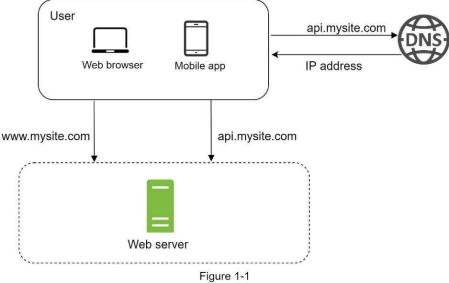
**CHƯƠNG 1: THIẾT KẾ HỆ THỐNG TỪ SỐ KHÔNG ĐẾN TRIỆU NGƯỜI DÙNG**

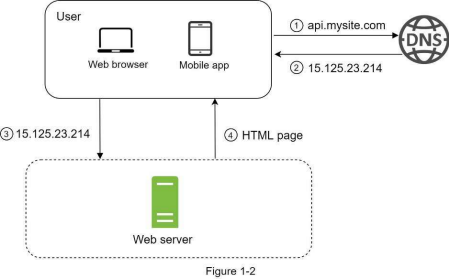
Thiết kế một hệ thống cho hàng triệu người dùng là một thách thức không hề dễ dàng và đó là một hành trình đòi hỏi sự cải tiến không ngừng. Trong chương này, chúng ta sẽ xây dựng một hệ thống và từ từ mở rộng quy mô để phục vụ hàng triệu người dùng. Sau chương này, bạn sẽ nắm vững một số kỹ thuật giúp bạn vượt qua các câu hỏi phỏng vấn về thiết kế hệ thống.

**Single server setup – Thiết lập máy chủ đơn**

Cuộc hành trình nào cũng đều bắt từ bước cơ bản nhất và việc xây dựng một hệ thống phục vụ hàng triệu người dùng không ngoại lệ. Bắt đầu với điều đơn giản nhất, chạy mọi thứ trên một máy chủ. **Figure 1-1** minh họa một hệ thống mà trong đó mọi thứ đang chạy trên một máy chủ: web app, cơ sở dữ liệu, cache,…



Để hiểu thiết lập này, việc điều tra luồng yêu cầu và nguồn lưu lượng truy cập sẽ rất hữu ích. Đầu tiên chúng ta hãy nhìn vào luồng yêu cầu (**Figure 1-2**).

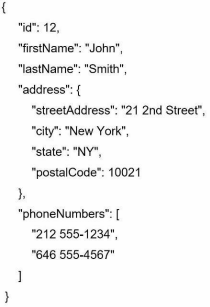


1. Người dùng truy cập trang web thông qua tên miền, chẳng hạn như api.mysite.com. Thông thường hệ thống tên miền (DNS) là dịch vụ trả phí do bên thứ ba cung cấp và không phải do máy chủ lưu trữ.
2. Địa chỉ Giao thức Internet (IP) được trả về trình duyệt hoặc ứng dụng di động. Trong ví dụ, địa chỉ IP 15.125.23.214 được trả về máy chủ.
3. Sau khi nhận được địa chỉ IP, Giao thức truyền siêu văn bản (HTTP) [1] sẽ gửi các yêu cầu trực tiếp đến máy chủ web.
4. Máy chủ web trả về các trang HTML hoặc phản hồi JSON để hiển thị.

Tiếp theo, chúng ta hãy kiểm tra nguồn lưu lượng truy cập. Lưu lượng truy cập vào máy chủ web của bạn đến từ hai nguồn: ứng dụng web và ứng dụng di động.

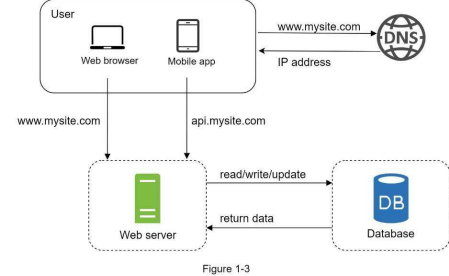
* Ứng dụng web: sử dụng kết hợp các ngôn ngữ phía máy chủ (Java, Python, v.v.) để xử lý logic nghiệp vụ, lưu trữ, v.v. và các ngôn ngữ phía máy khách (HTML và JavaScript) để trình bày.
* Ứng dụng di động: Giao thức HTTP là giao thức giao tiếp giữa thiết bị di động ứng dụng và máy chủ web. Ký hiệu đối tượng JavaScript (JSON) là định dạng phản hồi API thường được sử dụng để truyền dữ liệu do tính đơn giản của nó. Một ví dụ về phản hồi API ở định dạng JSON được hiển thị bên dưới:

GET /users/12 – Truy xuất đối tượng người dùng cho id = 12



**Database – Cơ sở dữ liệu**

Với sự phát triển của cơ sở người dùng, một máy chủ là không đủ và chúng ta cần nhiều máy chủ: một máy chủ dành cho lưu lượng truy cập web/di động, máy chủ kia dành cho cơ sở dữ liệu (**Figure 1-3**). Việc tách biệt các máy chủ lưu lượng truy cập web/di động (tầng web) và máy chủ cơ sở dữ liệu (tầng dữ liệu) cho phép chúng được mở rộng quy mô một cách độc lập.



**Nên sử dụng cơ sở dữ liệu nào?**

Bạn có thể chọn giữa cơ sở dữ liệu quan hệ truyền thống và cơ sở dữ liệu phi quan hệ. Hãy xem xét sự khác biệt của họ.

Cơ sở dữ liệu quan hệ còn được gọi là cơ sở dữ liệu SQL hoặc hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS). Những cái phổ biến nhất là MySQL, cơ sở dữ liệu Oracle, PostgreSQL, v.v. Cơ sở dữ liệu quan hệ biểu diễn và lưu trữ dữ liệu trong các bảng và hàng. Bạn có thể thực hiện các thao tác nối bằng SQL trên các bảng cơ sở dữ liệu khác nhau.

Cơ sở dữ liệu phi quan hệ còn được gọi là cơ sở dữ liệu NoSQL. Những cái phổ biến là CouchDB, Neo4j, Cassandra, HBase, Amazon DynamoDB, v.v. [2]. Các cơ sở dữ liệu này được nhóm thành bốn loại: lưu trữ khóa-giá trị, lưu trữ biểu đồ, lưu trữ cột và lưu trữ tài liệu. Các hoạt động tham gia thường không được hỗ trợ trong cơ sở dữ liệu không quan hệ.

Đối với hầu hết các nhà phát triển, cơ sở dữ liệu quan hệ là lựa chọn tốt nhất vì chúng đã tồn tại hơn 40 năm và trong lịch sử, chúng đã hoạt động tốt. Tuy nhiên, nếu cơ sở dữ liệu quan hệ không phù hợp với trường hợp sử dụng cụ thể của bạn thì điều quan trọng là phải khám phá những cơ sở dữ liệu quan hệ khác. Cơ sở dữ liệu phi quan hệ có thể là lựa chọn phù hợp nếu:

* Ứng dụng của bạn yêu cầu độ trễ cực thấp.
* Dữ liệu của bạn không có cấu trúc hoặc bạn không có bất kỳ dữ liệu quan hệ nào.
* Bạn chỉ cần tuần tự hóa và giải tuần tự hóa dữ liệu (JSON, XML, YAML, v.v.).
* Bạn cần lưu trữ một lượng lớn dữ liệu.

**So sánh Vertical scaling và Horizontal scaling**

Chia tỷ lệ theo chiều dọc, còn được gọi là "tăng quy mô", nghĩa là quá trình bổ sung thêm năng lượng (CPU, RAM, v.v.) cho máy chủ của bạn. Chia tỷ lệ theo chiều ngang, được gọi là “mở rộng quy mô”, cho phép bạn mở rộng quy mô bằng cách thêm nhiều máy chủ hơn vào nhóm tài nguyên của mình.

Khi lưu lượng truy cập thấp, chia tỷ lệ theo chiều dọc là một lựa chọn tuyệt vời và tính đơn giản của việc chia tỷ lệ theo chiều dọc là ưu điểm chính của nó. Thật không may, nó đi kèm với những hạn chế nghiêm trọng.

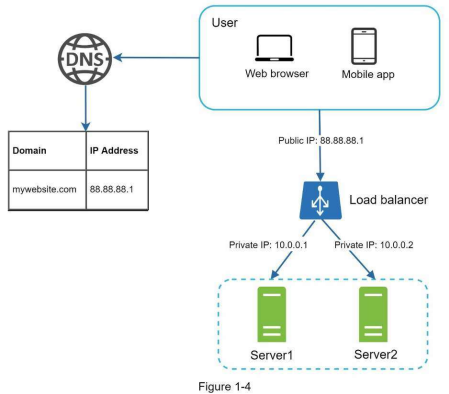
* Chia tỷ lệ theo chiều dọc có giới hạn cố định. Không thể thêm CPU và bộ nhớ không giới hạn vào một máy chủ duy nhất.
* Chia tỷ lệ theo chiều dọc không có khả năng chuyển đổi dự phòng và dự phòng. Nếu một máy chủ ngừng hoạt động, trang web/ứng dụng bị hỏng hoàn toàn.

Chia tỷ lệ theo chiều ngang được mong muốn hơn cho các ứng dụng quy mô lớn do những hạn chế của tỷ lệ theo chiều dọc.

