**Chương II: Tổng quan về cân bằng tải**

**1. Khái niệm cân bằng tải**

Cân bằng tải là một phương pháp phân phối khối lượng tải trên nhiều máy tính hoặc một cụm máy tính để có thể sử dụng tối ưu các nguồn lực, tối đa hóa thông lượng , giảm thời gian đáp ứng và tránh tình trạng quá tải trên máy chủ. Là cơ chế định tuyến các gói tin qua các đường có metric bằng nhau. Cân bằng tải dùng để chia sẽ dữ liệu truyền trên mạng giúp cho việc truyền tải thông suốt, không bị nghẽn mạng do quá tải hay do một sự cố nào đó. Hoặc khi có một máy server nào đó bị trục trặc sẽ có máy server khác thay thế giúp nhận dữ liệu thay thế cho server bị trục trặc đó, giúp cho việc truyền tải không bị ngừng do máy server bị lỗi đó gây ra.

**2. Lợi ích cân bằng tải**

\* Tăng khả năng đáp ứng , tránh tình trạng quá tải trên máy chủ , đảm bảo tính linh hoạt và mở rộng cho hệ thống.

\* Tăng độ tin cậy và khả năng dự phòng cho hệ thống: Sử dụng cân bằng tải giúp tăng tính HA ( High Availability ) cho hệ thống, đồng thời đảm bảo cho người dùng không bị gián đoạn dịch vụ khi xảy ra lỗi sự cố lỗi tại một thời điểm cung cấp dịch vụ.

\* Tăng tính bảo mật cho hệ thống: Thông thường khi người dùng gửi yêu cầu dịch vụ hệ thống, yêu cầu đó sẽ được xử lý trên bộ cân bằng tải, sau đó thành phần cân bằng tải mới chuyển tiếp các yêu cầu cho các máy chủ bên trong. Quá trình trả lờ cho khách hàng thông qua thành phần cân bằng tải , vì vậy mà người dùng không thể biết được chính xác các máy chủ bên trong cũng như như phương pháp phân tải được sử dụng. Bằng cách này có thể ngăn chặn người dùng giao tiếp trực tiếp với các máy chủ , ẩn các thông tin và cấu trúc mạng nội bộ, ngăn ngừa các cuộc tấn công trên mạng hoặc các dịch vụ không liên quan đang hoạt động trên các cổng khác.

**3. Kỷ thuật cân bằng tải**

Như chúng ta đã biết , bộ cân bằng tải có nhiệm vụ kết nối giữa người dùng và server, do đó nó có thể hoạt động như một proxy hoặc gateway. Một proxy có nhiệm vụ luân chuyển yêu cầu và dữ liệu đáp trã giữa người dùng và server, trong khi đó một gateway chỉ có nhiệm vụ tạo ra một kết nối hai đối tượng này và không làm gì thêm. Có thể sử dụng phần cứng hoặc phần mềm được cài đặt trên một front server hoặc trên chính web server. Thêm nữa, khi số lượng ngườì dùng tăng lên , để tránh SPOF ( Single point of failure), cần thiết phải cài đặt hai bộ cân bằng tải song song , hoạt động theo cơ chế active-active hoặc active-standby.

**3.1 Kiểm tra trạng thái server**

Để chọn được server phù hợp để gửi request, bộ cân bằng tảu cần phải biết được server nào đang có sẵn. Vì vậy, nó cần phải dùng biện pháp nào đó để kiểm tra trạng thái của server, chẳng hạn như gửi lệnh ping, các yêu cầu, thử kết nối hay bất cứ phương pháp nào mà người quản trị nghĩ là dùng được. Kỹ thuật kiểm tra này thường được gọi là “ health checks” .

Một server bị down có thể trả lời lệnh ping nhưng không thể trả lời các kết nối TCP, một server bị treo có khả năng trả lời kết nối TCP nhưng không thể trả lời các yêu cầu HTTP. Khi một ứng dụng web nhiều lớp được kích hoạt, một số yêu cầu HTTP có thể trả lời ngay lập tức trong khi số khác sẽ thất bại .

