Chương 4: Deploying and scaling SignalR

Nếu chúng ta phải tóm tắt quá trình triển khai bằng một câu, chúng ta có thể nói rằng đó là tập hợp các hành động cần được thực hiện để cài đặt phần mềm trong các môi trường mà nó có thể được sử dụng bởi người dùng. Trong bối cảnh các ứng dụng SignalR, các kịch bản triển khai có thể thay đổi rất nhiều do có nhiều nền tảng mà chúng ta có thể tìm thấy ở cả phía khách hàng và phía máy chủ và sự bùng nổ tổ hợp của các nền tảng này. Rõ ràng, việc triển khai các dịch vụ SignalR bên trong một ứng dụng ASP.NET thực hiện trên IIS không giống như việc thực hiện nó trong một dịch vụ hệ điều hành hoặc trong vai trò của một Windows Azure. Thủ tục triển khai ứng dụng Windows Phone 8 cũng không phải là cách sử dụng máy khách SignalR để truy cập vào các thông báo thời gian thực giống như quá trình triển khai ứng dụng cho Windows 8 hoặc một JavaScript client đơn giản.

Mỗi một trong những nền tảng này đều có các thủ tục triển khai riêng, sẽ vẫn còn nguyên vẹn với SignalR. Nghĩa là, bao gồm các thành phần SignalR trong một ứng dụng phía máy chủ không có ảnh hưởng gì đến cách mà nó phải được triển khai, cũng không ảnh hưởng đến phía client.

Ví dụ: trong các hệ thống web cổ điển, triển khai bao gồm cài đặt ứng dụng trên máy chủ, thường là theo các cách sau:

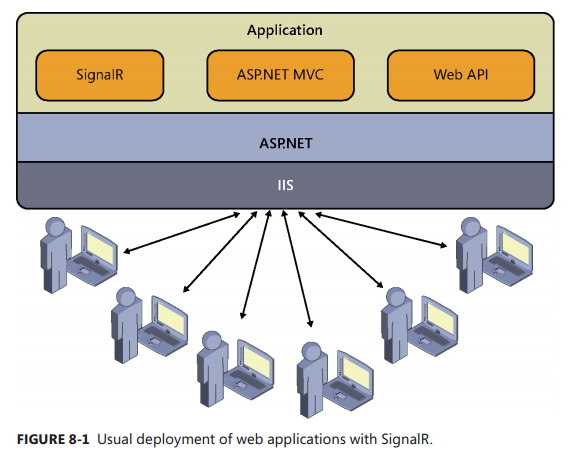
■ Sử dụng các công cụ xuất bản được tích hợp trong môi trường phát triển, chẳng hạn như tiện ích xuất bản Visual Studio.

■ Trong trường hợp các quá trình tự động, sử dụng Web Deploy 1 hoặc FTP, …

■ Bằng tay, sử dụng bất kỳ loại giao thức truyền tải nào cho phép chúng ta sao chép trực tiếp các tập tin vào máy chủ.

Tuy nhiên, thường không có hành động cụ thể để tạo thuận lợi cho việc triển khai các bộ phận đến phía khách hàng. Chúng sẽ được phân phối theo yêu cầu khi truy cập vào các trang hoặc các tính năng khác được cung cấp, và chúng sẽ được thực hiện trên trình duyệt của người dùng.

Trong các ứng dụng web với SignalR, nó là chính xác như nhau. Trên thực tế, vì tất cả các thành phần máy chủ của các ứng dụng SignalR được biên dịch (các hub, các kết nối liên tục, các lớp cấu hình ...), thì phải tuân theo bất kỳ quy trình xuất bản nào mà chúng ta thường sử dụng để nhận các dịch vụ thời gian thực của chúng tôi hoạt động trên các môi trường hoạt động . Các thành phần client - các tệp JavaScript cơ bản - sẽ được bao gồm trong gói cài đặt và sẽ được nhập vào ứng dụng dưới dạng tệp tĩnh.