Trong thiết kế trước, người dùng được kết nối trực tiếp với máy chủ web. Người dùng sẽ không thể truy cập trang web nếu máy chủ web ngoại tuyến. Trong một trường hợp khác, nếu nhiều người dùng truy cập máy chủ web đồng thời và nó đạt đến giới hạn tải của máy chủ web, thì người dùng thường phản hồi chậm hơn hoặc không kết nối được với máy chủ. Cân bằng tải là kỹ thuật tốt nhất để giải quyết những vấn đề này.

**Load balancer – Cân bằng tải**

Bộ cân bằng tải phân phối đồng đều lưu lượng truy cập đến giữa các máy chủ web được xác định trong bộ cân bằng tải. **Figure 1-4** cho thấy cách hoạt động của bộ cân bằng tải.



Như được hiển thị trong **Figure 1-4**, người dùng kết nối trực tiếp với IP công cộng của bộ cân bằng tải. Với thiết lập này, máy chủ web không thể truy cập trực tiếp được bởi khách hàng nữa. Để bảo mật tốt hơn, IP riêng được sử dụng để liên lạc giữa các máy chủ. IP riêng là địa chỉ IP chỉ có thể truy cập được giữa các máy chủ trong cùng một mạng; tuy nhiên, nó không thể truy cập được qua internet. Bộ cân bằng tải giao tiếp với các máy chủ web thông qua IP riêng.

Trong **Figure 1-4**, sau khi thêm bộ cân bằng tải và máy chủ web thứ hai, chúng tôi đã giải quyết thành công không có vấn đề chuyển đổi dự phòng và cải thiện tính khả dụng của tầng web. Chi tiết được giải thích dưới đây:

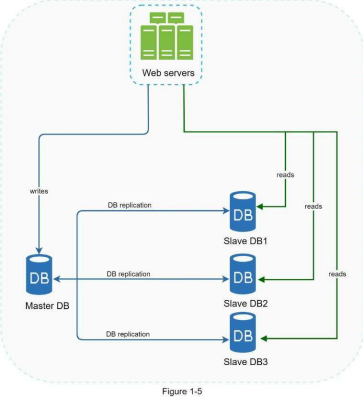
* Nếu máy chủ 1 ngoại tuyến, tất cả lưu lượng truy cập sẽ được chuyển đến máy chủ 2. Điều này ngăn cản trang web khỏi việc ngoại tuyến. Chúng tôi cũng sẽ thêm một máy chủ web mới vào nhóm máy chủ để cân bằng tải.
* Nếu lưu lượng truy cập trang web tăng nhanh và hai máy chủ không đủ để xử lý lưu lượng, bộ cân bằng tải có thể xử lý vấn đề này một cách khéo léo. Bạn chỉ cần thêm nhiều máy chủ hơn vào nhóm máy chủ web và bộ cân bằng tải sẽ tự động bắt đầu gửi yêu cầu đến chúng.

Bây giờ tầng web có vẻ ổn, còn tầng dữ liệu thì sao? Thiết kế hiện tại có một cơ sở dữ liệu,vì vậy nó không hỗ trợ chuyển đổi dự phòng và dự phòng. Sao chép cơ sở dữ liệu là một kỹ thuật phổ biến để giải quyết những vấn đề đó. Chúng ta hãy xem xét.

**Database replication – Nhân bản cơ sở dữ liệu**

Trích dẫn từ Wikipedia: “Sao chép cơ sở dữ liệu có thể được sử dụng trong nhiều hệ thống quản lý cơ sở dữ liệu, thường có mối quan hệ chủ/nô lệ giữa bản gốc (chủ) và bản sao (nô)” [3].

Cơ sở dữ liệu chủ thường chỉ hỗ trợ các thao tác ghi. Cơ sở dữ liệu phụ nhận các bản sao dữ liệu từ cơ sở dữ liệu chính và chỉ hỗ trợ các thao tác đọc. Tất cả các lệnh sửa đổi dữ liệu như chèn, xóa hoặc cập nhật phải được gửi đến cơ sở dữ liệu chính. Hầu hết các ứng dụng yêu cầu tỷ lệ đọc và ghi cao hơn nhiều; do đó, số lượng cơ sở dữ liệu nô lệ trong một hệ thống thường lớn hơn số lượng cơ sở dữ liệu chính. **Figure 1-5** hiển thị một cơ sở dữ liệu chính với nhiều cơ sở dữ liệu nhân bản.



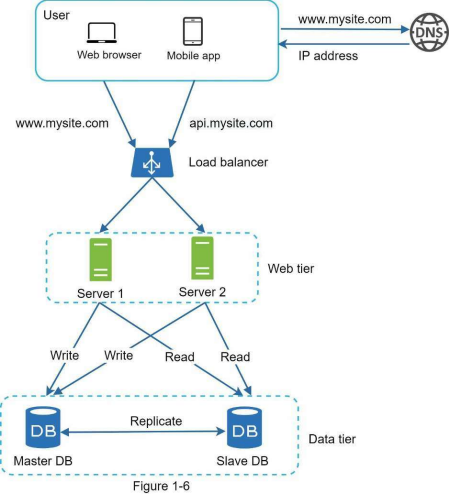
Ưu điểm của việc sao chép cơ sở dữ liệu:

* Hiệu suất tốt hơn: Trong mô hình chủ-nô, tất cả việc ghi và cập nhật đều diễn ra trong mô hình chính. điểm giao; trong khi đó, các hoạt động đọc được phân phối trên các nút phụ. Mô hình này cải thiện hiệu suất vì nó cho phép nhiều truy vấn được xử lý song song hơn.
* Độ tin cậy: Nếu một trong các máy chủ cơ sở dữ liệu của bạn bị phá hủy do thiên tai, chẳng hạn như một cơn bão hay một trận động đất, dữ liệu vẫn được bảo tồn. Bạn không cần lo lắng về việc mất dữ liệu vì dữ liệu được sao chép trên nhiều vị trí.
* Tính sẵn sàng cao: Bằng cách sao chép dữ liệu trên các vị trí khác nhau, trang web của bạn vẫn ở trạng thái ổn định hoạt động ngay cả khi cơ sở dữ liệu ngoại tuyến vì bạn có thể truy cập dữ liệu được lưu trữ trong máy chủ cơ sở dữ liệu khác.

Trong phần trước, chúng ta đã thảo luận về cách cân bằng tải giúp cải thiện tính khả dụng của hệ thống. Chúng tôi đặt câu hỏi tương tự ở đây: điều gì sẽ xảy ra nếu một trong các cơ sở dữ liệu ngoại tuyến? Thiết kế kiến ​​trúc được thảo luận trong **Figure 1-5** có thể giải quyết trường hợp này:

* Nếu chỉ có một cơ sở dữ liệu phụ và nó ngoại tuyến, các thao tác đọc sẽ được hướng dẫn vào cơ sở dữ liệu chủ tạm thời. Ngay sau khi tìm thấy sự cố, cơ sở dữ liệu phụ mới sẽ thay thế cơ sở dữ liệu cũ. Trong trường hợp có nhiều cơ sở dữ liệu phụ, các thao tác đọc được thực hiện được chuyển hướng đến cơ sở dữ liệu nô lệ khỏe mạnh khác. Một máy chủ cơ sở dữ liệu mới sẽ thay thế máy chủ cũ.
* Nếu cơ sở dữ liệu chính ngoại tuyến, cơ sở dữ liệu phụ sẽ được nâng cấp thành cơ sở dữ liệu mới bậc thầy. Tất cả các hoạt động cơ sở dữ liệu sẽ được thực hiện tạm thời trên cơ sở dữ liệu chính mới. Cơ sở dữ liệu nô lệ mới sẽ thay thế cơ sở dữ liệu cũ để sao chép dữ liệu ngay lập tức. Trong các hệ thống sản xuất, việc quảng bá một bản gốc mới phức tạp hơn vì dữ liệu trong cơ sở dữ liệu nô lệ có thể không được cập nhật. Dữ liệu bị thiếu cần được cập nhật bằng cách chạy tập lệnh khôi phục dữ liệu. Mặc dù một số phương pháp sao chép khác như đa bản chính và sao chép vòng tròn có thể hữu ích nhưng những thiết lập đó phức tạp hơn; và các cuộc thảo luận của họ nằm ngoài phạm vi của cuốn sách này. Bạn đọc quan tâm có thể tham khảo các tài liệu tham khảo được liệt kê [4] [5].

**Figure 1-6** cho thấy thiết kế hệ thống sau khi thêm bộ cân bằng tải và sao chép cơ sở dữ liệu.



Chúng ta hãy nhìn vào thiết kế:

* Người dùng lấy địa chỉ IP của bộ cân bằng tải từ DNS.
* Người dùng kết nối cân bằng tải với địa chỉ IP này.
* Yêu cầu HTTP được định tuyến tới Máy chủ 1 hoặc Máy chủ 2.
* Máy chủ web đọc dữ liệu người dùng từ cơ sở dữ liệu phụ.
* Máy chủ web định tuyến mọi hoạt động sửa đổi dữ liệu tới cơ sở dữ liệu chính. Điêu nay bao gồm thao tác ghi, cập nhật và xóa.