Chính vì thế, việc chọn một phương pháp test phù hợp được chấp nhận bởi ứng dụng web và bộ cân bằng tải là rất thú vị. Một số test đôi khi phải cần truy suất dữ liệu database nhằm đảm bảo rằng toàn bộ quá trình đó là đúng. Hạn chế lớn nhất là những phương pháp kiểm tra này sẽ chiếm tài nguyên của hệ thống như là CPU, threads…

Do đó, cân bằng thời gian kiểm tra chính là vấn đề khó nhất trong kỹ thuật lựa chọn server. Khoảng thời gian giữa 2 lần test liên tiếp phải đủ dài để không tốn quá nhiều tài nguyên của hệ thống và cũng cần đủ ngắn để nhanh chóng phát hiện ra những server “chết “ . Vì “ health checks” là một trong những khía cạnh phức tạp nhất của kỹ thuật cân bằng tải, nên thường sau một vài kiểm tra , các nhà phát triển ứng dụng sẽ thực thi một yêu cầu đặc biêt dành riêng cho bộ cân bằng tảu, giúp cho nó thực hiện một số kiểm tra nội bộ.

Phần mềm cân bằng tải có khả năng cung cấp scripting , do đó nó đạt được độ linh hoat rất cao. Thêm nữa, nếu như một bài kiểm tra nào đó đòi hỏi phải chỉnh sửa code nó có thể thực hiện trong một khoảng thời gian ngắn.

**3.2 Lựa chọn server tốt nhất**

Phương pháo dễ nhất thường dược sử dụng nhất trong các hệ thống nhỏ là Round Robin , các server được lựa chọn quay vòng, tuy nhiên phương pháp này có nhược điểm là hai yêu cầu liên tục từ một người dùng sẽ vào 2 servers khác nhau , thông tin giữa 2 yêu cầu liên tiếp sẽ bị mất, như vậy sẽ không thể tối ưu hóa được sư dụng tài nguyên. Đặc biệt là khi cần phải cài đặt kết nối cho có phiên chạy - ví dụ như SSL key negociation - sẽ rất tốn thời gian.

Một cách khắc phục nhược điểm này là sử dụng một hàm băm theo địa chỉ IP, như vậy yêu cầu từ cùng một địa chỉ IP sẽ chỉ vào một server duy nhất. Tuy vậy phương pháp này đòi hỏi người dùng phải có IP tĩnh. Vậy cách khắc phục cho những hạn chế trên là gì ? Đó chính là kỹ thuật Persistence.

**3.3 Kỷ thuật Session Persistence**

Như đã đề cập ở trên , vấn đề cần giải quyết chính là làm sao để giữ cho các yêu cầu của một người dùng được gửi vào một máy duy nhất trong suốt phiên làm việc của người đó. Tất cả các yêu cầu của người dùng này cần phải được chuyển vào cùng một server. Nế server bị chết , hoặc dừng để bảo trì, cần phải có cơ chế để chuyển session của người dùng này sang máy server khác. Đó chính là kỹ thuật Session Persistence. Có một số giải pháp đưa ra để tiếp cận kỹ thuật này, chẳng hạn như sử dụng một respone HTTP 302 hay tạo ra liên kết giữa người dùng - server. Tuy vậy hai phương pháp này đều có những hân chế , sử dung6 HTTP 302 sẽ khiến người dùng luôn luôn tìm cách kết nối vối một server duy nhất, kể cả khi server này đã “ chết “. Dùng cách tạo liên kết đòi hỏi user phải có IP tĩnh trong suốt phiên làm việc.