Chúng ta thường gặp ứng dụng ASP.NET được triển khai theo cách này, với cấu trúc thể hiện trong hình 8-1

Ứng dụng của chúng tôi có thể được viết bằng cách sử dụng các framework như ASP.NET MVC, ASP.NET Web Forms, hoặc Web API, những trang mà tải các thành phần của khách hàng SignalR để tiêu thụ các dịch vụ được cung cấp qua các hub hoặc các kết nối liên tục được định nghĩa trong các lớp được biên dịch bên trong ứng dụng. Tất nhiên, nó có thể được đi kèm với các hội đồng khác, chẳng hạn như các thành phần thị giác, các chức năng tiện ích, các lớp mô hình, hoặc bất kỳ loại nội dung nào, nhưng lược đồ sẽ vẫn như cũ. Đây là kiến trúc triển khai đơn giản nhất mà chúng ta có thể tìm thấy.

**Growing pains**

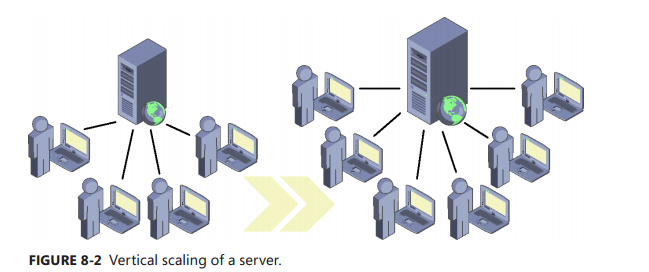
**Vâng, sau nhiều giờ làm việc vô tận, nhiều tuần lễ với hầu như không có đêm nghỉ ngơi và không ngủ, cuối cùng chúng tôi đã tung ra hệ thống xã hội cách mạng của chúng tôi, trong đó, tất nhiên, chúng tôi đã sử dụng SignalR để thực hiện các tính năng ngoạn mục sử dụng truyền thông đẩy và thời gian thực như nhân loại Chưa bao giờ được biết đến trước đây. Bây giờ chúng ta phải tiết lộ nó với thế giới và bắt đầu thu hút người dùng bởi hàng trăm. Có thể nói rằng các vấn đề của chúng ta đã kết thúc. (Hoặc bạn có thể nói rằng họ vừa mới bắt đầu.)**

**Từ điểm này, về cơ bản hai điều có thể xảy ra, trạng thái trung gian sang một bên. Trong trường hợp đầu tiên, ứng dụng siêu của chúng tôi sẽ trở nên không cần thiết như chúng tôi nghĩ, hoặc có lẽ người dùng không hiểu được giá trị đích thực của nó, và nó sẽ không bị ai sử dụng và cuối cùng biến mất trong sự quên lãng**

**Hoặc chúng ta có thể có trường hợp ngược lại: ứng dụng của chúng tôi là một hit thực sự. Thu hút bởi tính mới lạ và lời hứa cuối cùng là nhận được những trải nghiệm trực tuyến được chờ đợi suốt đời của họ, người dùng bắt đầu đăng ký một cách nhút nhát. Lời truyền miệng và truyền nhanh qua mạng xã hội khiến số lượng đăng ký tăng lên theo cấp số nhân sau vài tuần. Người dùng bắt đầu có số trong hàng trăm, sau đó hàng ngàn, và sau đó đến những khiếu nại đầu tiên về sự chậm trễ và các vấn đề với dịch vụ. Tác động này thường được gọi là "cái chết do sự thành công" và đó là lý do cho sự sụp đổ của nhiều dự án thú vị. Đây là một khoảnh khắc tinh tế, và sự sống sót của sinh vật của chúng ta sẽ phụ thuộc phần lớn vào khả năng của chúng ta để giải quyết những vấn đề này.**