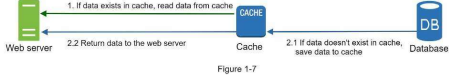
Bây giờ, bạn đã hiểu rõ về web và các tầng dữ liệu, đã đến lúc cải thiện thời gian tải/phản hồi. Điều này có thể được thực hiện bằng cách thêm lớp bộ đệm và chuyển nội dung tĩnh (tệp JavaScript/CSS/hình ảnh/video) sang mạng phân phối nội dung (CDN)

**Cache – Bộ nhớ đệm**

Bộ đệm là vùng lưu trữ tạm thời lưu trữ kết quả của các phản hồi đắt tiền hoặc dữ liệu được truy cập thường xuyên trong bộ nhớ để các yêu cầu tiếp theo được phục vụ nhanh hơn. Như minh họa trong **Figure 1-6**, mỗi khi tải một trang web mới, một hoặc nhiều lệnh gọi cơ sở dữ liệu sẽ được thực hiện để tìm nạp dữ liệu. Hiệu suất ứng dụng bị ảnh hưởng lớn khi gọi cơ sở dữ liệu nhiều lần. Bộ đệm có thể giảm thiểu vấn đề này.

**Cache tier – Các bậc của bộ nhớ đệm**

Tầng bộ đệm là lớp lưu trữ dữ liệu tạm thời, nhanh hơn nhiều so với cơ sở dữ liệu. Lợi ích của việc có một tầng bộ đệm riêng biệt bao gồm hiệu năng hệ thống tốt hơn, khả năng giảm khối lượng công việc cơ sở dữ liệu và khả năng mở rộng quy mô tầng bộ đệm một cách độc lập. **Figure 1-7** cho thấy khả năng thiết lập của máy chủ bộ đệm:



Sau khi nhận được yêu cầu, trước tiên máy chủ web sẽ kiểm tra xem bộ đệm có phản hồi khả dụng hay không. Nếu có, nó sẽ gửi dữ liệu trở lại máy khách. Nếu không, nó sẽ truy vấn cơ sở dữ liệu, lưu trữ phản hồi trong bộ đệm và gửi lại cho máy khách. Chiến lược bộ đệm này được gọi là bộ đệm đọc qua. Các chiến lược bộ nhớ đệm khác có sẵn tùy thuộc vào loại dữ liệu, kích thước và kiểu truy cập. Nghiên cứu trước đây giải thích cách hoạt động của các chiến lược bộ nhớ đệm khác nhau [6].

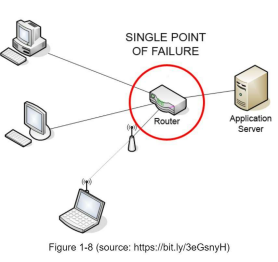
Tương tác với máy chủ bộ đệm rất đơn giản vì hầu hết máy chủ bộ đệm đều cung cấp API cho các ngôn ngữ lập trình phổ biến. Đoạn mã sau đây hiển thị các MemcachedAPI điển hình:



**Những cân nhắc khi sử dụng bộ đệm**

Dưới đây là một số cân nhắc khi sử dụng hệ thống bộ nhớ đệm:

* Quyết định khi nào nên sử dụng bộ nhớ đệm. Hãy cân nhắc việc sử dụng bộ đệm khi dữ liệu được đọc thường xuyên nhưng được sửa đổi không thường xuyên. Vì dữ liệu được lưu trong bộ nhớ đệm được lưu trữ trong bộ nhớ dễ thay đổi nên máy chủ bộ đệm không lý tưởng để lưu trữ dữ liệu. Ví dụ: nếu máy chủ bộ đệm khởi động lại, tất cả dữ liệu trong bộ nhớ sẽ bị mất. Vì vậy, dữ liệu quan trọng nên được lưu trong kho dữ liệu liên tục.
* Chính sách hết hạn. Đó là một thực hành tốt để thực hiện chính sách hết hạn. Sau khi được lưu vào bộ nhớ đệm dữ liệu hết hạn, nó sẽ bị xóa khỏi bộ đệm. Khi không có chính sách hết hạn, dữ liệu được lưu trong bộ nhớ đệm sẽ được lưu trữ vĩnh viễn trong bộ nhớ. Không nên đặt ngày hết hạn quá ngắn vì điều này sẽ khiến hệ thống tải lại dữ liệu từ cơ sở dữ liệu quá thường xuyên. Trong khi đó, không nên đặt ngày hết hạn quá dài vì dữ liệu có thể trở nên cũ.
* Tính nhất quán: Điều này liên quan đến việc giữ đồng bộ kho dữ liệu và bộ nhớ đệm. Sự không nhất quán có thể xảy ra do các hoạt động sửa đổi dữ liệu trên kho lưu trữ dữ liệu và bộ nhớ đệm không nằm trong một giao dịch duy nhất. Khi mở rộng quy mô trên nhiều vùng, việc duy trì tính nhất quán giữa việc lưu trữ dữ liệu và bộ đệm là một thách thức. Để biết thêm chi tiết, hãy tham khảo bài viết có tiêu đề “Scaling Memcache at Facebook” do Facebook xuất bản [7].
* Giảm thiểu lỗi: Một máy chủ bộ nhớ đệm đơn thể hiện một điểm lỗi tiềm ẩn (SPOF), được định nghĩa trong Wikipedia như sau: “Một điểm lỗi duy nhất (SPOF) là một phần của hệ thống mà nếu bị lỗi sẽ khiến toàn bộ hệ thống ngừng hoạt động” [8]. Kết quả là, nên sử dụng máy chủ bộ đệm trên các trung tâm dữ liệu khác nhau để tránh SPOF. Một cách tiếp cận khác được đề xuất là cung cấp quá mức bộ nhớ cần thiết theo một tỷ lệ nhất định. Điều này cung cấp một bộ đệm khi mức sử dụng bộ nhớ tăng lên.



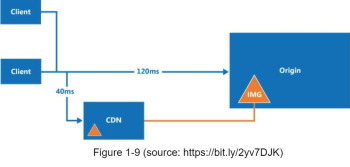
Chính sách Trục xuất: Khi bộ đệm đầy, mọi yêu cầu thêm mục vào bộ đệm có thể khiến các mục hiện có bị loại bỏ. Điều này được gọi là loại bỏ bộ nhớ đệm. Ít được sử dụng gần đây nhất (LRU) là chính sách loại bỏ bộ nhớ đệm phổ biến nhất. Các chính sách trục xuất khác, chẳng hạn như Chính sách ít được sử dụng thường xuyên nhất (LFU) hoặc Nhập trước xuất trước (FIFO), có thể được áp dụng để đáp ứng các trường hợp sử dụng khác nhau.

**Mạng phân phối nội dung (CDN)**

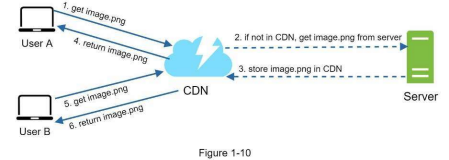
CDN là một mạng lưới các máy chủ phân tán về mặt địa lý được sử dụng để cung cấp nội dung tĩnh. Máy chủ CDN lưu trữ nội dung tĩnh như hình ảnh, video, CSS, tệp JavaScript, v.v.

Bộ nhớ đệm nội dung động là một khái niệm tương đối mới và nằm ngoài phạm vi của cuốn sách này. Nó cho phép lưu vào bộ đệm của các trang HTML dựa trên đường dẫn yêu cầu, chuỗi truy vấn, cookie và tiêu đề yêu cầu. Tham khảo bài viết được đề cập trong tài liệu tham khảo [9] để biết thêm về điều này. Cuốn sách này tập trung vào cách sử dụng CDN để lưu trữ nội dung tĩnh.

Đây là cách CDN hoạt động ở cấp độ cao: khi người dùng truy cập một trang web, máy chủ CDN gần người dùng nhất sẽ cung cấp nội dung tĩnh. Theo trực giác, càng có nhiều người dùng từ máy chủ CDN thì tốc độ tải trang web càng chậm. Ví dụ: nếu máy chủ CDN ở San Francisco, người dùng ở LosAngeles sẽ nhận được nội dung nhanh hơn người dùng ở Châu Âu. **Figure 1-9** là một ví dụ tuyệt vời cho thấy showshow CDN cải thiện thời gian tải.



**Figure 1-10** minh họa quy trình làm việc của CDN.



1. Người dùng A cố gắng lấy image.png bằng cách sử dụng URL hình ảnh. Tên miền của URL được cung cấp bởi nhà cung cấp CDN. Hai URL hình ảnh sau đây là các mẫu được sử dụng để minh họa URL hình ảnh trông như thế nào trên CDN của Amazon và Akamai:

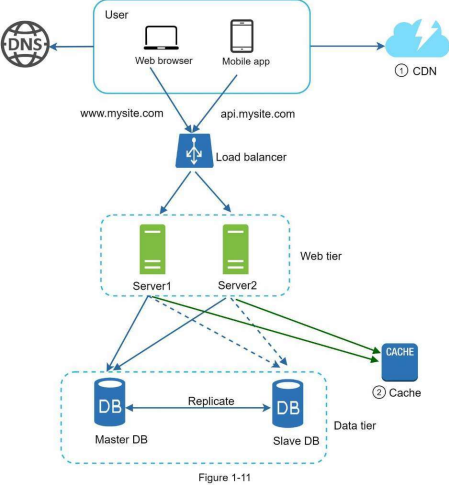
* <https://mysite.cloudfront.net/logo.jpg>
* <https://mysite.akamai.com/image-manager/img/logo.jpg>

1. Nếu máy chủ CDN không có image.png trong bộ đệm, máy chủ CDN sẽ yêu cầu tệp từ nguồn, có thể là máy chủ web hoặc bộ lưu trữ trực tuyến như Amazon S3.
2. Nguồn gốc trả về image.png cho máy chủ CDN, bao gồm tiêu đề HTTP Thời gian tồn tại (TTL) tùy chọn mô tả khoảng thời gian hình ảnh được lưu vào bộ nhớ đệm.
3. CDN lưu hình ảnh vào bộ nhớ đệm và trả lại cho Người dùng A. Hình ảnh vẫn được lưu vào bộ đệm trong CDN cho đến khi TTL hết hạn.
4. Người dùng B gửi yêu cầu lấy hình ảnh tương tự.
5. Hình ảnh được trả về từ bộ đệm miễn là TTL chưa hết hạn.

