

SECOND EDITION

SYSTEM DESIGN INTERVIEW



AN INSIDER'S GUIDE

Alex Xu

Phỏng vấn thiết kế hệ thống: Hư ứng dẫn của người trong cuộc

Mọi quyền được bảo lưu. Cuốn sách này hoặc bất kỳ phần nào của nó không được sao chép hoặc sử dụng dưới bất kỳ hình thức nào mà không có sự cho phép rõ ràng bằng văn bản của nhà xuất bản, ngoại trừ việc sử dụng các trích dẫn ngắn trong bài đánh giá sách.

Về tác giả:

Alex Xu là một kỹ sư phần mềm và doanh nhân giàu kinh nghiệm. Trước đây, anh đã làm việc tại Twitter, Apple, Zynga và Oracle. Anh đã nhận bằng MS từ Đại học Carnegie Mellon.

Anh ấy có niềm đam mê với việc thiết kế và triển khai các hệ thống phức tạp.

Vui lòng đăng ký danh sách email của chúng tôi nếu bạn muốn được thông báo khi có chương mới: <https://bit.ly/3dtTcsF>

Để biết thêm thông tin, hãy liên hệ systemdesigninsider@gmail.com

Biên tập viên: Paul Solomon

Mục lục

[Phỏng vấn thiết kế hệ thống: Hướng dẫn của người dùng cuộc](#)

[PHÍA TRƯỚC](#)

[CHƯƠNG 1: PHÁT TRIỂN TỪ SỐ KHÔNG ĐẾN HÀNG TRIỆU NGƯỜI DÙNG](#)

[CHƯƠNG 2: UỐC TÍNH TRÊN BÀI](#)

[CHƯƠNG 3: KHUNG CHO CÁC CUỘC PHỎNG VẤN THIẾT KẾ HỆ THỐNG](#)

[CHƯƠNG 4: THIẾT KẾ BỘ GIỚI HẠN TỐC ĐỘ](#)

[CHƯƠNG 5: THIẾT KẾ BASHING NHẤT QUÁN](#)

[CHƯƠNG 6: THIẾT KẾ MỘT CỦA HÀNG CHỈ NH-GIÁ TRỊ](#)

[CHƯƠNG 7: THIẾT KẾ MỘT BỘ TẠO ID DUY NHẤT TRONG CÁC HỆ THỐNG PHÂN TÁN](#)

[CHƯƠNG 8: THIẾT KẾ URL RÚT GỌN](#)

[CHƯƠNG 9: THIẾT KẾ WEB CRAWLER](#)

[CHƯƠNG 10: THIẾT KẾ HỆ THỐNG THÔNG BÁO](#)

[CHƯƠNG 11: THIẾT KẾ HỆ THỐNG TIN TỨC](#)

[CHƯƠNG 12: THIẾT KẾ HỆ THỐNG CHAT](#)

[CHƯƠNG 13: THIẾT KẾ HỆ THỐNG TỰ ĐỘNG HOÀN THÀNH TÌM KIẾM](#)

[CHƯƠNG 14: THIẾT KẾ YOUTUBE](#)

[CHƯƠNG 15: THIẾT KẾ GOOGLE DRIVE](#)

[CHƯƠNG 16: VIỆC HỌC TẬP VĂN TIẾP TỤC](#)

[LỜI BÊN KẾT](#)

PHÍ A TRƯỚC

Chúng tôi rất vui vì bạn đã quyết định tham gia cùng chúng tôi để tìm hiểu về phỏng vấn thiết kế hệ thống. Các câu hỏi phỏng vấn thiết kế hệ thống là khó trả lời nhất trong số tất cả các cuộc phỏng vấn kỹ thuật. Các câu hỏi yêu cầu người được phỏng vấn thiết kế kiến trúc cho một hệ thống phần mềm, có thể là nguồn cấp tin tức, tìm kiếm Google, hệ thống trò chuyện, v.v. Những câu hỏi này rất đáng sợ và không có khuôn mẫu nhất định nào để tuân theo. Các câu hỏi thường có phạm vi rất lớn và mơ hồ. Các quy trình là mở và không rõ ràng nếu không có tiêu chuẩn hoặc chính xác trả lời.

Các công ty áp dụng rộng rãi các cuộc phỏng vấn thiết kế hệ thống vì các kỹ năng giao tiếp và giải quyết vấn đề được kiểm tra trong các cuộc phỏng vấn này tương tự như những kỹ năng mà công việc hàng ngày của một kỹ sư phần mềm yêu cầu. Người được phỏng vấn được đánh giá dựa trên cách cô ấy phân tích một vấn đề mơ hồ và cách cô ấy giải quyết vấn đề từng bước. Các khả năng được kiểm tra cũng bao gồm cách cô ấy giải thích ý tưởng, thảo luận với người khác và đánh giá và tối ưu hóa hệ thống. Trong tiếng Anh, sử dụng "she" trôi chảy hơn "he hoặc she" hoặc nhảy giữa hai cách. Để việc đọc dễ dàng hơn, chúng tôi sử dụng đại từ nhân xưng giống cái trong suốt cuốn sách này. Không có ý định thiêu tôn trọng đối với các kỹ sư nam.

Các câu hỏi thiết kế hệ thống là câu hỏi mở. Giống như trong thế giới thực, có nhiều sự khác biệt và biến thể trong hệ thống. Kết quả mong muốn là đưa ra một kiến trúc để đạt được các mục tiêu thiết kế hệ thống. Các cuộc thảo luận có thể diễn ra theo nhiều cách khác nhau tùy thuộc vào người phỏng vấn. Một số người phỏng vấn có thể chọn kiến trúc cấp cao để bao quát mọi khía cạnh; trong khi một số người có thể chọn một hoặc nhiều lĩnh vực để tập trung vào. Thông thường, các yêu cầu, hạn chế và nút thắt của hệ thống phải được hiểu rõ để định hình hướng đi của cả người phỏng vấn và người được phỏng vấn.

Mục tiêu của cuốn sách này là cung cấp một chiến lược đáng tin cậy để tiếp cận các câu hỏi thiết kế hệ thống. Chiến lược và kiến thức đúng đắn là yếu tố quan trọng cho sự thành công của một cuộc phỏng vấn.

Cuốn sách này cung cấp kiến thức vững chắc trong việc xây dựng một hệ thống có thể mở rộng. Càng có nhiều kiến thức thu được từ việc đọc cuốn sách này, bạn càng được trang bị tốt hơn để giải quyết các câu hỏi về thiết kế hệ thống.

Cuốn sách này cũng cung cấp khuôn khổ từng bước về cách giải quyết câu hỏi thiết kế hệ thống. Nó cung cấp nhiều ví dụ để minh họa cho cách tiếp cận có hệ thống với các bước chi tiết mà bạn có thể làm theo. Với việc thực hành liên tục, bạn sẽ được trang bị tốt để giải quyết các câu hỏi phỏng vấn thiết kế hệ thống.

CHƯƠNG 1: QUY MÔ TỪ SỐ KHÔNG ĐẾN HÀNG TRIỆU

NGƯỜI DÙNG

Thiết kế một hệ thống hỗ trợ hàng triệu người dùng là một thách thức và là một hành trình đòi hỏi sự tinh chỉnh liên tục và cải tiến không ngừng. Trong chương này, chúng ta sẽ xây dựng một hệ thống hỗ trợ một người dùng duy nhất và dần dần mở rộng quy mô để phục vụ hàng triệu người dùng. Sau khi đọc chương này, bạn sẽ nắm vững một số kỹ thuật giúp bạn giải quyết các câu hỏi phỏng vấn thiết kế hệ thống.

Thiết lập máy chủ đơn

Là một hành trình ngần dặm bắt đầu bằng một bút ớc chén, và xây dựng một hệ thống phức tạp cũng không khác gì. Để bắt đầu với một điều đơn giản, mọi thứ đều chạy trên một máy chủ duy nhất. Hình 1-1 minh họa thiết lập máy chủ đơn lẻ, trong đó mọi thứ đều chạy trên một máy chủ: Ứng dụng web, cơ sở dữ liệu, bộ nhớ đệm, v.v.

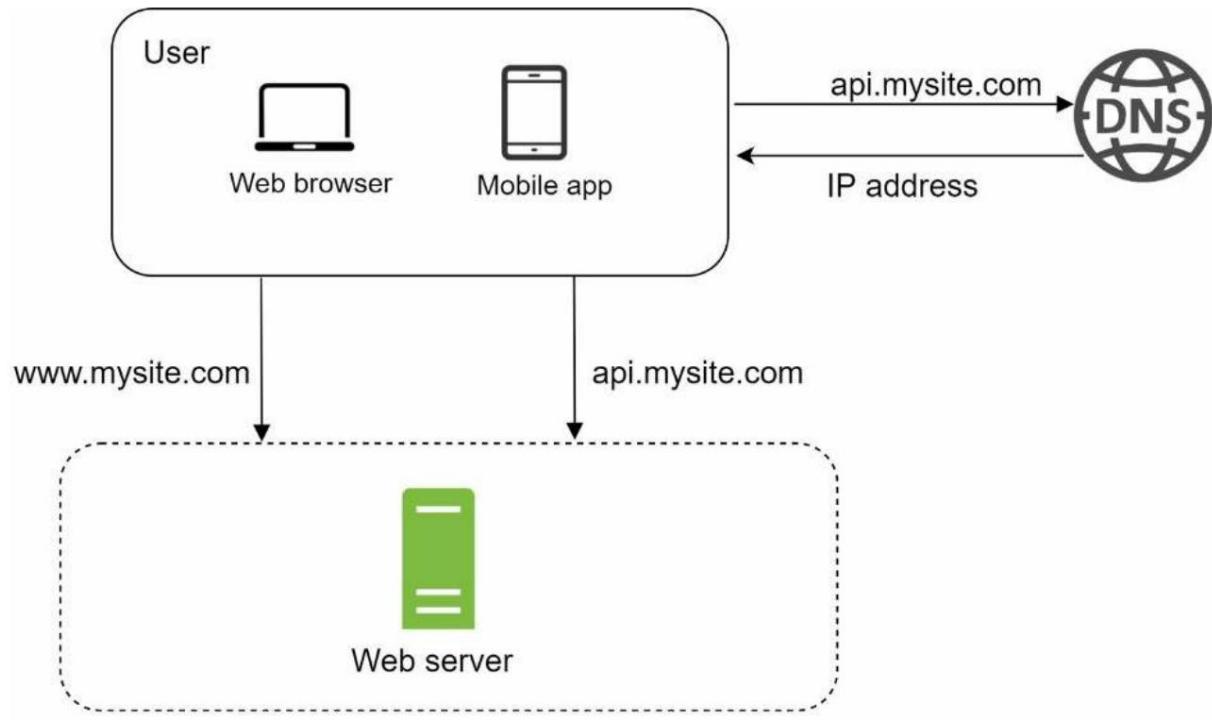


Figure 1-1

Để hiểu thiết lập này, việc nghiên cứu luồng yêu cầu và nguồn lưu trữ sẽ rất hữu ích. Trước tiên, chúng ta hãy xem luồng yêu cầu (Hình 1-2).

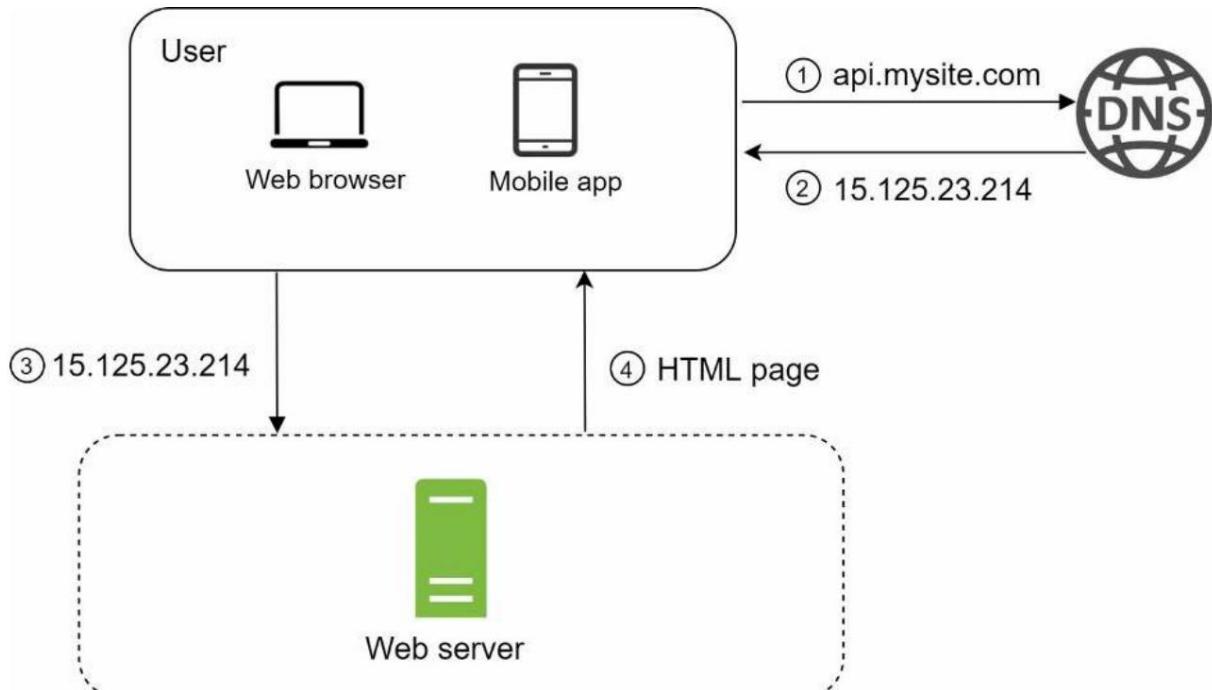


Figure 1-2

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 3 cung cấp và không được lưu trữ bởi máy chủ của chúng tôi.
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.
3. Sau khi có được địa chỉ IP, các yêu cầu Giao thức truyền siêu văn bản (HTTP) [1] sẽ được gửi trực tiếp đến máy chủ web của bạn.
4. Máy chủ web trả về các trang HTML hoặc phản hồi JSON để hiển thị.

Tiếp theo, chúng ta hãy xem xét nguồn lưu lượng truy cập. Lưu lượng truy cập đến 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 kinh doanh, lưu trữ, v.v. và các ngôn ngữ phía khách (HTML và JavaScript) để trình bày. • Ứng dụng di động: Giao thức HTTP là giao

thức truyền thông giữa ứng dụng di động và máy chủ web. JavaScript Object Notation (JSON) là định dạng phản hồi API thư ứng 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 - Lấy đối tượng người dùng cho id = 12
```

```
{
  "id": 12,
  "firstName": "John",
  "lastName": "Smith",
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [
    "212 555-1234",
    "646 555-4567"
  ]
}
```

Cơ sở dữ liệu

Với sự phát triển của cơ sở dữ liệu, một máy chủ là không đủ và chúng ta cần nhiều máy chủ: một cho lưu trữ web/di động, một cho cơ sở dữ liệu (Hình 1-3). Việc tách biệt các máy chủ lưu trữ web/di động (tầng web) và cơ sở dữ liệu (tầng dữ liệu) cho phép chúng được mở rộng độc lập.

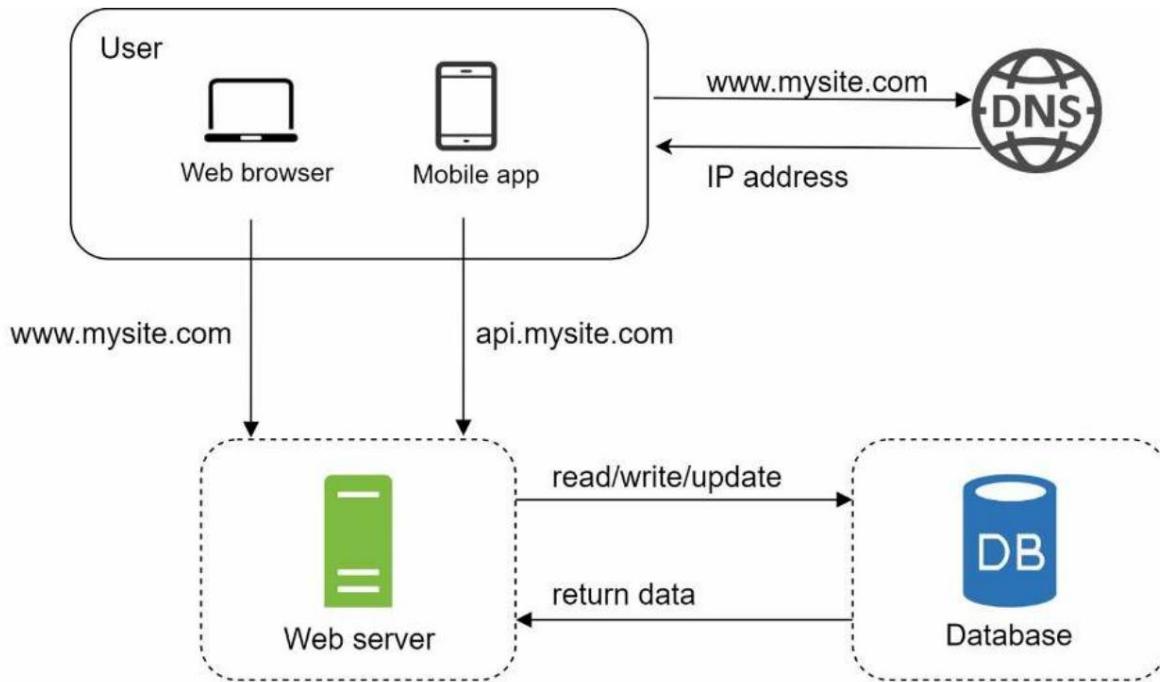


Figure 1-3

Sử dụng cơ sở dữ liệu nào?

Bạn có thể lựa chọn giữa cơ sở dữ liệu quan hệ truyền thống và cơ sở dữ liệu phi quan hệ. Chúng ta hãy xem xét sự khác biệt của chúng.

Cơ sở dữ liệu quan hệ còn được gọi là hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) hoặc cơ sở dữ liệu SQL.

Các hệ thống 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 hoạt động nối bằng SQL trên các bảng cơ sở dữ liệu khác nhau.

Cơ sở dữ liệu phi quan hệ cũng được gọi là cơ sở dữ liệu NoSQL. Các cơ sở dữ liệu phổ biến là CouchDB, Neo4j, Cassandra, HBase, Amazon DynamoDB, v.v. [2]. Các cơ sở dữ liệu này được nhóm thành bốn loại: kho lưu trữ khóa-giá trị, kho lưu trữ đồ thị, kho lưu trữ cột và kho lưu trữ tài liệu. Các hoạt động liên kết thường không được hỗ trợ trong cơ sở dữ liệu phi 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 trong hơn 40 năm và về mặt lịch sử, chúng hoạt động tốt. Tuy nhiên, nếu cơ sở dữ liệu quan hệ không phù hợp với các trường hợp sử dụng cụ thể của bạn, điều quan trọng là phải khám phá ngoài cơ sở dữ liệu quan hệ. Cơ sở dữ liệu không 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 hủy tuần tự hóa dữ liệu (JSON, XML, YAML, v.v.).
- Bạn cần lưu trữ một lượng dữ liệu lớn.

Mở rộng theo chiều dọc so với mở rộng theo chiều ngang

Mở rộng theo chiều dọc, được gọi là "tăng quy mô", có nghĩa là quá trình thêm nhiều năng lực hơn (CPU, RAM, v.v.) vào máy chủ của bạn. Mở rộng 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 trữ và CPU, việc mở rộng theo chiều dọc là một lựa chọn tuyệt vời và tính đơn giản của việc mở rộng theo chiều dọc là lợi thế chính của nó. Thật không may, nó đi kèm với những hạn chế nghiêm trọng.

- Mở rộng theo chiều dọc có giới hạn cứng. Không thể thêm CPU và bộ nhớ không giới hạn vào một máy chủ duy nhất. • Mở rộng 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 sẽ ngừng hoạt động hoàn toàn.

Việc mở rộng theo chiều ngang được ưu ái chuộng hơn cho các ứng dụng quy mô lớn do những hạn chế của việc mở rộng theo chiều dọc.

Trong thiết kế truy cập, 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 kịch bản khác, nếu nhiều người dùng truy cập máy chủ web cùng lúc và đạt đến giới hạn tải của máy chủ web, người dùng thường gặp phải phản hồi chậm hơn hoặc không kết nối được với máy chủ. Bộ cân bằng tải là kỹ thuật tốt nhất để giải quyết những vấn đề này.

Bộ cân bằng tải

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

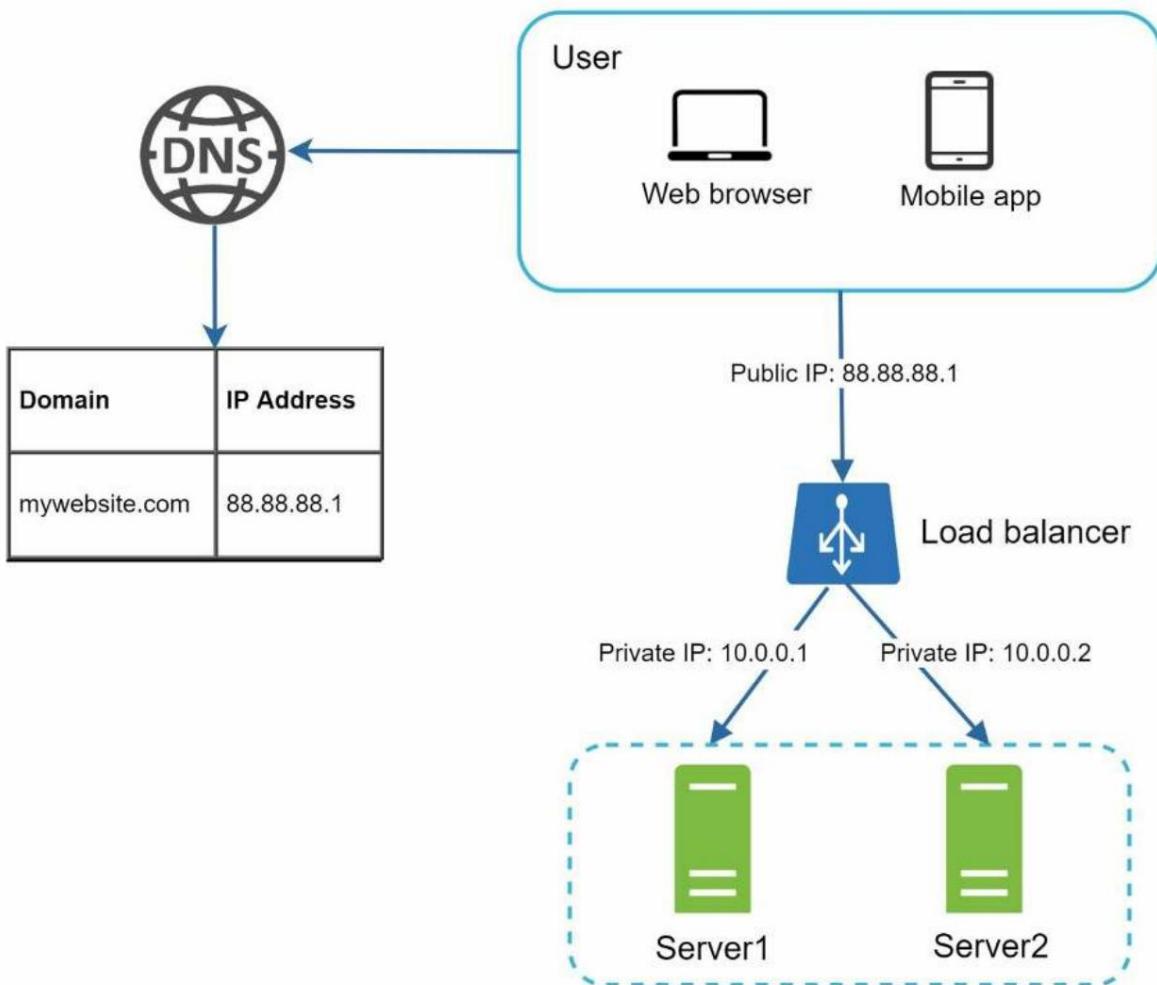


Figure 1-4

Như thể hiện trong Hình 1-4, người dùng kết nối trực tiếp đến IP công khai của bộ cân bằng tải. Với thiết lập này, máy chủ web không thể truy cập trực tiếp từ máy khách nữa. Để bảo mật tốt hơn, IP riêng được sử dụng để giao tiếp 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, không thể truy cập được qua internet. Bộ cân bằng tải giao tiếp với máy chủ web thông qua IP riêng.

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

- Nếu máy chủ 1 ngoại tuyến, tất cả lưu lượng sẽ được định tuyến đến máy chủ 2. Điều này ngăn không cho trang web ngoại tuyến. Chúng tôi cũng sẽ thêm một máy chủ web mới khỏe mạnh 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 truy cập, bộ cân bằng tải có thể xử lý vấn đề này một cách nhẹ nhàng. 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 các máy chủ đó.

Bây giờ tầng web trông 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 cùng xem xét.

Sao chép 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/tớ giữa bản gốc (chủ) và các bản sao (tớ)" [3].

Cơ sở dữ liệu chính thường chỉ hỗ trợ các thao tác ghi. Cơ sở dữ liệu phụ lấy 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 so với ghi cao hơn nhiều; do đó, số lượng cơ sở dữ liệu phụ trong một hệ thống thường lớn hơn số lượng cơ sở dữ liệu chính. Hình 1-5 cho thấy một cơ sở dữ liệu chính có nhiều cơ sở dữ liệu phụ.

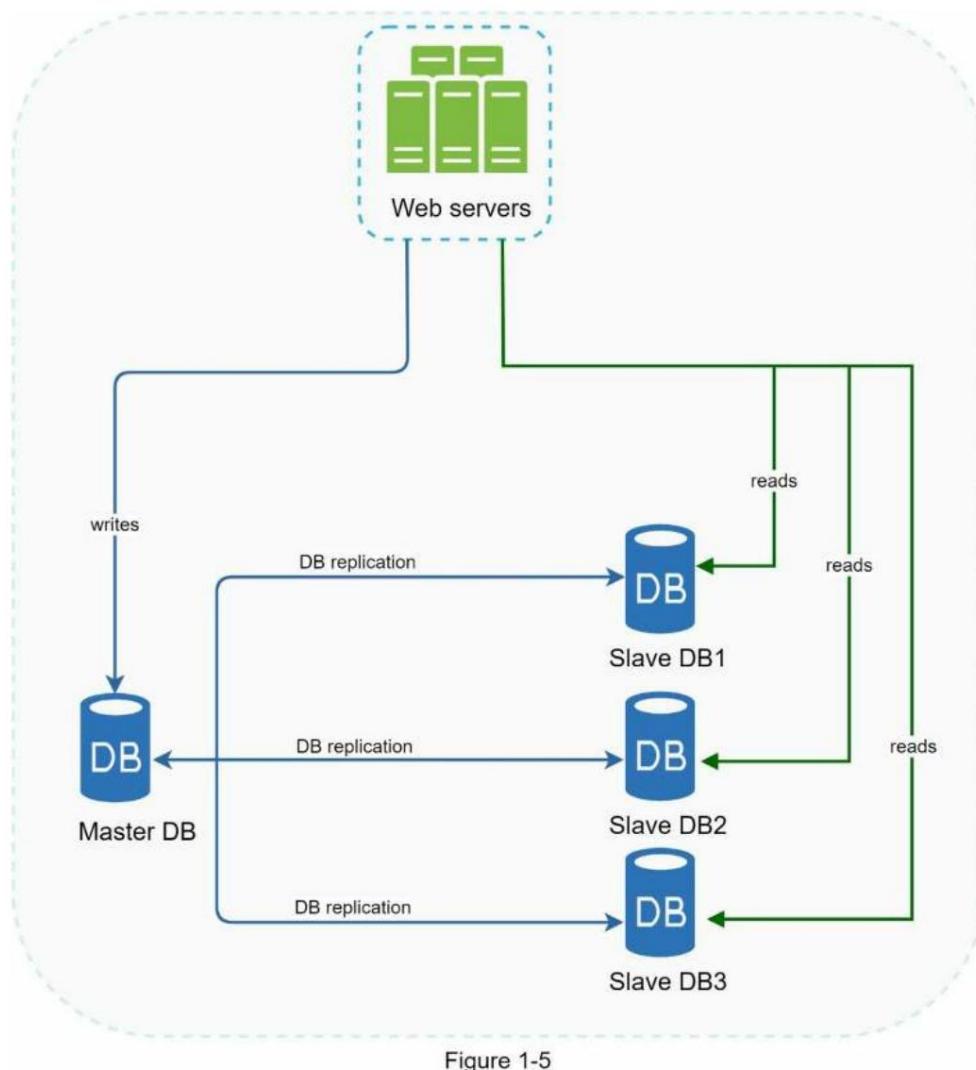


Figure 1-5

Ưu điểm của việc sao chép cơ sở dữ liệu:

- Hiệu suất tốt hơn: Trong mô hình chủ-tớ, tất cả các thao tác ghi và cập nhật đều diễn ra ở các nút chủ; trong khi đó, các thao tác đọc được phân bổ trên các nút tớ. Mô hình này cải thiện hiệu suất vì nó cho phép xử lý nhiều truy vấn 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 bão hoặc động đất, dữ liệu vẫn được bảo toàn. Bạn không cần phải lo lắng về việc mất dữ liệu vì dữ liệu được sao chép ở nhiều vị trí.
- Tính khả dụng cao: Bằng cách sao chép dữ liệu ở nhiều vị trí khác nhau, trang web của bạn vẫn ở

hoạt động ngay cả khi cơ sở dữ liệu ngoại tuyến vì bạn có thể truy cập dữ liệu được lưu trữ trong cơ sở dữ liệu khác máy chủ.

Trong phần trước, chúng ta đã thảo luận về cách bộ cân bằng tài giúp cải thiện tính khả dụng của hệ thống. Chúng ta đặt cùng một câu hỏi 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 Hình 1-5 có thể xử lý trường hợp này:

- Nếu chỉ có một cơ sở dữ liệu phụ khả dụng và nó ngoại tuyến, các hoạt động đọc sẽ được chuyển hướng tạm thời đến cơ sở dữ liệu chính. Ngay khi phát hiện ra sự cố, một cơ sở dữ liệu phụ mới sẽ thay thế cơ sở dữ liệu cũ. Trong trường hợp có nhiều cơ sở dữ liệu phụ khả dụng, các hoạt động đọc sẽ được chuyển hướng đến các cơ sở dữ liệu phụ 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, một cơ sở dữ liệu phụ sẽ được thăng cấp thành máy chủ chính mới. Tất cả các hoạt động cơ sở dữ liệu sẽ được thực hiện tạm thời trên cơ sở dữ liệu chính mới. Một cơ sở dữ liệu phụ mới sẽ thay thế cơ sở dữ liệu cũ để sao chép dữ liệu ngay lập tức.

Trong các hệ thống sản xuất, việc thúc đẩy một máy chủ mới phức tạp hơn vì dữ liệu trong cơ sở dữ liệu phụ có thể không được cập nhật. Dữ liệu bị thiếu cần được cập nhật bằng cách chạy các tập lệnh khôi phục dữ liệu. Mặc dù một số phương pháp sao chép khác như nhiều máy chủ và sao chép vòng tròn có thể hữu ích, nhưng các thiết lập đó phức tạp hơn; và các cuộc thảo luận của họ nằm ngoài phạm vi của cuốn sách này. Độc giả quan tâm nên tham khảo các tài liệu tham khảo được liệt kê [4] [5].

Hình 1-6 hiển thị thiết kế hệ thống sau khi thêm bộ cân bằng tải và sao chép cơ sở dữ liệu.

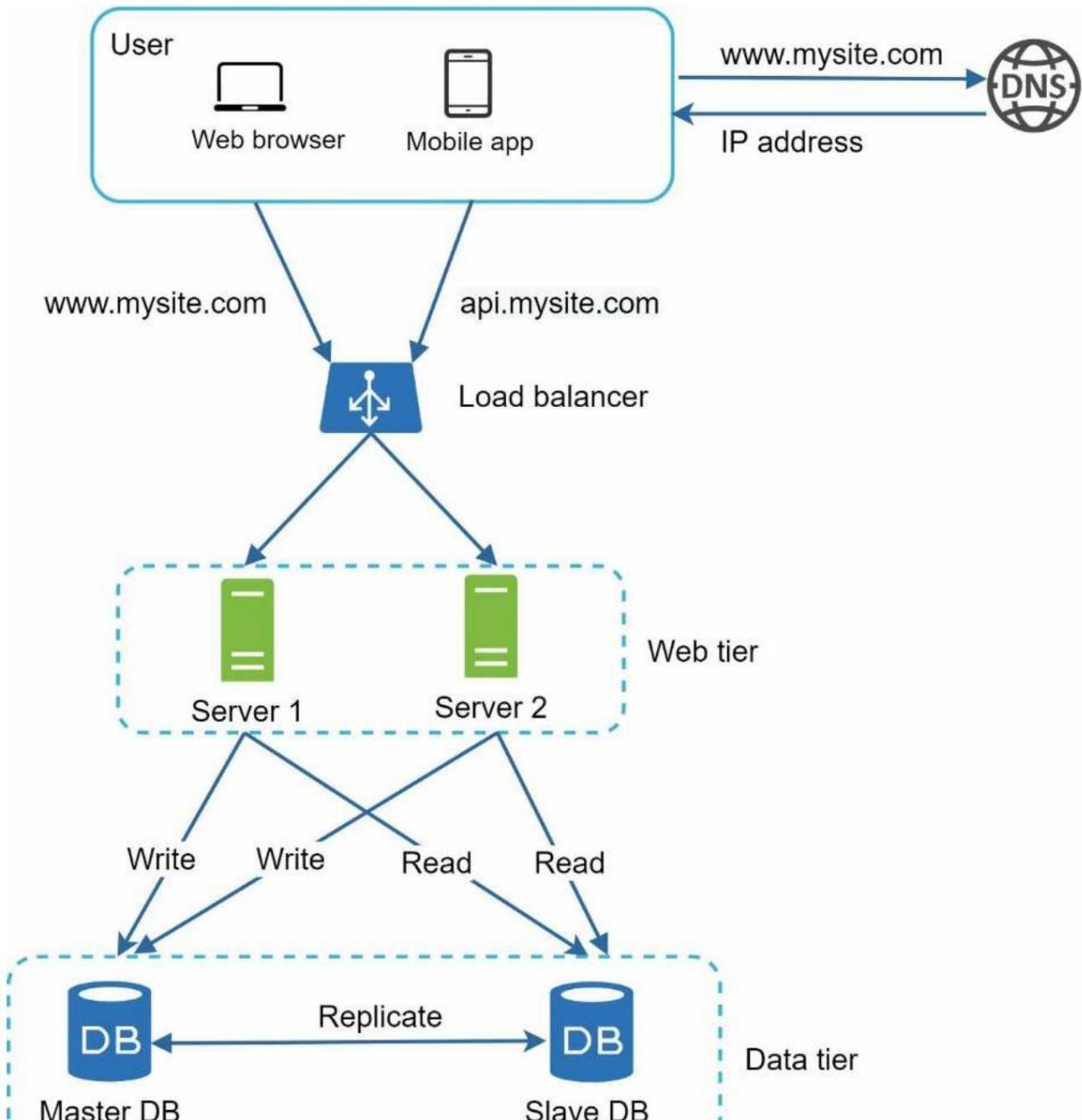


Figure 1-6

Chúng ta hãy xem xét thiết kế:

Người dùng nhận được địa chỉ IP của bộ cân bằng tải từ DNS. • Người dùng kết nối bộ cân bằng tải với địa chỉ IP này.

- Yêu cầu HTTP được định tuyến đến 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 đến cơ sở dữ liệu chính. Bao gồm các hoạt động ghi, cập nhật và xóa.

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

Bộ nhớ đệm

Bộ nhớ đệm là vùng lưu trữ tạm thời lưu trữ kết quả của các phản hồi tốn kém hoặc dữ liệu thường xuyên truy cập trong bộ nhớ để các yêu cầu tiếp theo được phục vụ nhanh hơn. Như minh họa trong Hình 1-6, mỗi lần tải một trang web mới, một hoặc nhiều lệnh gọi cơ sở dữ liệu được thực hiện để lấy dữ liệu. Hiệu suất ứng dụng bị ảnh hưởng rất nhiều khi gọi cơ sở dữ liệu nhiều lần.

Bộ nhớ đệm có thể giảm thiểu vấn đề này.

Tầng bộ nhớ đệm

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

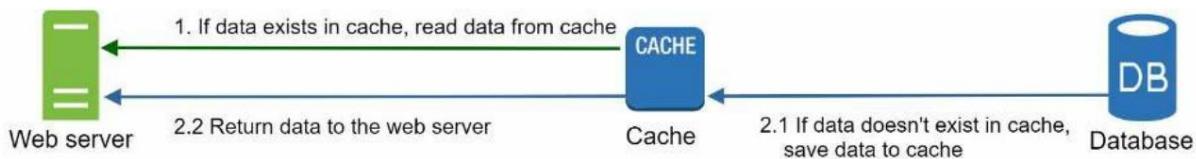


Figure 1-7

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

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

SECONDS = 1