**Đó là thời gian để quy mô - nghĩa là, để tìm cách để cung cấp chất lượng dịch vụ cho một số lượng ngày càng tăng của người sử dụng**

**Thật may mắn, luôn luôn có thể mở rộng máy chủ mà chúng tôi đã thuê để lưu trữ ứng dụng của chúng tôi, vì vậy cách tiếp cận đầu tiên của chúng tôi trong trường hợp này là đầu tư ở đó: tối đa hóa bộ nhớ RAM, cài đặt nhiều bộ vi xử lý như bảng hoặc máy ảo cho phép và cải tiến Lưu trữ liên tục về tốc độ và không gian. Cách nâng cao năng lực của cơ sở hạ tầng của chúng tôi để hỗ trợ nhiều người dùng được gọi là khả năng mở rộng theo chiều dọc. (Xem Hình 8-2.) Nó đòi hỏi hầu như không có thay đổi phần mềm hoặc các biện pháp phòng ngừa đặc biệt trong quá trình phát triển; Nó chỉ đơn giản liên quan đến việc mua hoặc thuê thêm kim loại hoặc thêm nhiều nguồn lực vào VM của chúng tôi.**

**Chú ý Quy mô sử dụng phương pháp tiếp cận này thường được gọi là tăng quy mô.**

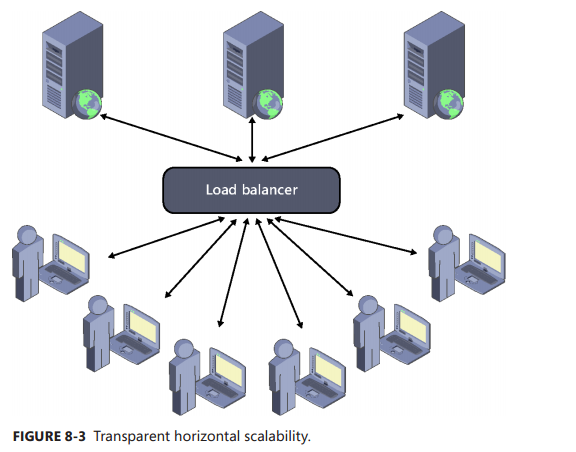
Các thành viên lạc quan hơn của nhóm có thể nghĩ rằng việc đầu tư lớn vào phần cứng này sẽ vượt qua được vấn đề. Và nó sẽ ... ít nhất trong một khoảng thời gian. Vấn đề với khả năng mở rộng theo chiều dọc là do hạn chế vốn có: tuy nhiên chúng ta muốn mở rộng một máy chủ, sẽ có một điểm vượt ra ngoài nó là thể chất không thể tăng khả năng của nó. Ở đây chúng tôi sẽ chuyển sang máy chủ có kiến trúc cho phép mở rộng nhiều hơn, nhưng với tốc độ tăng trưởng của ứng dụng siêu của chúng tôi,

Điều này sẽ chỉ phục vụ cho chúng ta một khoảng trống trong khi chúng ta tìm kiếm một giải pháp dứt khoát hơn. Và chúng ta không nên quên chi phí: thêm bộ nhớ hoặc CPU vào một hệ thống có xu hướng đắt tiền, và bình thường có một sự trở lại khá hạn chế về đầu tư. Nó cũng không phải là một lựa chọn rất linh hoạt. Nếu chúng ta biết rằng người dùng của chúng tôi kết nối chủ yếu vào giờ làm việc, trong suốt thời gian còn lại của ngày, tất cả sức mạnh và tài nguyên của máy chủ sẽ không được sử dụng, vì vậy chúng tôi sẽ không nhận được nhiều nhất từ đầu tư.