**Những cân nhắc khi sử dụng CDN**

* Chi phí: CDN do nhà cung cấp bên thứ ba điều hành và bạn bị tính phí truyền dữ liệu vào và ra khỏi CDN. Việc lưu vào bộ nhớ đệm những nội dung không được sử dụng thường xuyên không mang lại lợi ích đáng kể nào nên bạn nên cân nhắc việc chuyển chúng ra khỏi CDN.
* Đặt thời hạn bộ nhớ đệm thích hợp: Đối với nội dung nhạy cảm về thời gian, đặt thời hạn bộ nhớ đệm thời gian là quan trọng. Thời gian hết hạn của bộ đệm không được quá dài cũng không quá ngắn. Nếu quá dài, nội dung có thể không còn mới mẻ nữa. Nếu quá ngắn, nó có thể khiến nội dung được tải lại nhiều lần từ máy chủ gốc sang CDN.
* Dự phòng CDN: Bạn nên xem xét cách ứng phó với trang web/ứng dụng của bạn với CDN sự thất bại. Nếu CDN ngừng hoạt động tạm thời, khách hàng sẽ có thể phát hiện sự cố và yêu cầu tài nguyên từ nguồn.
* Vô hiệu hóa tệp: Bạn có thể xóa tệp khỏi CDN trước khi nó hết hạn bằng cách thực hiện một trong các thao tác sau:
* Vô hiệu hóa đối tượng CDN bằng API do nhà cung cấp CDN cung cấp.
* Sử dụng phiên bản đối tượng để phục vụ một phiên bản khác của đối tượng. Để phiên bản một đối tượng, bạn có thể thêm tham số vào URL, chẳng hạn như số phiên bản. Ví dụ: số phiên bản 2 được thêm vào chuỗi truy vấn: image.png?v=2.

**Figure 1-11** cho thấy thiết kế sau khi CDN và bộ đệm được thêm vào.



1. Nội dung tĩnh (JS, CSS, hình ảnh, v.v.) không còn được máy chủ web phục vụ nữa. Chúng được tìm nạp từ CDN để có hiệu suất tốt hơn.
2. Tải cơ sở dữ liệu được giảm nhẹ nhờ dữ liệu được lưu vào bộ nhớ đệm.

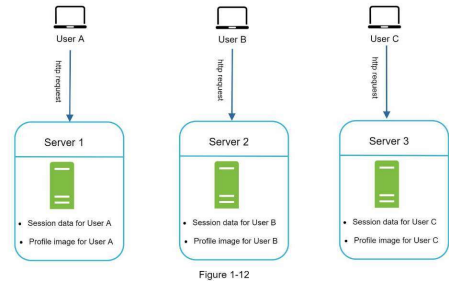
**Tầng web không quốc tịch**

Bây giờ là lúc xem xét mở rộng tầng web theo chiều ngang. Để làm được điều này, chúng ta cần chuyển trạng thái (ví dụ: dữ liệu phiên của người dùng) ra khỏi tầng web. Một cách thực hành tốt là lưu trữ dữ liệu phiên trong bộ lưu trữ liên tục như cơ sở dữ liệu quan hệ hoặc NoSQL. Mỗi máy chủ web trong cụm có thể truy cập dữ liệu trạng thái từ cơ sở dữ liệu. Đây được gọi là tầng web không trạng thái.

**Kiến trúc trạng thái**

Máy chủ có trạng thái và máy chủ không có trạng thái có một số điểm khác biệt chính. Máy chủ có trạng thái ghi nhớ dữ liệu (trạng thái) của khách hàng từ yêu cầu này sang yêu cầu tiếp theo. Một máy chủ không trạng thái không giữ thông tin trạng thái.

**Figure 1-12** cho thấy một ví dụ về kiến ​​trúc có trạng thái.

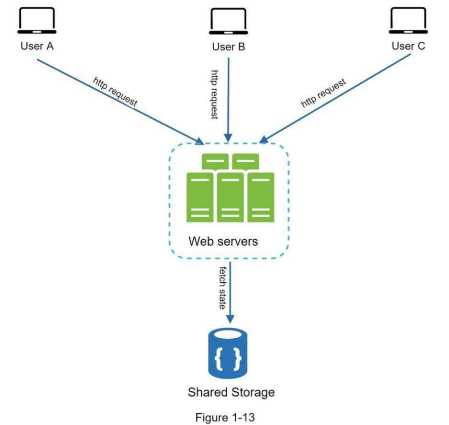


Trong **Figure 1-12**, dữ liệu phiên và hình ảnh hồ sơ của người dùng A được lưu trữ trong Máy chủ 1. Để xác thực Người dùng A, các yêu cầu HTTP phải được định tuyến đến Máy chủ 1. Nếu một yêu cầu được gửi đến các máy chủ khác như Máy chủ 2, việc xác thực sẽ thất bại vì Máy chủ 2 thực hiện điều đó không chứa dữ liệu phiên của Người dùng A. Tương tự, tất cả yêu cầu HTTP từ Người dùng B phải được định tuyến đến Máy chủ 2; tất cả các yêu cầu từ UserC phải được gửi đến Máy chủ 3.

Vấn đề là mọi yêu cầu từ cùng một máy khách phải được chuyển đến cùng một máy chủ. Điều này có thể được thực hiện bằng các phiên cố định trong hầu hết các bộ cân bằng tải [10]; tuy nhiên, điều này làm tăng thêm chi phí. Việc thêm hoặc xóa máy chủ khó khăn hơn nhiều với phương pháp này. Việc xử lý các lỗi máy chủ cũng là một thách thức.

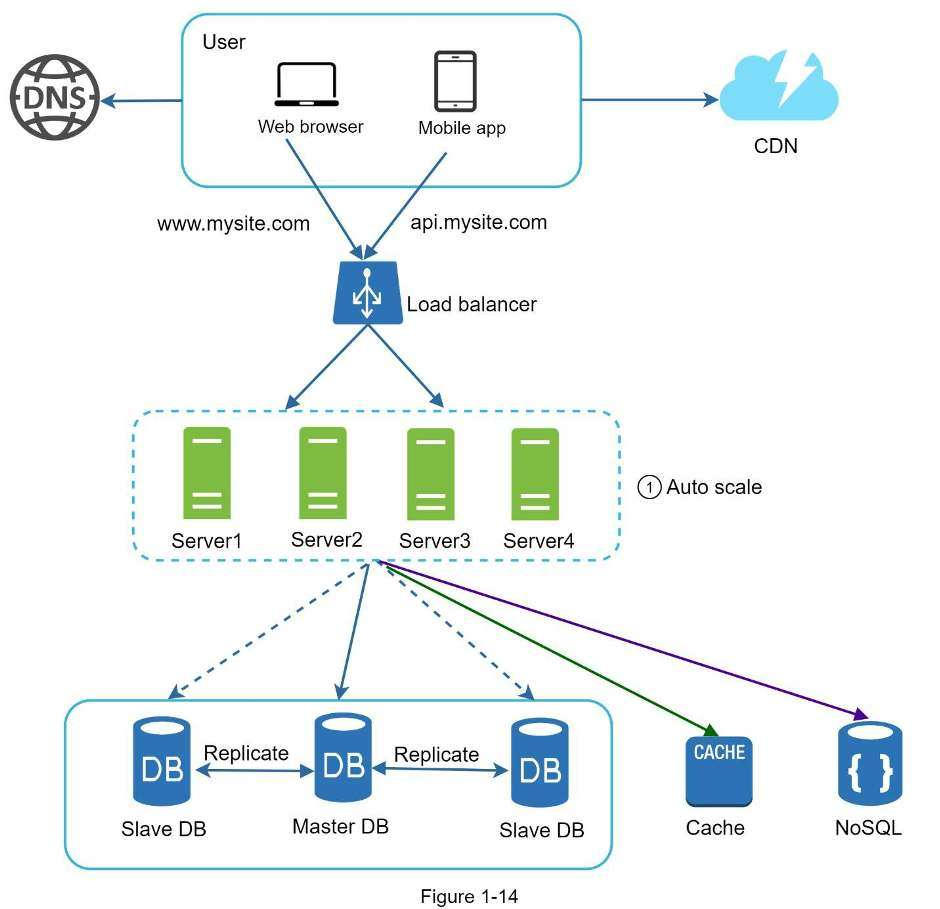
**Kiến trúc phi quốc tịch**

**Figure 1-13** cho thấy kiến ​​trúc phi trạng thái.



Trong kiến ​​trúc không trạng thái này, các yêu cầu HTTP từ người dùng có thể được gửi đến bất kỳ máy chủ web nào để lấy dữ liệu trạng thái từ kho lưu trữ dữ liệu dùng chung. Dữ liệu trạng thái được lưu trữ trong kho dữ liệu dùng chung và không được đưa vào máy chủ web. Một hệ thống không trạng thái đơn giản hơn, mạnh mẽ hơn và có thể mở rộng.

