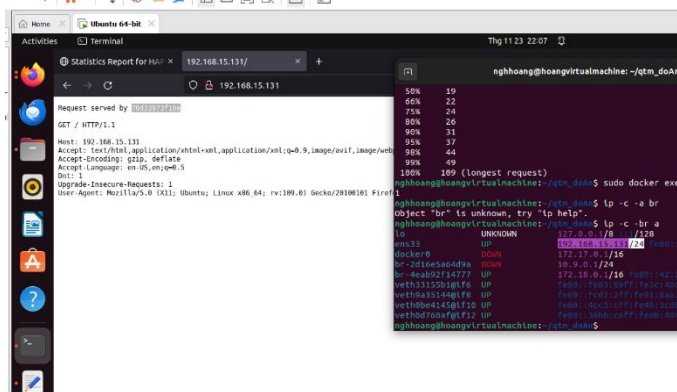
Khi một yêu cầu đến, HAProxy sẽ sử dụng địa chỉ nguồn của yêu cầu (thường là địa chỉ IP của client) để chọn server để chuyển hướng yêu cầu đến. Điều này có nghĩa là cùng một địa chỉ nguồn sẽ luôn được chuyển hướng đến cùng một máy chủ, giúp duy trì trạng thái và kết nối.

Config:

```
# các phần trên không thay đổi.
backend webservers
    balance source
    hash-type consistent # chọn kiểu hash
    server s1 192.168.15.131:8081 check
```

server s2 192.168.15.131:8082 check
server s3 192.168.15.131:8083 check

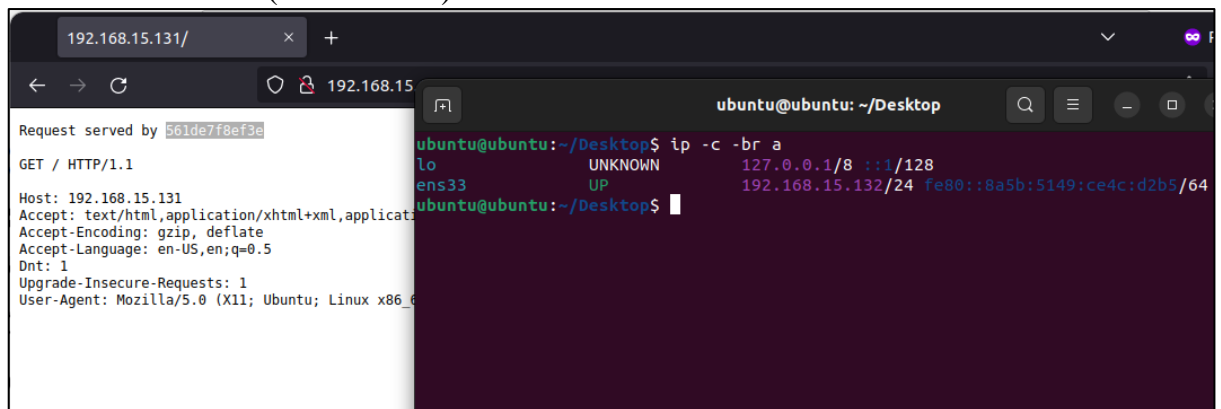
Thực hiện demo:



Frontend		0	1	-	0	1	100	1	
webservers									
		Queue			Session rate			Sessions	
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max
s1		0	0	-	0	1	-	0	1
s2		0	0	-	0	0	-	0	0
s3		0	0	-	0	0	-	0	0
Backend		0	0	-	0	1	-	10	1

Hình 26. Source- Thực hiện truy cập ở máy có IP: 192.168.15.131

Thử kết nối (3 lần để test)



Hình 27. Source- Thực hiện truy cập ở máy có IP: 192.168.15.132

	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	
Frontend				0	1	-	1	2	100	3			1 222	1 790	0	
webservers																
	Queue			Session rate			Sessions						Bytes		Denied	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp
s1	0	0	-	0	1		0	1	-	1	1	5m19s	317	459		0
s2	0	0	-	0	0		0	0	-	0	0	?	0	0		0
s3	0	0	-	0	2		0	1	-	3	3	9s	905	1 331		0
Backend	0	0		0	2		0	1	10	4	4	9s	1 222	1 790	0	0

Hình 28. Source-IP: 192.168.15.132- Thực hiện truy cập 3 lần

Từ nhiều lần thực nghiệm, demo cho thấy cùng 01 địa chỉ IP (Source) cho cùng 1 kết quả là: HAProxy cân bằng tải lên các server cùng 1 địa chỉ IP thì tải đều cùng 1 web Server.

Thuật toán uri

Khi một yêu cầu đến, HAProxy sẽ sử dụng URI của yêu cầu (phần đường dẫn sau domain) để hash và tải đến các server. Điều này có nghĩa là cùng một URI sẽ luôn được chuyển hướng đến cùng một server.

Config:

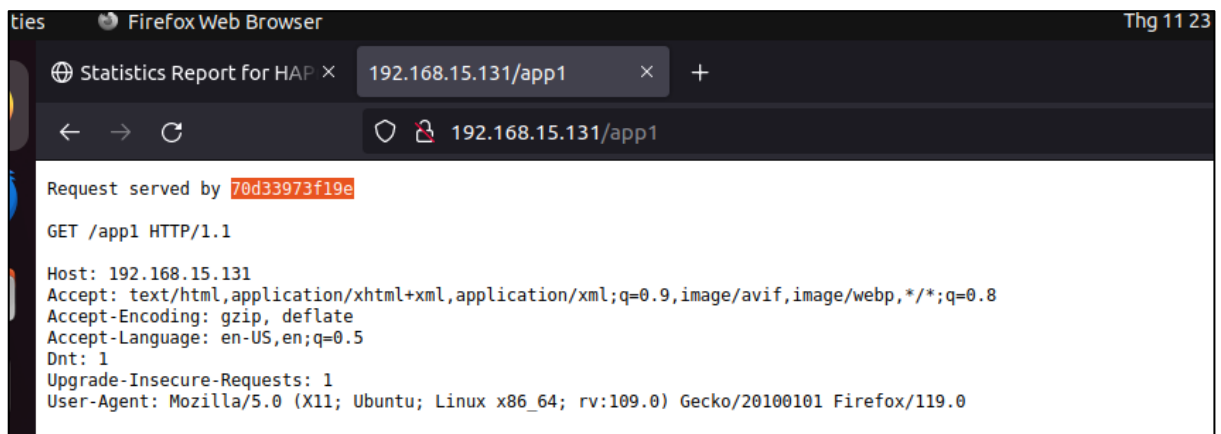
```
# các phần trên không thay đổi.
backend webserver
    balance uri
    hash-type consistent # chọn kiểu hash
    server s1 192.168.15.131:8081 check
    server s2 192.168.15.131:8082 check
    server s3 192.168.15.131:8083 check
```

Tiến hành demo:

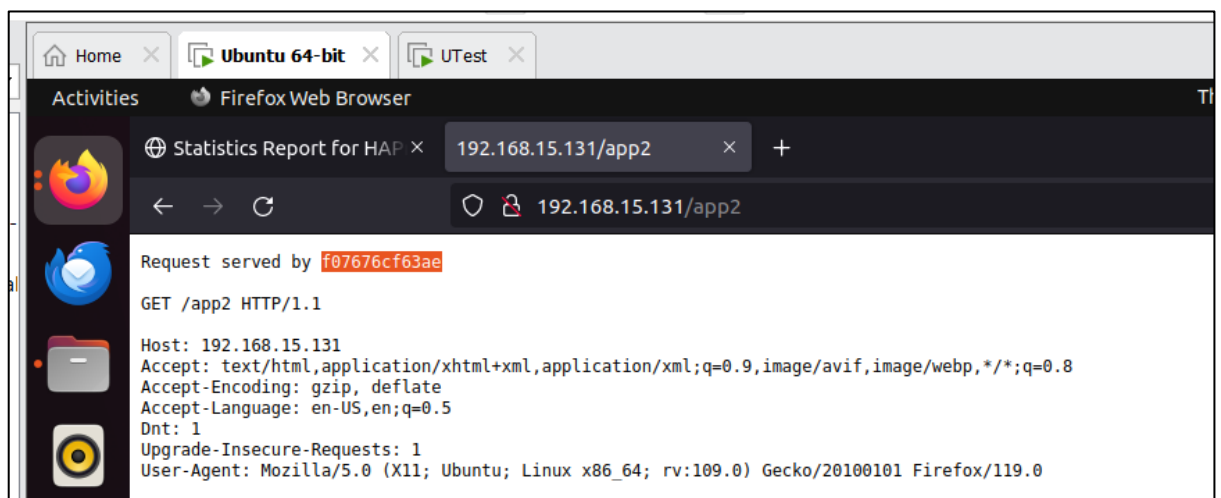
Web1_ID: 70d33973f19e

Web2_ID: f07676cf63ae

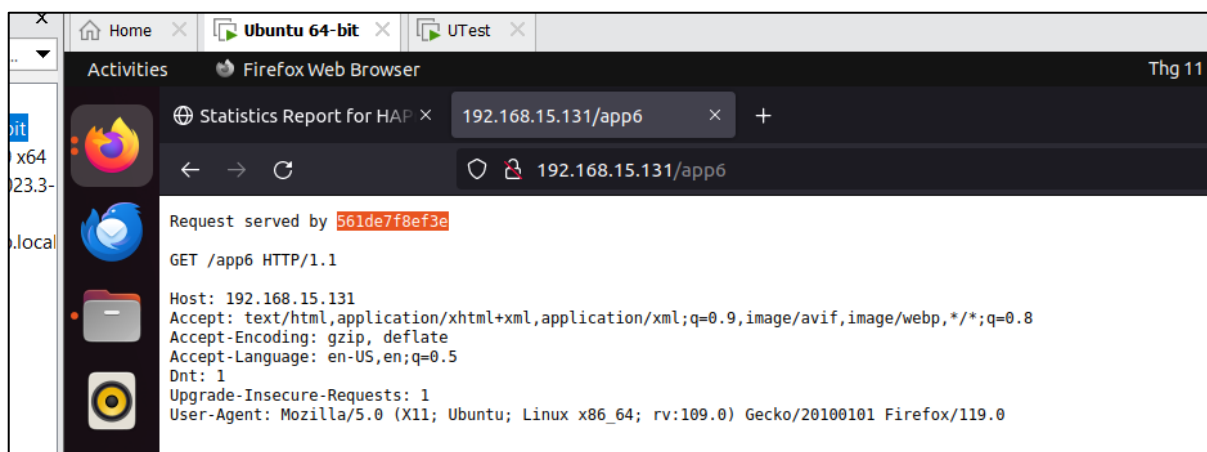
Web3_ID: 561de7f8ef3e



Hình 29. Uri- tải đến web1



Hình 30. Uri- tải đến web2



Hình 31. Uri- tải đến web3

Sau khi thực nghiệm demo nhiều lần, đã chứng tỏ được với cùng 1 Uri sẽ được HAProxy tải đến cùng 01 web server.

Thuật toán url_param

URL parameters (tham số URL) là các thông tin được truyền đi trong một URL sau dấu "?" và được ngăn cách bởi dấu "&". Các tham số này thường được sử dụng để truyền dữ liệu giữa client và server qua URL.

Ví dụ: https://example.com/page?name=John&age=25

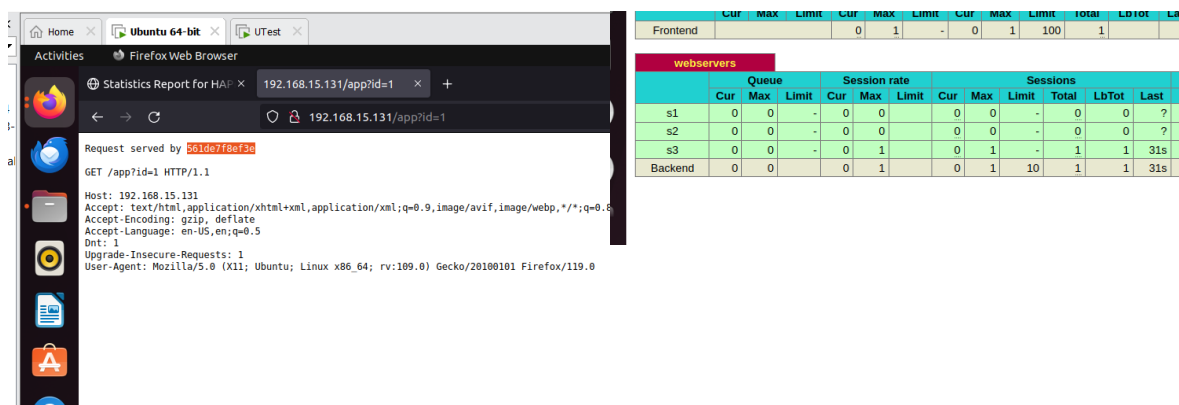
Đối với thuật toán url_param, khi một yêu cầu đến HAProxy sẽ sử dụng phần parameter trong url của yêu cầu để hash và tải đến các server. Điều này có nghĩa là cùng một URI sẽ luôn được chuyển hướng đến cùng một server.

Config:

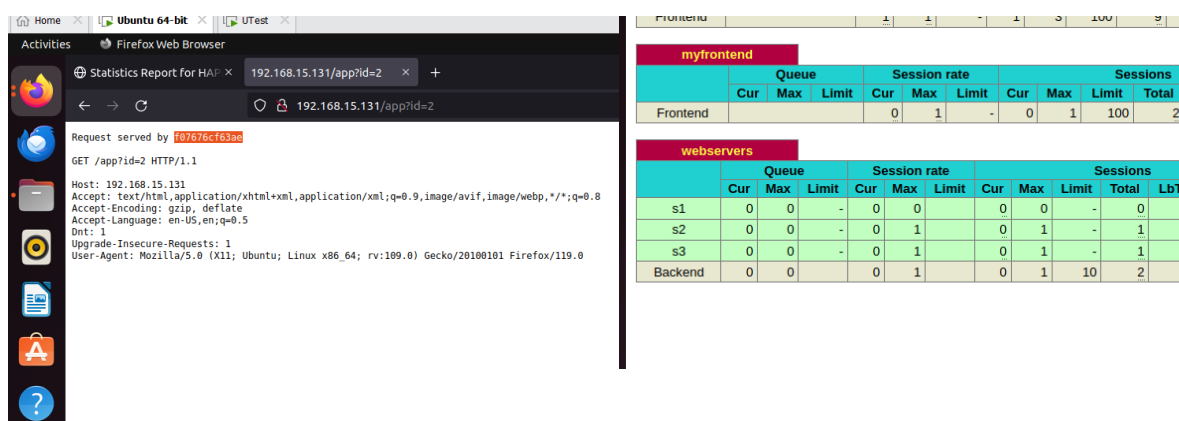
```
# các phần trên không thay đổi.
backend webservers
    balance url_param id
    hash-type cosistent # chọn kiểu hash
    server s1 192.168.15.131:8081 check
    server s2 192.168.15.131:8082 check
    server s3 192.168.15.131:8083 check
```

Tại đây nhóm test với param là id

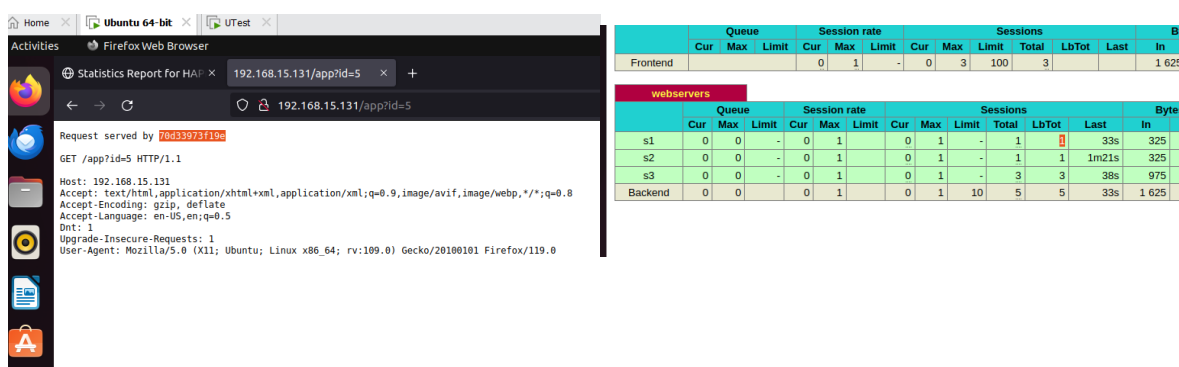
Thực hiện demo:



Hình 32. url_param- id=1 – tải đến web3



Hình 33. url_param- id=2 – tải đến web2



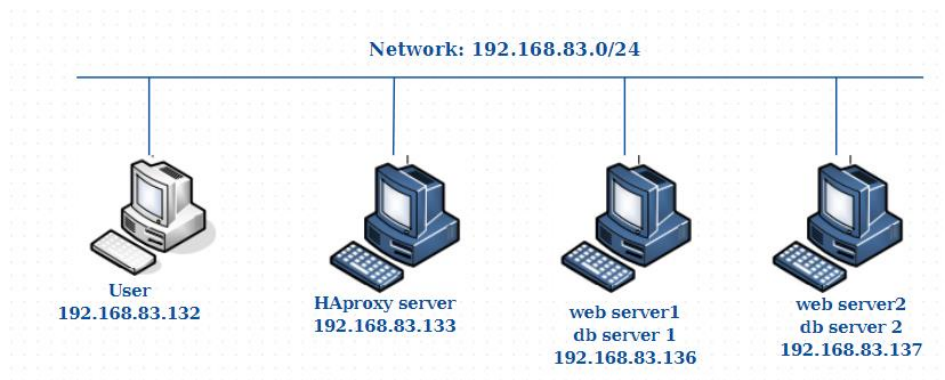
Hình 34. url_param- id=5 – tải đến web1

Sau nhiều lần thực nghiệm thì vẫn đảm bảo chứng minh được cùng url_param thì HAProxy sẽ tải đến cùng 01 server.

➤ Link thực nghiệm demo 06 thuật toán: <https://tinyurl.com/yr2dq17y>

3.2 Demo mô hình cân bằng tải với Webserver và Database

3.2.1 Tài nguyên và setup hệ thống



Hình 35. Mô hình đơn giản HAProxy

a) HAProxy server

- Os: Ubuntu 22.04 LTS
- Service: HAProxy, Mysql

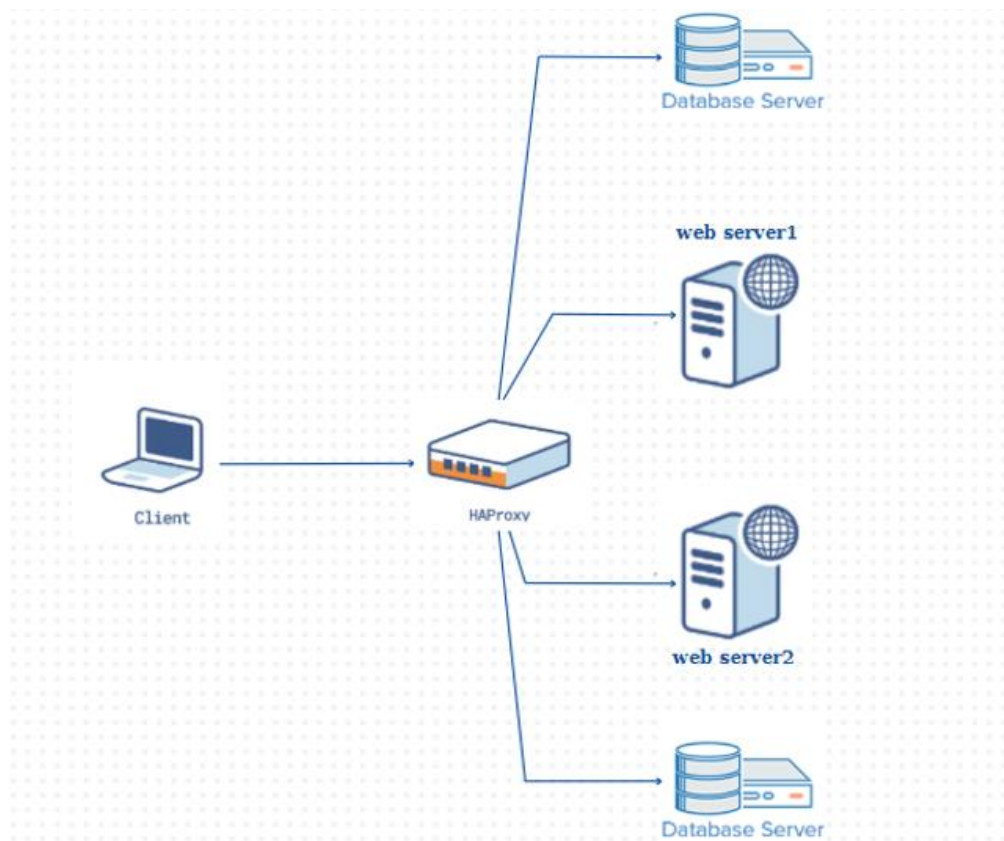
b) Server 1

- OS: Ubuntu 22.04 LTS
- Service: PHP, Mysql, Apache2, Docker

c) Server 2

- OS: Ubuntu 22.04 LTS
- Service: PHP, Mysql, Apache2, Docker

3.2.2. Cài đặt chi tiết



Hình 36. Mô hình HAProxy triển khai cho webserver và database

3.2.2.1. Thiết lập và đồng bộ database - mysql master to master replication

a. Database server 1

- Tạo thư mục gồm các thành phần sau :

```
mysql1/
  config/
    my.cnf
  Dockerfile
```

Hình 37. Thư mục

Trong đó **my.cnf** có nội dung như sau :


```

1  [mysqld]
2  bind-address = 0.0.0.0
3
4  server-id = 1
5  log_bin = /var/log/mysql/mysql-bin.log
6  expire_logs_days = 10
7  max_binlog_size = 100M
8  binlog_do_db = my_db
9  replicate-do-db = my_db
10
11 auto_increment_increment = 10
12 auto_increment_offset = 5

```

Hình 38. my.cnf

Dockerfile:

```

1  FROM mysql:5.7
2  COPY config/my.cnf /etc/mysql/conf.d/my.cnf
3  RUN mkdir /var/log/mysql && chown -R mysql:mysql /var/log/mysql/

```

Hình 39. Dockerfile

- Tạo image:

```
$ docker build -t mysql-my-db:latest .
```

- Sau khi đã tạo image, ta dùng image trên để tạo container

```

$ docker run --name mysql-my-db -d -p 3407:3306
-e MYSQL_ROOT_PASSWORD=pass987
--restart unless-stopped
-v mysql-my-db:/var/lib/mysql mysql-my-db:latest

```

- Kiểm tra kết nối đến mysql:

Dùng lệnh ipconfig để xem Docker IP

```

$ ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255

```

Sử dụng lệnh để kết nối đến mysql:

```
$ mysql -u root -p -h 172.17.0.1 -P 3407
```

Sử dụng password đã tạo ở các bước trước đó để đăng nhập.

- Lưu ý : có thể sử dụng IP của máy chủ để kiểm tra kết nối đến mysql

- Trên máy user, sử dụng câu lệnh : `mysql -u root -p -h 192.168.83.136 -P 3407`

b. Database server 2

- Tạo thư mục gồm các thành phần sau :

```
mysql2/  
    config/  
        my.cnf  
    Dockerfile
```

Hình 40. Thư mục db2

Trong đó **my.cnf** có nội dung như sau :

```
1  [mysqld]  
2  bind-address = 0.0.0.0  
3  
4  server-id = 2  
5  log_bin = /var/log/mysql/mysql-bin.log  
6  expire_logs_days = 10  
7  max_binlog_size = 100M  
8  binlog_do_db = my_db  
9  replicate-do-db = my_db  
10  
11 auto_increment_increment = 10  
12 auto_increment_offset = 1
```

Hình 41. db2- my.cnf

Dockerfile

```
1  FROM mysql:5.7  
2  COPY config/my.cnf /etc/mysql/conf.d/my.cnf  
3  RUN mkdir /var/log/mysql && chown -R mysql:mysql /var/log/mysql/
```

Hình 42. Dockerfile- db2

- Tạo image:

```
$ docker build -t mysql-my-db:latest .
```

- Sau khi đã tạo image, ta dùng image trên để tạo container

```
docker run --name mysql-my-db -d -p 3407:3306  
-e MYSQL_ROOT_PASSWORD=pass987  
--restart unless-stopped  
-v mysql-my-db:/var/lib/mysql mysql-my-db:latest
```

- Kiểm tra kết nối đến mysql:

Dùng lệnh ipconfig để xem Docker IP

```
$ ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
```

Hình 43. docker IP check

Sử dụng lệnh để kết nối đến mysql:

```
mysql -u root -p -h 172.17.0.1 -P 3407
```

Sử dụng password đã tạo ở các bước trước đó để đăng nhập.