Tóm lại, khả năng mở rộng theo chiều dọc có giá trị đối với các môi trường khép kín hoặc kiểm soát chặt chẽ, chẳng hạn như các ứng dụng hoặc hệ thống của công ty có ít người dùng vì bản chất của dịch vụ và ngữ cảnh của nó. Nói chung, chúng tôi không cần nhiều hơn nữa nếu chúng tôi tạo một ứng dụng trò chuyện dành cho những người hâm mộ khó tính của nhóm trẻ sơ sinh bơi đồng bộ trong khu phố lân cận - chọn cho các giải pháp phức tạp hơn trong kịch bản này có thể được coi là quá mức, quá kỹ thuật , Hoặc chỉ cần overkill

Khi khả năng mở rộng theo chiều dọc là không đủ, giải pháp được đưa ra thông qua một cách tiếp cận mạnh mẽ hơn: khả năng mở rộng theo chiều ngang. Quy mô theo chiều ngang bao gồm việc tăng số lượng máy chủ cho đến khi họ có thể đáp ứng nhu cầu của người dùng

Trong các môi trường web truyền thống, giải pháp lý tưởng là giới thiệu một cơ chế để tự động phân phối tải giữa các nút khác nhau theo cách minh bạch đối với người dùng, hành động vì mọi ý định và mục đích giống như ứng dụng đã làm việc trên một máy chủ duy nhất, Yêu cầu (thậm chí cả những người từ cùng một người dùng) có thể được xử lý bởi một nút khác trong mạng. Các yếu tố này được gọi là bộ cân bằng tải được đặt tại lối vào trang trại máy chủ và chuyển hướng yêu cầu đến máy chủ ít nhất bị bận để được xử lý ở đó. Xem Hình 8-3



Và nhờ các công nghệ hiện có, chúng ta thậm chí có thể sử dụng các dịch vụ đám mây như những gì được cung cấp bởi Windows Azure 2 để tự động điều chỉnh số lượng máy chủ theo nhu cầu thực tế tại mỗi thời điểm, do đó đảm bảo rằng ứng dụng của chúng tôi sẽ luôn sẵn sàng phục vụ Người sử dụng cũng như đạt được sự dễ dàng quản lý cơ sở hạ tầng và mức độ chịu lỗi tốt

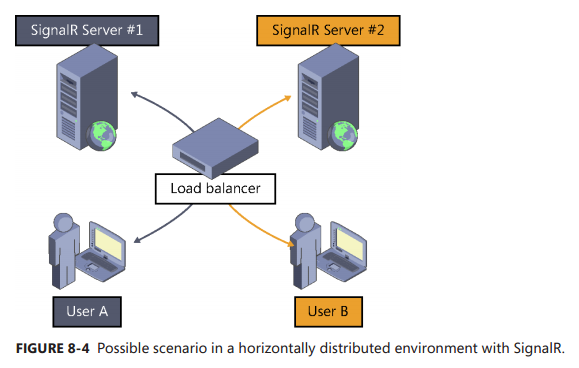
Ngoài ra, để giảm độ trễ và tăng hiệu suất, có thể đưa nội dung và dịch vụ gần hơn với người dùng tiêu thụ chúng, sử dụng CDN 3 (Mạng phân phối nội dung) hoặc các máy chủ có chiến lược. Lợi thế chính so với mở rộng quy mô là trong quy mô-out có một giới hạn về thể chất priorino: chúng ta luôn có thể thêm các máy chủ mới để chịu tải lớn hơn. Tuy nhiên, để làm việc trên kiến trúc này, các ứng dụng phải được thiết kế để dự đoán khả năng này, bởi vì mỗi yêu cầu có thể được xử lý bởi một máy chủ khác, bạn không thể sử dụng các tài nguyên cục bộ của bộ nhớ thứ cấp (như bộ nhớ hoặc hệ thống tập tin) để lưu trữ Dữ liệu liên tục hoặc để lưu trữ trạng thái của ứng dụng

Là một lưu ý phụ, có một sự thay thế được gọi là sự liên quan, bao gồm việc cân bằng tải luôn gán các yêu cầu đến từ một người dùng nhất định vào cùng một máy chủ, theo đó thông tin trạng thái có thể được lưu trữ trong các máy chủ. Còn được gọi là phiên dính, kỹ thuật này tốt hơn so với quy mô theo chiều dọc nhưng tồi tệ hơn quy mô theo chiều ngang với cách tiếp cận không quốc tịch.