**Figure 1-14** cho thấy thiết kế được cập nhật với tầng web không trạng thái.

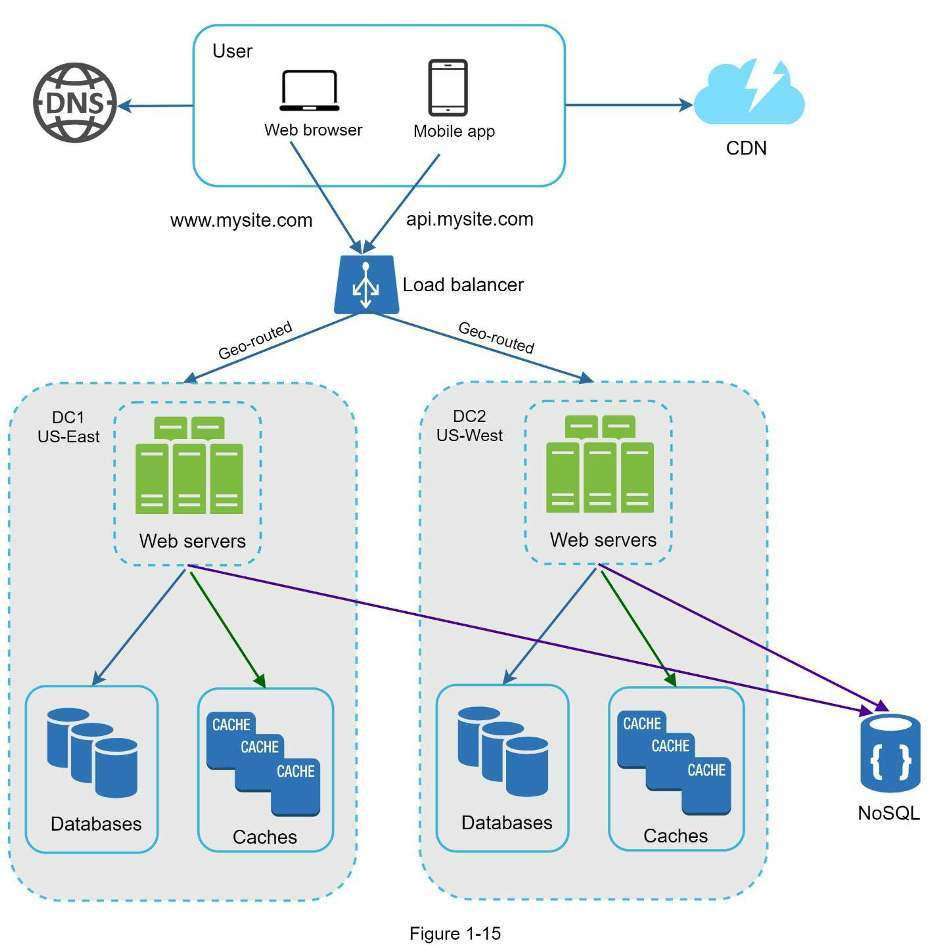


Trong **Figure 1-14**, chúng tôi di chuyển dữ liệu phiên ra khỏi tầng web và lưu trữ chúng trong kho dữ liệu liên tục. Kho dữ liệu dùng chung có thể là cơ sở dữ liệu quan hệ, Memcached/Redis, NoSQL, v.v. Kho dữ liệu NoSQL được chọn vì nó dễ mở rộng quy mô. Tự động điều chỉnh có nghĩa là tự động thêm hoặc xóa máy chủ web dựa trên lưu lượng truy cập. Sau khi dữ liệu trạng thái được xóa khỏi máy chủ web, bạn có thể dễ dàng tự động mở rộng cấp độ web bằng cách thêm hoặc xóa máy chủ dựa trên tải lưu lượng truy cập.

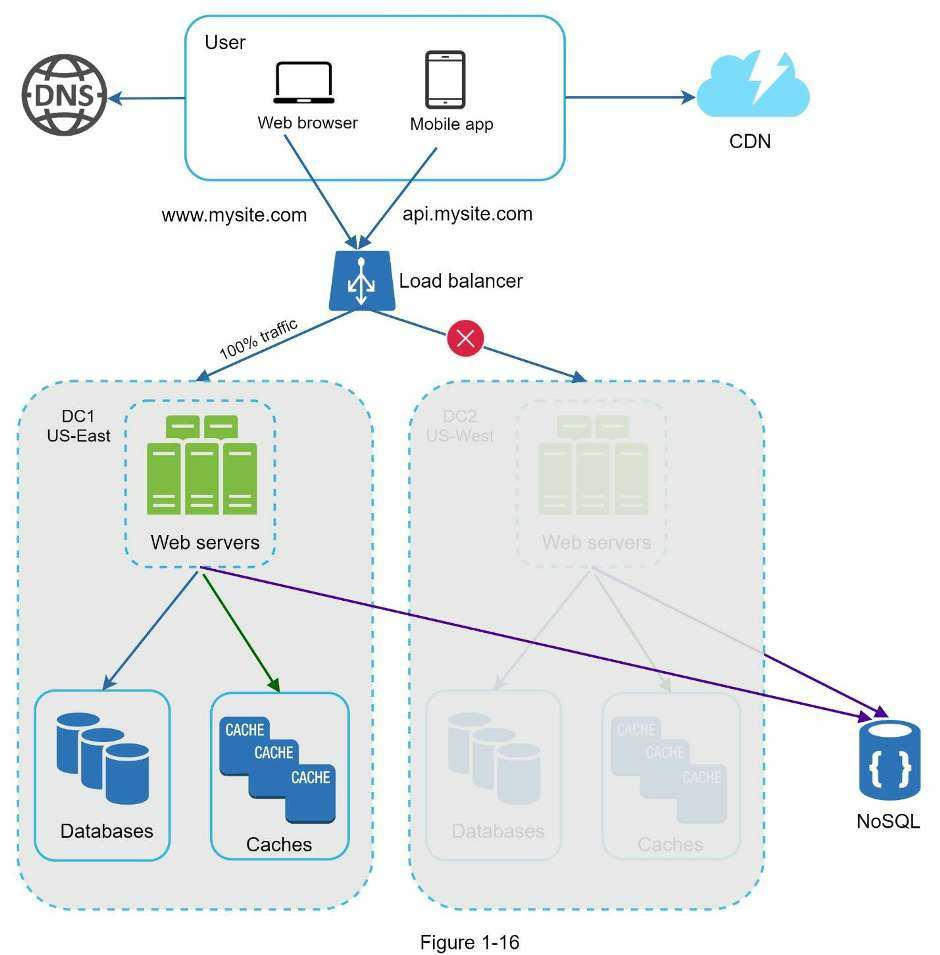
Trang web của bạn phát triển nhanh chóng và thu hút một lượng lớn người dùng quốc tế. Để cải thiện tính khả dụng và cung cấp trải nghiệm người dùng tốt hơn trên các khu vực địa lý rộng hơn, việc hỗ trợ nhiều trung tâm dữ liệu là rất quan trọng.

**Các trung tâm dữ liệu**

**Figure 1-15** hiển thị một ví dụ thiết lập với hai trung tâm dữ liệu. Trong hoạt động bình thường, người dùng được định tuyến theo địa lýDNS, còn được gọi là định tuyến theo địa lý, đến trung tâm dữ liệu gần nhất, với lưu lượng truy cập được phân chia x% ở Miền Đông Hoa Kỳ và (100 – x)% ở Miền Tây Hoa Kỳ. GeoDNS là dịch vụ DNS cho phép phân giải tên miền thành địa chỉ IP dựa trên vị trí của người dùng.



Trong trường hợp có bất kỳ sự cố ngừng hoạt động nghiêm trọng nào của trung tâm dữ liệu, chúng tôi sẽ hướng tất cả lưu lượng truy cập đến một trung tâm dữ liệu tốt. Trong **Figure 1-16**, trung tâm dữ liệu 2 (US-West) đang ngoại tuyến và 100% lưu lượng truy cập được chuyển đến trung tâm dữ liệu 1 (US-West) -Phía đông).