- Lưu ý : có thể sử dụng IP của máy chủ để kiểm tra kết nối đến mysql

- Trên máy user, sử dụng câu lệnh :

```
mysql -u root -p -h 192.168.83.137 -P 3407
```

c. Tạo database trên cả hai mysql server

Tạo 1 database tên my_db trên cả 2 server mysql:

```
mysql> create database my_db;
Query OK, 1 row affected (0.01 sec)
```

Hình 44. Tạo DB trên 2 MYSQL Server

Kiểm tra bằng lệnh: show master status;

```
mysql> show master status;
+-----+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+-----+
| mysql-bin.000003 | 321      | my_db        |                   |                   |
+-----+-----+-----+-----+-----+
```

Hình 45. Check MYSQL

d. Tạo replication user và cấp quyền truy cập

- Trên mysql 1, sử dụng câu lệnh :

```
CREATE USER 'replication'@'192.168.83.137' IDENTIFIED BY 'rep987';
```

- Trong đó 192.168.83.137 là IP address của mysql server 2, 'rep987' là password để đăng nhập.

```
GRANT REPLICATION SLAVE ON *.* TO 'replication'@'192.168.83.137';
```

- Tương tự trên mysql 2:

```
CREATE USER 'replication'@'192.168.83.136' IDENTIFIED BY 'rep987';
```

- Trong đó 192.168.83.136 là IP address của mysql server 1, 'rep987' là password để đăng nhập.

```
GRANT REPLICATION SLAVE ON *.* TO 'replication'@'192.168.83.136';
```

e. Đồng bộ dữ liệu sử dụng **CHANGE MASTER TO**

- CHANGE MASTER TO là câu lệnh dùng để sao chép dữ liệu.

Phương pháp pulling (lấy dữ liệu binlog). Hiểu đơn giản nó hoạt động giống như git pull.

- Trên mysql 1, chạy lệnh sau:

```
- CHANGE MASTER TO MASTER_HOST = '192.168.83.137',
- MASTER_USER = 'replication',
- MASTER_PASSWORD = 'rep987',
- MASTER_PORT = 3407,
- MASTER_LOG_FILE = 'mysql-bin.000003',
- MASTER_LOG_POS = 321;
-
```

- Trên mysql 2, chạy lệnh sau:

```
- CHANGE MASTER TO MASTER_HOST = '192.168.83.136',
- MASTER_USER = 'replication',
- MASTER_PASSWORD = 'rep987',
- MASTER_PORT = 3407,
- MASTER_LOG_FILE = 'mysql-bin.000003',
- MASTER_LOG_POS = 321;
```

- Kiểm tra cấu hình trên cả 2 mysql server bằng lệnh : `show slave status\G;`
- Kiểm tra xem giá trị của các trường thông tin: Master_Host, Master_User, Master_Port, Read_Master_Log_Pos, Relay_Master_Log_File, Replicate_Do_DB .
- Start slave trên cả 2 mysql server bằng lệnh: `start slave;`
- Hiện tại 2 mysql server đã đồng bộ với nhau. Để kiểm tra, ta có thể thêm, xóa, sửa dữ liệu trên 1 server và kiểm tra ở server còn lại.

* Tạo tài khoản cho việc triển khai HAProxy sau này:

```
Insert into mysql.user(Host,User,ssl_cipher,x509_issuer,x509_subject) values ('IP_HAproxy','USER','abc','abc','abc');
```

- Ở đây, IP_HAproxy là 192.168.83.133
- Chúng ta sẽ tạo 2 tài khoản:
 - + haproxy_checkstatus : dùng cho HAproxy check status
 - + super : dùng để connect và sử dụng database mysql.
- Với user 'super', ta sẽ cấp đầy đủ quyền bằng lệnh:

```
GRANT ALL PRIVILEGES ON *.* TO 'super'@'192.168.83.133' IDENTIFIED BY '12345' WITH GRANT OPTION;
```

Lưu ý : chạy lệnh `FLUSH PRIVILEGES;` để cập nhật.

3.2.2.2. Tạo trang web cơ bản sử dụng mysql database.

Ở trên web server 1 và web server 2, sau khi đã cài đặt các dịch vụ như yêu cầu.

Ta tiến hành tạo html file như sau :

```
$ sudo nano /var/www/html/form.html
```

```
<head>
<title>
Test Page
</title>
</head>
<body>
<form action="form_submit.php" class="alt" method="POST">
<div class="row uniform">
<div class="name">
<input name="name" id="" placeholder="Name" type="text">
</div>
<div class="email">
<input name="email" placeholder="Email" type="email">
</div>
<div class="message">
<textarea name="message" placeholder="Message" rows="4"></textarea>
</div>
</div>
<br/>
<input class="alt" value="Submit" name="submit" type="submit">
</form>
</body>
```

Hình 46. Page-html

Ở trên mysql server, tạo table tên test trong database my_db đã tạo trước đó:

```

use my_db;
create table test(
    id int NOT NULL AUTO_INCREMENT,
    name varchar(255),
    email varchar(255),
    message text,
    PRIMARY KEY (id)
);

```

Tạo file php để xử lý chức năng cho trang web trên :

```
$ sudo nano /var/www/html/form_submit.php
```

```

1 <?php
2 $host = "192.168.83.133";
3 $db_name = "my_db";
4 $username = "super";
5 $password = "12345";
6 $connection = null;
7 try{
8     $connection = new PDO("mysql:host=" . $host . ";dbname=" . $db_name, $username, $password);
9     $connection->exec("set names utf8");
10 }catch(PDOException $exception){
11     echo "Connection error: " . $exception->getMessage();
12 }
13
14 function saveData($name, $email, $message){
15     global $connection;
16     $query = "INSERT INTO test(name, email, message) VALUES( :name, :email, :message)";
17
18     $callToDb = $connection->prepare( $query );
19     $name=htmlspecialchars(strip_tags($name));
20     $email=htmlspecialchars(strip_tags($email));
21     $message=htmlspecialchars(strip_tags($message));
22     $callToDb->bindParam(":name",$name);
23     $callToDb->bindParam(":email",$email);
24     $callToDb->bindParam(":message",$message);
25
26     if($callToDb->execute()){
27         return '<h3 style="text-align:center;">We will get back to you very shortly!</h3>';
28     }
29 }
30
31 if( isset($_POST['submit'])){
32     $name = htmlentities($_POST['name']);
33     $email = htmlentities($_POST['email']);
34     $message = htmlentities($_POST['message']);
35
36     //then you can use them in a PHP function.
37     $result = saveData($name, $email, $message);
38     echo $result;
39 }
40 else{
41     echo '<h3 style="text-align:center;">A very detailed error message ( ◡ ◡ )</h3>';
42 }
43 ?>

```

Hình 47. File .php

Trong đó:

- \$host = "HA proxy server IP";
- \$db_name = "database name";
- \$username = "user"; (ở đây sử dụng user super đã tạo ở bước trước đó)
- \$password = "password"; (sử dụng mật khẩu của user super – 12345)