**Scalability in SignalR**

Chúng tôi đã giải thích rằng trong các môi trường web được điều chỉnh theo chiều ngang, chúng ta không nên sử dụng các tài nguyên máy chủ cục bộ như bộ nhớ hoặc đĩa. Những điều cần được thậm chí nhiều hơn omplicated khi nói đến các ứng dụng SignalR.

Khi chúng ta xem xét các kỹ thuật bảo trì của nhà nước tại máy chủ trong Chương 5, "Hubs", chúng tôi mô tả các vấn đề về việc có nhiều máy chủ SignalR phục vụ song song và chúng tôi đưa ra một số hướng dẫn để giải quyết chúng. Tuy nhiên, như chúng tôi đã giải thích, SignalR đặt ra một thách thức thực sự trong các kịch bản phân tán theo chiều ngang, đặc biệt là bởi vì mỗi máy chủ chỉ nhận thức được các khách hàng trực tiếp kết nối với nó. Vấn đề trình bày này được minh họa trong ví dụ "tối giản" thể hiện trong hình 8-4: người dùng A được kết nối với dịch vụ SignalR, và bộ cân bằng tải phân bổ nó đến máy chủ số 1. Sau đó đến người dùng B và nó được cấp phát cho máy chủ # 2

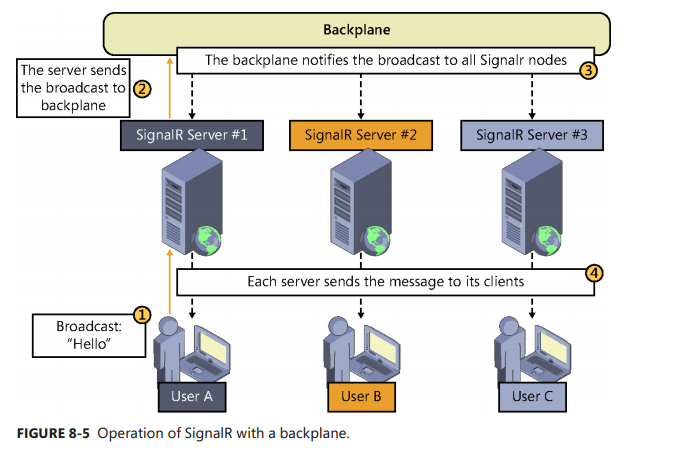


Mặc dù cả hai người dùng về mặt lý thuyết sẽ kết nối với cùng một kết nối hoặc trung tâm liên tục - sẽ chỉ có một URL truy cập-bộ cân bằng sẽ ủy quyền xử lý cho một máy chủ cụ thể, một máy chủ khác trong mỗi trường hợp. Và bởi vì mỗi một kết nối vật lý với một máy chủ khác, thông điệp phát sóng do máy chủ số 1 sẽ chỉ nhận được bởi người dùng A và cũng giống với máy chủ khác.

Ngoài ra, việc phân bổ khách hàng này cho các máy chủ có thể thay đổi mỗi lần kết nối vật lý mới mở ra, gây ra tình huống khó quản lý do chuyển vùng. Ví dụ: hãy tưởng tượng rằng người dùng A được kết nối với máy chủ số 1 và trải qua thời gian chờ do sự cố mạng, kết nối lại và được phân bổ cho máy chủ số 2. Từ quan điểm của máy chủ số 1, người dùng A đã rời ứng dụng, trong khi từ máy chủ # 2, người dùng A sẽ được coi là người dùng mới. Hơn nữa, trong thời gian thực hiện cho quá trình kết nối lại, các tin nhắn mới có giá trị được gửi đến người dùng có thể đã vào hệ thống.