```
cache.set('myKey', 'hi there', 3600 * SECONDS)
cache.get('myKey')
```

Những cân nhắc khi sử dụng bộ nhớ đệm

Sau đây là một số lưu ý khi sử dụng hệ thống bộ nhớ đệm:

- Quyết định khi nào sử dụng bộ nhớ đệm. Cân nhắc sử dụng bộ nhớ đệm khi dữ liệu được đọc thường xuyên như ít khi được sửa đổi. Vì dữ liệu được lưu trong bộ nhớ đệm được lưu trữ trong bộ nhớ dễ bay hơi nên máy chủ bộ nhớ đệm không lý tưởng để lưu trữ dữ liệu. Ví dụ, nếu máy chủ bộ nhớ đệm khởi động lại, tất cả dữ liệu trong bộ nhớ sẽ bị mất. Do đó, dữ liệu quan trọng nên được lưu trong kho dữ liệu liên tục.
- Chính sách hết hạn. Thực hiện chính sách hết hạn là một biện pháp tốt. Khi dữ liệu được lưu trong bộ nhớ đệm hết hạn, dữ liệu đó sẽ bị xóa khỏi bộ nhớ đệm. Khi không có chính sách hết hạn, dữ liệu được lưu trong bộ nhớ sẽ được lưu trữ vĩnh viễn trong bộ nhớ. Nên không nên để 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 để ngày hết hạn quá dài vì dữ liệu có thể bị cũ.

- Tính nhất quán: Điều này liên quan đến việc giữ cho kho dữ liệu và bộ nhớ đệm đồng bộ. Sự không nhất quán có thể xảy ra vì các hoạt động sửa đổi dữ liệu trên kho 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

kho dữ liệu và bộ nhớ đệm là một thách thức. Để biết thêm chi tiết, hãy tham khảo bài báo có tiêu đề "Scaling Memcache at Facebook" do Facebook xuất bản [7]. • Giảm thiểu lỗi: Một máy chủ bộ nhớ đệm duy nhất đại diện cho một điểm lỗi duy nhất tiềm ẩn (SPOF), được định nghĩa trong Wikipedia như sau: "Điểm lỗi duy nhất (SPOF) là một phần của hệ thống, nếu nó bị lỗi, sẽ khiến toàn bộ hệ thống ngừng hoạt động" [8]. Do đó, nhiều máy chủ bộ nhớ đệm trên các trung tâm dữ liệu khác nhau được khuyến nghị để tránh SPOF. Một cách tiếp cận được khuyến nghị khác là cung cấp quá mức bộ nhớ cần thiết theo một số phần trăm nhất định.

Tính năng này cung cấp vùng đệm khi mức sử dụng bộ nhớ tăng lên.

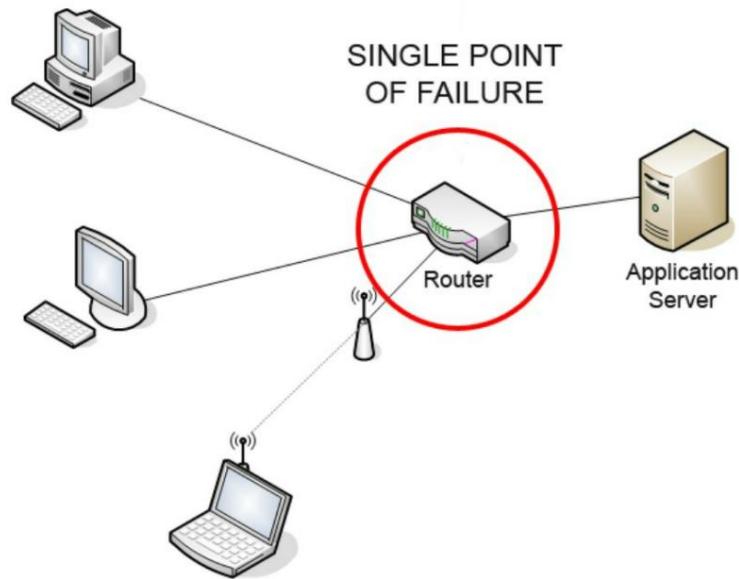


Figure 1-8 (source: <https://bit.ly/3eGsnyH>)

- Chính sách xóa: Khi bộ nhớ đệm đầy, bất kỳ yêu cầu nào để thêm mục vào bộ nhớ đệm có thể khiến các mục hiện có bị xóa. Đây được gọi là xóa bộ nhớ đệm. Ít được sử dụng gần đây nhất (LRU) là chính sách xóa bộ nhớ đệm phổ biến nhất. Các chính sách xóa khác, chẳng hạn như Ít được sử dụng thường xuyên nhất (LFU) hoặc Vào trự ớc ra trự ớc (FIFO), có thể được áp dụng để đáp ứng các mục đích sử dụng khác nhau

tương hợp.

Mạng phân phối nội dung (CDN)

CDN là mạng lưới các máy chủ phân tán về mặt địa lý được sử dụng để phân phối nội dung tĩnh.

Máy chủ CDN lưu trữ nội dung tĩnh như hình ảnh, video, tệp CSS, 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 bộ nhớ đệm 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.

Sau đây là cách CDN hoạt động ở cấp độ cao: khi người dùng truy cập trang web, máy chủ CDN gần người dùng nhất sẽ phân phối nội dung tĩnh. Theo trực giác, người dùng càng xa máy chủ CDN thì trang web tải càng chậm. Ví dụ, nếu máy chủ CDN ở San Francisco, người dùng ở Los Angeles sẽ nhận được nội dung nhanh hơn người dùng ở Châu Âu. Hình 1-9 là một ví dụ tuyệt vời cho thấy cách CDN cải thiện thời gian tải.

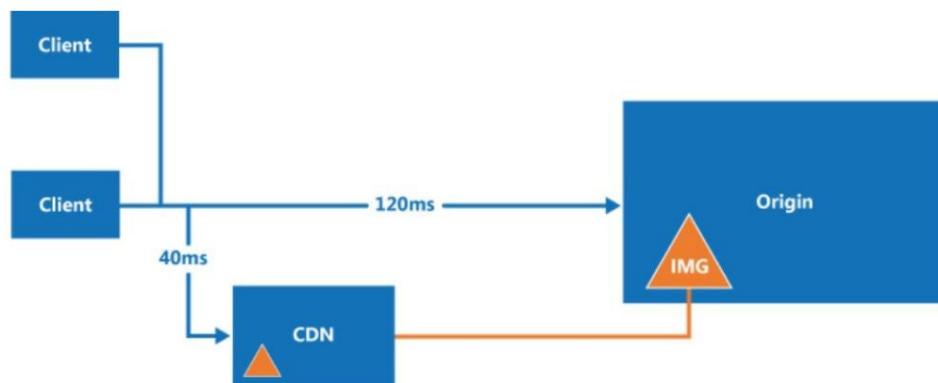


Figure 1-9 (source: <https://bit.ly/2yv7DJK>)

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

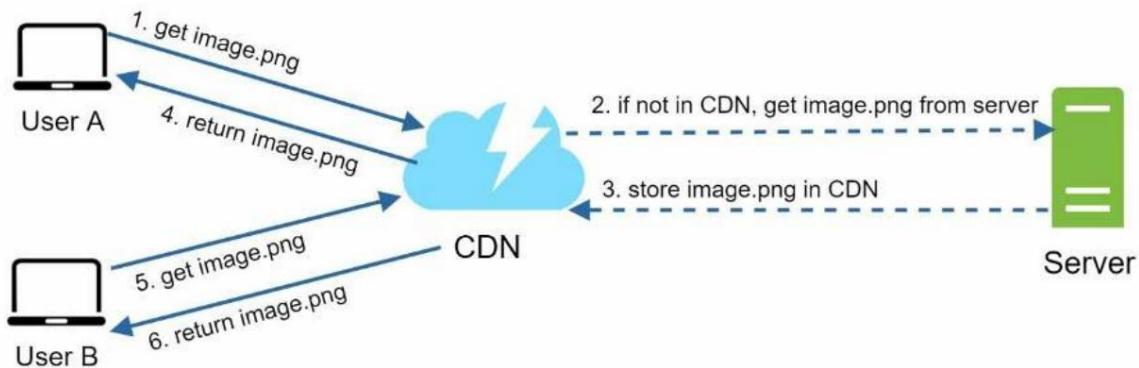


Figure 1-10

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 chứng minh URL hình ảnh trông như thế nào trên Amazon và Akamai CDN:

- `https://mysite.cloudfront.net/logo.jpg`
- `https://mysite.akamai.com/image-manager/img/logo.jpg`

2. Nếu máy chủ CDN không có `image.png` trong bộ nhớ đệm, máy chủ CDN sẽ yêu cầu tệp từ nguồn gốc, có thể là máy chủ web hoặc bộ lưu trữ trực tuyến như Amazon S3.

3. Nguồn gốc trả về `image.png` cho máy chủ CDN, bao gồm tiêu đề HTTP tùy chọn Thời gian tồn tại (TTL) mô tả thời gian lưu trữ bộ nhớ đệm của hình ảnh.

4. CDN lưu trữ hình ảnh và trả về cho Người dùng A. Hình ảnh vẫn được lưu trữ trong CDN cho đến khi TTL hết hạn.
5. Người dùng B gửi yêu cầu để lấy cùng một hình ảnh.
6. Hình ảnh được trả về từ bộ nhớ đệm miễn là TTL chưa hết hạn.

Những cân nhắc khi sử dụng CDN

- Chi phí: CDN do các nhà cung cấp bên thứ ba điều hành và bạn phải trả phí cho việc truyền dữ liệu vào và ra khỏi CDN. Việc lưu trữ đệm các tài sản ít được sử dụng không mang lại lợi ích đáng kể nào nên bạn nên cân nhắc di chuyển chúng ra khỏi CDN.
- Đặt thời gian hết hạn bộ đệm phù hợp: Đôi với nội dung nhạy cảm với thời gian, việc đặt thời gian hết hạn bộ đệm là rất quan trọng. Thời gian hết hạn bộ đệm không được quá dài cũng không được quá ngắn. Nếu quá dài, nội dung có thể không còn mới nữa. Nếu quá ngắn, nội dung có thể phải tải lại nhiều lần từ máy chủ gốc đến CDN.
- Dự phòng CDN: Bạn nên cân nhắc cách trang web/ứng dụng của mình đối phó với lỗi CDN. Nếu có sự cố CDN tạm thời, máy khách sẽ có thể phát hiện ra sự cố và yêu cầu tài nguyên từ nguồn gốc.

Làm mất hiệu lực các tệp: Bạn có thể xóa

tệp khỏi CDN trước khi tệp đó hết hạn bằng cách thực hiện một trong các thao tác sau:

- Làm mất hiệu lực đối tượng CDN bằng API do nhà cung cấp CDN cung cấp.

Sử dụng phiên bản đối tượng để phục vụ phiên bản khác của đối tượng. Để phiên bản 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.

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

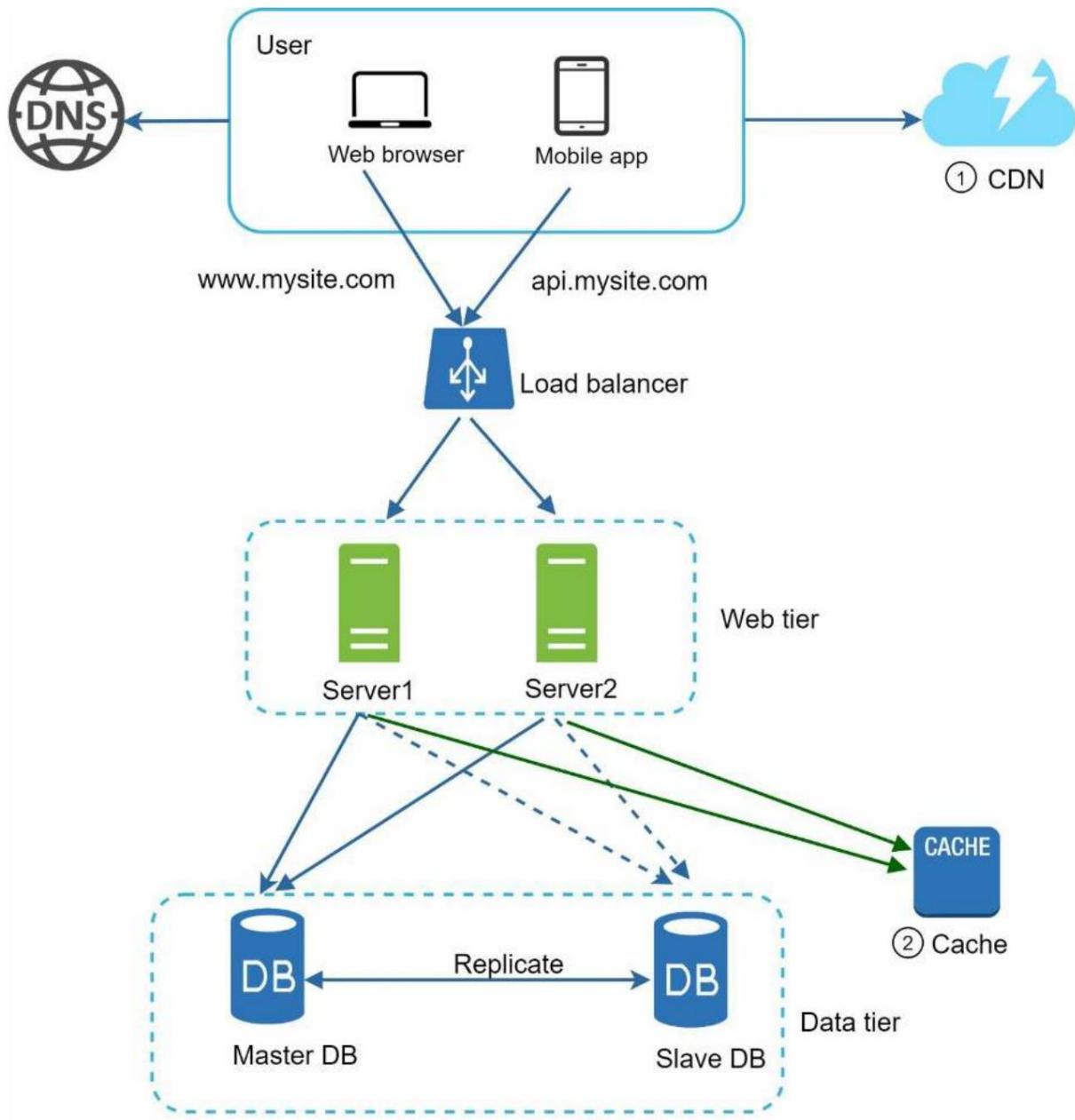


Figure 1-11

1. Các tài sản tĩnh (JS, CSS, hình ảnh, v.v.) không còn được phục vụ bởi máy chủ web nữa. Chúng được lấy từ CDN để có hiệu suất tốt hơn.
2. Giảm tải cơ sở dữ liệu bằng cách lưu trữ dữ liệu đệm.

Tầng web không trạng thái

Bây giờ là lúc cân nhắc việc mở rộng tầng web theo chiều ngang. Để làm được điều này, chúng ta cần di chuyển trạng thái (ví dụ dữ liệu phiên người dùng) ra khỏi tầng web. Một cách làm 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 có trạng thái

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

Hình 1-12 cho thấy một ví dụ về kiến trúc có trạng thái.

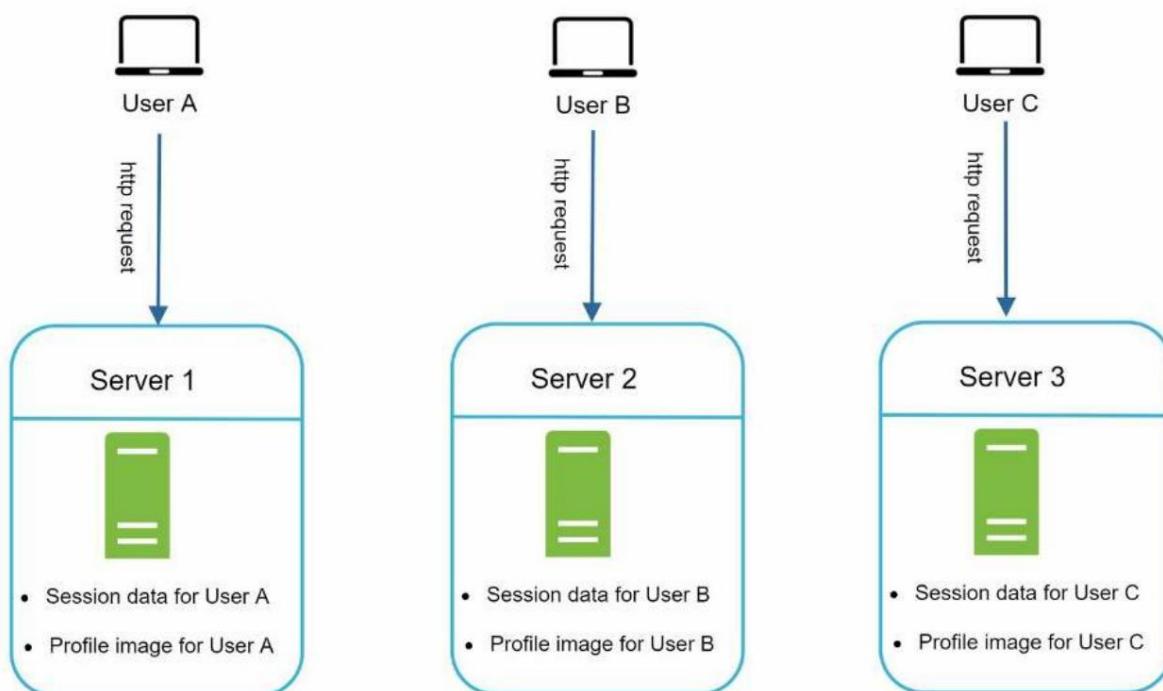


Figure 1-12

Trong Hình 1-12, dữ liệu phiên và 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, xác thực sẽ không thành công vì Máy chủ 2 không chứa dữ liệu phiên của Người dùng A.

Tương tự như vậy, tất cả các yêu cầu HTTP từ Người dùng B phải được chuyển đến Máy chủ 2; tất cả các yêu cầu từ Người dùng C phải được gửi đến Máy chủ 3.

Vấn đề là mọi yêu cầu từ cùng một máy khách phải được định tuyến đến cùng một máy chủ. Điều này có thể thực hiện được với 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 hơn nhiều với cách tiếp cận này. Việc xử lý các lỗi máy chủ cũng rất khó khăn.

Kiến trúc không quốc tịch

Hình 1-13 cho thấy kiến trúc không trạng thái.

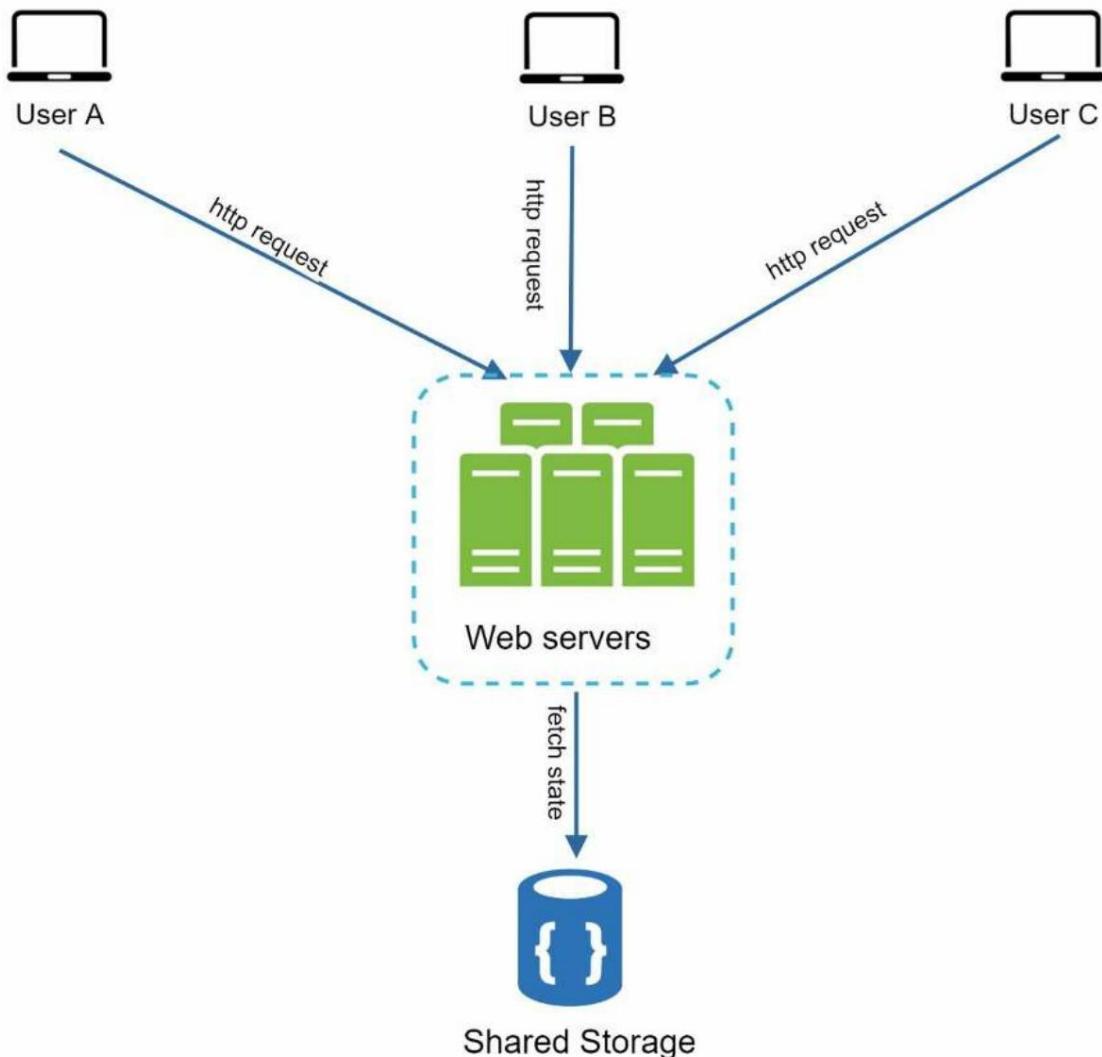


Figure 1-13

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, nơi lấy dữ liệu trạng thái từ kho dữ liệu chung. Dữ liệu trạng thái được lưu trữ trong kho dữ liệu chung và không được đưa vào máy chủ web. Hệ thống không trạng thái đơn giản hơn, mạnh mẽ hơn và có khả năng mở rộng.

Hình 1-14 hiển thị thiết kế được cập nhật với tầng web không trạng thái.

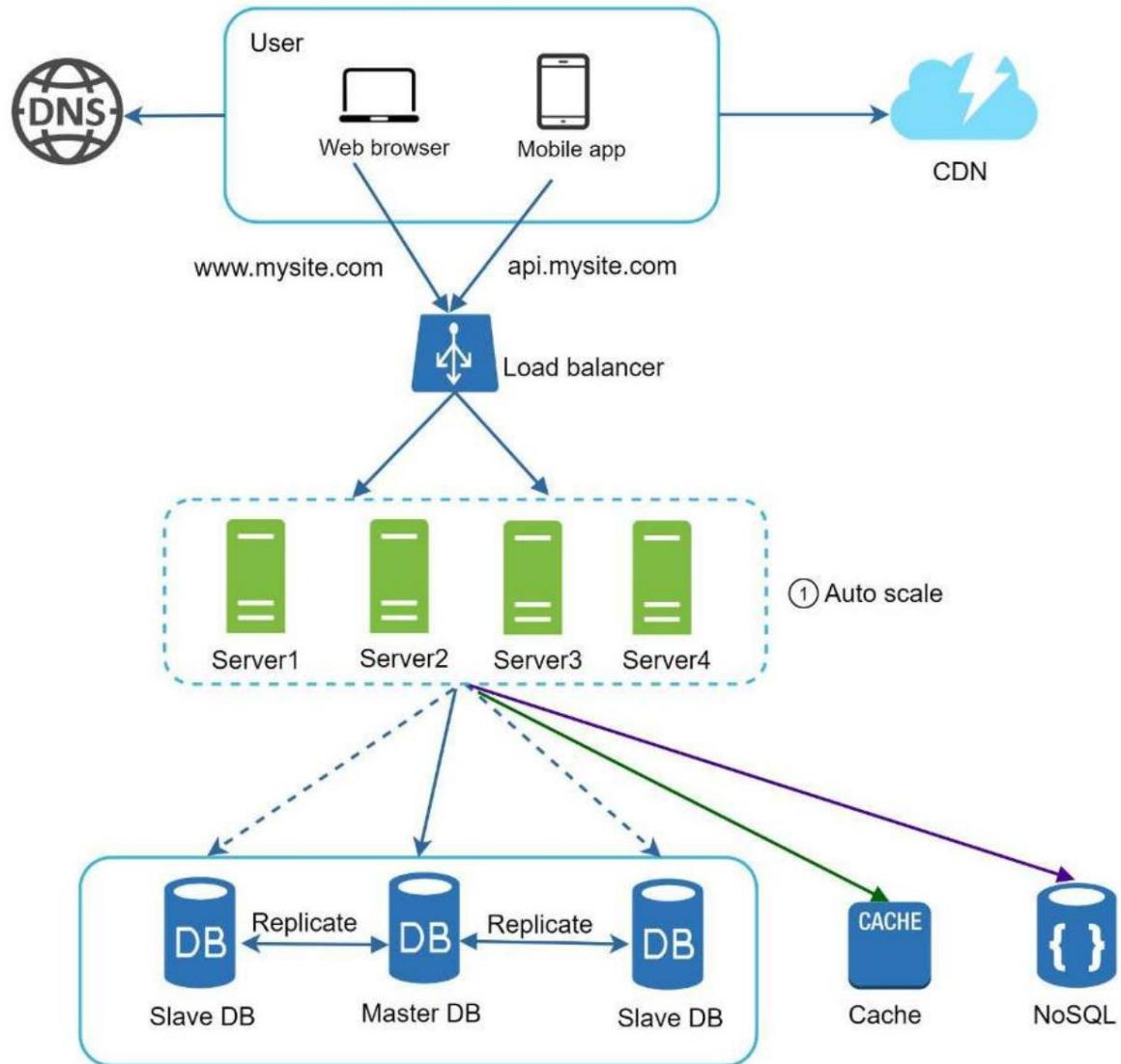


Figure 1-14

Trong Hình 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 được chia sẻ có thể là cơ sở dữ liệu quan hệ, Memcached/Redis, NoSQL, v.v. Kho dữ liệu NoSQL được chọn vì dễ mở rộng quy mô. Tự động mở rộng quy mô có nghĩa là tự động thêm hoặc xóa máy chủ web dựa trên tải lưu lượng truy cập. Sau khi dữ liệu trạng thái được xóa khỏi máy chủ web, tự động mở rộng quy mô của tầng web có thể dễ dàng đạt được 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 trên toàn thế giới. Để cải thiện tính khả dụng và cung cấp trải nghiệm người dùng tốt hơn trên nhiều khu vực địa lý hơn, việc hỗ trợ nhiều trung tâm dữ liệu là rất quan trọng.

Trung tâm dữ liệu

Hình 1-15 cho thấy một ví dụ thiết lập với hai trung tâm dữ liệu. Trong hoạt động bình thường, người dùng được định tuyến geoDNS, còn được gọi là định tuyến địa lý, đến trung tâm dữ liệu gần nhất, với lưu lượng chia tách là $x\%$ ở phía Đông Hoa Kỳ và $(100 - x)\%$ ở phía Tây Hoa Kỳ. geoDNS là một dịch vụ DNS cho phép phân giải tên miền thành địa chỉ IP dựa trên vị trí của người dùng.

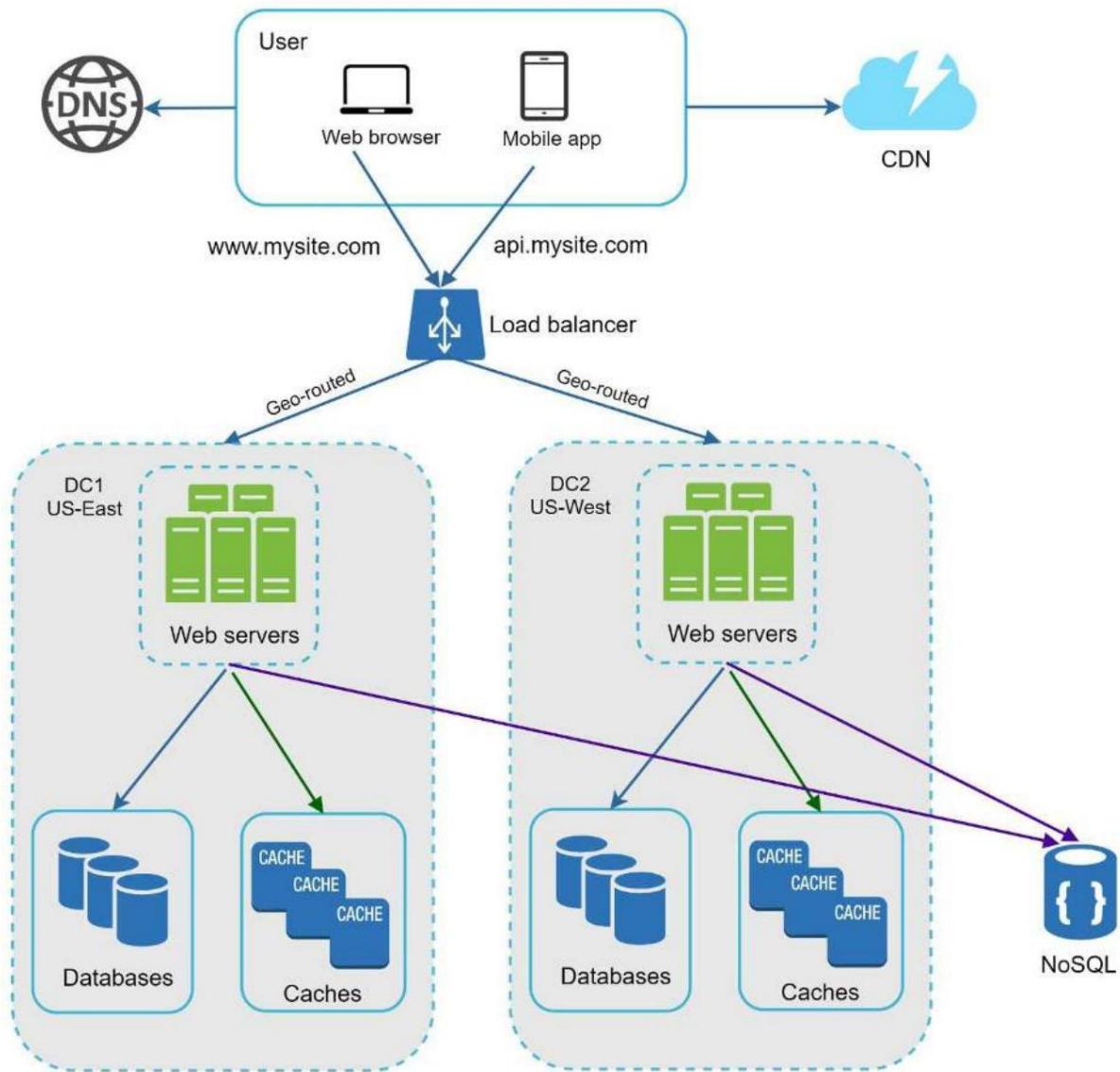


Figure 1-15

Trong trường hợp trung tâm dữ liệu bị ngừng hoạt động đáng kể, chúng tôi sẽ chuyển hướng toàn bộ lưu lượng truy cập đến một trung tâm dữ liệu đang hoạt động bình thường. Trong Hình 1-16, trung tâm dữ liệu 2 (US-West) đang ngoại tuyến và 100% lưu lượng được định tuyến đến trung tâm dữ liệu 1 (US-East).

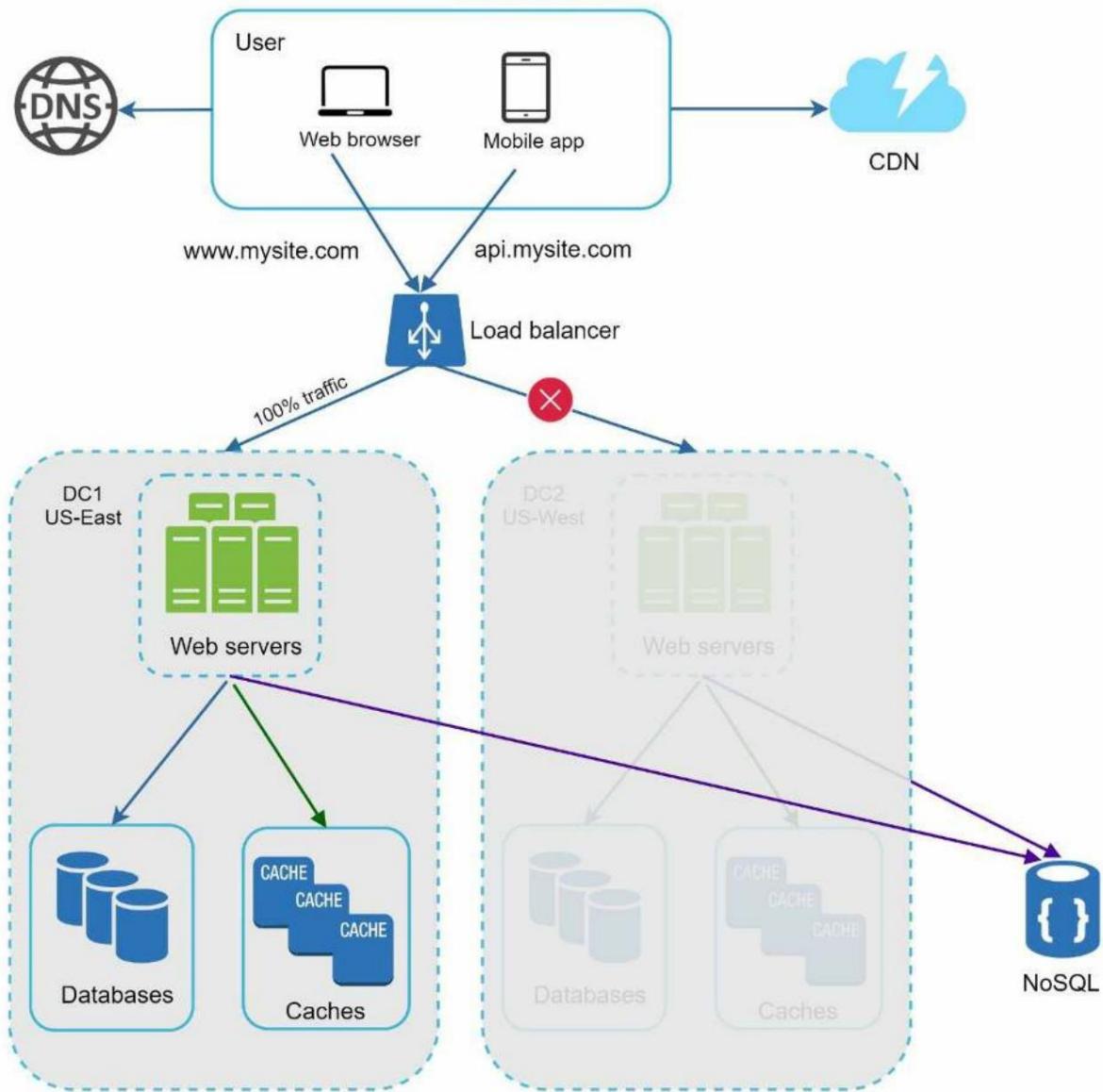


Figure 1-16

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

- Chuyển hướng lưu lượng truy cập: Cần có các công cụ hiệu quả để chuyển hướng lưu lượng truy cập đến đúng trung tâm dữ liệu.
- GeodNS có thể được sử dụng để chuyển 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 vùng khác nhau có thể sử dụng các cơ sở dữ liệu hoặc bộ nhớ đệm cục bộ khác nhau. Trong trường hợp chuyển đổi dự phòng, lưu lượng có thể được định tuyến đến một trung tâm dữ liệu nơi dữ liệu không khả dụng.