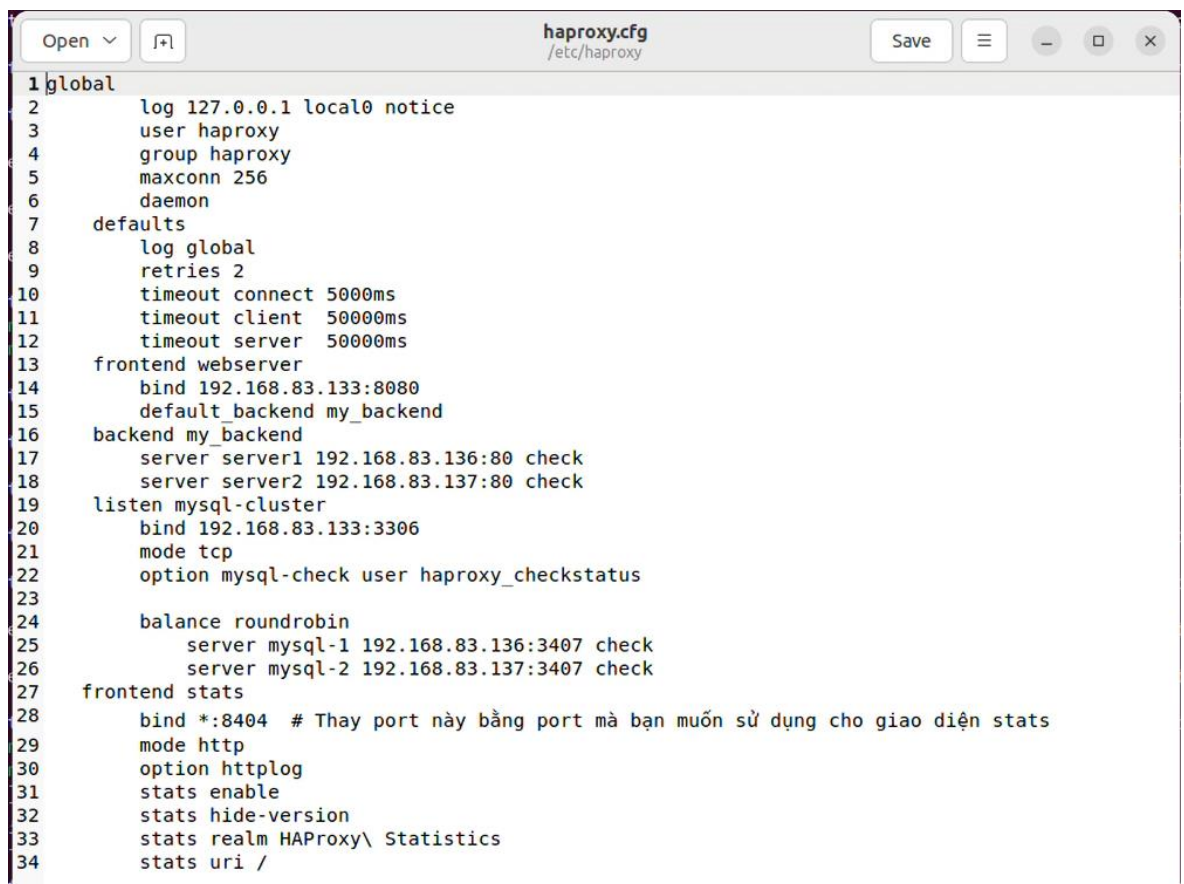
Sau khi đã cấu hình xong các file nói trên, khởi động apache2 để có thể truy cập trang web.

3.2.2.3. Cấu hình HAProxy cho web server và database

Trên máy chủ HAProxy, sau khi đã cài đặt các dịch vụ như yêu cầu. Ta sẽ khởi động HAProxy: `sudo service haproxy start`

Tiến hành cấu hình bằng cách thay đổi nội dung file haproxy.cfg :

```
$ sudo nano /etc/haproxy/haproxy.cfg
```

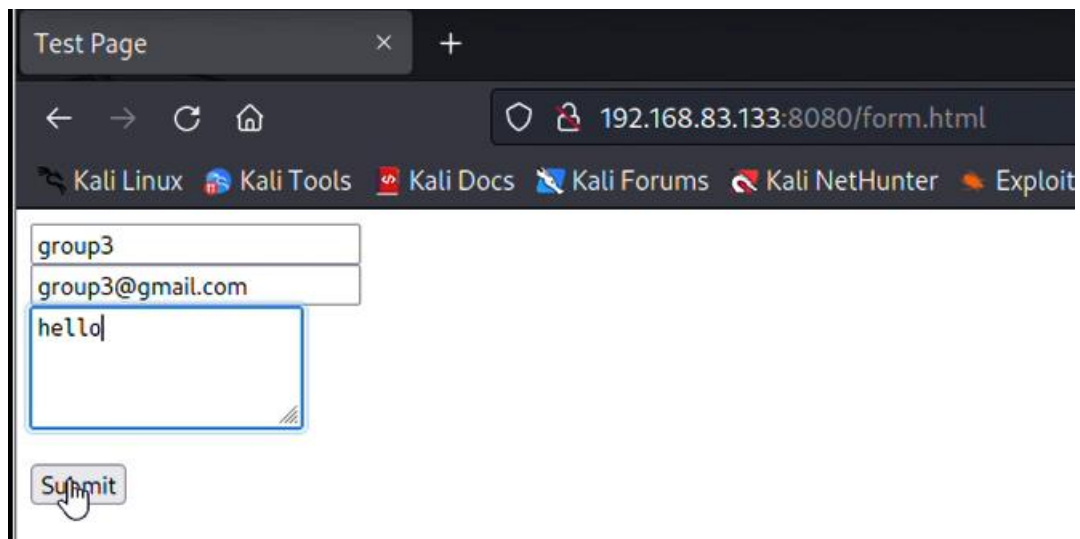


```
1 global
2     log 127.0.0.1 local0 notice
3     user haproxy
4     group haproxy
5     maxconn 256
6     daemon
7
8 defaults
9     log global
10    retries 2
11    timeout connect 5000ms
12    timeout client 50000ms
13    timeout server 50000ms
14
15 frontend webserver
16     bind 192.168.83.133:8080
17     default_backend my_backend
18
19 backend my_backend
20     server server1 192.168.83.136:80 check
21     server server2 192.168.83.137:80 check
22
23 listen mysql-cluster
24     bind 192.168.83.133:3306
25     mode tcp
26     option mysql-check user haproxy_checkstatus
27
28 balance roundrobin
29     server mysql-1 192.168.83.136:3407 check
30     server mysql-2 192.168.83.137:3407 check
31
32 frontend stats
33     bind *:8404 # Thay port này bằng port mà bạn muốn sử dụng cho giao diện stats
34     mode http
35     option httplog
36     stats enable
37     stats hide-version
38     stats realm HAProxy\ Statistics
39     stats uri /
```

Hình 48. haproxy.cfg- demo2

Khởi động lại dịch vụ haproxy và bắt đầu chạy thử nghiệm.

Sử dụng máy user để truy cập vào trang web thông qua địa chỉ IP của haproxy với port **8080** đã cấu hình và thực hiện các thao tác truy vấn dữ liệu.



Hình 49. Test user

Có thể kiểm tra tình trạng của các web server và mysql server bằng cách truy cập tại: <http://192.168.83.133:8404/>

Statistics Report for HAProxy

192.168.83.133:8040

☆

HAProxy

Statistics Report for pid 5188

> General process information

active UP

active UP going down

active DOWN

active or backup DOWN

active or backup DOWN for maintenance (MAINT)

active or backup SOFT STOPPED for maintenance

Note: "NOLEB/DRAIN" = UP with load-balancing disabled.

backup UP

backup UP going down

backup DOWN, going up

not checked

Display option:

Scope:

Hide DOWN servers

Refresh page

CSV export

JSON export (schemas)

External resources:

HAProxy site

Updater (v2.5)

Online manual

webserver

Queue				Session rate				Sessions				Bytes		Denied		Errors		Warnings		Status		Server						
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle
Frontend																												
0	0	-	0	0	-	0	0	-	0	0	256	0	0	0	0	0	0	0	0	0	OPEN							

my_backend

Queue				Session rate				Sessions				Bytes		Denied		Errors		Warnings		Status		Server						
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle
server1																												
0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	4s UP	L7OK in 1ms	1/1	Y	-	0	0	0s	-
server2																												
0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	4s UP	L7OK in 1ms	1/1	Y	-	0	0	0s	-
Backend																												
0	0	-	0	0	-	0	0	-	0	0	26	0	0	?	0	0	0	0	0	4s UP		2/2	2	0	0	0	0s	-

mysql-cluster

Queue				Session rate				Sessions				Bytes		Denied		Errors		Warnings		Status		Server						
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle
Frontend																												
0	0	-	0	0	-	0	0	-	0	0	256	0	0	0	0	0	0	0	0	OPEN								
mysql-1																												
0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	4s UP	L7OK in 3ms	1/1	Y	-	0	0	0s	-
mysql-2																												
0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	4s UP	L7OK in 2ms	1/1	Y	-	0	0	0s	-
Backend																												
0	0	-	0	0	-	0	0	-	0	0	26	0	0	?	0	0	0	0	0	4s UP		2/2	2	0	0	0	0s	-

stats

Queue				Session rate				Sessions				Bytes		Denied		Errors		Warnings		Status		Server						
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle
Frontend																												
1	1	-	1	1	-	1	1	-	1	1	256	1	0	0	45	198	0	0	0	0	OPEN							

Hình 50. Truy cập trang quản trị HAProxy- demo2

3.2.2.4 Thử nghiệm hệ thống

Đối với mô hình trên, có 4 kịch bản thử nghiệm được đặt ra, bao gồm:

- Truy cập và sử dụng dịch vụ khi cả 2 web server và 2 database server hoạt động.
- Truy cập và sử dụng dịch vụ khi 2 web server và chỉ 1 database server còn hoạt động.