Vậy thì câu trả lời cuối cùng là gì ? Đó chính là sử dụng Cookie. Cookie là một đối tượng được điều khiển bởi Web Servers. Trong kết quả trã về cho người dùng web servers sẽ chèn thêm một số thông tin của cookie này, server sẽ đọc các cookie và biết phải làm gì với các yêu cầu nà.

**3.4 Cân bằng tải sử dụng phần cứng**

Bộ cân bằng tải bằng phần cứng sẽ thể hiện một địa chỉ IP ảo đối với mạng bên ngoài, địa chỉ này bản đồ hóa đến các địa chỉ của mỗi máy trong một cluster. Chính vì vậy bộ các máy tính trong cluster sẽ chỉ được xem như là một máy duy nhất đối với bên ngoài. Bộ cân bằng tải sử dụng phần cứng thường hoạt động ở tầng mạng và hoạt động dựa trên sự định tuyến, sử dụng một trong các phương pháp: Định tuyến trực tiếp (direct routing ) , tunnelling , IP address translation (NAT)

Khi một request đến bộ cân bằng tải, nó sẽ ghi lại header của request để trỏ đến các máy khác trong cluster. Nếu một máy nào đó bị gợ bõ từ cluster thì request sẽ không chạy một cách rủi ro việc " hit " vào máy server đã chết này, vì tất cả các máy server khác trong cluster xuất hiện đều có cùng địa chỉ IP. Địa chỉ này duy trì giống nhau thậm chí nếu một chút nào đó trong cluster bị hỏng. Khi một đáp trả được trả về, client sẽ xem đáp trả đang đến từ bộ cân bằng tải phần cứng. Hay nói theo cách khác thì người dùng sẽ xử lý với một máy tính đó là bộ cân bằng tải sử dụng phần cứng.

**Ưu điểm của phương pháp này là :**

Mối quan hệ giữa các máy chủ. Bộ cân bằng tải phần cứng đọc cookie hoặc các URL đang được đọc trên mỗi một request bởi máy client. Dựa trên các thông tin này , nó có thể ghi lại các thông tin header và gửi request đến nút thích hợp trong cluster, nơi session của nó được duy trì

Các bộ cần bằng tải này có thể cung cấp mối quan hệ giữa các máy server trong truyền thông HTTP, nhưng không thông qua kênh an toàn như HTTPS. Trong kẹnh an toàn, các thông báo được mã hóa SSL và có thể tránh bộ cân bằng tải đọc các thông tin session.

Khả năng có sẵn thông qua hệ thống tự động chuyển đổi dự phòng. Việc chuyển đổi dự phòng xảy ra khi một nút trong cluster không thể xử lý một request và chuyến hướng nó đến một nút khác. Có hai kiểu tự động chuyển đỗi dự phòng:

Yều cầu mức chuyển đổi dự phòng. Khi một nút trong cluster không thể xử lý một request ( thường là vì bị bỏng ) thì nó sẽ chuyển request này sang một nút khác

Chuyển đổi dự phòng session một cách trong suốt. Khi một lời triệu gọi thất bại, nó sẽ được định tuyến một cách trong suốt đến một nút khác trong cluster để hoàn tất công việc.

Bộ cân bằng kiểu này cung cấp chuyển đổi dự phòng mức request; tức là khi nó phát hiện có một nút nào đó bị sự cồ thì bộ cần bằng này sẽ chuyển hướng tất cả các request theo sau được gửi đến nút này sang một nút tích cực khác trong cluster. Mặc dù vậy, bất kỳ một thông tin session nào trên nút chết sẽ bị mất các request được chuyển hướng đến một nút mới.

Chuyển đổi dự phòng session trong suốt yêu cầu một số kiến thức về sự thực thi trong một quá trình trong một nút, vì bộ cân bằng tải phần cứng chỉ có thể phát hiện các vấn đề mức mạng, không có lỗi. Để thực thi một cách trong suốt về vấn đề chuyển đổi dự phòng, các nút trong cluster phải kết hợp với các nút khác và có vùng bộ nhớ chia sẽ hoặc cơ dở dữ liệu chung để lưu tất cả các dữ liệu session. Cũng chính vì vậy nếu một nút trong cluster có vấn đề thì một session có thể tiếp tục trong một nút khác.