- Một 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 nhiều trung tâm dữ liệu không đồng bộ [11].
- Kiểm tra và triển khai: Với thiết lập nhiều trung tâm dữ liệu, điều quan trọng là phải kiểm tra trang web/ứng dụng của bạn ở các vị trí khác nhau. Các công cụ triển khai tự động rất quan trọng để duy trì các dịch vụ nhất quán trên tất cả các trung tâm dữ liệu [11].

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

Hàng đợi tin nhắn

Hàng đợi tin nhắn là một thành phần bền vững, được lưu trữ trong bộ nhớ, hỗ trợ giao tiếp không đồng bộ. Nó đóng vai trò 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 các tin nhắn. Mô hình được thể hiện ở Hình 1-17.



Figure 1-17

Việc tách rời làm cho hàng đợi tin nhắn trở thành kiến trúc được ưa thích để xây dựng ứng dụng có khả năng mở rộng và đáng tin cậy. Với hàng đợi tin nhắn, nhà sản xuất có thể đăng tin nhắn vào hàng đợi khi người tiêu dùng không có mặt để xử lý tin nhắn đó. 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, làm sắc nét, làm mờ, v.v. Các tác vụ tùy chỉnh đó mất thời gian để hoàn thành. Trong Hình 1-18, máy chủ web xuất bản các tác vụ xử lý ảnh vào hàng đợi tin nhắn. Các công nhân xử lý ảnh nhận các tác vụ từ hàng đợi tin nhắn và thực hiện các tác vụ tùy chỉnh ảnh không đồng bộ. Nhà sản xuất và người tiêu dùng có thể được mở rộng độc lập. Khi kích thước của hàng đợi trở nên lớn, nhiều công nhân hơn được thêm vào để giảm thời gian xử lý.

Tuy nhiên, nếu hàng đợi thư ờng xuyên trống thì có thể giảm số lượng công nhân.

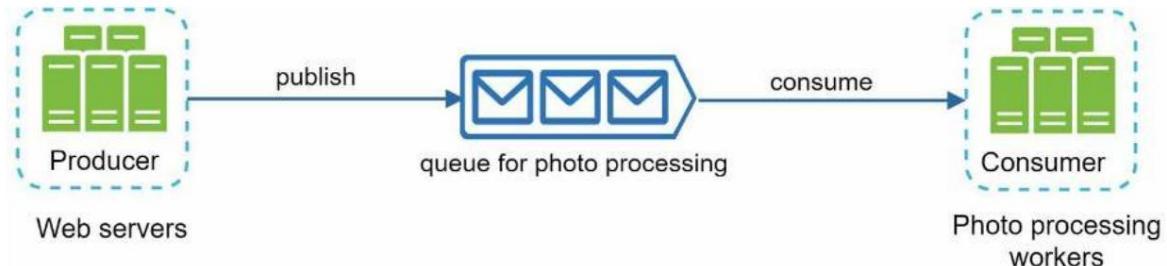


Figure 1-18

Ghi nhật ký, số liệu, tự động hóa Khi làm

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

Ghi nhật ký: Theo dõi nhật ký lỗi rất quan trọng vì nó giúp xác định 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: Thu thập các loại số liệu khác nhau giúp chúng tôi có được thông tin chi tiết về doanh nghiệp và hiểu được tình trạng sức khỏe của hệ thống. Một số số liệu sau đây hữu ích:

- Số liệu cấp máy chủ: CPU, Bộ nhớ, I/O đĩa, v.v. • Số liệu tổng hợp: ví dụ: hiệu suất của toàn bộ tầng cơ sở dữ liệu, tầng bộ đệm, v.v. • Số liệu kinh doanh chính: người dùng hoạt động hàng ngày, tỷ lệ duy trì, 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 hoạt động tốt, trong đó mỗi lần kiểm tra mã được xác minh thông qua tự động hóa, cho phép các nhóm phát hiện sớm các vấn đề. Bên cạnh đó, tự động hóa quy 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

Hình 1-19 cho thấy thiết kế đã 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 hàng đợi tin nhắn, giúp hệ thống kết nối lồng léo hơn và có khả năng chống lỗi tốt hơn.
2. Bao gồm các công cụ ghi nhật ký, giám sát, đo lưu lượng và tự động hóa.

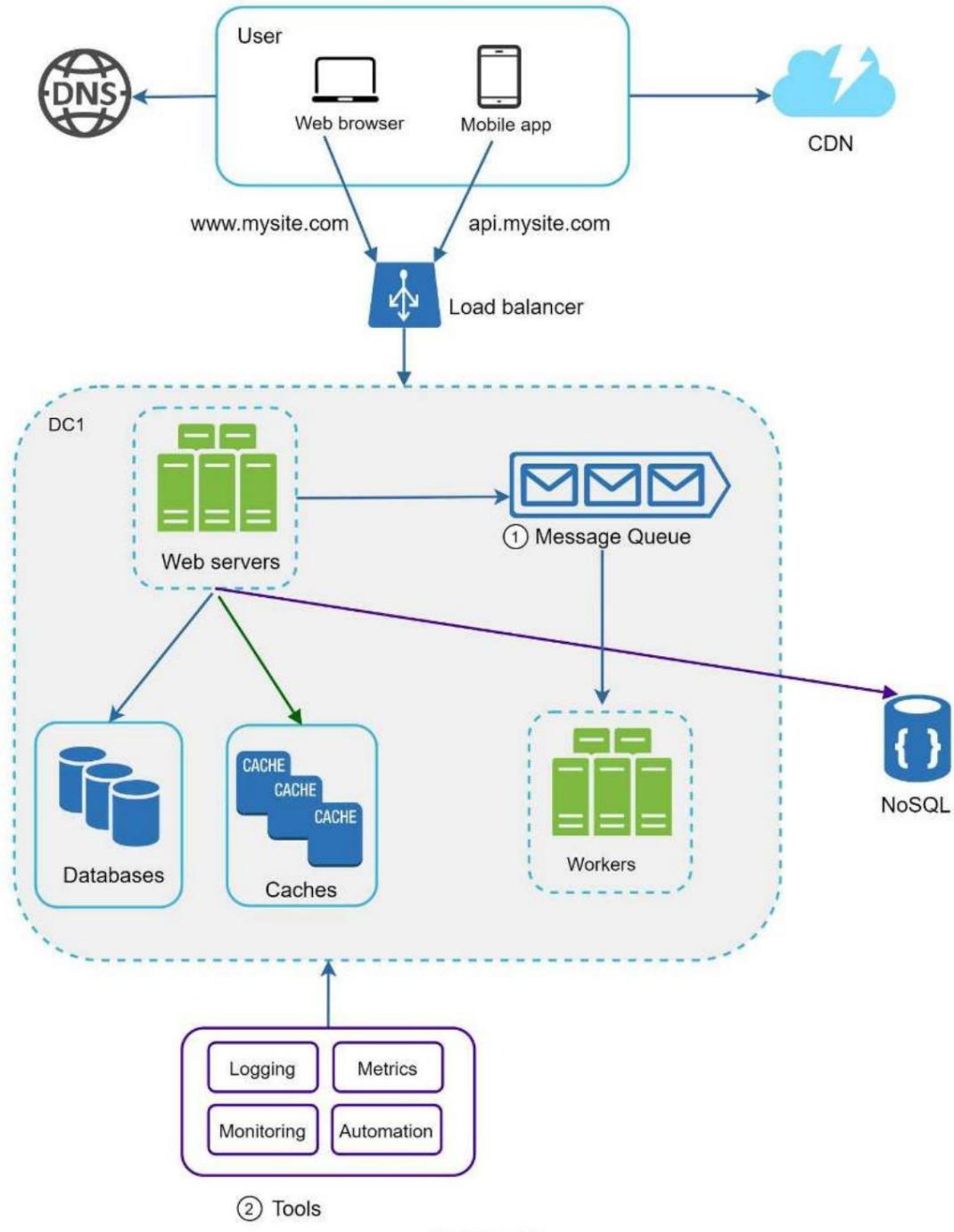


Figure 1-19

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

Mở rộng quy mô cơ sở dữ liệu

Có hai cách tiếp cận chung để mở rộng quy mô cơ sở dữ liệu: mở rộng theo chiều dọc và mở rộng theo chiều ngang.

Mở rộng theo chiều dọc

Mở rộng theo chiều dọc, còn được gọi là tăng quy mô, là việc mở rộng bằng cách thêm nhiều năng lực hơn (CPU, RAM, ĐĨA, v.v.) vào một máy hiện có. Có một số máy chủ cơ sở dữ liệu mạnh mẽ. Theo Amazon Relational Database Service (RDS) [12], bạn có thể có một máy chủ cơ sở dữ liệu với 24 TB RAM. Loại máy chủ cơ sở dữ liệu mạnh mẽ này có thể lưu trữ và xử lý nhiều dữ liệu. Ví dụ: stackoverflow.com vào năm 2013 có hơn 10 triệu lượt truy cập duy nhất hàng tháng, nhưng chỉ có 1 cơ sở dữ liệu chính [13]. Tuy nhiên, mở rộng theo chiều dọc đi kèm với một số nhược điểm nghiêm trọng:

- Bạn có thể thêm nhiều CPU, RAM, v.v. vào máy chủ cơ sở dữ liệu của mình, nhưng có giới hạn về phần cứng. Nếu bạn có lượng người dùng lớn, thì một máy chủ duy nhất là không đủ.
- Rủi ro cao hơn về điểm lỗi đơn.
- Tổng chi phí mở rộng theo chiều dọc là cao. Máy chủ mạnh mẽ đắt hơn nhiều.

Mở rộng theo chiều ngang

Mở rộng theo chiều ngang, còn được gọi là phân mảnh, là phương pháp thêm nhiều máy chủ hơn. Hình 1-20 so sánh mở rộng theo chiều dọc với mở rộng theo chiều ngang.

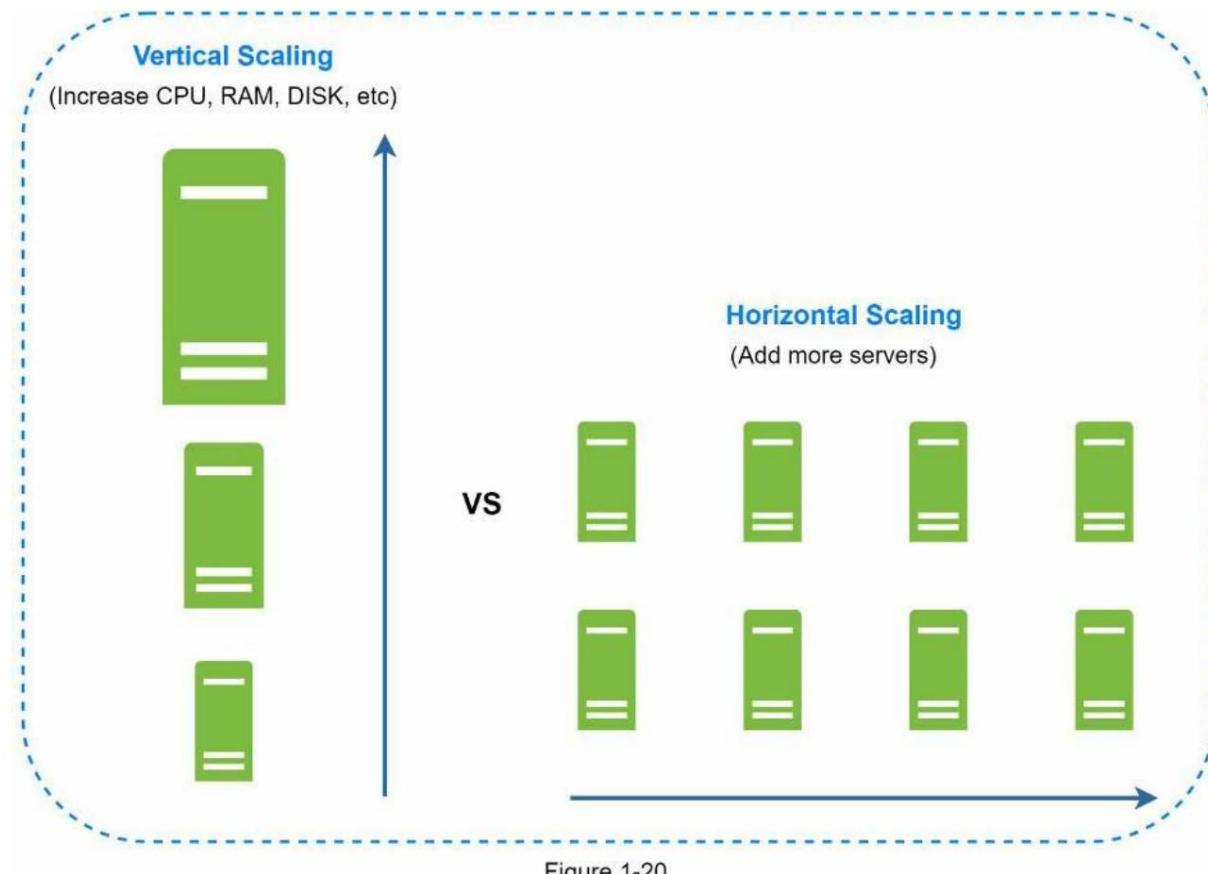


Figure 1-20

Phân mảnh phân chia cơ sở dữ liệu lớn thành các phần nhỏ hơn, dễ quản lý hơn gọi là phân mảnh.

Mỗi phân đoạn chia sẻ cùng một lược đồ, mặc dù dữ liệu thực tế trên mỗi phân đoạn là duy nhất đối với phân đoạn đó.

Hình 1-21 cho thấy một ví dụ về cơ sở dữ liệu phân mảnh. Dữ liệu người dùng được phân bổ cho máy chủ cơ sở dữ liệu dựa trên ID người dùng. Bất cứ khi nào bạn truy cập dữ liệu, một hàm băm được sử dụng để tìm phân mảnh tương ứng. Trong ví dụ của chúng tôi, `user_id % 4` được sử dụng làm hàm băm. Nếu kết quả

bảng 0, phân đoạn 0 được sử dụng để lưu trữ và lấy dữ liệu. Nếu kết quả bảng 1, phân đoạn 1 được sử dụng. Logic tương tự cũng áp dụng cho các mảnh vỡ khác.

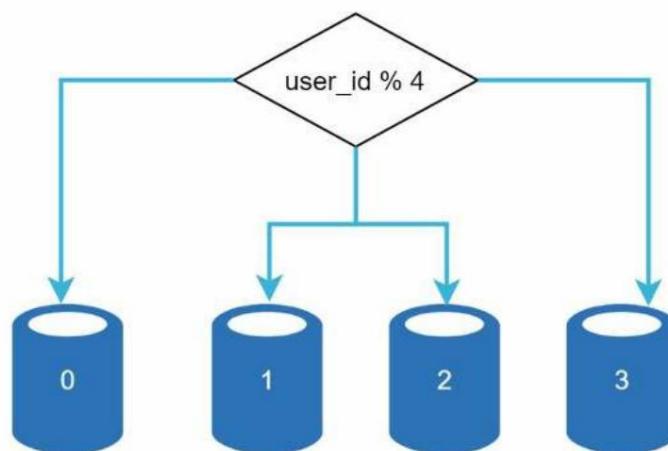


Figure 1-21

Hình 1-22 hiển thị bảng ngẫu nhiên dùng trong cơ sở dữ liệu phân mảnh.

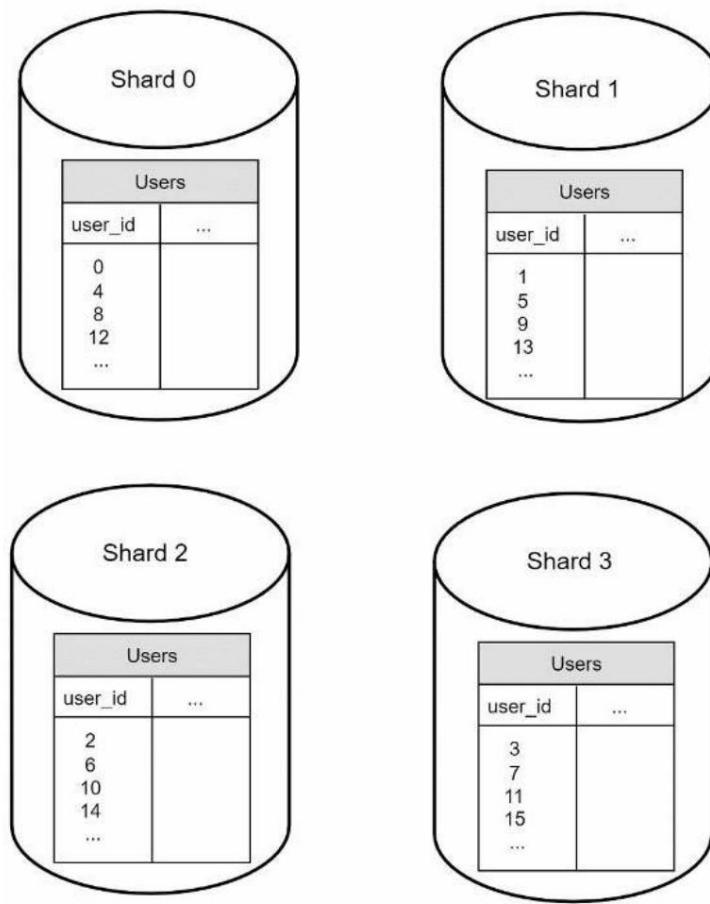


Figure 1-22

Yếu tố quan trọng nhất cần xem xét khi triển khai chiến lược phân mảnh là lựa chọn khóa phân mảnh. Khóa phân mảnh (được gọi là khóa phân vùng) bao gồm một hoặc nhiều cột xác định cách dữ liệu được phân phối. Như thể hiện trong Hình 1-22, "user_id" là khóa phân mảnh. Khóa phân mảnh cho phép bạn truy xuất và sửa đổi dữ liệu hiệu quả bằng cách định tuyến các truy vấn cơ sở dữ liệu đến đúng cơ sở dữ liệu. Khi chọn khóa phân mảnh, một trong những điều quan trọng nhất

Tiêu chí là chọn một khóa có thể phân bổ dữ liệu đồng đều.

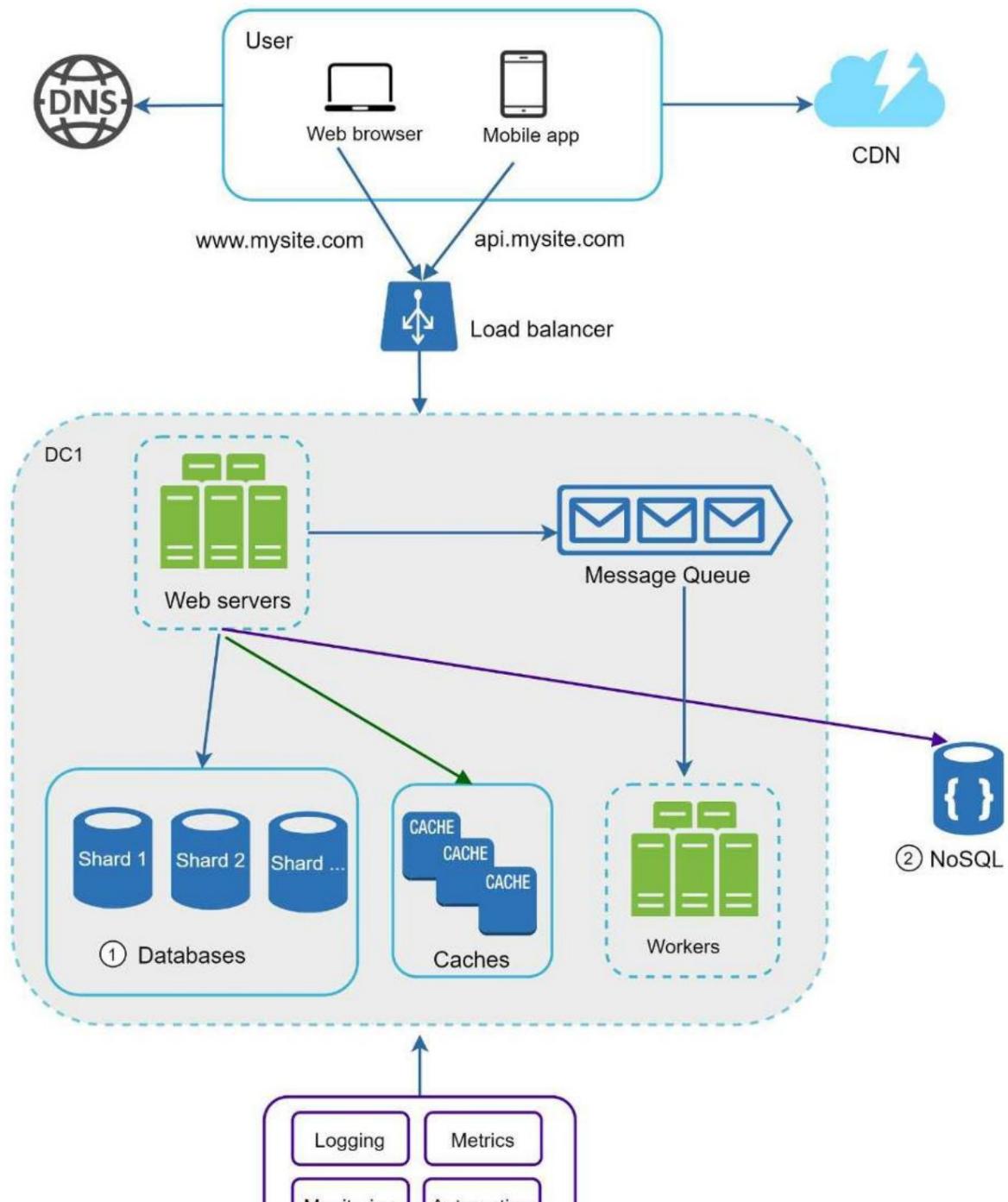
Phân mảnh là một kỹ thuật tuyệt vời để mở rộng cơ sở dữ liệu như nó không phải là giải pháp hoàn hảo. Nó tạo ra sự phức tạp và thách thức mới cho hệ thống:

Phân mảnh lại dữ liệu: Phân mảnh lại dữ liệu là cần thiết khi 1) một mảnh đơn lẻ không còn có thể chứa nhiều dữ liệu hơn do tăng trưởng nhanh. 2) Một số mảnh nhất định có thể bị cạn kiệt mảnh nhanh hơn những mảnh khác do phân phối dữ liệu không đồng đều. Khi tình trạng cạn kiệt mảnh xảy ra, cần phải cập nhật hàm phân mảnh và di chuyển dữ liệu xung quanh. Bước 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: Vấn đề này cũng được gọi là vấn đề khóa điểm truy cập. Truy cập quá mức vào một phân đoạn cụ thể có thể gây quá tải máy chủ. Hãy thử ống tư ống dữ liệu của Katy Perry, Justin Bieber và Lady Gaga đều nằm trên cùng một phân đoạn. Đối với các ứng dụng xã hội, phân đoạn đó sẽ bị quá tải với các hoạt động 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.

Join và de-normalization: Khi một cơ sở dữ liệu đã được phân mảnh trên nhiều máy chủ, sẽ rất khó để thực hiện các hoạt động join trên các phân mảnh cơ sở dữ liệu. Một giải pháp thay thế phổ biến là de-normalize cơ sở dữ liệu để có thể thực hiện các truy vấn trong một bảng duy nhất.

Trong Hình 1-23, chúng tôi phân mảnh cơ sở dữ liệu để hỗ trợ lưu lượng dữ liệu tăng nhanh. Đồng thời, một số chức năng không quan hệ được chuyển đến kho dữ liệu NoSQL để giảm tải cơ sở dữ liệu. Sau đây là một bài viết đề cập đến nhiều trường hợp sử dụng NoSQL [14].



Tools

Figure 1-23

Hàng triệu người dùng và hơn nữa Việt

mở rộng quy mô hệ thống là một quá trình lắp đặt lắp lại. Lắp lại những gì chúng ta đã học được trong chương này có thể giúp chúng ta tiến xa hơn. Cần phải tinh chỉnh nhiều hơn và các chiến lược mới để mở rộng quy mô vượt ra ngoài hàng triệu người dùng. Ví dụ, bạn có thể cần tối ưu hóa hệ thống của mình và tách hệ thống ra thành các dịch vụ thậm chí còn nhỏ hơn. Tất cả các kỹ thuật được học trong chương này sẽ cung cấp một nền tảng tốt để giải quyết những thách thức mới. Để kết thúc chương này, chúng tôi cung cấp một 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 trữ dữ liệu đậm nhiều nhất có thể • Hỗ trợ nhiều trung tâm dữ liệu • Lưu trữ tài sản tĩnh trong CDN
- Mở rộng quy mô tầng dữ liệu của bạn bằng cách phân mảnh • Chia các tầng thành các dịch vụ riêng lẻ • Giám sát hệ thống của bạn và sử dụng các công cụ tự động hóa

Xin chúc mừng vì đã đi đến đây! Bây giờ hãy tự khen mình nhé. 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 không?: <https://blog.teamtreehouse.com/should-you-go-beyond-relational-databases> [3] Sao chép: [https://en.wikipedia.org/wiki/Replication_\(computing\)](https://en.wikipedia.org/wiki/Replication_(computing))

[4] Sao chép đa chủ: https://en.wikipedia.org/wiki/Multi-master_replication [5] Sao chép cụm
NDB: Sao chép đa chủ và sao chép tuần hoàn: <https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-replication-multi-master.html> [6] Chiến lược lưu trữ 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, mở rộng Memcache tại," Hội nghị chuyên đề USENIX lần
thứ 10 về Thiết kế và triển khai hệ thống mạng (NSDI '13).

[8] Điểm lỗi đơn: https://en.wikipedia.org/wiki/Single_point_of_failure [9] Amazon CloudFront
Dynamic Content Delivery: <https://aws.amazon.com/cloudfront/dynamic-content/> [10] Cấu hình Sticky Sessions cho Classic Load
Balancer của bạn: <https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-sticky-sessions.html> [11] Active-Active cho khả năng phục hồi đa vùng: <https://netflixtechblog.com/active-active-for-multi-regional-resiliency-c47719f6685b> [12]