Tác động tương tự này có thể xảy ra với các kết nối song song được thực hiện bởi khách hàng để gửi thông tin tới máy chủ trong các cuộc vận chuyển đẩy một chiều như các sự kiện do máy chủ gửi hoặc các cuộc thăm dò lâu dài. Mỗi một có thể được xử lý bởi một máy chủ khác nhau, nhưng kết nối push chính sẽ vẫn hoạt động và cố định từ đầu. Như bạn có thể đoán, những vấn đề này không phải là dễ dàng để giải quyết.

Để giải quyết các tình huống này, SignalR đi kèm với một cơ chế mạnh mẽ cho phép triển khai các thành phần máy chủ của nó trong các môi trường phân tán ngang thông qua các lớp nền. Như trong hình 8-5, bảng nối đa năng là một thành phần hoạt động như một hệ thống nhắn tin giữa các nút SignalR, tương tự như một bus để liên lạc nội bộ giữa các hệ thống. Khi được kích hoạt trên máy chủ, tất cả các thông điệp được tạo ra từ nó sẽ được gửi qua bảng nối tiếp, nơi mà các máy chủ còn lại sẽ lắng nghe để chuyển tiếp chúng tới các khách hàng của họ.



Có nhiều loại backplanes khác nhau, mỗi nhóm sử dụng một công nghệ khác nhau để quản lý các ấn phẩm và đăng ký sử dụng trong hệ thống nhắn tin. Hiện tại, SignalR chính thức cung cấp backplanes cho Windows Azure Service Bus, SQL Server và Redis, nhưng cộng đồng nhà phát triển đã cung cấp một số bổ sung. Chúng tôi sẽ mô tả từng chi tiết sau.

Mặc dù sử dụng backplane cho phép sử dụng SignalR trong một trang trại máy chủ và phân phối tải người dùng giữa các máy chủ của nó, nhưng làm như vậy cũng có những hạn chế của nó. Hạn chế chính là nó chắc chắn sẽ dẫn đến sự gia tăng thời gian cần thiết để gửi mỗi tin nhắn và do đó giảm đáng kể số lượng tin nhắn có thể di chuyển qua hệ thống - đó là thông lượng. Nghịch lý, giải pháp tương tự được sử dụng để cho phép khả năng mở rộng ngang của các dịch vụ SignalR có thể trở thành một nút cổ chai trong các kiến trúc đã được thu nhỏ. Trên thực tế, khả năng mở rộng không đồng nghĩa với việc tăng hiệu suất trong trường hợp này.

Do đó, chúng ta không nên nghĩ đến các backplanes như là giải pháp cuối cùng cho các vấn đề về khả năng mở rộng của tất cả các loại ứng dụng SignalR, nhưng chỉ như là cơ chế cho phép sử dụng khuôn khổ này với các trang trại máy chủ cân bằng, trong trường hợp đây sẽ là kiến trúc triển khai cuối cùng Chọn cho hệ thống của chúng tôi.

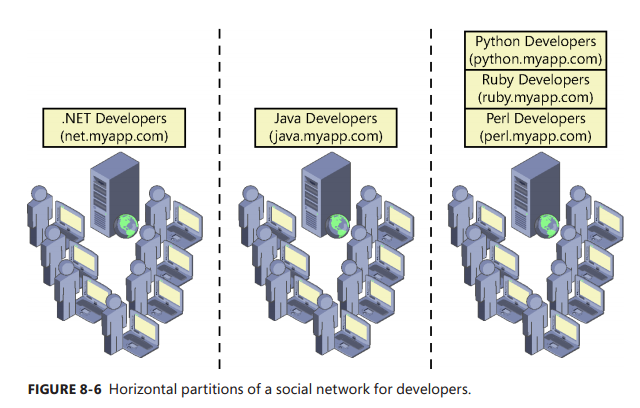
Việc sử dụng backplanes được khuyến cáo trong các ứng dụng có một máy chủ phát sóng cùng một thông điệp cho nhiều khách hàng hoặc nhóm khách hàng. Điều này là do, trong SignalR, các thư được xử lý sao cho gửi cùng một thông điệp tới nhiều khách hàng sẽ không làm say đắm cho chiếc xe buýt. Việc sử dụng của họ cũng thích hợp khi có tần suất gửi rất thấp hoặc nếu số tiền của họ không liên quan trực tiếp đến sự gia tăng số lượng khách hàng. Trong tất cả các trường hợp này, bảng nối đa năng có thể mang lại hiệu suất tốt và có khả năng đồng thời khả năng mở rộng gần như không giới hạn.