Metrics, Vì tất cả các yêu cầu tới một ựng dụng web đều phải qua hệ thống cân bằng tải, hệ thống có thể quyết định số lượng session hoạt động , số lượng session hoat động được kết nối trong các trường hợp khác nhau , các khoảng thời gian đáp ứng, thời gian tối đa điện áp , số lượng session trong suốt khoảng tối đa điện áp , số lượng session trong suốt khoảng tối thiểu điện áp... Tất cả các thông tin kiểm định này được sử dụng để tinh chỉnh toàn bộ hệ thống nhằm tối ưu hiệu suất.

**3.5 Các thuật toán cân bằng tải**

**3.5.1 Thuật toán Round Robin**

Đây gọi là thuật toán luân chuyển vòng, các máy chủ sẽ được xem ngang hàng và sắp xếp theo một vòng quay. Các truy vấn dịch vụ sẽ lần lượt được gửi tới các máy chủ theo thứ tự sắp xếp.

Ví dụ :

Cấu hình một cụm Cluster bao gồm 3 máy chủ : A , B . C

Yêu cầu dịch vụ thứ nhất sẽ được gửi đến máy chủ A.

Yêu cầu dịch vụ thứ hai sẽ được gửi đến máy chủ B.

Yêu cầu dịch vụ thứ ba sẽ được gửi đến máy chủ C.

Yêu cầu dịch vụ thứ tư sẽ được gửi đến máy chủ A...

**3.5.2 Thuật toán Weighted Round Robin:**

Bản chất giống như thuật toán Round Robin, tuy nhiên chúng ta có thể cấu hình cho một máy chủ đó thường xuyên được sử dụng hơn.

**3.5.3. Thuật toán Least Connection**

Đây là thuật toán dựa trên tính toán số lượng kết nối để thực hiện cân bằng tải cho máy chủ, nó sẽ tự động lựa chọn máy chủ với số lượng kết nối đang hoạt động là nhỏ nhất.

**3.5.4. Thuật toán Weights Least Connection:**

Bản chất giống thuật toán Least Connection , nhưng chúng ta có thể cấu hình ưu tiên cho một máy chủ trong cụm máy chủ hoạt động.

**3.5.5. Thuật toán Least Response Time:**

Đây là thuật toán sử dụng phương pháp thời gian đáp ứng ít nhất, lựa chọn dịch vụ trên máy chủ với thời gian đáp ứng là thấp nhất.

Ngoài ra còn có rất nhiều thuật toán cân bằng tải khác tùy theo phần mềm hoặc phần cứng cân bằng tải được sử dụng.

**3.5.6. Hàm băm:**

Hàm băm là giải thuật nhằm sinh ra các giá trị băm tương ứng với mỗi database server. Giá trị băm đóng vai gần như một khóa để phân biệt các database server cũng giống với giải thuật Round-robin, hàm băm chỉ định server trả lời truy vấn cho client mà không quan tâm đến trạng thái hiện tại của server đó ( có ít hay nhiều truy vấn ).

**3.5.7. Giải thuật xác định tổng số kết nối nhỏ nhất :**

Giải thuật này xác định tổng số kết nối hiện tại trên các database server, nếu server nào có tổng kết nối nhỏ nhất thì server đó sẽ được chỉ định là trả lời truy vấn tiếp theo của client. Trong giải thuật này có quan tâm đến trạng thái của server. Đây là một giải thuật tối ưu để cân bằng tải cho các database server. Tuy nhiên, thời gian thực hiện bài toán là lâu hơn hai giải thuật trên, bởi vì nó phải ghi lại và tính số kết nối đến các server hiện thời.