Amazon EC2 High Memory Instances: <https://aws.amazon.com/ec2/instance-types/high-memory/> [13] Những gì
cần có để chạy Stack Overflow: <http://nickcraver.com/blog/2013/11/22/what-it-takes-to-run-stack-overflow> [14] Bạn thực sự
đang sử dụng NoSQL để làm gì: <http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html>

CHƯƠNG 2: ƯỚC TÍNH TRÊN BÌA

Trong một cuộc phỏng vấn thiết kế hệ thống, đôi khi bạn được yêu cầu ước tính năng lực hệ thống hoặc các yêu cầu về hiệu suất bằng cách sử dụng ước tính sơ bộ. Theo Jeff Dean, Nghiên cứu viên cao cấp của Google, “các phép tính sơ bộ là các ước tính bạn tạo ra bằng cách kết hợp các thí nghiệm tư duy và các số liệu hiệu suất chung để có cảm nhận tốt về thiết kế nào sẽ đáp ứng các yêu cầu của bạn” [1].

Bạn cần có hiểu biết tốt về các nguyên tắc cơ bản về khả năng mở rộng để thực hiện ước tính sơ bộ một cách hiệu quả. Các khái niệm sau đây cần được hiểu rõ: lũy thừa của hai [2], số độ trễ mà mọi lập trình viên nên biết và số khả dụng.

Sức mạnh của hai

Mặc dù khối lượng dữ liệu có thể trở nên rất lớn khi xử lý các hệ thống phân tán, nhưng tất cả các phép tính đều được tóm gọn lại thành những điều cơ bản. Để có được các phép tính chính xác, điều quan trọng là phải biết đơn vị khối lượng dữ liệu bằng cách sử dụng lũy thừa của 2. Một byte là một chuỗi 8 bit. Một ký tự ASCII sử dụng một byte bộ nhớ (8 bit). Dưới đây là bảng giải thích đơn vị khối lượng dữ liệu (Bảng 2-1).

Power	Approximate value	Full name	Short name
10	1 Thousand	1 Kilobyte	1 KB
20	1 Million	1 Megabyte	1 MB
30	1 Billion	1 Gigabyte	1 GB
40	1 Trillion	1 Terabyte	1 TB
50	1 Quadrillion	1 Petabyte	1 PB

Table 2-1

Số độ trễ mà mọi lập trình viên nên biết Tiến sĩ Dean từ

Google tiết lộ độ dài của các hoạt động máy tính thông thường vào năm 2010 [1]. Một số con số đã lỗi thời vì máy tính trở nên nhanh hơn và mạnh hơn. Tuy nhiên, những con số đó vẫn có thể cung cấp cho chúng ta ý tưởng về tốc độ và độ chậm của các hoạt động máy tính khác nhau.

Operation name	Time
L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns = 10 µs
Send 2K bytes over 1 Gbps network	20,000 ns = 20 µs
Read 1 MB sequentially from memory	250,000 ns = 250 µs
Round trip within the same datacenter	500,000 ns = 500 µs
Disk seek	10,000,000 ns = 10 ms
Read 1 MB sequentially from the network	10,000,000 ns = 10 ms
Read 1 MB sequentially from disk	30,000,000 ns = 30 ms
Send packet CA (California) ->Netherlands->CA	150,000,000 ns = 150 ms

Table 2-2

Ghi chú

ns = nano giây, µs = micro giây, ms = mili giây 1 ns = 10^{-9} giây
1 µs = 10^{-6} giây =

1.000 ns 1 ms = 10^{-3} giây = 1.000 µs
= 1.000.000 ns

Một kỹ sư phần mềm của Google đã xây dựng một công cụ để trực quan hóa các con số của Tiến sĩ Dean. Công cụ này cũng tính đến yếu tố thời gian. Hình 2-1 cho thấy các con số độ trễ trực quan hóa tính đến năm 2020 (nguồn số liệu: tài liệu tham khảo [3]).

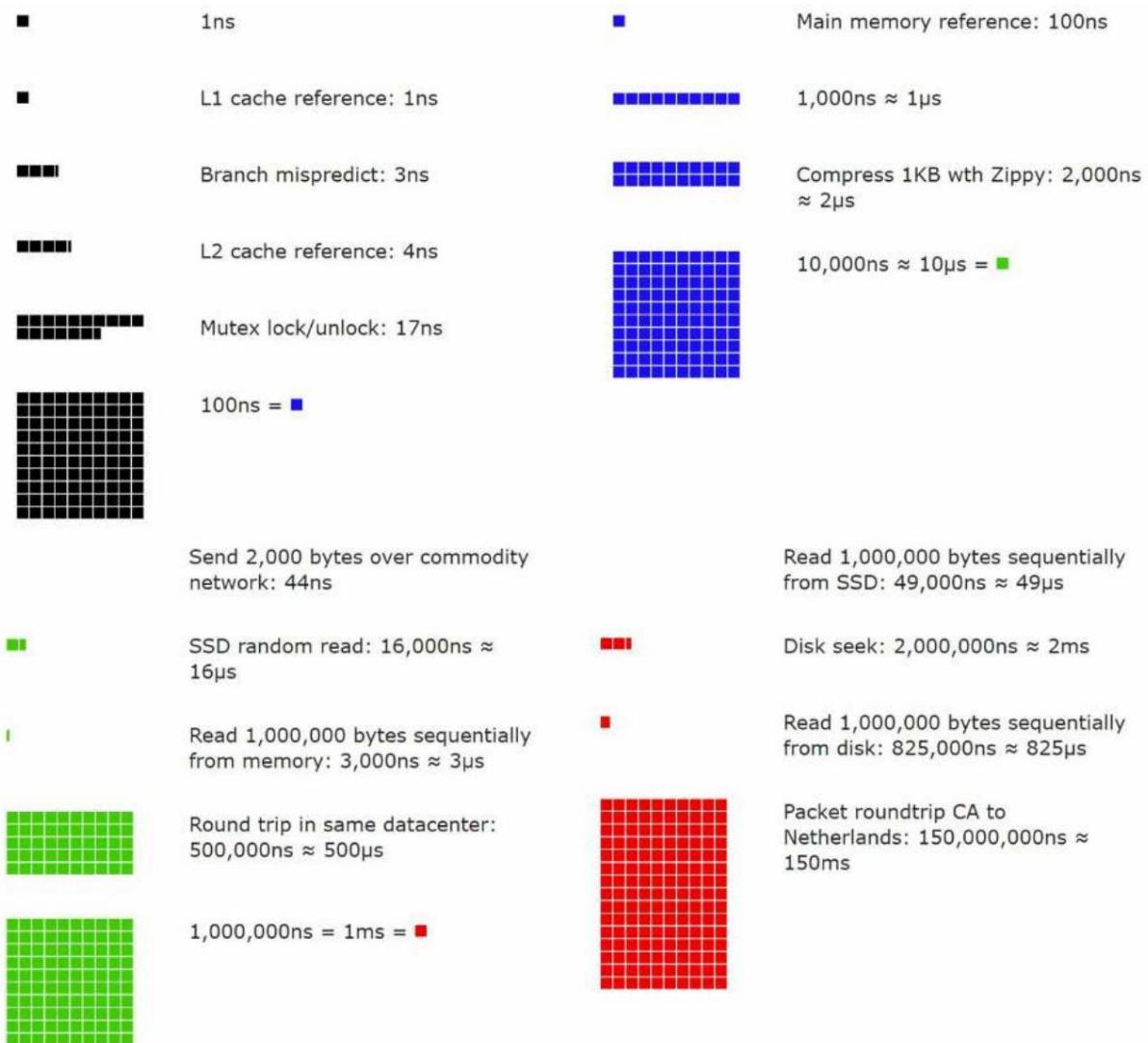


Figure 2-1

Qua phân tích các số liệu trong Hình 2-1, chúng ta rút ra được những kết luận sau:

- Bộ nhớ nhanh như ổ đĩa chùm. • Tránh tìm kiếm ổ đĩa nếu có thể. • Thuật toán nén đơn giản nhanh. • Nén dữ liệu trước khi gửi qua internet nếu có thể. • Các trung tâm dữ liệu thường ở các khu vực khác nhau và mất thời gian để gửi dữ liệu giữa các trung tâm.

Số liệu khả dụng Khả năng khả

dụng cao là khả năng của một hệ thống liên tục hoạt động trong một khoảng thời gian dài mong muốn. Khả năng khả dụng cao được đo bằng phần trăm, với 100% có nghĩa là dịch vụ không có thời gian chết. Hầu hết các dịch vụ nằm trong khoảng từ 99% đến 100%.

Thỏa thuận mức dịch vụ (SLA) là thuật ngữ thường được sử dụng cho các nhà cung cấp dịch vụ. Đây là thỏa thuận giữa bạn (nhà cung cấp dịch vụ) và khách hàng của bạn, và thỏa thuận này chính thức xác định mức thời gian hoạt động mà dịch vụ của bạn sẽ cung cấp. Các nhà cung cấp dịch vụ đám mây Amazon [4], Google [5] và Microsoft [6] đặt SLA của họ ở mức 99,9% trở lên. Thời gian hoạt động theo truyền thống được đo bằng số chín. Số chín càng nhiều thì càng tốt. Như thể hiện trong Bảng 2-3, số lượng số chín tương quan với thời gian ngừng hoạt động dự kiến của hệ thống.

Availability %	Downtime per day	Downtime per year
99%	14.40 minutes	3.65 days
99.9%	1.44 minutes	8.77 hours
99.99%	8.64 seconds	52.60 minutes
99.999%	864.00 milliseconds	5.26 minutes
99.9999%	86.40 milliseconds	31.56 seconds

Table 2-3

Ví dụ: Ước tính QPS của Twitter và yêu cầu lưu trữ Xin lưu ý rằng các số liệu sau chỉ dành cho bài tập này vì chúng không phải là số liệu thực từ Twitter.

Giả định: • 300

- triệu người dùng hoạt động hàng tháng.
- 50% người dùng sử dụng Twitter hàng ngày.
- Người dùng đăng trung bình 2 tweet mỗi ngày.
- 10% tweet chứa nội dung phuơng tiện.
- Dữ liệu được lưu trữ trong 5 năm.

Ước tính:

Ước tính số lượng truy vấn mỗi giây (QPS):

- Người dùng hoạt động hàng ngày (DAU) = 300 triệu * 50% = 150 triệu • QPS của Tweet = 150 triệu * 2 tweet / 24 giờ / 3600 giây = ~3500 • QPS của Peak = 2 * QPS = ~7000

Chúng tôi chỉ ước tính dung lượng lưu trữ phuơng tiện ở đây.

- Kích thước tweet trung bình:
 - tweet_id 64 byte 140 byte
 - chữ
 - phuơng tiện truyền thông 1 MB
- Lưu trữ phuơng tiện: 150 triệu * 2 * 10% * 1 MB = 30 TB mỗi ngày • Lưu trữ phuơng tiện 5 năm: 30 TB * 365 * 5 = ~55 PB

Mẹo

Ước tính sơ bộ là tất cả về quy trình. Giải quyết vấn đề quan trọng hơn là đạt được kết quả. Người chơi phỏng vấn có thể kiểm tra kỹ năng giải quyết vấn đề của bạn. Sau đây là một số mẹo cần tuân theo:

- Làm tròn và Xấp xỉ. Thật khó để thực hiện các phép toán phức tạp trong buổi phỏng vấn. Ví dụ, kết quả của "99987 / 9.1" là bao nhiêu? Không cần phải dành thời gian quý báu để giải các bài toán phức tạp. Không cần độ chính xác. Sử dụng số tròn và phép xấp xỉ để có lợi cho bạn. Câu hỏi chia có thể được đơn giản hóa như sau: "100.000 / 10". • Viết ra các giả định của bạn. Viết ra các giả định của bạn là một ý tưởng hay để tham khảo sau.

- Ghi nhận đơn vị của bạn. Khi bạn viết "5", có nghĩa là 5 KB hay 5 MB? Bạn có thể tự làm mình bối rối với điều này. Hãy viết ra các đơn vị vì "5 MB" giúp loại bỏ sự mơ hồ. • Các ước tính thường được hỏi:

QPS, QPS định, lưu trữ, bộ nhớ đệm, số lượng máy chủ, v.v. Bạn có thể thực hành các phép tính này khi chuẩn bị cho một cuộc phỏng vấn. Thực hành tạo nên sự hoàn hảo.

Xin chúc mừng vì đã đi đến đây! Nay giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

[1] J. Dean. Google Pro Tip: Sử dụng Back-Of-The-Envelope-Calculations để chọn thiết kế tốt nhất: <http://highscalability.com/blog/2011/1/26/google-pro-tip-use-back-of-the-envelope-calculation-to-choose.html>

[2] Tài liệu hướng dẫn thiết kế hệ thống: <https://github.com/donnemartin/system-design-primer> [3]

Số liệu độ trễ mà mọi lập trình viên nên biết: https://colin-scott.github.io/personal_website/research/interactive_latency.html [4]

Thỏa thuận mức dịch vụ của Amazon Compute: <https://aws.amazon.com/compute/sla/> [5]

Thỏa thuận mức dịch vụ (SLA) của Compute

Engine: <https://cloud.google.com/compute/sla> [6]

Tóm tắt SLA cho các dịch vụ Azure: <https://azure.microsoft.com/en-us/support/legal/sla/summary/>

CHƯƠNG 3: KHUNG THIẾT KẾ HỆ THỐNG

PHỎNG VẤN

Bạn vừa có được một cuộc phỏng vấn tại chỗ đáng mơ ước tại công ty mơ ước của mình. Người điều phối tuyển dụng gửi cho bạn lịch trình cho ngày hôm đó. Khi lướt qua danh sách, bạn cảm thấy khá hài lòng cho đến khi mắt bạn dừng lại ở buổi phỏng vấn này - Phỏng vấn thiết kế hệ thống.

Phỏng vấn thiết kế hệ thống thường rất đáng sợ. Nó có thể mơ hồ như "thiết kế một sản phẩm X nổi tiếng?". Các câu hỏi mơ hồ và có vẻ rỗng một cách vô lý. Sự mệt mỏi của bạn là điều dễ hiểu. Rốt cuộc, làm sao ai đó có thể thiết kế một sản phẩm phổ biến trong một giờ mà hàng trăm nếu nói là hàng nghìn kỹ sư đã xây dựng?

Tin tốt là không ai mong đợi bạn làm điều đó. Thiết kế hệ thống trong thế giới thực cực kỳ phức tạp. Ví dụ, tìm kiếm trên Google có vẻ đơn giản một cách đánh lừa; tuy nhiên, luring công nghệ hỗ trợ cho sự đơn giản đó thực sự đáng kinh ngạc. Nếu không ai mong đợi bạn thiết kế một hệ thống trong thế giới thực trong một giờ, thì lợi ích của một cuộc phỏng vấn thiết kế hệ thống là gì?

Phỏng vấn thiết kế hệ thống mô phỏng việc giải quyết vấn đề trong đời thực, trong đó hai đồng nghiệp hợp tác giải quyết một vấn đề mơ hồ và đưa ra giải pháp đáp ứng mục tiêu của họ. Vấn đề là mở và không có câu trả lời hoàn hảo. Thiết kế cuối cùng ít quan trọng hơn so với công sức bạn bỏ ra trong quá trình thiết kế. Điều này cho phép bạn thể hiện kỹ năng thiết kế của mình, bảo vệ các lựa chọn thiết kế của mình và phản hồi phản hồi theo cách xây dựng.

Chúng ta hãy lật ngược lại và xem xét những gì diễn ra trong đầu người phỏng vấn khi cô ấy bước vào phòng họp để gặp bạn. Mục tiêu chính của người phỏng vấn là đánh giá chính xác khả năng của bạn. Điều cuối cùng cô ấy muốn là đưa ra đánh giá không có kết luận vì buổi họp diễn ra không tốt và không có đủ tín hiệu. Người phỏng vấn tìm kiếm điều gì ở một cuộc phỏng vấn thiết kế hệ thống?