- Truy cập và sử dụng dịch vụ khi chỉ 1 web server và 2 database server còn hoạt động.
- Truy cập và sử dụng dịch vụ khi chỉ 1 web server và 1 database server còn hoạt động.

a) Cả 2 web server và 2 database server hoạt động

Truy cập HAProxy statistics report để kiểm tra tình trạng của các server

Statistics Report for HAProxy

192.168.83.133:8404

Statistics Report for pid 4858

General process information

pid = 4858 (process #1, nbproc = 1, nbthread = 4)
uptime = 0d 0h0m0s
system limits: memmax = unlimited, ulimit-n = 563
maxsock = 563, maxconn = 256, maxpipes = 0
current conns = 1, current pipes = 0, conn rate = 1/sec, bit rate = 1.087 kbps
Running tasks: 0/25, idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLEAF DRAIN" = UP with load-balancing disabled.

Display option:
Scope:
Hide DOWN servers
Refresh now
CSV export
JSON export (schema)

External resources:
Primary site
Updates (v2.4)
Online manual

webserver

	Queue		Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		LastChk		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LibTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Sts	LastChk	Wght	Act	Bck	Chk	Den	Downtime	Thrtle
Frontend	0	0	-	0	0	-	0	0	0	256	0	0	0	0	0	0	0	0	0	0	0	OPEN								

my backend

	Queue		Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		LastChk		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LibTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Sts	LastChk	Wght	Act	Bck	Chk	Den	Downtime	Thrtle
server1	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7s UP	L7OK in 2ms	1/1	Y	-	0	0	0s	-
server2	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7s UP	L7OK in 1ms	1/1	Y	-	0	0	0s	-
Backend	0	0	-	0	0	-	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	7s UP		2/2	2	0	0	0	0s	-

mysql-cluster

	Queue		Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		LastChk		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LibTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Sts	LastChk	Wght	Act	Bck	Chk	Den	Downtime	Thrtle
Frontend	0	0	-	0	0	-	0	0	0	256	0	0	0	0	0	0	0	0	0	0	0	OPEN								
mysql-1	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7s UP	L7OK in 2ms	1/1	Y	-	0	0	0s	-
mysql-2	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7s UP	L7OK in 4ms	1/1	Y	-	0	0	0s	-
Backend	0	0	-	0	0	-	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	7s UP		2/2	2	0	0	0	0s	-

stats

	Queue		Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		LastChk		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LibTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Sts	LastChk	Wght	Act	Bck	Chk	Den	Downtime	Thrtle
Frontend	1	1	-	1	1	-	1	1	1	256	1	0	0	0	0	0	0	0	0	0	0	OPEN								

Hình 51. HAProxy stat report 2 web- 2 server

Khi đã thấy các server hoạt động bình thường, thực hiện truy cập web

Test Page

192.168.83.133:8080/form.html

group3

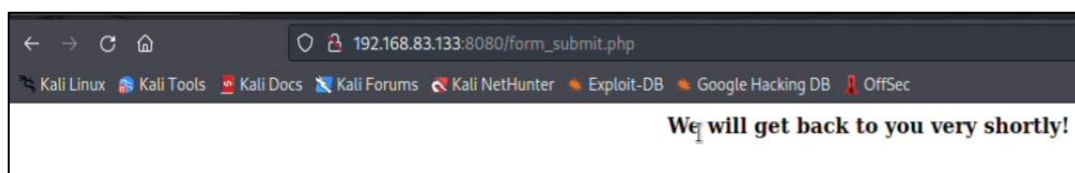
group3@gmail.com

hello

Submit

Hình 52. Test web server

Thực hiện nhập thông tin và gửi đi nhằm ghi dữ liệu vào database:



Hình 53. Kết quả- submit thông tin trên web

Kiểm tra cả 2 database server và nhận thấy dữ liệu vừa nhập đã được thêm vào ở cả 2 server:

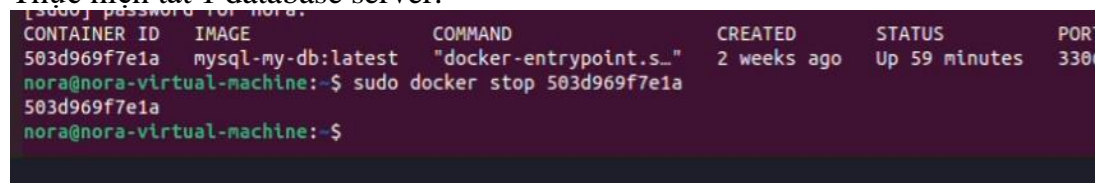


Hình 54. Check 2 database

Như vậy, hệ thống hoạt động tốt và 2 database đã được đồng bộ với nhau.

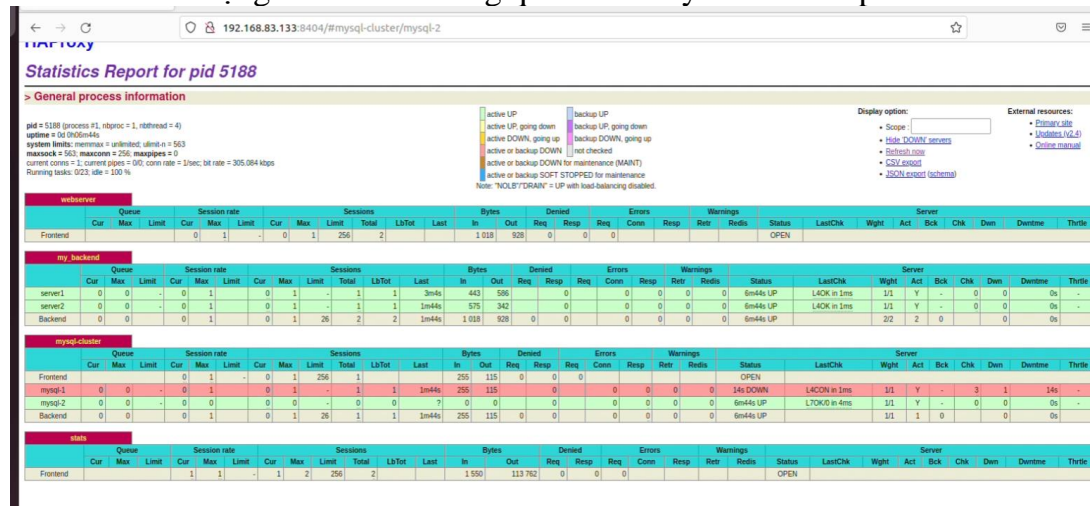
b) 2 web server và chỉ 1 database server còn hoạt động.

Thực hiện tắt 1 database server:



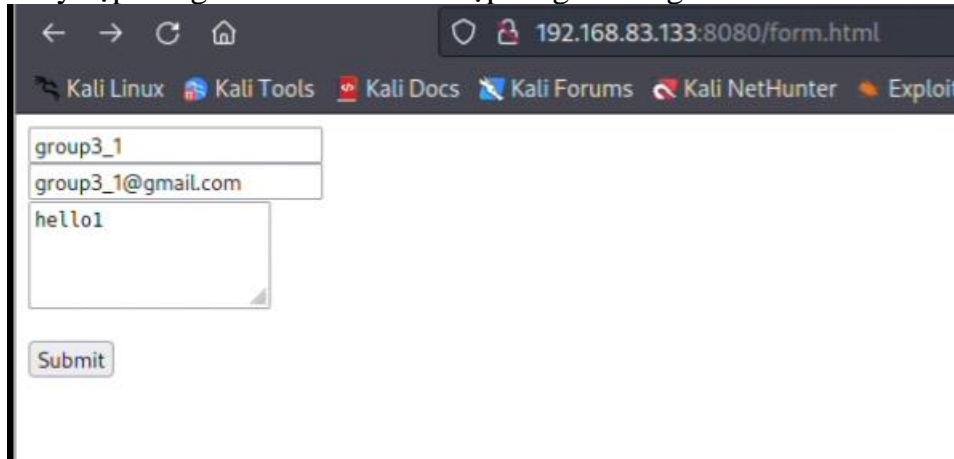
Hình 55. Tắt 1 database server

Kiểm tra tình trạng các server thông qua HAProxy statistics report



Hình 56. Check status khi tắt 1 Database server

Truy cập trang web và thao tác nhập và gửi thông tin



The screenshot shows a web browser window with the address bar displaying `192.168.83.133:8080/form.html`. The browser's tab bar includes links to 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', and 'Exploit'. The form itself has three input fields: the first contains 'group3_1', the second contains 'group3_1@gmail.com', and the third contains 'hello1'. Below these fields is a 'Submit' button.