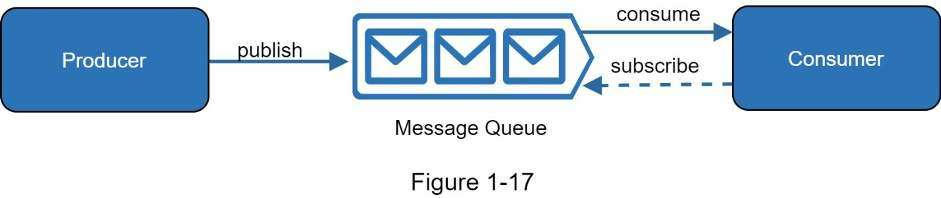
Một số thách thức kỹ thuật phải được giải quyết để đạt được thiết lập trung tâm đa dữ liệu:

* Chuyển hướng lưu lượng truy cập: Cần có các công cụ hiệu quả để hướng lưu lượng truy cập đến đúng trung tâm dữ liệu. GeoDNS có thể được sử dụng để hướng lưu lượng truy cập đến trung tâm dữ liệu gần nhất tùy thuộc vào vị trí của người dùng.
* Đồng bộ hóa dữ liệu: Người dùng từ các khu vực khác nhau có thể sử dụng cơ sở dữ liệu cục bộ khác nhau hoặc bộ nhớ đệm. Trong trường hợp chuyển đổi dự phòng, lưu lượng truy cập có thể được định tuyến đến trung tâm dữ liệu nơi không có dữ liệu. Chiến lược phổ biến là sao chép dữ liệu trên nhiều trung tâm dữ liệu. Một nghiên cứu trước đây cho thấy cách Netflix triển khai sao chép trung tâm đa dữ liệu không đồng bộ [11].
* Kiểm tra và triển khai: Với thiết lập trung tâm đa dữ liệu, điều quan trọng là phải kiểm tra trang web/ứng dụng tại các địa điểm khác nhau. Các công cụ triển khai tự động rất quan trọng để giữ cho các dịch vụ nhất quán trong tất cả các trung tâm dữ liệu [11].

Để mở rộng quy mô hệ thống của mình hơn nữa, chúng tôi cần tách riêng các thành phần khác nhau của hệ thống để chúng có thể được mở rộng quy mô một cách độc lập. Hàng đợi nhắn tin là một chiến lược quan trọng được nhiều hệ thống phân tán trong thế giới thực sử dụng để giải quyết vấn đề này.

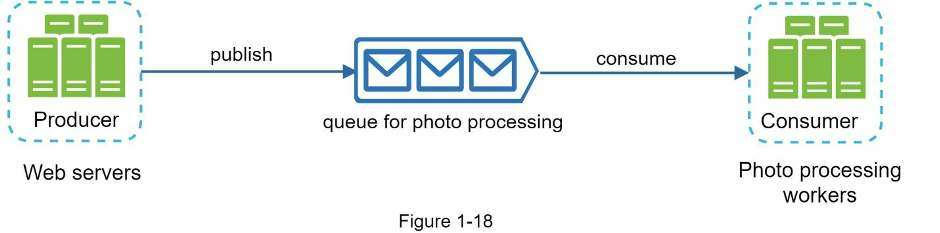
**Tin nhắn xếp hàng**

Hàng đợi tin nhắn là một thành phần lâu bền, được lưu trữ trong bộ nhớ, hỗ trợ giao tiếp không đồng bộ. Nó phục vụ như một bộ đệm và phân phối các yêu cầu không đồng bộ. Kiến trúc cơ bản của hàng đợi tin nhắn rất đơn giản. Các dịch vụ đầu vào, được gọi là nhà sản xuất/nhà xuất bản, tạo tin nhắn và xuất bản chúng vào hàng đợi tin nhắn. Các dịch vụ hoặc máy chủ khác, được gọi là người tiêu dùng/người đăng ký, kết nối với hàng đợi và thực hiện các hành động được xác định bởi tin nhắn. Mô hình được thể hiện trong **Figure 1-17**.



Việc tách rời làm cho hàng đợi tin nhắn trở thành một kiến ​​trúc được ưu tiên để xây dựng một ứng dụng đáng tin cậy và có thể mở rộng. Với hàng đợi tin nhắn, nhà sản xuất có thể đăng một tin nhắn lên hàng đợi khi người tiêu dùng không có mặt để xử lý nó. Người tiêu dùng có thể đọc tin nhắn từ hàng đợi ngay cả khi nhà sản xuất không có mặt.

Hãy xem xét trường hợp sử dụng sau: ứng dụng của bạn hỗ trợ tùy chỉnh ảnh, bao gồm cắt xén, làm sắc nét, làm mờ, v.v. Những tác vụ tùy chỉnh đó cần có thời gian để hoàn thành. Trong **Figure 1-18**, máy chủ web xuất bản các công việc xử lý ảnh vào hàng đợi tin nhắn. Công nhân xử lý ảnh nhận công việc từ hàng đợi tin nhắn và thực hiện các tác vụ tùy chỉnh ảnh một cách không đồng bộ. Nhà sản xuất và người tiêu dùng có thể được mở rộng quy mô một cách độc lập. Khi kích thước của hàng đợi trở nên lớn, nhiều công nhân sẽ được thêm vào để giảm thời gian xử lý. Tuy nhiên, nếu hàng đợi trống hầu hết thời gian, số lượng công nhân có thể giảm.



**Ghi nhật ký, số liệu, tự động hóa**

Khi làm việc với một trang web nhỏ chạy trên một vài máy chủ, việc hỗ trợ ghi nhật ký, số liệu và tự động hóa là những phương pháp hay nhưng không cần thiết. Tuy nhiên, giờ đây trang web của bạn đã phát triển để phục vụ một doanh nghiệp lớn, việc đầu tư vào những công cụ đó là điều cần thiết.

Ghi nhật ký: Việc theo dõi nhật ký lỗi rất quan trọng vì nó giúp xác định các lỗi và sự cố trong hệ thống. Bạn có thể theo dõi nhật ký lỗi ở cấp độ máy chủ hoặc sử dụng các công cụ để tổng hợp chúng thành một dịch vụ tập trung để dễ dàng tìm kiếm và xem.

Số liệu: Việc thu thập các loại số liệu khác nhau giúp chúng tôi hiểu rõ hơn về hoạt động kinh doanh và hiểu được tình trạng hoạt động của hệ thống. Một số số liệu sau đây rất hữu ích:

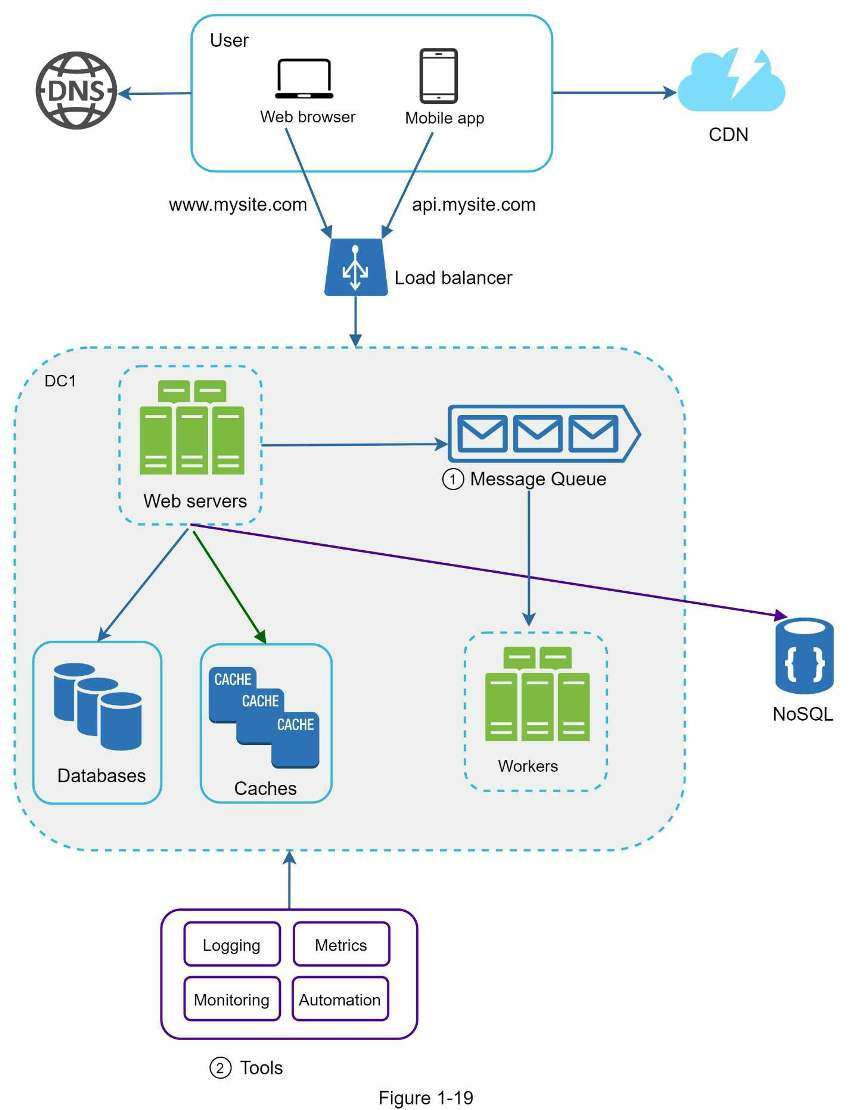
* Số liệu cấp độ máy chủ: CPU, Bộ nhớ, I/O đĩa, v.v.
* Số liệu cấp tổng hợp: ví dụ: hiệu suất của toàn bộ tầng cơ sở dữ liệu, bộ nhớ đệm cấp, v.v.
* Các số liệu kinh doanh chính: số người dùng hoạt động hàng ngày, tỷ lệ giữ chân, doanh thu, v.v.