Nhiều người nghĩ rằng phỏng vấn thiết kế hệ thống chỉ là về kỹ năng thiết kế kỹ thuật của một người. Nó còn hơn thế nữa. Một cuộc phỏng vấn thiết kế hệ thống hiệu quả sẽ đưa ra những tín hiệu mạnh mẽ về khả năng cộng tác, làm việc dưới áp lực và giải quyết sự mơ hồ một cách xây dựng của một người. Khả năng đặt câu hỏi hay cũng là một kỹ năng thiết yếu và nhiều người phỏng vấn đặc biệt tìm kiếm kỹ năng này.

Một người phỏng vấn giỏi cũng tìm kiếm những dấu hiệu cảnh báo. Kỹ thuật quá mức là một căn bệnh thực sự của nhiều kỹ sư vì họ thích sự tinh khiết của thiết kế và bỏ qua sự đánh đổi. Họ thường không biết về chi phí gộp của các hệ thống được thiết kế quá mức và nhiều công ty phải trả giá cao cho sự thiếu hiểu biết đó. Bạn chắc chắn không muốn thể hiện xu hướng này trong một cuộc phỏng vấn thiết kế hệ thống. Những dấu hiệu cảnh báo khác bao gồm tư duy hẹp hòi, bướng bỉnh, v.v.

Trong chương này, chúng ta sẽ xem xét một số mẹo hữu ích và giới thiệu một khuôn khổ đơn giản và hiệu quả để giải quyết các vấn đề phỏng vấn thiết kế hệ thống.

Quy trình 4 bư ớc để phỏng vấn thiết kế hệ thống hiệu quả Mỗi cuộc phỏng

vấn thiết kế hệ thống đều khác nhau. Một cuộc phỏng vấn thiết kế hệ thống tuyệt vời là cuộc phỏng vấn mở và không có giải pháp nào phù hợp với tất cả mọi người. Tuy nhiên, có các bư ớc và điểm chung cần đề cập trong mọi cuộc phỏng vấn thiết kế hệ thống.

Bư ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế "Tại sao hỏi lại
gần?"

Một cánh tay giơ lên ở phía cuối lớp.

"Vâng, Jimmy?", cô giáo đáp.

"Bởi vì anh ấy ĐÓI".

"Tốt lắm Jimmy."

Trong suốt thời thơ ấu, Jimmy luôn là người đầu tiên trả lời câu hỏi trong lớp.

Bất cứ khi nào giáo viên đặt câu hỏi, luôn có một đứa trẻ trong lớp thích trả lời câu hỏi, bất kể nó có biết câu trả lời hay không. Đó chính là Jimmy.

Jimmy là một học sinh giỏi. Cậu ấy tự hào vì biết tất cả các câu trả lời nhanh chóng. Trong các kỳ thi, cậu ấy thường là người đầu tiên hoàn thành các câu hỏi. Cậu ấy là sự lựa chọn hàng đầu của giáo viên cho bất kỳ cuộc thi học thuật nào.

ĐÙNG giống như Jimmy.

Trong một cuộc phỏng vấn thiết kế hệ thống, việc đưa ra câu trả lời nhanh chóng mà không suy nghĩ sẽ không mang lại cho bạn điểm thư ờng. Trả lời mà không hiểu rõ các yêu cầu là một dấu hiệu cảnh báo lớn vì cuộc phỏng vấn không phải là cuộc thi đó vui. Không có câu trả lời đúng.

Vì vậy, đừng vội đưa ra giải pháp. Hãy chậm lại. Hãy suy nghĩ sâu sắc và đặt câu hỏi để làm rõ các yêu cầu và giả định. Điều này cực kỳ quan trọng.

Là một kỹ sư, chúng tôi thích giải quyết các vấn đề khó và bắt tay vào thiết kế cuối cùng; tuy nhiên, cách tiếp cận này có thể khiến bạn thiết kế sai hệ thống. Một trong những kỹ năng quan trọng nhất của một kỹ sư là đặt câu hỏi đúng, đưa ra các giả định phù hợp và thu thập mọi thông tin cần thiết để xây dựng một hệ thống. Vì vậy, đừng ngại đặt câu hỏi.

Khi bạn đặt câu hỏi, người phỏng vấn sẽ trả lời trực tiếp câu hỏi của bạn hoặc yêu cầu bạn đưa ra giả định. Nếu trao đổi hợp sau xảy ra, hãy viết ra những giả định của bạn trên bảng trắng hoặc giấy. Bạn có thể cần chúng sau này.

Nên hỏi những loại câu hỏi nào? Hãy đặt câu hỏi để hiểu chính xác các yêu cầu. Sau đây là danh sách các câu hỏi giúp bạn bắt đầu:

- Chúng ta sẽ xâ

dựng những tính năng cụ thể nào? • Sản phẩm có bao

nhiều người dùng? • Công ty dự kiến mở rộng

quy mô nhanh như thế nào? Quy mô dự kiến trong 3 tháng, 6 tháng và 1 năm là bao nhiêu? • Công ty có nền tảng công nghệ nào? Bạn có

thể tận dụng những dịch vụ hiện có nào để đơn giản hóa thiết kế?

Ví dụ: Nếu

bạn được yêu cầu thiết kế một hệ thống nguồn cấp tin tức, bạn muốn đặt những câu hỏi giúp bạn làm rõ các yêu cầu. Cuộc trò chuyện giữa bạn và người phỏng vấn có thể như sau:

Ứng viên: Đây có phải là ứng dụng di động không? Hay ứng dụng web? Hoặc cả hai?

Người phỏng vấn: Cả hai.

Ứng viên: Tính năng quan trọng nhất của sản phẩm là gì?

Người phỏng vấn: Khả năng đăng bài và xem tin tức của bạn bè.

Ứng viên: Nguồn cấp tin tức được sắp xếp theo thứ tự thời gian người hay theo thứ tự cụ thể? Thứ tự cụ thể có nghĩa là mỗi bài đăng được gắn một trọng số khác nhau. Ví dụ, bài đăng từ bạn bè thân thiết của bạn quan trọng hơn bài đăng từ một nhóm.

Người phỏng vấn: Để đơn giản, chúng ta hãy giả sử nguồn cấp dữ liệu được sắp xếp theo thứ tự thời gian người lại.

Ứng viên: Một người dùng có thể có bao nhiêu bạn bè?

Người phỏng vấn: 5000

Ứng viên: Lượng giao thông là bao nhiêu?

Người phỏng vấn: 10 triệu người dùng hoạt động hàng ngày (DAU)

Ứng viên: Nguồn cấp dữ liệu có thể chứa hình ảnh, video hoặc chỉ văn bản không?

Người phỏng vấn: Nó có thể chứa các tập tin phương tiện, bao gồm cả hình ảnh và video.

Trên đây là một số câu hỏi mẫu mà bạn có thể hỏi người phỏng vấn. Điều quan trọng là phải hiểu các yêu cầu và làm rõ những điều mơ hồ

Bước 2 - Đề xuất thiết kế cấp cao và nhận được sự đồng thuận Ở bước này, chúng

tôi đặt mục tiêu phát triển thiết kế cấp cao và đạt được thỏa thuận với người phỏng vấn về thiết kế. Sẽ là một ý tưởng tuyệt vời nếu hợp tác với người phỏng vấn trong quá trình
quá trình.

- Đưa ra bản thiết kế ban đầu. Yêu cầu phản hồi. Đối xử với người phỏng vấn như một đồng đội và cùng nhau làm việc. Nhiều người phỏng vấn giỏi thích nói chuyện và tham gia.

- Vẽ sơ đồ hộp với các thành phần chính trên bảng trắng hoặc giấy. Có thể bao gồm máy khách (di động/web), API, máy chủ web, kho dữ liệu, bộ nhớ đệm, CDN, hàng đợi tin nhắn, v.v. • Thực hiện các phép tính sơ bộ để đánh giá xem bản thiết kế của bạn có phù hợp với các ràng buộc về quy mô hay không. Nghĩ lớn tiếng. Trao đổi với người phỏng vấn nếu cần sơ bộ trước khi bắt tay vào thực hiện.

Nếu có thể, hãy xem qua một vài trang hợp sử dụng cụ thể. Điều này sẽ giúp bạn định hình hình thiết kế cấp cao. Cũng có khả năng là các trang hợp sử dụng sẽ giúp bạn khám phá ra các trang hợp ngoại lệ mà bạn chưa xem xét.

Chúng ta có nên đưa điểm cuối API và lược bỏ cơ sở dữ liệu vào đây không? Điều này phụ thuộc vào vấn đề.

Đối với các vấn đề thiết kế lớn như "Thiết kế công cụ tìm kiếm Google", mức này hơi thấp.

Đối với một vấn đề như thiết kế phần mềm trợ cho một trò chơi poker nhiều người chơi thì đây là một trò chơi công bằng. Giao tiếp với người phỏng vấn bạn.

Ví dụ Chúng

ta hãy sử dụng "Thiết kế hệ thống nguồn cấp tin tức" để chứng minh cách tiếp cận thiết kế cấp cao. Ở đây bạn không cần phải hiểu hệ thống thực sự hoạt động như thế nào. Tất cả các chi tiết sẽ được giải thích trong Chương 11.

Ở cấp độ cao, thiết kế được chia thành hai luồng: xuất bản nguồn cấp dữ liệu và xây dựng nguồn cấp dữ liệu tin tức.

• Xuất bản

nguồn cấp dữ liệu: khi người dùng xuất bản bài đăng, dữ liệu tương ứng sẽ được ghi vào bộ nhớ đệm/cơ sở dữ liệu và bài đăng sẽ được đưa vào nguồn cấp dữ liệu tin tức của bạn bè.

- Xây dựng nguồn cấp tin tức: nguồn cấp tin tức được xây dựng bằng cách tổng hợp các bài đăng của bạn bè theo thứ tự thời gian ngược lại.

Hình 3-1 và Hình 3-2 trình bày các thiết kế cấp cao cho luồng xuất bản nguồn cấp dữ liệu và xây dựng nguồn cấp dữ liệu tin tức.

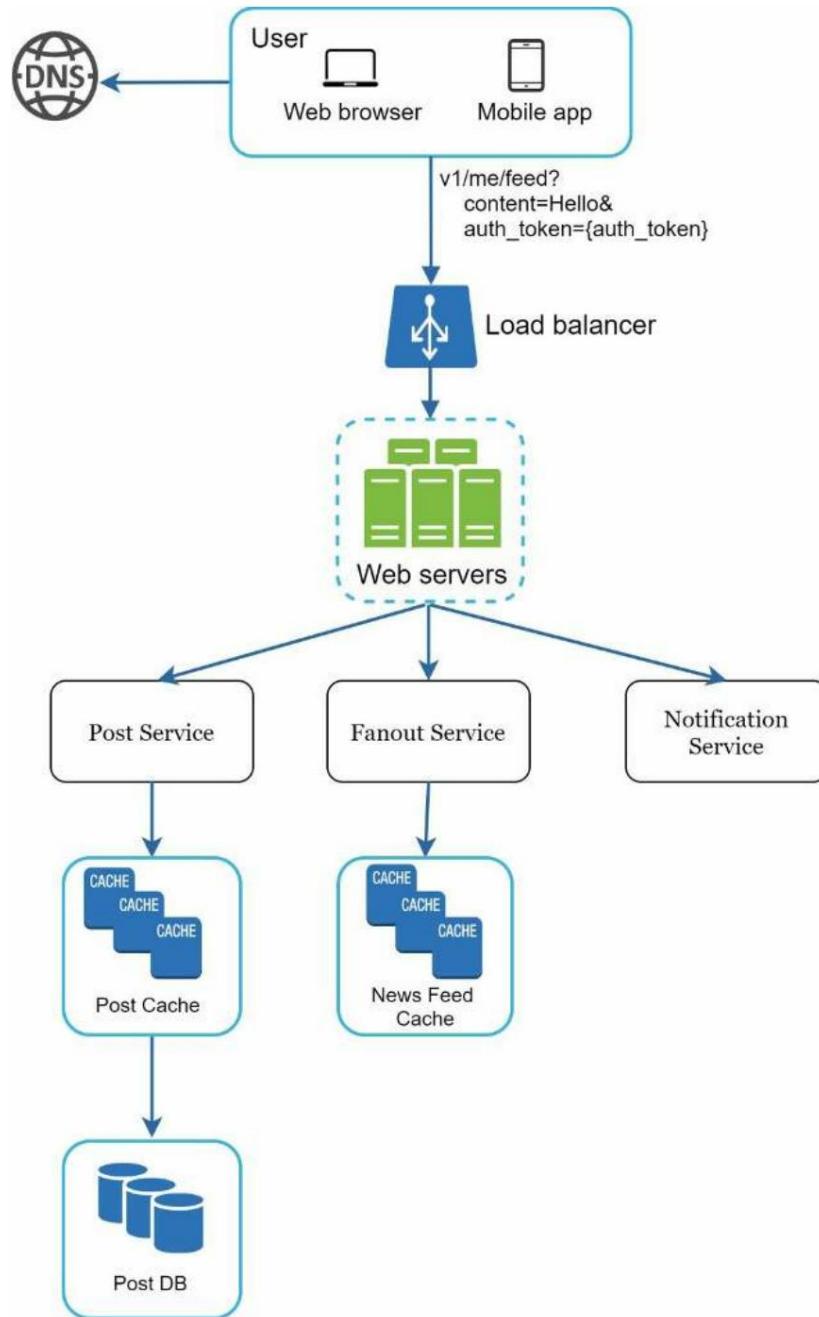


Figure 3-1

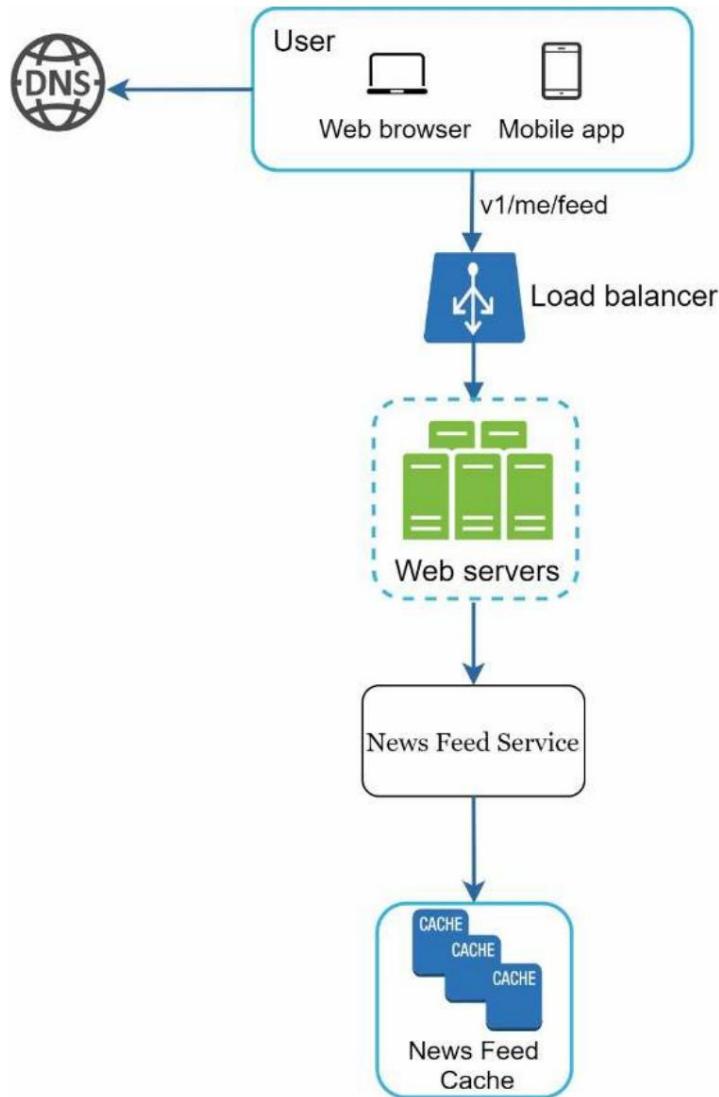


Figure 3-2

Bút ớc 3 - Thiết kế sâu

Ở bút ớc này, bạn và ngư ời phỏng vấn phải đạt đư ợc các mục tiêu sau:

- Thông nhất về mục tiêu chung và phạm vi tính năng
- Phác thảo bản thiết kế cấp cao cho thiết kế tổng thể • Nhận

phản hồi từ ngư ời phỏng vấn về thiết kế cấp cao • Có một số ý tư ớng ban

đầu về các lĩnh vực cần tập trung vào khi phân tích sâu dựa trên phản hồi của cô ấy Bạn sẽ

làm việc với ngư ời phỏng vấn để xác định và ưu tiên các thành phần trong kiến trúc.

Điều đáng nhấn mạnh là mỗi cuộc phỏng vấn đều khác nhau. Đôi khi, ngư ời phỏng vấn có thể đưa ra gợi ý rằng cô ấy thích tập trung vào thiết kế cấp cao. Đôi khi, đối với một cuộc phỏng vấn ứng viên cao cấp, cuộc thảo luận có thể là về các đặc điểm hiệu suất hệ thống, có khả năng tập trung vào các nút thắt cổ chai và Ước tính tài nguyên. Trong hầu hết các trường hợp, ngư ời phỏng vấn có thể muốn bạn đào sâu vào chi tiết của một số thành phần hệ thống. Đối với trình rút gọn URL, thật thú vị khi tìm hiểu sâu về thiết kế hàm băm chuyển đổi URL dài thành URL ngắn. Đối với hệ thống trò chuyện, cách giảm độ trễ và cách hỗ trợ trạng thái trực tuyến/ngoại tuyến là hai chủ đề thú vị.

Quản lý thời gian là điều cần thiết vì bạn dễ bị cuốn theo những chi tiết nhỏ không thể hiện đư ợc khả năng của mình. Bạn phải đư ợc trang bị những tín hiệu để cho ngư ời phỏng vấn thấy. Cố gắng không

đi sâu vào các chi tiết không cần thiết. Ví dụ, nói về thuật toán EdgeRank của bảng xếp hạng nguồn cấp dữ liệu Facebook một cách chi tiết không phải là lý tưởng trong một cuộc phỏng vấn thiết kế hệ thống vì điều này tốn nhiều thời gian quý báu và không chứng minh được khả năng thiết kế hệ thống có thể mở rộng của bạn.

Ví dụ Tại

thời điểm này, chúng ta đã thảo luận về thiết kế cấp cao cho hệ thống nguồn cấp tin tức và người dùng phỏng vấn hài lòng với đề xuất của bạn. Tiếp theo, chúng ta sẽ điều tra hai trong số những điều quan trọng nhất truy cập hợp sử dụng:

1. Xuất bản nguồn cấp dữ liệu
2. Truy xuất nguồn cấp dữ liệu tin tức