Hình 57. Truy cập web- nhập lại thông tin

Truy vấn dữ liệu trên database 2, ta thấy dữ liệu vừa gửi đã xuất hiện



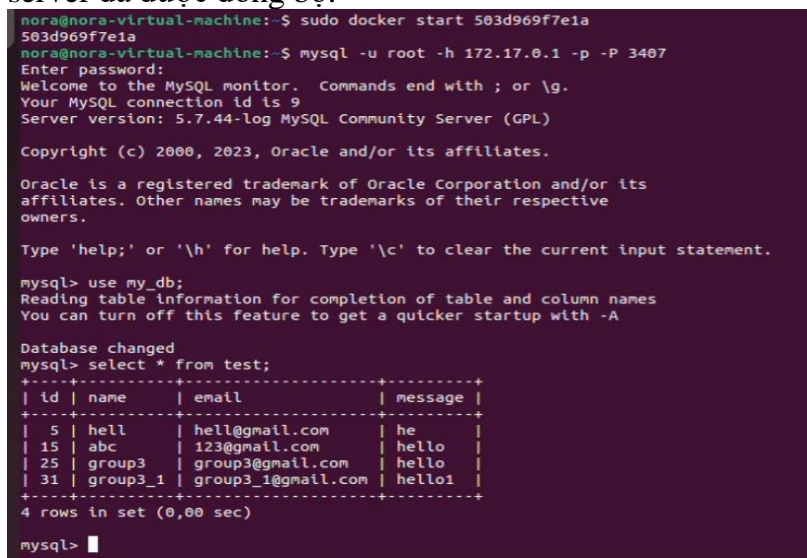
```
mysql> select * from test;
```

id	name	email	message
5	hell	hell@gmail.com	he
15	abc	123@gmail.com	hello
25	group3	group3@gmail.com	hello
31	group3_1	group3_1@gmail.com	hello1

4 rows in set (0,00 sec)

Hình 58. Truy vấn lại dữ liệu ở database 2

Khởi động lại database 1 sau đó truy vấn dữ liệu, ta thấy dữ liệu được thêm vào khi database server 1 ngừng hoạt động cũng đã xuất hiện do 2 database server đã được đồng bộ.



```
nora@nora-virtual-machine:~$ sudo docker start 503d969f7e1a
503d969f7e1a
nora@nora-virtual-machine:~$ mysql -u root -h 172.17.0.1 -p -P 3407
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.44-log MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use my_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from test;
```

id	name	email	message
5	hell	hell@gmail.com	he
15	abc	123@gmail.com	hello
25	group3	group3@gmail.com	hello
31	group3_1	group3_1@gmail.com	hello1

4 rows in set (0,00 sec)

```
mysql>
```

Hình 59. Check lại database 1

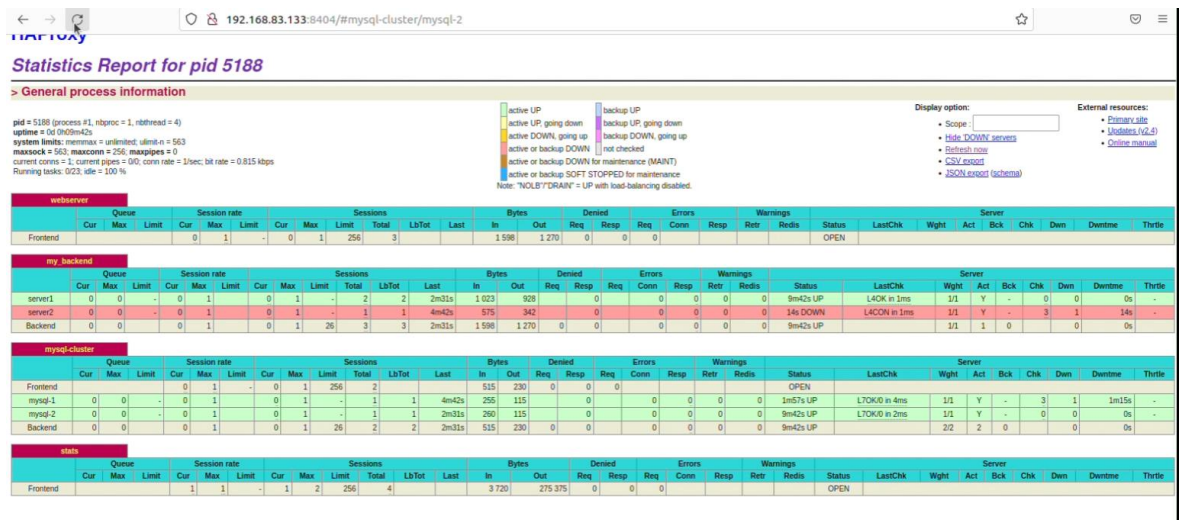
c) Khi chỉ 1 web server và 2 database server còn hoạt động.

Tất 1 web server

```
nora@nora-virtual-machine:~$ service apache2 stop
nora@nora-virtual-machine:~$ service apache2 status
○ apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Wed 2023-11-22 17:05:45 +07; 4s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 812 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Process: 3774 ExecStop=/usr/sbin/apachectl graceful-stop (code=exited, status=0/SUCCESS)
   Main PID: 1161 (code=exited, status=0/SUCCESS)
     CPU: 1.255s

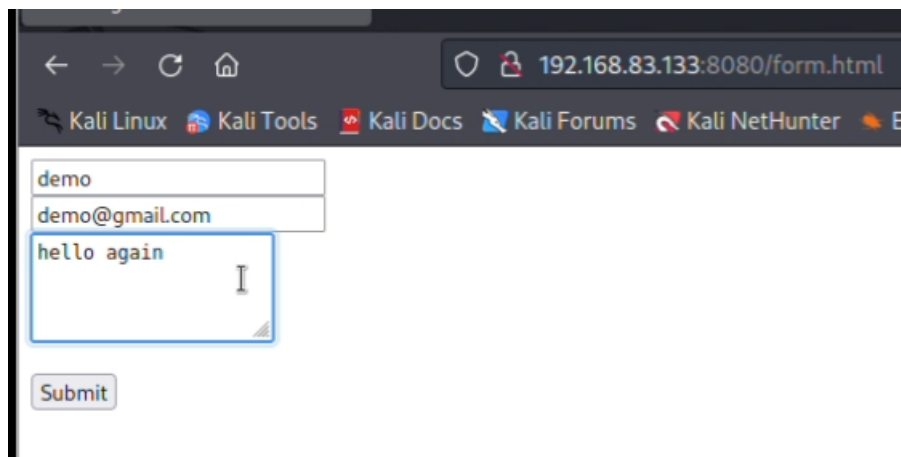
Thg 11 22 16:02:58 nora-virtual-machine systemd[1]: Starting The Apache HTTP Server...
Thg 11 22 16:03:01 nora-virtual-machine apachectl[845]: AH00558: apache2: Could not reliably determine the server's
Thg 11 22 16:03:01 nora-virtual-machine systemd[1]: Started The Apache HTTP Server.
```