Tự động hóa: Khi một hệ thống trở nên lớn và phức tạp, chúng ta cần xây dựng hoặc tận dụng các công cụ tự động hóa để cải thiện năng suất. Tích hợp liên tục là một phương pháp hay, trong đó mỗi lần kiểm tra mã được xác minh thông qua tự động hóa, cho phép các nhóm phát hiện sớm sự cố. Ngoài ra, việc tự động hóa quá trình xây dựng, thử nghiệm, triển khai, v.v. của bạn có thể cải thiện đáng kể năng suất của nhà phát triển.

**Thêm hàng đợi tin nhắn và các công cụ khác nhau**

**Figure 1-19** thể hiện thiết kế được cập nhật. Do hạn chế về không gian, chỉ có một trung tâm dữ liệu được hiển thị trong hình.

1. Thiết kế bao gồm một hàng đợi tin nhắn, giúp làm cho hệ thống được kết nối lỏng lẻo hơn và có khả năng phục hồi tốt hơn khi xảy ra lỗi.
2. Bao gồm các công cụ ghi nhật ký, giám sát, số liệu và tự động hóa



Khi dữ liệu tăng lên mỗi ngày, cơ sở dữ liệu của bạn sẽ bị quá tải hơn. Đã đến lúc mở rộng quy mô datatier

**Chia tỷ lệ cơ sở dữ liệu**

Có hai cách tiếp cận rộng rãi để chia tỷ lệ cơ sở dữ liệu: chia tỷ lệ theo chiều dọc và tỷ lệ theo chiều ngang.

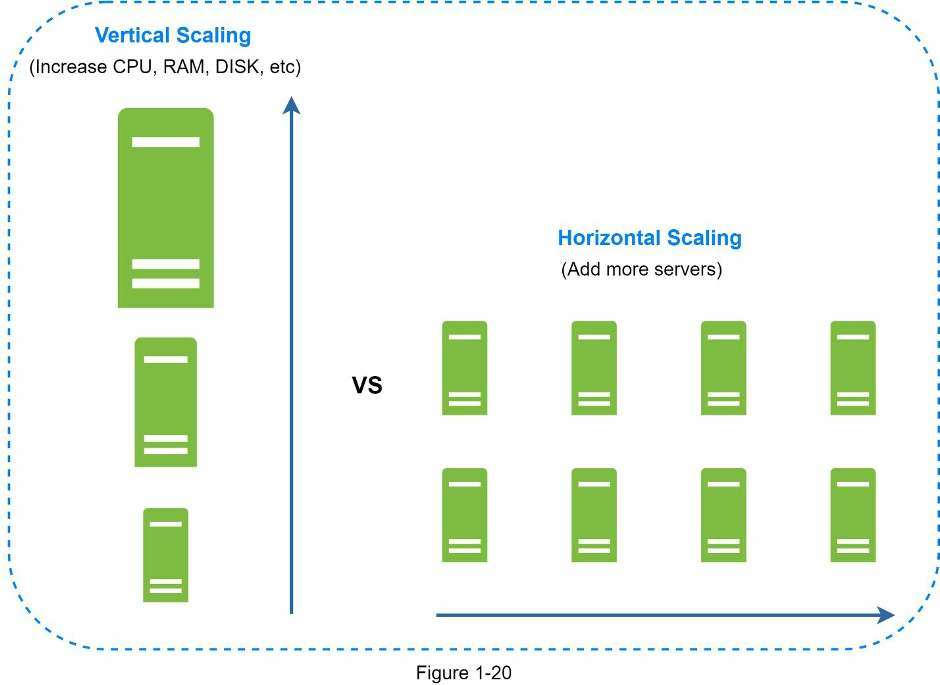
**Chia tỷ lệ dọc**

Chia tỷ lệ theo chiều dọc, còn được gọi là tăng tỷ lệ, là tỷ lệ bằng cách bổ sung thêm năng lượng (CPU, RAM, DISK, v.v.) cho máy hiện có. Có một số máy chủ cơ sở dữ liệu mạnh mẽ. Theo Dịch vụ cơ sở dữ liệu quan hệ của Amazon (RDS) [12], bạn có thể có một máy chủ cơ sở dữ liệu có RAM 24 TB. Loại máy chủ cơ sở dữ liệu mạnh mẽ này có thể lưu trữ và xử lý nhiều dữ liệu. Ví dụ: stackoverflow.com năm 2013 có hơn 10 triệu khách truy cập hàng tháng, nhưng nó chỉ có 1 cơ sở dữ liệu chủ [13]. Tuy nhiên, chia tỷ lệ theo chiều dọc có một số nhược điểm nghiêm trọng:

* Bạn có thể bổ sung thêm CPU, RAM, v.v. vào máy chủ cơ sở dữ liệu của mình, nhưng phần cứng cũng có giới hạn của nó. Nếu bạn có lượng người dùng lớn thì chỉ một máy chủ là không đủ.
* Nguy cơ xảy ra lỗi một điểm cao hơn.
* Tổng chi phí của việc mở rộng quy mô theo chiều dọc cao. Máy chủ mạnh mẽ đắt hơn nhiều.

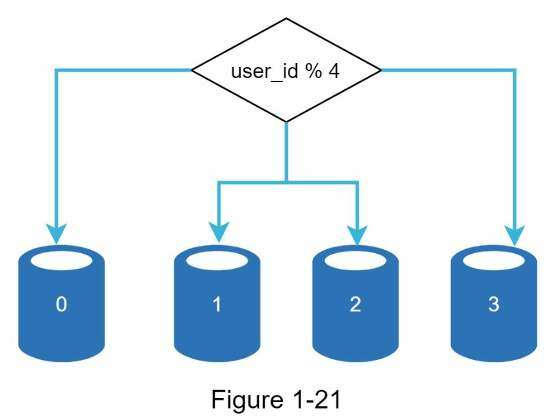
**Chia tỷ lệ theo chiều ngang**

Chia tỷ lệ theo chiều ngang, còn được gọi là sharding, là phương pháp thêm nhiều máy chủ hơn. Hình 1-20 so sánh tỷ lệ dọc và tỷ lệ ngang.

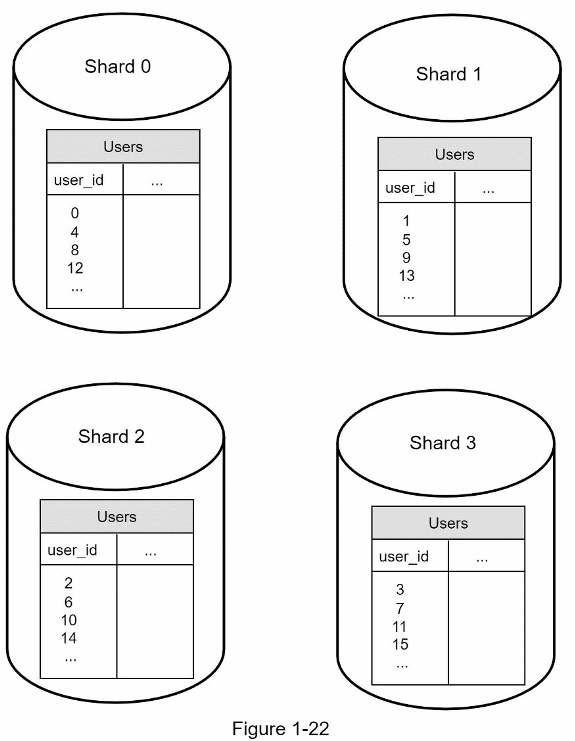


Sharding tách cơ sở dữ liệu lớn thành các phần nhỏ hơn, dễ quản lý hơn gọi là phân đoạn. Mỗi phân đoạn chia sẻ cùng một lược đồ, mặc dù dữ liệu thực tế trên mỗi phân đoạn là duy nhất cho phân đoạn đó.

**Figure 1-21** cho thấy một ví dụ về cơ sở dữ liệu được phân chia. Dữ liệu người dùng được phân bổ tới máy chủ cơ sở dữ liệu dựa trên ID người dùng. Bất cứ khi nào bạn truy cập dữ liệu, hàm băm sẽ được sử dụng để tìm phân đoạn tương ứng. Trong ví dụ của chúng tôi, user\_id % 4 được sử dụng làm hàm băm. Nếu kết quả bằng 0, phân đoạn 0 được sử dụng để lưu trữ và tìm nạp dữ liệu. Nếu kết quả bằng 1, phân đoạn 1 sẽ được sử dụng. Logic tương tự áp dụng cho các phân đoạn khác.



**Figure 1-22** hiển thị bảng người dùng trong cơ sở dữ liệu được phân chia.



Yếu tố quan trọng nhất cần xem xét khi thực hiện chiến lược sharding là lựa chọn khóa sharding. Khóa phân chia (được gọi là khóa phân vùng) bao gồm một hoặc nhiều cột xác định cách phân phối dữ liệu. Như được hiển thị trong **Figure 1-22**, “user\_id” là shardingkey. Khóa sharding cho phép bạn truy xuất và sửa đổi dữ liệu một cách hiệu quả bằng cách định tuyến các truy vấn cơ sở dữ liệu đến cơ sở dữ liệu chính xác. Khi chọn khóa sharding, một trong những điều quan trọng nhất tiêu chí là chọn một khóa có thể phân phối dữ liệu đồng đều.