Hình 3-3 và Hình 3-4 hiển thị thiết kế chi tiết cho hai truy cập hợp sử dụng, sẽ được giải thích chi tiết trong Chương 11.

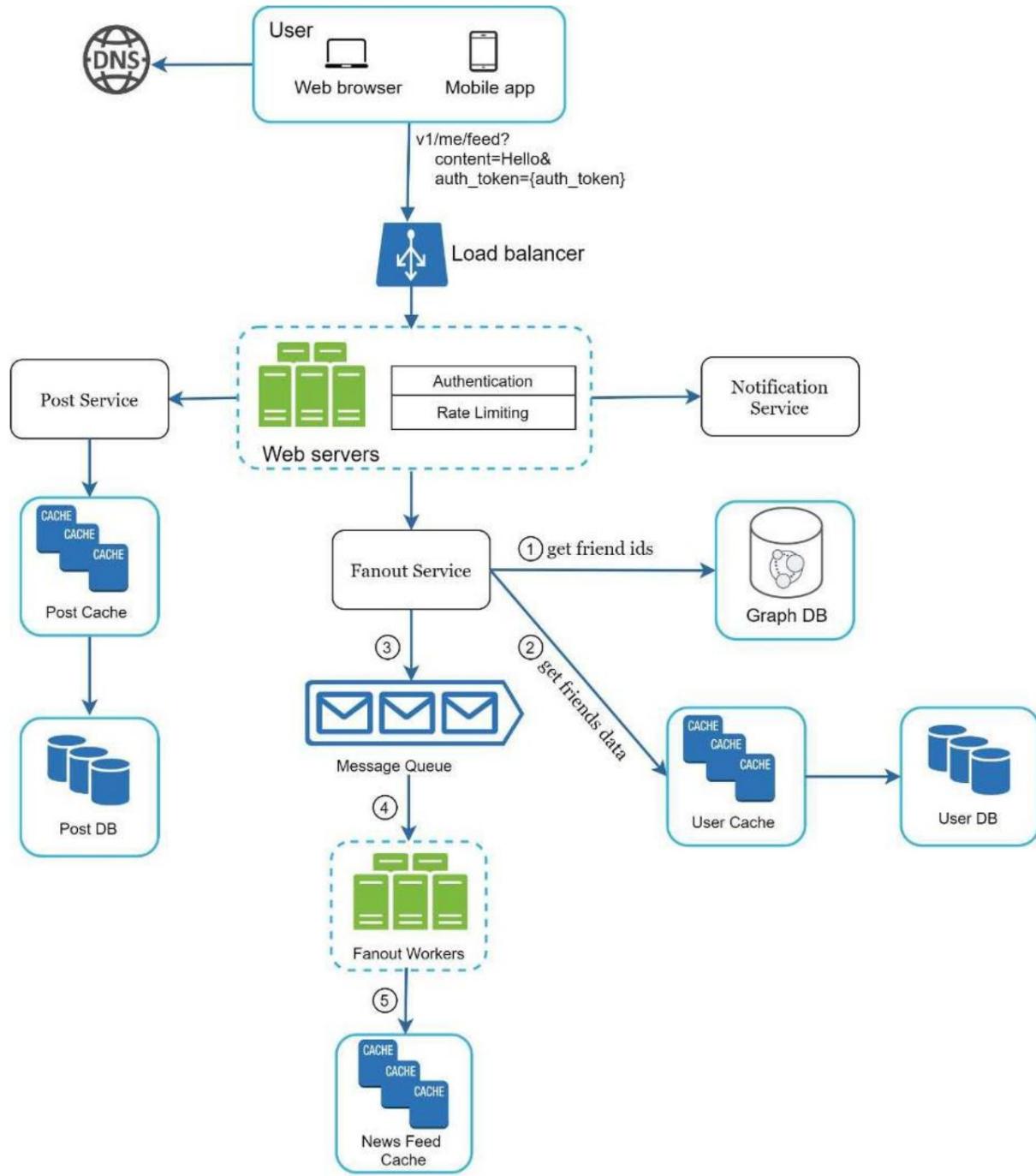


Figure 3-3

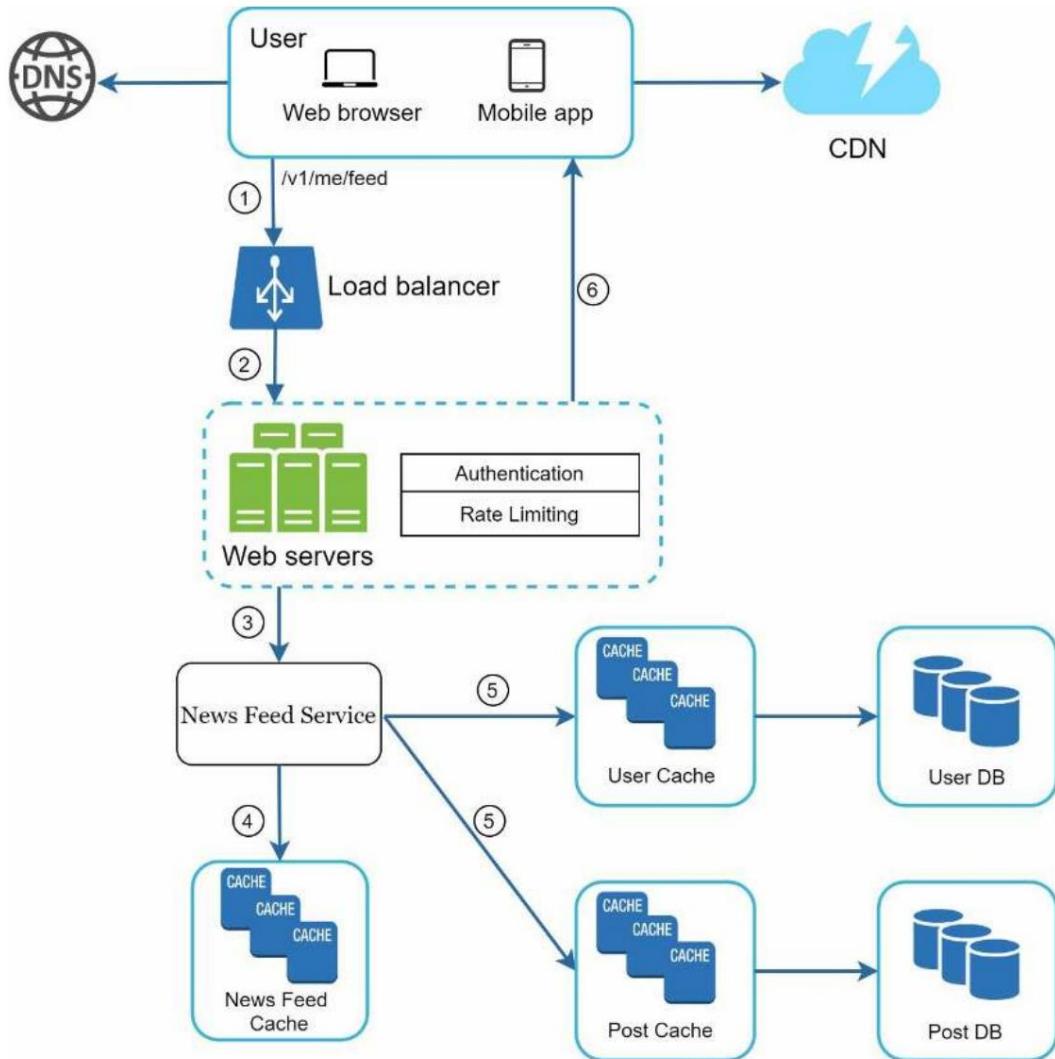


Figure 3-4

Bút ớc 4 - Kết thúc Ở

bút ớc cuối cùng này, người phỏng vấn có thể hỏi bạn một vài câu hỏi tiếp theo hoặc cho bạn tự do thảo luận các điểm bổ sung khác. Sau đây là một số hướng dẫn cần tuân theo:

- Người phỏng vấn có thể muốn bạn xác định các nút thắt của hệ thống và thảo luận về những cải tiến tiềm năng. Đừng bao giờ nói rằng thiết kế của bạn hoàn hảo và không thể cải thiện được gì. Luôn có điều gì đó cần cải thiện. Đây là cơ hội tuyệt vời để thể hiện tư duy phản biện của bạn và để lại ấn tượng tốt cuối cùng.
- Có thể hữu ích khi tóm tắt lại thiết kế

của bạn cho người phỏng vấn. Điều này đặc biệt quan trọng nếu bạn đề xuất một vài giải pháp. Làm mới lại trí nhớ của người phỏng vấn có thể hữu ích sau một buổi phỏng vấn dài. • Các trử ờng hợp lỗi (lỗi chủ, mất mạng, v.v.) rất thú vị

- để nói đến. • Các vấn đề về hoạt động đáng đư ợc đề cập. Bạn theo dõi số liệu và nhật ký lỗi như thế nào?

Làm thế nào để triển khai hệ

- thống? • Làm thế nào để xử lý đư ờng cong quy mô tiếp theo cũng là một chủ đề thú vị. Ví dụ, nếu thiết kế hiện tại của bạn hỗ trợ 1 triệu người dùng, bạn cần thực hiện những thay đổi nào để hỗ trợ 10 triệu người dùng?

- Đề xuất những cải tiến khác mà bạn cần nếu có nhiều thời gian hơn.

Để kết thúc, chúng tôi tóm tắt danh sách những điều nên làm và không nên làm.

Của

- Luôn yêu cầu làm rõ. Đừng cho rằng giả định của bạn là đúng. • Hiểu các yêu cầu của vấn đề. • Không có câu trả lời đúng hay câu trả lời tốt nhất. Một giải pháp được thiết kế để giải quyết các vấn đề của một công ty khởi nghiệp trẻ sẽ khác với giải pháp của một công ty đã thành danh với hàng triệu người dùng. Hãy đảm bảo rằng bạn hiểu các yêu cầu. • Hãy cho người phỏng vấn biết bạn đang nghĩ gì.
Trao đổi với người phỏng vấn của bạn. • Đề xuất nhiều cách tiếp cận nếu có thể. • Khi bạn đã đồng ý với người phỏng vấn về bản thiết kế, hãy đi vào chi tiết từng thành phần. Thiết kế các thành phần quan trọng nhất trước. • Trao đổi ý tư ởng với người phỏng vấn. Một người phỏng vấn giỏi sẽ làm việc với bạn như một đồng đội. • Đừng bao giờ bỏ cuộc.

Không nên

- Đừng thiếu chuẩn bị cho các câu hỏi phỏng vấn thông thường.
- Đừng vội đưa ra giải pháp mà không làm rõ các yêu cầu và giả định. • Đừng đi sâu vào quá nhiều chi tiết về một thành phần duy nhất ngay từ đầu. Đưa ra thiết kế cấp cao trước rồi mới đi sâu vào chi tiết. • Nếu bạn gặp khó khăn, đừng ngần ngại yêu cầu gợi ý. • Một lần nữa, hãy giao tiếp. Đừng suy nghĩ trong im lặng. • Đừng nghĩ rằng cuộc phỏng vấn của bạn đã kết thúc sau khi bạn đưa ra thiết kế. Bạn chưa hoàn thành cho đến khi người phỏng vấn nói rằng bạn đã hoàn thành. Hãy yêu cầu phản hồi sớm và thư ởng xuyênn.

Phân bổ thời gian cho từng bước

Các câu hỏi phỏng vấn thiết kế hệ thống thường rất rộng và 45 phút hoặc một giờ là không đủ để bao quát toàn bộ thiết kế. Quản lý thời gian là điều cần thiết. Bạn nên dành bao nhiêu thời gian cho từng bước? Sau đây là hướng dẫn rất sơ bộ về cách phân bổ thời gian của bạn trong buổi phỏng vấn kéo dài 45 phút. Xin lưu ý rằng đây chỉ là ước tính sơ bộ và việc phân bổ thời gian thực tế phụ thuộc vào phạm vi của vấn đề và các yêu cầu từ người phỏng vấn.

Bước 1 Hiểu vấn đề và thiết lập phạm vi thiết kế: 3 - 10 phút

Bước 2 Đề xuất thiết kế cấp cao và nhận được sự đồng thuận: 10 - 15 phút

Bước 3 Thiết kế chuyên sâu: 10 - 25 phút

Bước 4 Quán: 3 - 5 phút

CHƯƠNG 4: THIẾT KẾ BỘ GIỚI HẠN TỐC ĐỘ

Trong hệ thống mạng, bộ giới hạn tốc độ được sử dụng để kiểm soát tốc độ lưu lượng được gửi bởi máy khách hoặc dịch vụ.

Trong thế giới HTTP, bộ giới hạn tốc độ giới hạn số lượng yêu cầu của máy khách được phép gửi trong một khoảng thời gian nhất định. Nếu số lượng yêu cầu API vượt quá ngưỡng do bộ giới hạn tốc độ xác định, tất cả các cuộc gọi vượt quá sẽ bị chặn. Sau đây là một số ví dụ:

- Người dùng không được viết quá 2 bài đăng mỗi giây. • Bạn có thể tạo tối đa 10 tài khoản mỗi ngày từ cùng một địa chỉ IP. • Bạn có thể yêu cầu phần thư ởng không quá 5 lần mỗi tuần từ cùng một thiết bị.

Trong chương này, bạn được yêu cầu thiết kế một bộ giới hạn tốc độ. Trước khi bắt đầu thiết kế, trước tiên chúng ta hãy xem xét những lợi ích của việc sử dụng bộ giới hạn tốc độ API:

- Ngăn chặn tình trạng thiếu hụt tài nguyên do tấn công Từ chối dịch vụ (DoS) [1]. Hầu như tất cả các API do các công ty công nghệ lớn phát hành đều áp dụng một số hình thức giới hạn tốc độ. Ví dụ: Twitter giới hạn số lượng tweet ở mức 300 trong 3 giờ [2]. API Google Docs có giới hạn mặc định sau: 300 cho mỗi người dùng trong 60 giây đối với các yêu cầu đọc [3]. Bộ giới hạn tốc độ ngăn chặn các cuộc tấn công DoS, cố ý hoặc vô ý, bằng cách chặn các cuộc gọi thừa.
- Giảm chi phí. Giới hạn các yêu cầu thừa có nghĩa là ít máy chủ hơn và phân bổ nhiều tài nguyên hơn cho các API có mức độ ưu tiên cao. Giới hạn tốc độ cực kỳ quan trọng đối với các công ty sử dụng API của bên thứ ba trả phí. Ví dụ: bạn bị tính phí theo mỗi cuộc gọi đối với các API bên ngoài sau: kiểm tra tín dụng, thực hiện thanh toán, truy xuất hồ sơ sức khỏe, v.v.

Việc hạn chế số lượng cuộc gọi là điều cần thiết để giảm chi phí. • Ngăn chặn máy chủ bị quá tải. Để giảm tải máy chủ, bộ giới hạn tốc độ được sử dụng để lọc các yêu cầu dư thừa do bot hoặc hành vi sai trái của người dùng gây ra.

Bút ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế Giới hạn tỷ lệ có thể được triển khai bằng các thuật toán khác nhau, mỗi thuật toán có ưu và nhược điểm riêng. Tương tác giữa người dùng và ứng viên giúp làm rõ loại giới hạn tỷ lệ mà chúng ta đang cố gắng xây dựng.

Ứng viên: Chúng ta sẽ thiết kế loại giới hạn tốc độ nào? Đó là giới hạn tốc độ phía máy khách hay giới hạn tốc độ API phía máy chủ?

Người dùng: Câu hỏi hay. Chúng tôi tập trung vào bộ giới hạn tốc độ API phía máy chủ.

Ứng viên: Bộ giới hạn tốc độ có giới hạn các yêu cầu API dựa trên IP, ID người dùng hay các thuộc tính khác không?

Người dùng: Bộ giới hạn tốc độ phải đủ linh hoạt để hỗ trợ các bộ quy tắc điều tiết khác nhau.

Ứng viên: Quy mô của hệ thống như thế nào? Hệ thống được xây dựng cho một công ty khởi nghiệp hay một công ty lớn có lượng người dùng lớn?

Người dùng: Hệ thống phải có khả năng xử lý số lượng lớn yêu cầu.

Ứng viên: Hệ thống có hoạt động được trong môi trường phân tán không?

Người dùng: Có.

Ứng viên: Bộ giới hạn tốc độ có phải là một dịch vụ riêng biệt hay nên được triển khai trong mã ứng dụng?

Người dùng: Quyết định thiết kế là tùy thuộc vào bạn.

Ứng viên: Chúng ta có cần thông báo cho người dùng đang bị hạn chế thông tin không?

Người dùng: Có.

Yêu cầu sau đây là

tóm tắt các yêu cầu đối với hệ thống:

- Giới hạn chính xác các yêu cầu quá mức.
- Độ trễ thấp. Bộ giới hạn tốc độ không được làm chậm thời gian phản hồi HTTP.
- Sử dụng càng ít bộ nhớ càng tốt.
- Giới hạn tốc độ phân tán. Bộ giới hạn tốc độ có thể được chia sẻ trên nhiều máy chủ hoặc quy trình.
- Xử lý ngoại lệ. Hiển thị các ngoại lệ rõ ràng cho người dùng khi yêu cầu của họ bị hạn chế.
- Khả năng chịu lỗi cao. Nếu có bất kỳ sự cố nào với bộ giới hạn tốc độ (ví dụ: máy chủ bộ nhớ đệm ngoại tuyến), điều đó sẽ không ảnh hưởng đến toàn bộ hệ thống.

Bức 2 - Đề xuất thiết kế cấp cao và nhận được sự đồng thuận

Chúng ta hãy giữ mọi thứ đơn giản và sử dụng mô hình máy khách và máy chủ cơ bản để giao tiếp.

Đặt bộ giới hạn tốc độ ở đâu?

Theo trực giác, bạn có thể triển khai bộ giới hạn tốc độ ở cả phía máy khách hoặc phía máy chủ.

- Triển khai phía máy khách. Nói chung, máy khách là nơi không đáng tin cậy để thực thi giới hạn tốc độ vì các yêu cầu của máy khách có thể dễ dàng bị làm giả bởi các tác nhân độc hại. Hơn nữa, chúng ta có thể không kiểm soát được việc triển khai máy khách.
- Triển khai phía máy chủ. Hình 4-1 cho thấy một bộ giới hạn tốc độ được đặt ở phía máy chủ.



Figure 4-1

Bên cạnh các triển khai phía máy khách và máy chủ, còn có một cách thay thế. Thay vì đặt bộ giới hạn tốc độ tại các máy chủ API, chúng tôi tạo một phần mềm gian giới hạn tốc độ, điều chỉnh các yêu cầu đến API của bạn như thể hiện trong Hình 4-2.

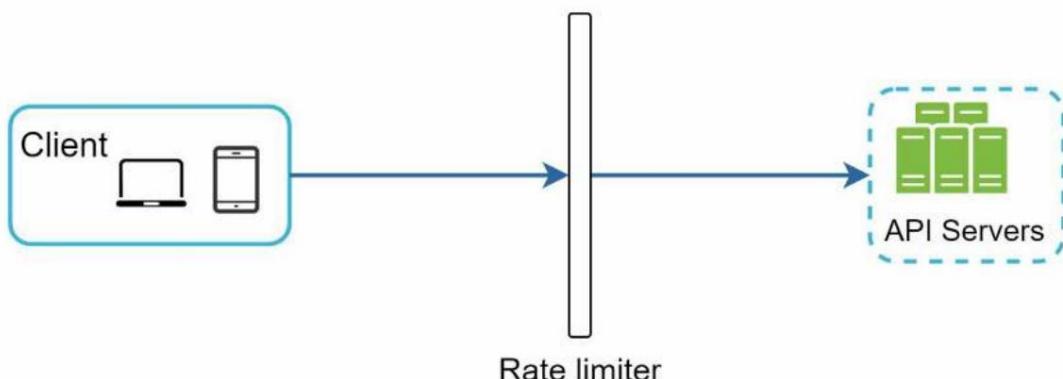


Figure 4-2

Chúng ta hãy sử dụng ví dụ trong Hình 4-3 để minh họa cách giới hạn tốc độ hoạt động trong thiết kế này.

Giả sử API của chúng ta cho phép 2 yêu cầu mỗi giây và một máy khách gửi 3 yêu cầu đến máy chủ trong vòng một giây. Hai yêu cầu đầu tiên được định tuyến đến máy chủ API. Tuy nhiên, phần mềm trung gian giới hạn tốc độ sẽ điều chỉnh yêu cầu thứ ba và trả về mã trạng thái HTTP 429. Mã trạng thái phản hồi HTTP 429 cho biết người dùng đã gửi quá nhiều yêu cầu.

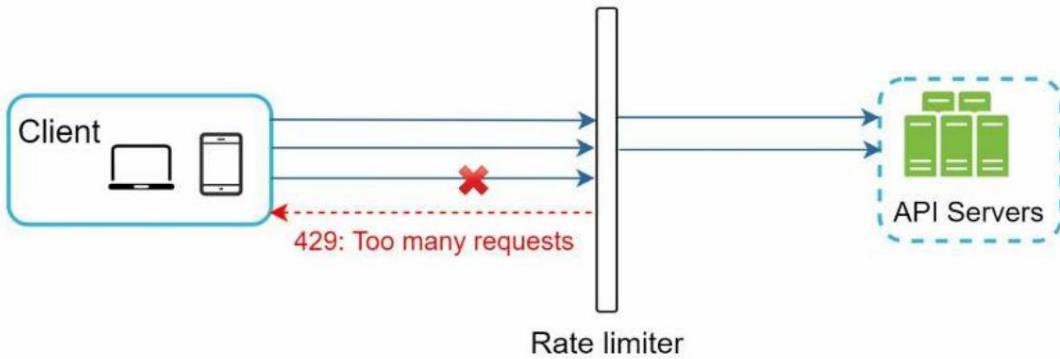


Figure 4-3

Các dịch vụ vi mô đám mây [4] đã trở nên phổ biến rộng rãi và việc giới hạn tốc độ thường được triển khai trong một thành phần gọi là cổng API. Cổng API là một dịch vụ được quản lý hoàn toàn hỗ trợ giới hạn tốc độ, chấm dứt SSL, xác thực, danh sách trắng IP, phục vụ nội dung tĩnh, v.v. Hiện tại, chúng ta chỉ cần biết rằng cổng API là một phần mềm trung gian hỗ trợ giới hạn tốc độ.