Tuy nhiên, ngay khi thông tin được cá nhân hóa cần được gửi đến từng khách hàng hoặc ngay lập tức là rất quan trọng, chúng ta phải dùng đến các giải pháp khác

Khi thiết kế chiến lược khả năng mở rộng cung cấp hiệu suất phù hợp cho loại ứng dụng được đề cập, có nhiều giải pháp như các loại hệ thống và các nhóm phát triển đang cố gắng thực hiện chúng. Điều gì sẽ là cách tốt nhất để thiết kế một ứng dụng SignalR mở rộng? Câu trả lời tốt nhất mà chúng tôi có thể cho là "nó phụ thuộc".

Nói chung, công thức để nhân rộng các hệ thống của chúng tôi luôn luôn bao gồm phân phối tải trên nhiều máy chủ, nhưng nếu chúng ta muốn duy trì hiệu năng tốt trong các dịch vụ thời gian thực dựa trên SignalR, chúng ta phải thiết kế và thực hiện các giải pháp của riêng mình. Và trong trường hợp này, chúng hầu như luôn luôn liên quan đến việc tìm kiếm các kỹ thuật phân vùng hoặc phân phối người dùng theo các tiêu chí logic cụ thể cho miền của ứng dụng mà chúng tôi đang phát triển để nhóm chúng vào cùng một máy chủ. Nếu tất cả những người dùng mà chúng tôi muốn truyền thông như một nhóm đều được tìm thấy trên cùng một máy chủ, các bản đệ trình sẽ được trực tiếp và chúng tôi sẽ tránh những người trung gian có thể giới thiệu độ trễ, chẳng hạn như khi sử dụng backplanes. Chúng tôi cũng sẽ đạt được hiệu quả và chúng tôi sẽ có thể mở rộng quy mô của mình theo chiến lược nhóm mà chúng tôi đã lựa chọn cho phép chúng tôi, mặc dù thường có chi phí tăng nỗ lực phát triển.

Cách tiếp cận đơn giản và cơ bản nhất có thể nảy sinh nếu chúng ta biết trước và chắc chắn về những gì mà người dùng chúng tôi đang xử lý và phân phối của họ, bởi vì chúng tôi có thể định nghĩa một kiến trúc tĩnh cho họ. Chúng ta sẽ phải chuẩn bị cơ sở hạ tầng của các máy chủ được cấu hình và thích nghi với các nhu cầu ban đầu của hệ thống, như thể hiện trong hình 8-6, và thích ứng nó theo sự tiến triển của nhu cầu.



Trong ví dụ này, chúng tôi có thể tạo các dịch vụ chat mới vì các công nghệ mới xuất hiện trong tương lai và lưu trữ các dịch vụ này trên các máy chủ có dung lượng dự phòng hoặc thêm các máy chủ mới. Tuy nhiên, để làm cho tùy chọn này có thể, do tính chất của dịch vụ, chúng ta nên đảm bảo rằng không có phân vùng sẽ phát triển vượt ra ngoài những gì đã được giả định khi quy hoạch kích thước của cơ sở hạ tầng.