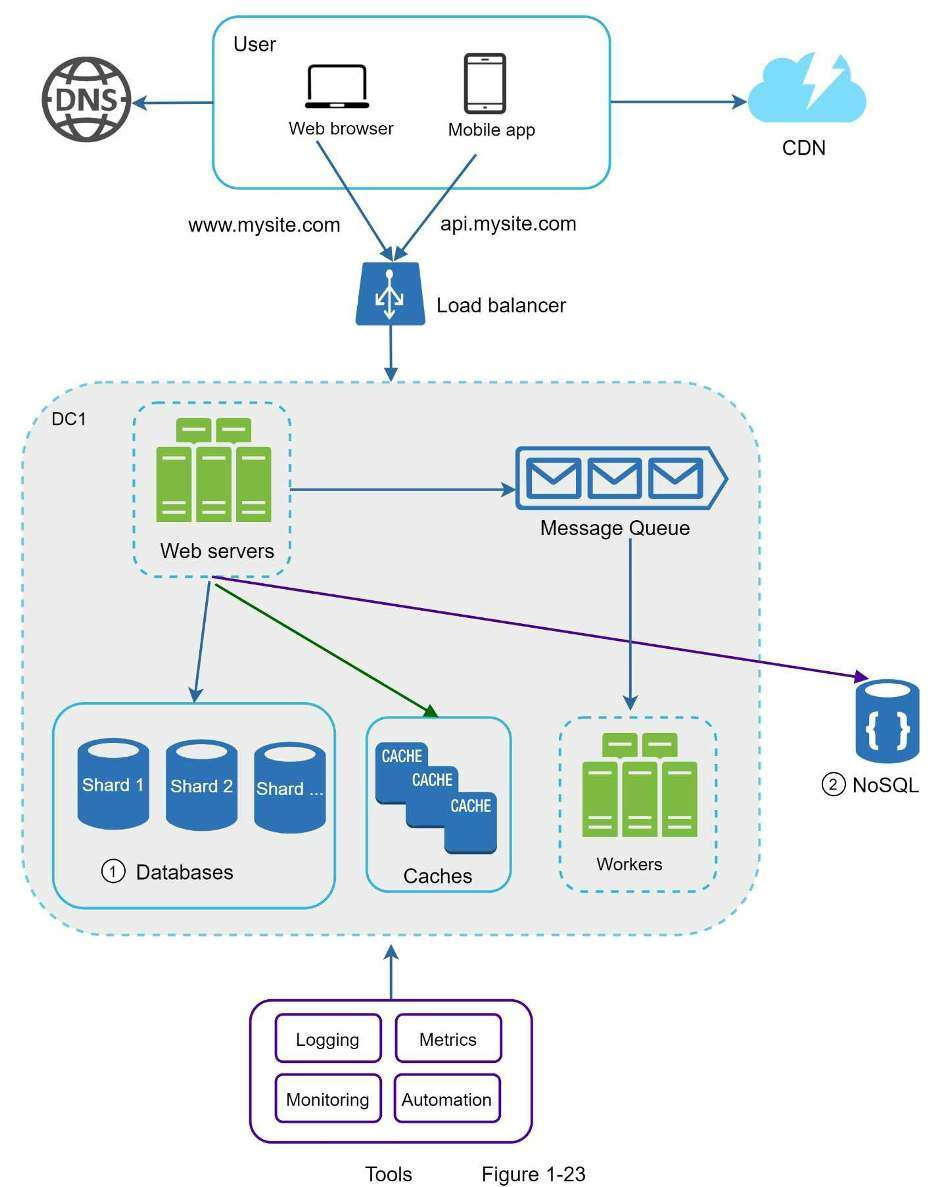
Sharding là một kỹ thuật tuyệt vời để mở rộng quy mô cơ sở dữ liệu nhưng nó vẫn chưa phải là một giải pháp hoàn hảo. Nó giới thiệu sự phức tạp và thách thức mới cho hệ thống:

**Phân chia lại dữ liệu**: Cần phân chia lại dữ liệu khi 1) một phân đoạn đơn lẻ không thể chứa được nhiều dữ liệu hơn do tốc độ tăng trưởng nhanh chóng. 2) Một số phân đoạn nhất định có thể bị cạn kiệt phân đoạn nhanh hơn các phân đoạn khác do phân phối dữ liệu không đồng đều. Khi tình trạng cạn kiệt phân đoạn xảy ra, nó yêu cầu cập nhật chức năng phân chia và di chuyển dữ liệu xung quanh. Băm nhất quán, sẽ được thảo luận trong Chương 5, là một kỹ thuật thường được sử dụng để giải quyết vấn đề này.

**Vấn đề về người nổi tiếng**: Đây còn được gọi là vấn đề về điểm phát sóng. Việc truy cập quá mức vào một phân đoạn cụ thể có thể gây ra tình trạng quá tải máy chủ. Hãy tưởng tượng dữ liệu của Katy Perry, Justin Bieber và LadyGaga đều xuất hiện trên cùng một phân đoạn. Đối với các ứng dụng xã hội, phân đoạn đó sẽ tràn ngập các thao tác đọc. Để giải quyết vấn đề này, chúng ta có thể cần phân bổ một phân đoạn cho mỗi người nổi tiếng. Mỗi phân đoạn thậm chí có thể yêu cầu phân vùng thêm.

**Tham gia và khử chuẩn hóa**: Khi cơ sở dữ liệu đã được phân chia trên nhiều máy chủ, thật khó để thực hiện các thao tác kết hợp trên các phân đoạn cơ sở dữ liệu. Một cách giải quyết phổ biến là khử chuẩn hóa cơ sở dữ liệu để có thể thực hiện các truy vấn trong một bảng duy nhất.

Trong **Figure 1-23**, chúng tôi phân chia cơ sở dữ liệu để hỗ trợ lưu lượng dữ liệu ngày càng tăng nhanh. Đồng thời, một số chức năng phi quan hệ được chuyển sang kho dữ liệu NoSQL để giảm tải cơ sở dữ liệu. Đây là một bài viết đề cập đến nhiều trường hợp sử dụng NoSQL [14].



**Hàng triệu người dùng và hơn thế nữa**

Mở rộng quy mô hệ thống là một quá trình lặp đi lặp lại. Lặp lại những gì chúng ta đã học được trong chương này có thể giúp chúng ta tiến xa hơn. Cần có nhiều chiến lược mới và tinh chỉnh hơn để mở rộng quy mô ra ngoài hàng triệu người dùng. Ví dụ: bạn có thể cần tối ưu hóa hệ thống của mình và tách hệ thống thành các dịch vụ thậm chí còn nhỏ hơn. Tất cả các kỹ thuật học được trong chương này sẽ cung cấp nền tảng tốt để giải quyết những thách thức mới. Để kết thúc chương này, chúng tôi cung cấp bản tóm tắt về cách chúng tôi mở rộng quy mô hệ thống của mình để hỗ trợ hàng triệu người dùng:

* Giữ tầng web không có trạng thái
* Xây dựng dự phòng ở mọi tầng
* Lưu vào bộ nhớ đệm dữ liệu nhiều nhất có thể
* Hỗ trợ nhiều trung tâm dữ liệu
* Lưu trữ nội dung tĩnh trong CDN • Mở rộng cấp dữ liệu của bạn bằng cách phân chia
* Chia tầng thành các dịch vụ riêng lẻ
* Giám sát hệ thống của bạn và sử dụng các công cụ tự động hóa

Chúc mừng bạn đã tiến xa đến mức này! Làm tốt lắm!

**TÀI LIỆU THAM KHẢO**

[1] Giao thức truyền siêu văn bản:

https://en.wikipedia.org/wiki/Hypertext\_Transfer\_Protocol

[2] Bạn có nên vượt ra ngoài cơ sở dữ liệu quan hệ?:

https://blog.teamtreehouse.com/Should-you-go-beyond-relational-databases

[3] Sao chép:https://en.wikipedia.org/wiki/Replication\_(máy tính)

[4] Sao chép đa chủ: <https://en.wikipedia.org/wiki/Multi-master_replication>

[5] Sao chép cụm NDB: Sao chép nhiều chủ và sao chép tròn:

https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-replication-multi-master.html

[6] Chiến lược bộ nhớ đệm và cách chọn chiến lược phù hợp: <https://codeahoy.com/2017/08/11/caching-strategies-and-how-to-choose-the-right-one/>

[7] R. Nishtala, "Facebook, Scaling Memcache at," 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI’13).

[8] Điểm lỗi duy nhất: https://en.wikipedia.org/wiki/Single\_point\_of\_failure

[9] Phân phối nội dung động của Amazon CloudFront:

https://aws.amazon.com/cloudfront/dynamic-content/

[10] Định cấu hình Phiên cố định cho Cân bằng tải cổ điển của bạn:

https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-sticky-sessions.html

[11] Active-Active cho khả năng phục hồi đa khu vực:

https://netflixtechblog.com/active-active-for-multi-khu-vực-resiliency-c47719f6685b

[12] Phiên bản bộ nhớ cao Amazon EC2: https://aws.amazon.com/ec2/instance-types/high-memory/

[13] Cần những gì để chạy Stack Overflow:

http://nickcraver.com/blog/2013/11/22/what-it-takes-to-run-stack-overflow

[14] Thực ra bạn đang sử dụng NoSQL để làm gì:

http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-USE-nosql for.html