Hình 60. Thực hiện tắt 1 web server

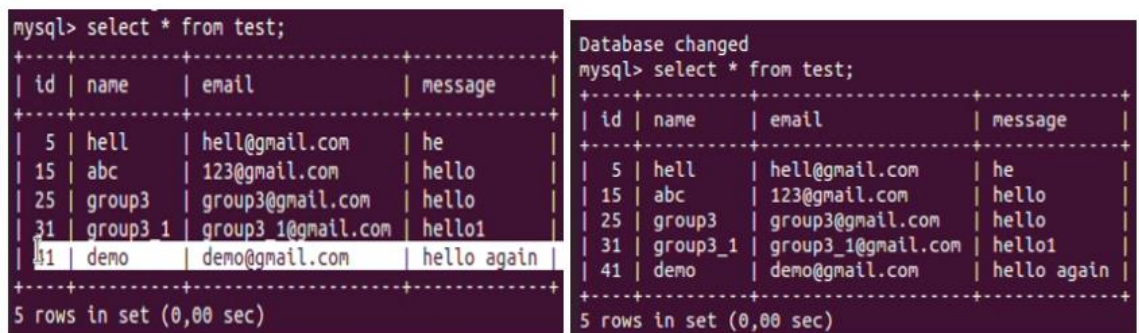


Hình 61. Check status sau khi tắt web server

Sau khi nhập và gửi thông tin khi truy cập web, dữ liệu được ghi vào database và được đồng bộ ở cả 2 server



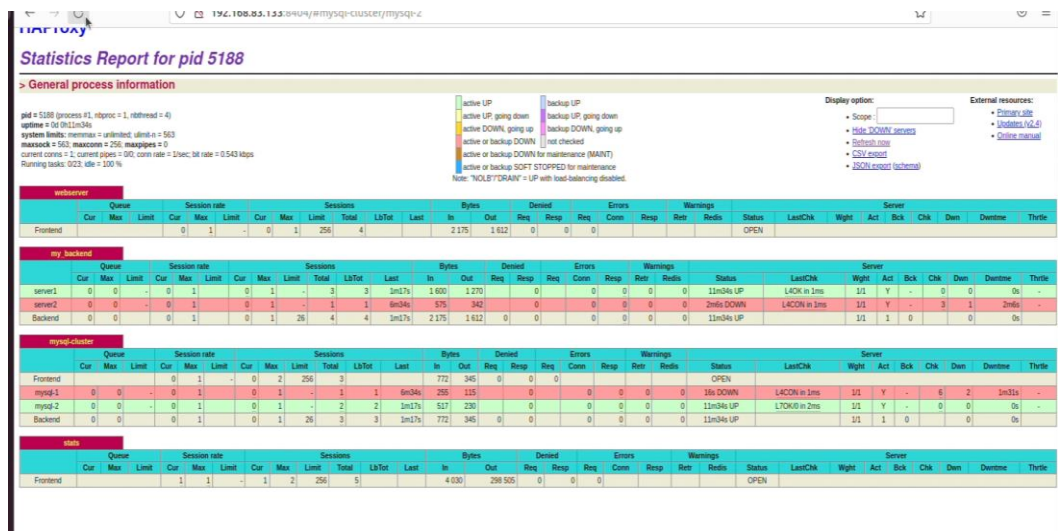
Hình 62. Thực hiện submit dữ liệu lên web



Hình 63. Kiểm tra 2 database

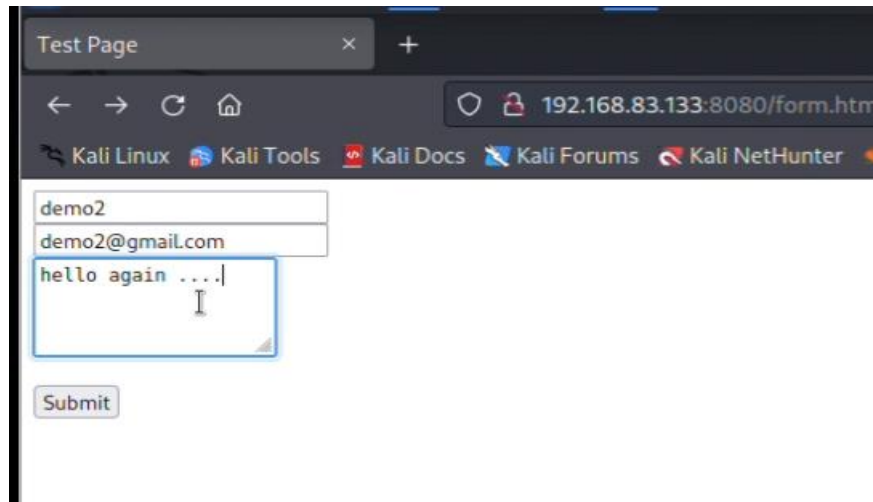
d) Khi chỉ 1 web server và 1 database server còn hoạt động.

Thực hiện tắt 1 web server và 1 database server:



Hình 64. Check status khi tắt 1 web và 1 server

Sau khi đã kiểm tra tình trạng của các server trong hệ thống, thực hiện truy cập web và gửi thông tin.



Test Page

192.168.83.133:8080/form.htm

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter

demo2

demo2@gmail.com

hello again

Submit

Hình 65. Submit dữ liệu lên web

Truy xuất và đối chiếu dữ liệu trên database server

```
mysql> select * from test;
```

id	name	email	message
5	hell	hell@gmail.com	he
15	abc	123@gmail.com	hello
25	group3	group3@gmail.com	hello
31	group3_1	group3_1@gmail.com	hello1
41	demo	demo@gmail.com	hello again
51	demo2	demo2@gmail.com	hello again

```
6 rows in set (0,00 sec)
```

Hình 66. Check trên database server

REFERENCES

- Hap. Technologies, “Backends,” HAProxy config tutorials, <https://www.haproxy.com/documentation/haproxy-configuration-tutorials/core-concepts/backends/> (accessed Dec. 1, 2023).
- “Summary,” HAProxy Enterprise Documentation version 2.8r1 (1.0.0-307.317) - Configuration Manual, <https://www.haproxy.com/documentation/haproxy-configuration-manual/latest/#2.1> (accessed Dec. 1, 2023).
- M. Anicas, “An introduction to HAProxy and load balancing concepts,” DigitalOcean, <https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts> (accessed Dec. 1, 2023).
- D. Nek, “How to setup haproxy as load balancer for Apache on ubuntu,” Linux Tutorials for Beginners, <https://webhostinggeeks.com/howto/how-to-setup-haproxy-as-load-balancer-for-apache-on-ubuntu/> (accessed Dec. 1, 2023).
- “How to setup haproxy load balance server for sharing web traffic,” Online Tutorials, Courses, and eBooks Library, <https://www.tutorialspoint.com/how-to-setup-haproxy-load-balance-server-for-sharing-web-traffic> (accessed Dec. 1, 2023).
- A. Hasbiyatmoko, “MySQL master to master replication,” latcoding.com, <https://www.latcoding.com/2023/07/18/sharing-pengalaman-setup-mysql-master-to-master-replication-dengan-docker-mysql-5-7/> (accessed Dec. 1, 2023).
- Devart, “How to install mysql on ubuntu: Advanced manual,” Devart, <https://www.devart.com/dbforge/mysql/how-to-install-mysql-on-ubuntu/> (accessed Dec. 1, 2023).
- A. Alam, “Create a web server and save form data into mysql database using PHP (Beginners Guide),” DEV Community, <https://dev.to/satellitebots/create-a-web-server-and-save-form-data-into-mysql-database-using-php-beginners-guide-fah> (accessed Dec. 1, 2023).
- M. Dancuk, “How to configure and use HAProxy for load balancing,” Knowledge Base by phoenixNAP, <https://phoenixnap.com/kb/haproxy-load-balancer> (accessed Dec. 1, 2023).
- Đỗ Hoàng Minh Hưng, “Xây dựng Loadbalancer Cho Các Server mysql Với HAproxy Trên Ubuntu,” Viblo, <https://viblo.asia/p/xay-dung-loadbalancer-cho-cac-server-mysql-voi-haproxy-tren-ubuntu-Az45bpeOZxY> (accessed Dec. 1, 2023).
- N. Ramirez, “How to run HAProxy with Docker (in-depth guide),” HAProxy Technologies, <https://www.haproxy.com/blog/how-to-run-haproxy-with-docker> (accessed Nov. 29, 2023).
- M. Mhedhbi, “Dynamic configuration with the Haproxy runtime API,” HAProxy Technologies, <https://www.haproxy.com/blog/dynamic-configuration-haproxy-runtime-api> (accessed Nov. 29, 2023).