Chúng ta cũng có thể có một cách tiếp cận năng động hơn và linh hoạt, điều này sẽ được khuyến khích trong hầu hết các kịch bản. Theo khái niệm, điều này có nghĩa là tạo ra một hệ thống cân bằng "thông minh", nhạy cảm với miền của ứng dụng, có thể bao gồm một dịch vụ phía trước có thể chuyển hướng khách hàng đến máy chủ phù hợp, dựa trên các tiêu chí nhất định.

Ví dụ: trong dịch vụ chỉnh sửa văn bản cộng tác, tiêu chí nhóm người dùng có thể là chính tài liệu. Nếu người dùng nhập vào ứng dụng và quyết định mở một tài liệu, một máy chủ SignalR có thể được gán cho chúng trên máy bay - máy chủ bận rộn nhất trong một tập hợp các máy chủ có sẵn. Lập bản đồ này sẽ được lưu trữ trong một hệ thống tồn tại tập trung, và người sử dụng sẽ được chuyển hướng đến máy chủ nói rằng cô ấy có thể bắt đầu làm việc trên tài liệu. Nếu một số người dùng khác truy cập vào cùng một tài liệu sau đó, ứng dụng sẽ biết rằng đã có một máy chủ thời gian thực được gán cho nó và sẽ chuyển hướng người dùng mới này tới nó. Trong trường hợp này, tiêu chí phân chia hoặc phân phối người dùng sẽ là tài liệu

Chúng ta có thể tìm thấy một ứng dụng khác trong một hệ thống nhắn tin nội bộ trên một hệ thống ERP đa cấp được phân phối như là SaaS. Trong trường hợp này, tiêu chí phân bổ người dùng có thể là chính người thuê nhà - nghĩa là công ty sử dụng dịch vụ - bởi vì mỗi người trong số họ hoàn toàn độc lập với những người khác, và do đó các dịch vụ thời gian thực có thể được cung cấp trong các máy chủ riêng biệt

Lưu ý rằng một cách tiếp cận tương tự hiện đang được sử dụng trong nhiều ứng dụng. Ví dụ: có nhiều trò chơi trực tuyến, nơi người dùng có khu vực có thể sử dụng để tìm đối thủ hoặc thực hiện các hoạt động khác, nhưng tại thời điểm bắt đầu trò chơi, một máy chủ hiện có được chỉ định và tất cả người dùng liên quan đến cùng một trò chơi được chuyển hướng tới nó. Trong trường hợp này, tiêu chí phân vùng sẽ là trò chơi

Cũng không khả thi khi sử dụng máy tính xách tay trong các hệ thống khi cần truyền thông theo thời gian thực với tần suất cập nhật cố định mỗi giây, chẳng hạn như trong một game bắn súng nhiều người chơi. Kịch bản này, có lẽ là một trong những điểm cực đoan nhất mà chúng ta phải đối mặt khi phát triển ứng dụng thời gian thực, có thể được tiếp cận theo cách tương tự: làm cho tất cả người dùng tham gia vào hành động hoặc trò chơi kết nối với cùng một máy chủ để có thể gửi tin nhắn của họ trực tiếp Và không có người trung gian

Một khả năng chúng ta có thể xem xét khi sử dụng backplanes đang cố gắng tối ưu hóa lưu lượng truy cập được tạo ra trong bus nhắn tin dựa trên nhu cầu của ứng dụng của chúng tôi. Ví dụ, sẽ có thể tạo ra một hệ thống để giữ cho các thuê bao người dùng nhận được cập nhật trên mỗi nút, và thực hiện một hệ thống lọc ngăn chặn các thông điệp mà một nút không cần đến từ nút đó qua bảng nối đa năng

Tất nhiên, chúng ta có thể kết hợp một số kỹ thuật này hoặc đưa ra các chiến lược mới phù hợp với nhu cầu chính xác của từng ứng dụng, nhưng khác với các backplanes, chúng ta sẽ phải thực hiện giải pháp của riêng mình