Khi thiết kế bộ giới hạn tốc độ, một câu hỏi quan trọng cần tự hỏi là: bộ giới hạn tốc độ nên được triển khai ở đâu, trên máy chủ hay trong cổng? Không có câu trả lời tuyệt đối.

Điều này phụ thuộc vào công nghệ hiện tại, nguồn lực kỹ thuật, ưu tiên, mục tiêu, v.v. của công ty bạn. Sau đây là một số hướng dẫn chung:

- Đánh giá công nghệ hiện tại của bạn, chẳng hạn như ngôn ngữ lập trình, dịch vụ bộ nhớ đệm, v.v. Đảm bảo ngôn ngữ lập trình hiện tại của bạn có hiệu quả để triển khai giới hạn tốc độ ở phía máy chủ.
- Xác định thuật toán giới hạn tốc độ phù hợp với nhu cầu kinh doanh của bạn. Khi bạn triển khai mọi thứ trên phía máy chủ, bạn có toàn quyền kiểm soát thuật toán. Tuy nhiên, lựa chọn của bạn có thể bị hạn chế nếu bạn sử dụng cổng bên thứ ba.
- Nếu bạn đã sử dụng kiến trúc dịch vụ vi mô và đưa cổng API vào thiết kế để thực hiện xác thực, danh sách trắng IP, v.v., bạn có thể thêm bộ giới hạn tốc độ vào cổng API.
- Việc xây dựng dịch vụ giới hạn tốc độ của riêng bạn cần có thời gian. Nếu bạn không có đủ nguồn lực kỹ thuật để triển khai bộ giới hạn tốc độ, thì cổng API thương mại là lựa chọn tốt hơn.

Thuật toán giới hạn tốc độ Giới hạn tốc

độ có thể được triển khai bằng nhiều thuật toán khác nhau và mỗi thuật toán đều có ưu và nhược điểm riêng biệt. Mặc dù chúng này không tập trung vào thuật toán, nhưng việc hiểu chúng ở cấp độ cao sẽ giúp chọn đúng thuật toán hoặc kết hợp các thuật toán phù hợp với trường hợp sử dụng của chúng ta. Sau đây là danh sách các thuật toán phổ biến:

- Token bucket

- Xô bị rò rỉ • Cửa sổ cố định
- Cửa sổ trượt log • Cửa sổ trượt counter

Thuật toán thùng token Thuật

toán thùng token được sử dụng rộng rãi để giới hạn tốc độ. Nó đơn giản, dễ hiểu và

thư ờng đư ợc các công ty internet sử dụng. Cả Amazon [5] và Stripe [6] đều sử dụng thuật toán này để điều tiết các yêu cầu API của họ.

Thuật toán nhóm token hoạt động như sau:

- Một thùng token là một thùng chứa có sức chứa đư ợc xác định trước. Token đư ợc đưa vào thùng theo tỷ lệ đư ợc thiết lập trước theo định kỳ. Khi thùng đầy, không thêm token nào nữa. Như thể hiện trong Hình 4-4, sức chứa của thùng token là 4. Người nạp lại sẽ đưa 2 token vào thùng mỗi giây. Khi thùng đầy, token thừa sẽ tràn ra ngoài.

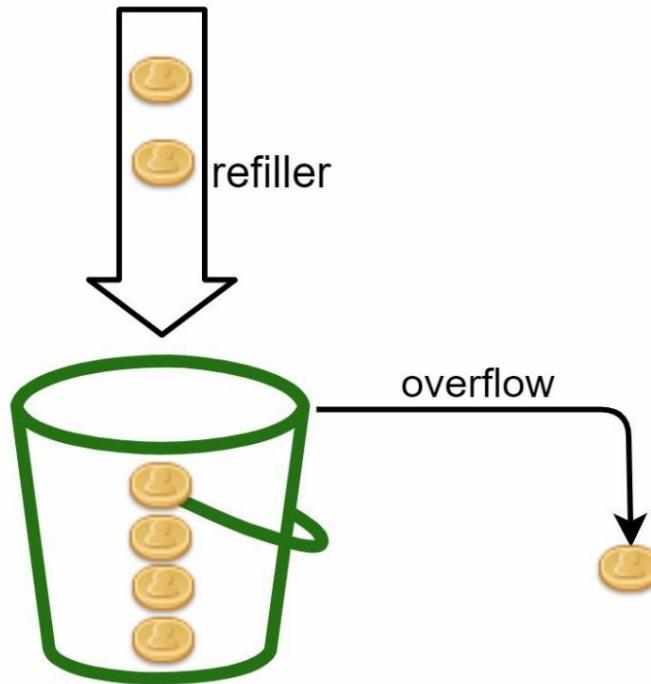


Figure 4-4

- Mỗi yêu cầu sử dụng một mã thông báo. Khi yêu cầu đến, chúng tôi kiểm tra xem có đủ mã thông báo trong thùng hay không. Hình 4-5 giải thích cách thức hoạt động.

- Nếu có đủ mã thông báo, chúng tôi sẽ lấy một mã thông báo cho mỗi yêu cầu và yêu cầu sẽ đư ợc xử lý. •

Nếu không đủ mã thông báo, yêu cầu sẽ bị hủy bỏ.

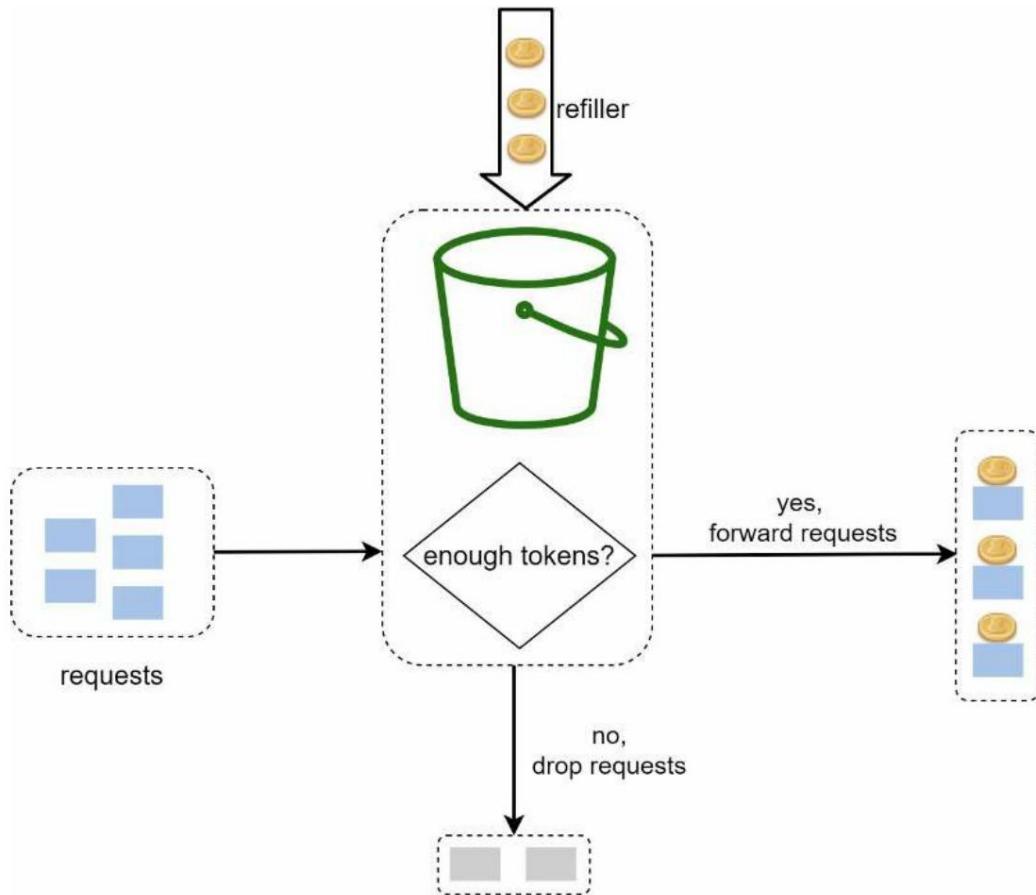


Figure 4-5

Hình 4-6 minh họa cách thức hoạt động của logic tiêu thụ token, nạp lại và giới hạn tốc độ.

Trong ví dụ này, kích thước buồng token là 4 và tốc độ nạp lại là 4 trong 1 phút.

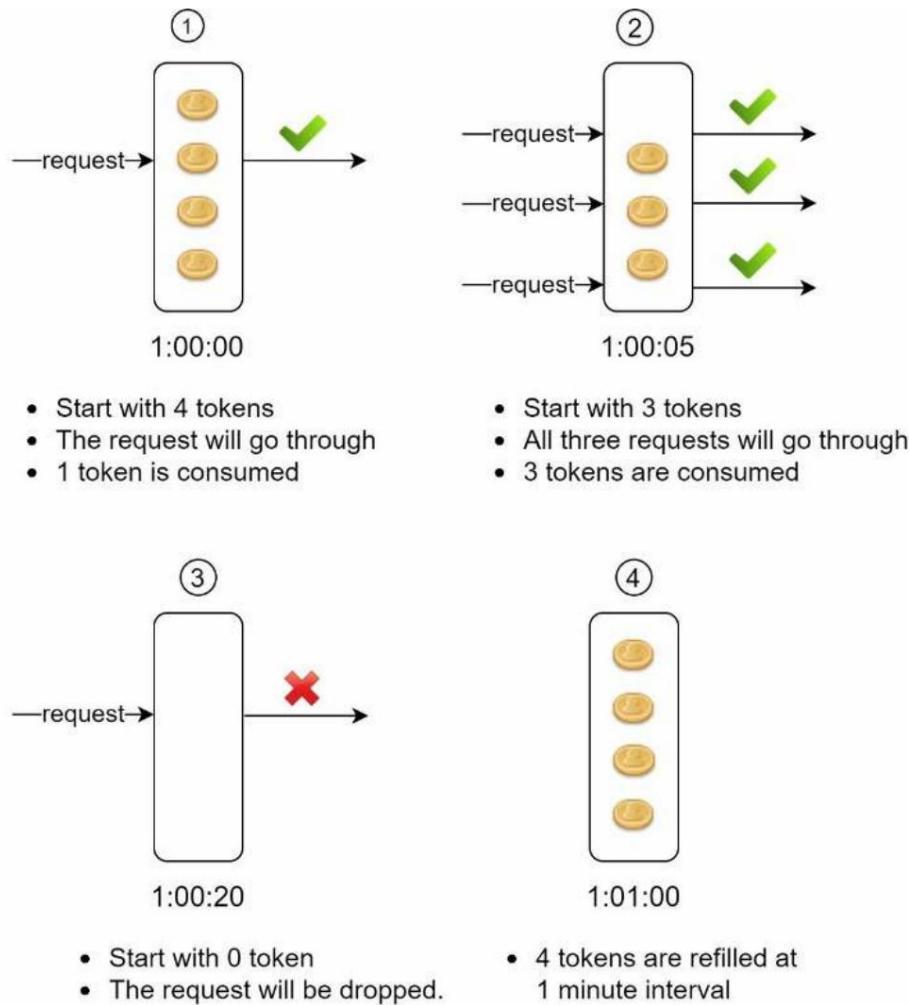


Figure 4-6

Thuật toán thùng mã thông báo có hai tham số:

- Kích thước thùng: số lượng mã thông báo tối đa được phép trong thùng
- Tỷ lệ nạp lại: số lượng token được đưa vào thùng mỗi giây

Chúng ta cần bao nhiêu thùng? Điều này thay đổi và phụ thuộc vào các quy tắc giới hạn tỷ lệ. Sau đây là một vài ví dụ.

- Thường cần có các bucket khác nhau cho các điểm cuối API khác nhau. Ví dụ, nếu người dùng đăng nhập 1 bài viết mỗi giây, thêm 150 bạn bè mỗi ngày và khoảng 5 bài viết mỗi giây, thì cần có 3 bucket cho mỗi người dùng.
- Nếu chúng ta cần điều chỉnh yêu cầu dựa trên địa chỉ IP, thì mỗi địa chỉ IP cần một bucket.
- Nếu hệ thống cho phép tối đa 10.000 yêu cầu mỗi giây, thì việc có một bucket toàn cục được chia sẻ cho tất cả các yêu cầu là hợp lý.

Ưu điểm:

- Thuật toán dễ triển khai.
- Hiệu quả về bộ nhớ.
- Token bucket cho phép lưu trữ truy cập bùng nổ trong thời gian ngắn. Yêu cầu có thể được thực hiện miễn phí vẫn còn token.

Nhược điểm:

- Hai tham số trong thuật toán là kích thước thùng và tỷ lệ nạp lại mã thông báo. Tuy nhiên, nó có thể

sẽ rất khó để điều chỉnh chúng cho đúng.

Thuật toán leaking bucket Thuật

toán leaks bucket tương tự như token bucket ngoại trừ việc các yêu cầu được xử lý ở tốc độ cố định. Thuật toán này thường được triển khai với hàng đợi first-in-first-out (FIFO). Thuật toán hoạt động như sau:

- Khi một yêu cầu đến, hệ thống sẽ kiểm tra xem hàng đợi có đầy không. Nếu không đầy, yêu cầu sẽ được thêm vào hàng đợi. • Nếu không, yêu cầu sẽ bị hủy. • Các yêu cầu sẽ được kéo ra khỏi hàng đợi và được xử lý theo các khoảng thời gian đều đặn.

Hình 4-7 giải thích cách thức hoạt động của thuật toán.

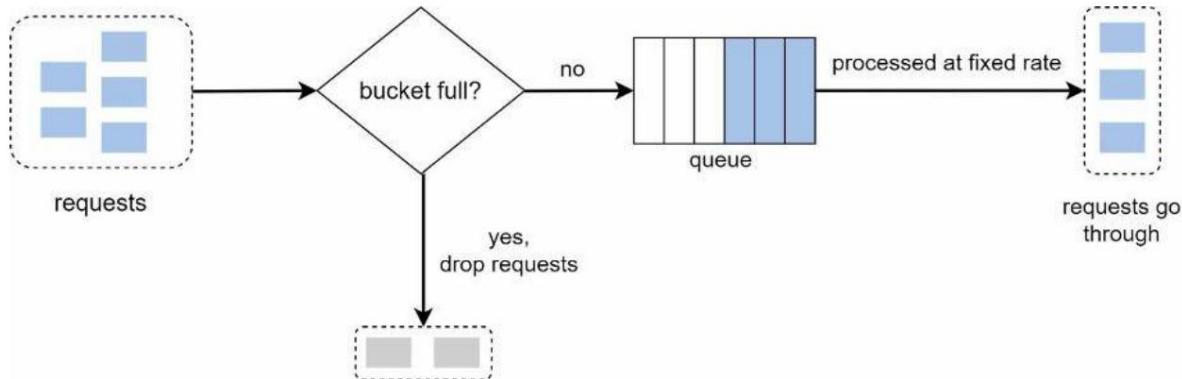


Figure 4-7

Thuật toán thùng rò rỉ sử dụng hai tham số sau:

- Kích thước thùng: bằng kích thước hàng đợi. Hàng đợi giữ các yêu cầu được xử lý ở tốc độ cố định.
- Tỷ lệ luồng ra: xác định số lượng yêu cầu có thể được xử lý theo một tỷ lệ cố định, thường tính bằng giây.

Shopify, một công ty thương mại điện tử, sử dụng các thùng rò rỉ để giới hạn tỷ lệ [7].

Ưu điểm:

- Hiệu quả về bộ nhớ do kích thước hàng đợi hạn chế. •

Các yêu cầu được xử lý ở tốc độ cố định, do đó phù hợp với các truy vấn hợp sử dụng cần tốc độ truyền ra ổn định.

Nhược điểm:

- Một luồng lưu lượng lớn sẽ lấp đầy hàng đợi bằng các yêu cầu cũ và nếu chúng không được xử lý kịp thời, các yêu cầu gần đây sẽ bị giới hạn tốc độ.
- Có hai tham số trong thuật toán. Có thể không dễ để điều chỉnh chúng một cách phù hợp.

Thuật toán đếm cửa sổ cố định Thuật toán

đếm cửa sổ cố định hoạt động như sau: • Thuật toán chia dòng

thời gian thành các cửa sổ thời gian có kích thước cố định và chỉ định một bộ đếm cho mỗi cửa sổ.

- Mỗi yêu cầu sẽ tăng bộ đếm lên một. • Khi bộ đếm đạt đến ngưỡng được xác định trước, các yêu cầu mới sẽ bị loại bỏ cho đến khi một cửa sổ thời gian mới bắt đầu.

Chúng ta hãy sử dụng một ví dụ cụ thể để xem cách thức hoạt động. Trong Hình 4-8, đơn vị thời gian là 1 giây và hệ thống cho phép tối đa 3 yêu cầu mỗi giây. Trong mỗi cửa sổ giây, nếu nhận được nhiều hơn 3 yêu cầu, các yêu cầu bổ sung sẽ bị loại bỏ như thể hiện trong Hình 4-8.

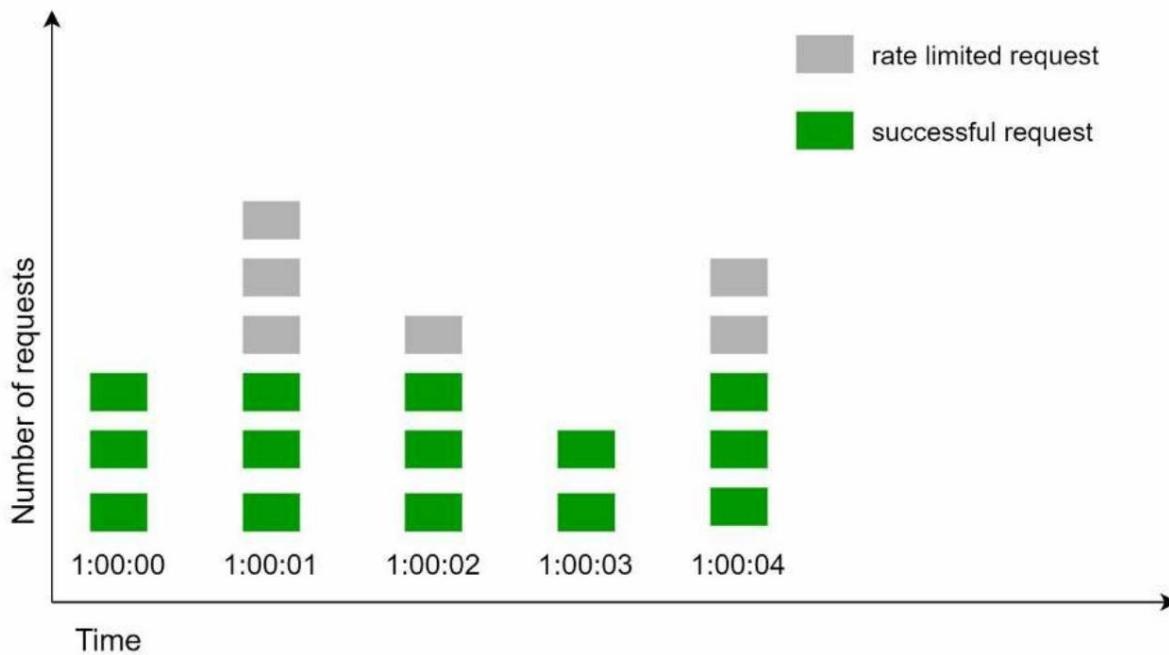


Figure 4-8

Một vấn đề lớn với thuật toán này là sự bùng nổ lưu lượng ở rìa cửa sổ thời gian có thể gây ra nhiều yêu cầu hơn hạn ngạch được phép đi qua. Hãy xem xét trường hợp sau:

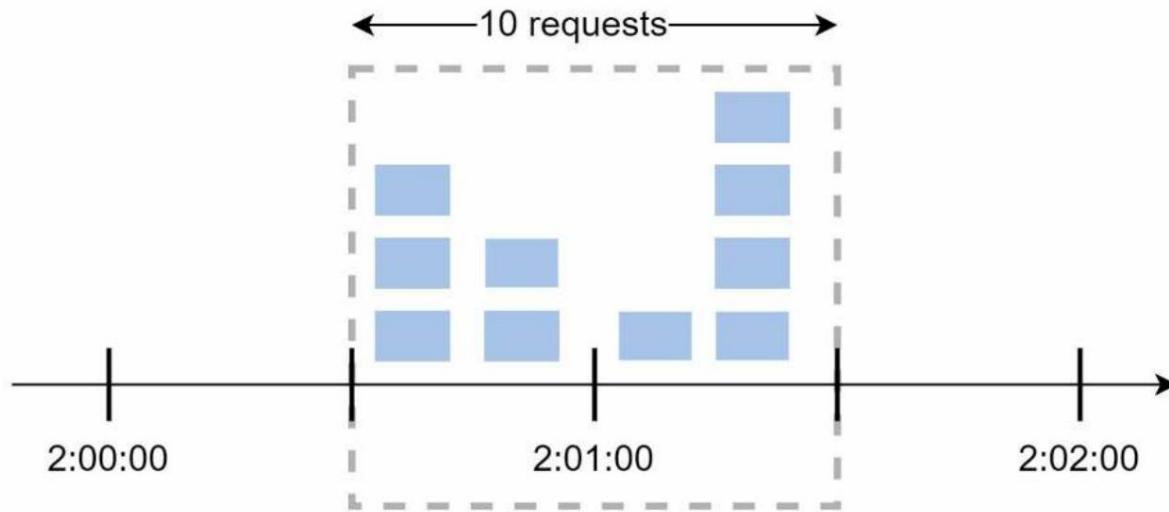


Figure 4-9

Trong Hình 4-9, hệ thống cho phép tối đa 5 yêu cầu mỗi phút và hạn ngạch khả dụng được đặt lại ở phút tròn thân thiện với con người. Như đã thấy, có năm yêu cầu giữa 2:00:00 và 2:01:00 và năm yêu cầu nữa giữa 2:01:00 và 2:02:00. Đối với khung thời gian một phút giữa 2:00:30 và 2:01:30, 10 yêu cầu được xử lý. Con số này gấp đôi số yêu cầu được phép.

Ưu điểm:

- Hiệu quả về bộ nhớ.

- Dễ hiểu. • Việc đặt lại

hạn ngạch khả dụng vào cuối cửa sổ thời gian đơn vị phù hợp với một số trường hợp sử dụng nhất định.

Như đọc điểm:

- Lưu lựợng truy cập tăng đột biến ở rìa cửa sổ có thể khiến số lưu lượng yêu cầu vượt quá hạn ngạch được phép xử lý.

Thuật toán nhật ký cửa sổ trượt Như

đã thảo luận trước đó, thuật toán bộ đếm cửa sổ có định có một vấn đề lớn: nó cho phép nhiều yêu cầu hơn đi qua các cạnh của cửa sổ. Thuật toán nhật ký cửa sổ trượt khắc phục vấn đề này. Nó hoạt động như sau:

- Thuật toán theo dõi dấu thời gian yêu cầu. Dữ liệu dấu thời gian thường được lưu trong bộ nhớ đệm, chẳng hạn như các tập hợp Redis được sắp xếp [8]. • Khi có yêu cầu mới, hãy xóa tất cả các dấu thời gian lỗi thời. Dấu thời gian lỗi thời được định nghĩa là những dấu thời gian cũ hơn thời điểm bắt đầu của cửa sổ thời gian hiện tại.
- Thêm dấu thời gian của yêu cầu mới vào nhật ký. • Nếu kích thước nhật ký bằng hoặc thấp hơn số lưu lượng được phép, yêu cầu sẽ được chấp nhận. Nếu không, nó sẽ bị từ chối.

Chúng tôi giải thích thuật toán bằng ví dụ như trong Hình 4-10.

Allow 2 requests per minute

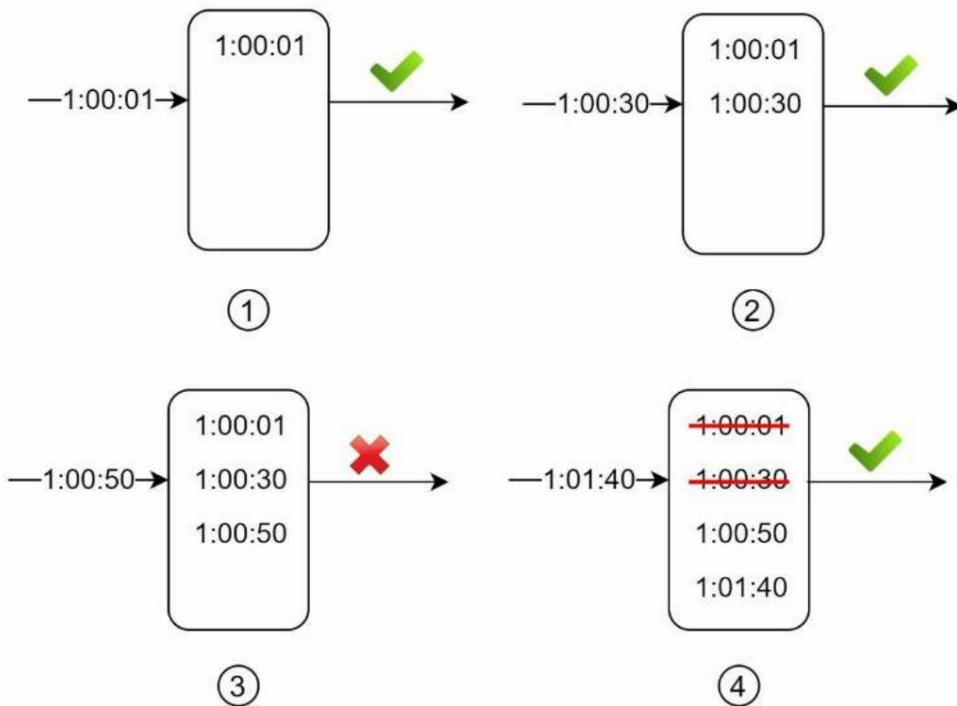


Figure 4-10

Trong ví dụ này, bộ giới hạn tốc độ cho phép 2 yêu cầu mỗi phút. Thông thường, dấu thời gian Linux được lưu trữ trong nhật ký. Tuy nhiên, biểu diễn thời gian có thể đọc được bằng con người được sử dụng trong ví dụ của chúng tôi để dễ đọc hơn.

Nhật ký trống khi có yêu cầu mới đến lúc 1:00:01. Do đó, yêu cầu được phép.

- Một yêu cầu mới đến lúc 1:00:30, dấu thời gian 1:00:30 được chèn vào nhật ký. Sau khi chèn, kích thước nhật ký là 2, không lớn hơn số lượng được phép. Do đó, yêu cầu được phép. • Một yêu cầu mới đến lúc 1:00:50 và dấu thời gian được chèn vào nhật ký. Sau khi chèn, kích thước nhật ký là 3, lớn hơn kích thước được phép là 2. Do đó, yêu cầu này bị từ chối mặc dù dấu thời gian vẫn còn trong nhật ký. • Một yêu cầu mới đến lúc 1:01:40. Các yêu cầu trong phạm vi [1:00:40,1:01:40) nằm trong khung thời gian mới nhất, nhưng các yêu cầu được gửi trước 1:00:40 đã lỗi thời. Hai dấu thời gian lỗi thời, 1:00:01 và 1:00:30, được xóa khỏi nhật ký. Sau thao tác xóa, kích thước nhật ký trở thành 2; do đó, yêu cầu được chấp nhận.

Ưu điểm:

- Giới hạn tốc độ được thực hiện bởi thuật toán này rất chính xác. Trong bất kỳ cửa sổ lăn nào, các yêu cầu sẽ không vượt quá giới hạn tốc độ.

Nhược điểm:

- Thuật toán này tiêu tốn rất nhiều bộ nhớ vì ngay cả khi yêu cầu bị từ chối, dấu thời gian của yêu cầu vẫn có thể được lưu trữ trong bộ nhớ.

Thuật toán đếm cửa sổ trượt Thuật toán đếm

cửa sổ trượt là một phương pháp lai kết hợp giữa bộ đếm cửa sổ cố định và nhật ký cửa sổ trượt. Thuật toán có thể được triển khai bằng hai phương pháp khác nhau. Chúng tôi sẽ giải thích một phương pháp triển khai trong phần này và cung cấp tài liệu tham khảo cho phương pháp triển khai còn lại ở cuối phần. Hình 4-11 minh họa cách thức hoạt động của thuật toán này.

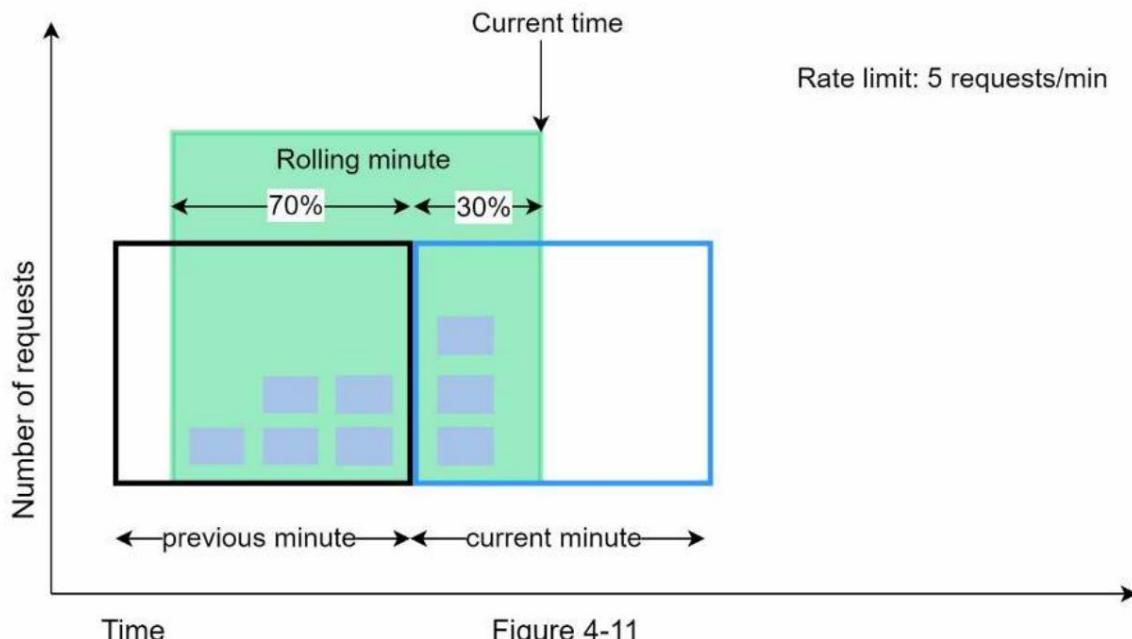


Figure 4-11

Giả sử bộ giới hạn tốc độ cho phép tối đa 7 yêu cầu mỗi phút và có 5 yêu cầu trong phút trứớc và 3 yêu cầu trong phút hiện tại. Đối với yêu cầu mới đến ở vị trí 30% trong phút hiện tại, số lượng yêu cầu trong cửa sổ lăn được tính bằng công thức sau:

- Các yêu cầu trong cửa sổ hiện tại + các yêu cầu trong cửa sổ trứớc * tỷ lệ chồng lấn của cửa sổ đang chạy và cửa sổ trứớc

- Sử dụng công thức này, chúng ta có $3 + 5 * 0,7\% = 6,5$ yêu cầu. Tùy thuộc vào truy ờng hợp sử dụng, số có thể được làm tròn lên hoặc xuống. Trong ví dụ của chúng tôi, nó được làm tròn xuống thành 6.

Vì bộ giới hạn tốc độ cho phép tối đa 7 yêu cầu mỗi phút nên yêu cầu hiện tại có thể được thông qua. Tuy nhiên, giới hạn sẽ đạt đến sau khi nhận thêm một yêu cầu nữa.

Do giới hạn không gian, chúng tôi sẽ không thảo luận về việc triển khai khác ở đây. Độc giả quan tâm nên tham khảo tài liệu tham khảo [9]. Thuật toán này không hoàn hảo. Nó có ưu và nhược điểm.

Ưu điểm

- Làm phẳng các đột biến về lưu lượng truy cập vì tốc độ được tính dựa trên tốc độ trung bình của cửa sổ trượt
- đó.
- Hiệu quả về bộ nhớ.

Nhược điểm

- Nó chỉ hoạt động đối với cửa sổ nhìn lại không quá nghiêm ngặt. Đây là một phép tính gần đúng của tỷ lệ thực tế vì nó giả định các yêu cầu trong cửa sổ trượt được phân bổ đều. Tuy nhiên, vẫn đề này có thể không tệ như vẻ bề ngoài. Theo các thí nghiệm do Cloudflare [10] thực hiện, chỉ có 0,003% yêu cầu được phép sai hoặc tỷ lệ bị giới hạn trong số 400 triệu yêu cầu.

Kiến trúc cấp cao Y tư ờng cơ

bản của thuật toán giới hạn tốc độ rất đơn giản. Ở cấp độ cao, chúng ta cần một bộ đếm để theo dõi số lưu lượng yêu cầu được gửi từ cùng một người dùng, địa chỉ IP, v.v. Nếu bộ đếm lớn hơn giới hạn, yêu cầu sẽ không được phép.

Chúng ta sẽ lưu trữ bộ đếm ở đâu? Sử dụng cơ sở dữ liệu không phải là một ý tư ờng hay do tốc độ truy cập đĩa chậm. Bộ đệm trong bộ nhớ được chọn vì nó nhanh và hỗ trợ chiến lược hết hạn theo thời gian. Ví dụ, Redis [11] là một tùy chọn phổ biến để triển khai giới hạn tốc độ. Đây là bộ lưu trữ trong bộ nhớ cung cấp hai lệnh: INCR và EXPIRE.

- INCR: Nó tăng bộ đếm được lưu trữ

lên 1.

- EXPIRE: Nó đặt thời gian chờ cho bộ đếm.

Nếu thời gian chờ hết hạn, bộ đếm sẽ tự động bị xóa.

Hình 4-12 cho thấy kiến trúc cấp cao để giới hạn tốc độ và hoạt động như sau:

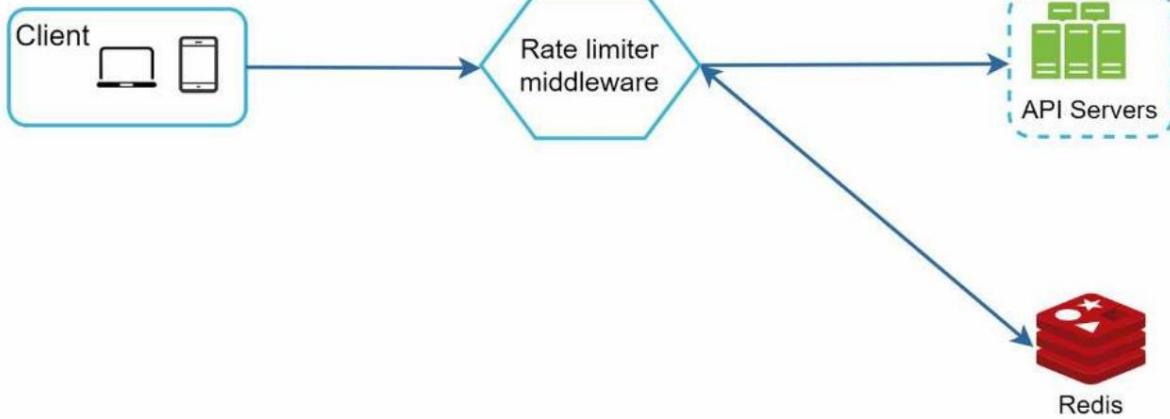


Figure 4-12

- Máy khách gửi yêu cầu đến phần mềm trung gian giới hạn tốc độ.
- Phần mềm trung gian giới hạn tốc độ lấy bộ đếm từ thùng tư ứng trong Redis và

kiểm tra xem giới hạn đã đạt đến hay chưa.

- Nếu đạt đến giới hạn, yêu cầu sẽ bị từ chối. • Nếu không đạt đến giới hạn, yêu cầu sẽ được gửi đến máy chủ API. Trong khi đó, hệ thống sẽ tăng bộ đếm và lưu lại vào Redis.

BƯỚC 3 - Thiết kế chuyên sâu

- Thiết kế cấp cao trong Hình 4-12 không trả lời các câu hỏi sau:
- Các quy tắc giới hạn tốc độ được tạo ra như thế nào? Các quy tắc được lưu trữ ở đâu?
 - Làm thế nào để xử lý các yêu cầu bị giới hạn tốc độ?

Trong phần này, trước tiên chúng ta sẽ trả lời các câu hỏi liên quan đến quy tắc giới hạn tốc độ và sau đó xem xét các chiến lược để xử lý các yêu cầu giới hạn tốc độ. Cuối cùng, chúng ta sẽ thảo luận về giới hạn tốc độ trong môi trường phân tán, thiết kế chi tiết, tối ưu hóa hiệu suất và giám sát.

Quy tắc giới hạn tốc độ

Lyft đã mã nguồn mở thành phần giới hạn tốc độ của họ [12]. Chúng ta sẽ xem xét bên trong thành phần và xem một số ví dụ về quy tắc giới hạn tốc độ:

miền: mô tả tin nhắn :

- khóa:

message_type

Giá trị: tỷ lệ tiếp

thị giới hạn:

đơn vị:

ngày yêu cầu trên mỗi đơn vị: 5

Trong ví dụ trên, hệ thống được cấu hình để cho phép tối đa 5 tin nhắn tiếp thị mỗi ngày. Sau đây là một ví dụ khác:

miền: auth mô

tả: - khóa:

auth_type

Giá trị: tỷ lệ

đăng nhập_giới

hạn: đơn vị:

phút yêu cầu_trên_một_đơn vị: 5

Quy tắc này cho thấy khách hàng không được phép đăng nhập quá 5 lần trong 1 phút. Quy tắc thường được viết trong tệp cấu hình và lưu trên đĩa.

Vượt quá giới hạn tốc độ Trong

trường hợp yêu cầu bị giới hạn tốc độ, API trả về mã phản hồi HTTP 429 (quá nhiều yêu cầu) cho máy khách. Tùy thuộc vào các trường hợp sử dụng, chúng tôi có thể xếp hàng các yêu cầu bị giới hạn tốc độ để xử lý sau. Ví dụ: nếu một số đơn hàng bị giới hạn tốc độ do hệ thống quá tải, chúng tôi có thể giữ các đơn hàng đó để xử lý sau.

Tiêu đề giới hạn tốc độ

Làm thế nào để máy khách biết được liệu nó có bị hạn chế không? Và làm thế nào để máy khách biết được số lượng yêu cầu còn lại được phép trước khi bị hạn chế? Câu trả lời nằm ở tiêu đề phản hồi HTTP. Bộ giới hạn tốc độ trả về các tiêu đề HTTP sau cho máy khách:

X-Ratelimit-Remaining: Số lượng yêu cầu còn lại được phép trong cửa sổ.

X-Ratelimit-Limit: Chỉ ra số lượng cuộc gọi mà khách hàng có thể thực hiện trong mỗi khung thời gian.

X-Ratelimit-Retry-After: Số giây phải chờ cho đến khi bạn có thể gửi yêu cầu lại mà không bị giới hạn.

Khi người dùng gửi quá nhiều yêu cầu, lỗi 429 quá nhiều yêu cầu và X-Ratelimit-

Tiêu đề Retry-After được trả về cho máy khách.

Thiết kế chi tiết Hình

4-13 trình bày thiết kế chi tiết của hệ thống.

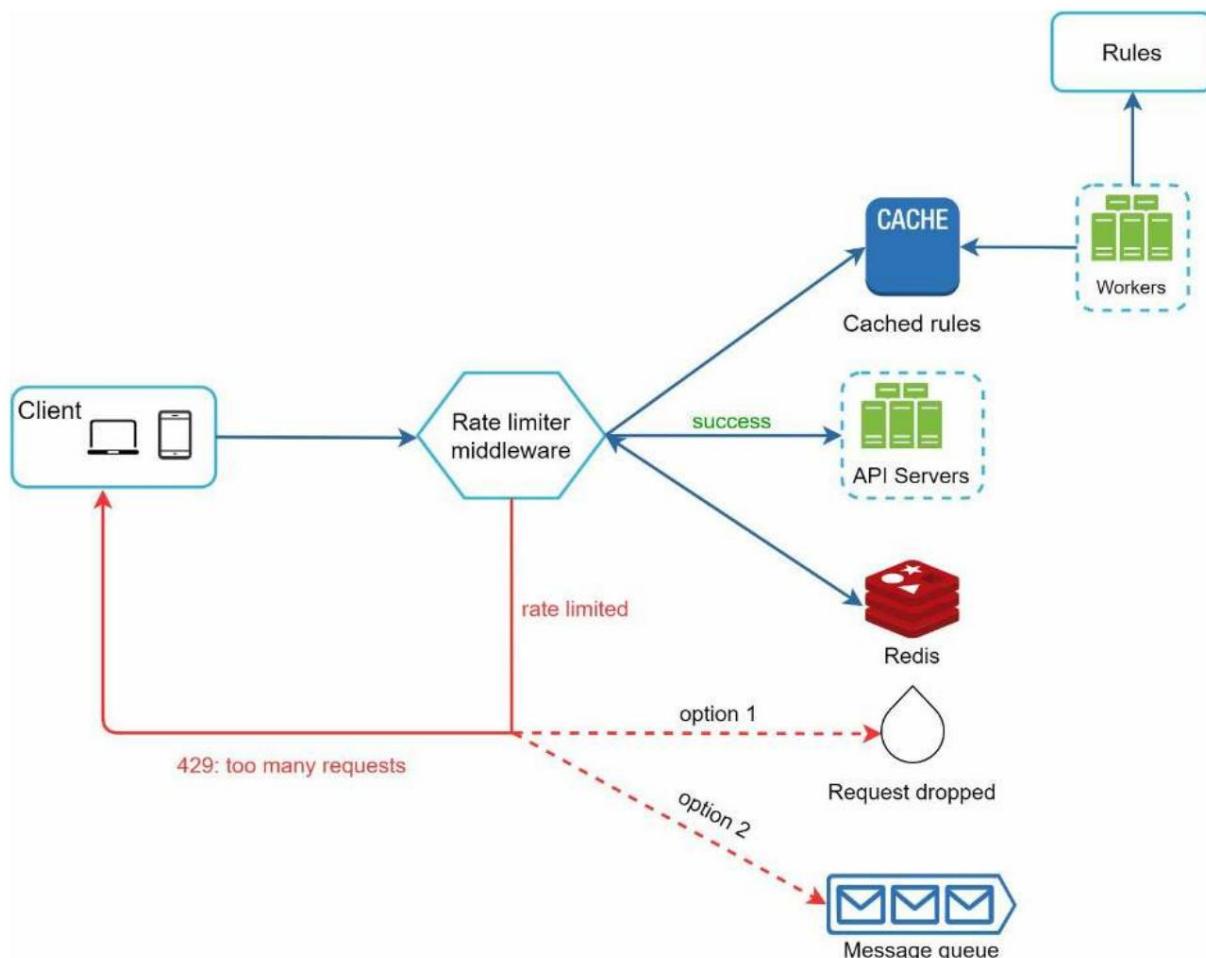


Figure 4-13

- Các quy tắc được lưu trữ trên đĩa. Người làm việc thường xuyên lấy các quy tắc từ đĩa và lưu trữ chúng trong bộ nhớ đệm.
- Khi máy khách gửi yêu cầu đến máy chủ, yêu cầu đó sẽ được gửi đến phần mềm trung gian giới hạn tốc độ trước.
- Phần mềm trung gian giới hạn tốc độ tải các quy tắc từ bộ nhớ đệm. Nó lấy các bộ đếm và dấu thời gian yêu cầu cuối cùng từ bộ nhớ đệm Redis. Dựa trên phản hồi, bộ giới hạn tốc độ quyết định:
 - nếu yêu cầu không bị giới hạn tốc độ, nó sẽ được chuyển tiếp đến máy chủ API.
 - nếu yêu cầu bị giới hạn tốc độ, bộ giới hạn tốc độ trả về lỗi 429 quá nhiều yêu cầu cho máy khách. Trong đó, yêu cầu sẽ bị hủy hoặc chuyển tiếp đến hàng đợi.

Bộ giới hạn tốc độ trong môi trường phân tán

Xây dựng một bộ giới hạn tốc độ hoạt động trong một môi trường máy chủ duy nhất không khó. Tuy nhiên, việc mở rộng hệ thống để hỗ trợ nhiều máy chủ và luồng đồng thời lại là một câu chuyện khác.

Có hai thách thức:

- Tình trạng chạy đua

- Vấn đề đồng bộ hóa

Tình trạng cuộc đua

Như đã thảo luận trước đó, bộ giới hạn tốc độ hoạt động như sau ở cấp độ cao:

- Đọc giá trị bộ đếm từ Redis.
- Kiểm tra xem (`counter + 1`) có vượt quá ngưỡng không.
- Nếu không, hãy tăng giá trị counter lên 1 trong Redis.

Tình trạng chạy đua có thể xảy ra trong môi trường có nhiều sự kiện xảy ra đồng thời như thể hiện trong Hình 4-14.

Original counter value: 3

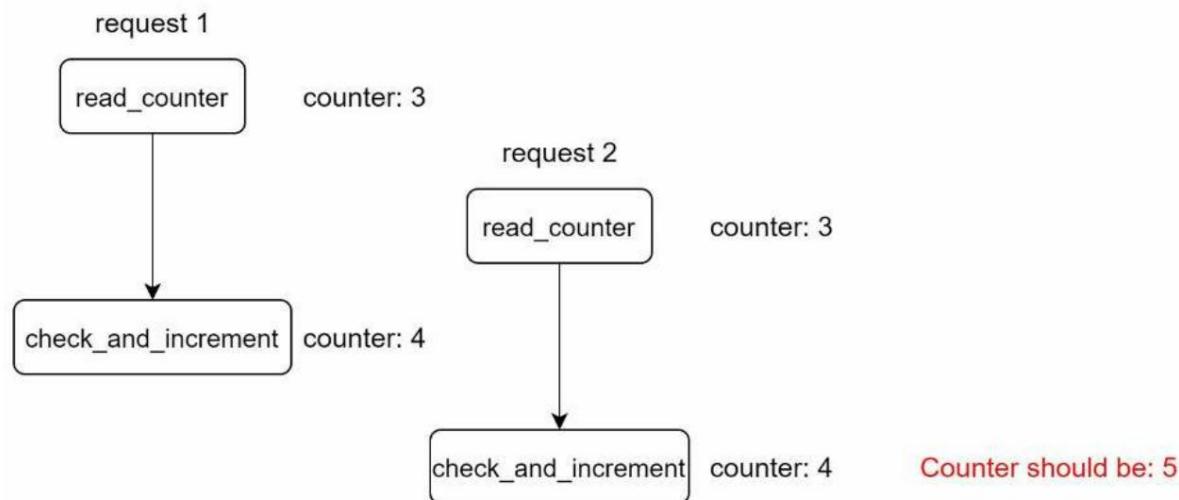


Figure 4-14

Giả sử giá trị bộ đếm trong Redis là 3. Nếu hai yêu cầu đồng thời đọc giá trị bộ đếm trước khi một trong hai yêu cầu ghi lại giá trị, mỗi yêu cầu sẽ tăng bộ đếm lên một và ghi lại mà không kiểm tra luồng kia. Cả hai yêu cầu (luồng) đều tin rằng chúng có giá trị bộ đếm chính xác là 4. Tuy nhiên, giá trị bộ đếm chính xác phải là 5.

Khóa là giải pháp rõ ràng nhất để giải quyết tình trạng chạy đua. Tuy nhiên, khóa sẽ làm chậm đáng kể hệ thống. Hai chiến lược thường được sử dụng để giải quyết vấn đề: tập lệnh Lua [13] và cấu trúc dữ liệu tập hợp được sắp xếp trong Redis [8]. Đối với độc giả quan tâm đến các chiến lược này, hãy tham khảo các tài liệu tham khảo tư duy ứng [8] [13].

Vấn đề đồng bộ hóa

Đồng bộ hóa là một yếu tố quan trọng khác cần xem xét trong môi trường phân tán. Để hỗ trợ hàng triệu người dùng, một máy chủ giới hạn tốc độ có thể không đủ để xử lý lưu lượng truy cập.

Khi sử dụng nhiều máy chủ giới hạn tốc độ, cần phải đồng bộ hóa. Ví dụ, ở phía bên trái của Hình 4-15, máy khách 1 gửi yêu cầu đến máy giới hạn tốc độ 1 và máy khách 2 gửi yêu cầu đến

bộ giới hạn tốc độ 2. Vì tầng web không có trạng thái, nên máy khách có thể gửi yêu cầu đến một bộ giới hạn tốc độ khác như được hiển thị ở phía bên phải của Hình 4-15. Nếu không có sự đồng bộ hóa nào xảy ra, bộ giới hạn tốc độ 1 không chứa bất kỳ dữ liệu nào về máy khách 2. Do đó, bộ giới hạn tốc độ không thể hoạt động bình thường.

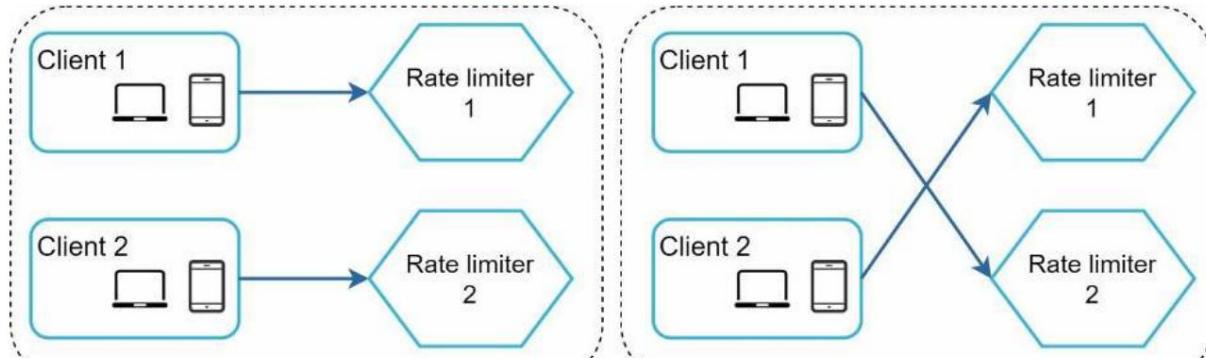


Figure 4-15

Một giải pháp khả thi là sử dụng các phiên cố định cho phép máy khách gửi lưu lượng đến cùng một bộ giới hạn tốc độ. Giải pháp này không được khuyến khích vì nó không có khả năng mở rộng cũng như không linh hoạt. Một cách tiếp cận tốt hơn là sử dụng các kho dữ liệu tập trung như Redis. Thiết kế được thể hiện trong Hình 4-16.

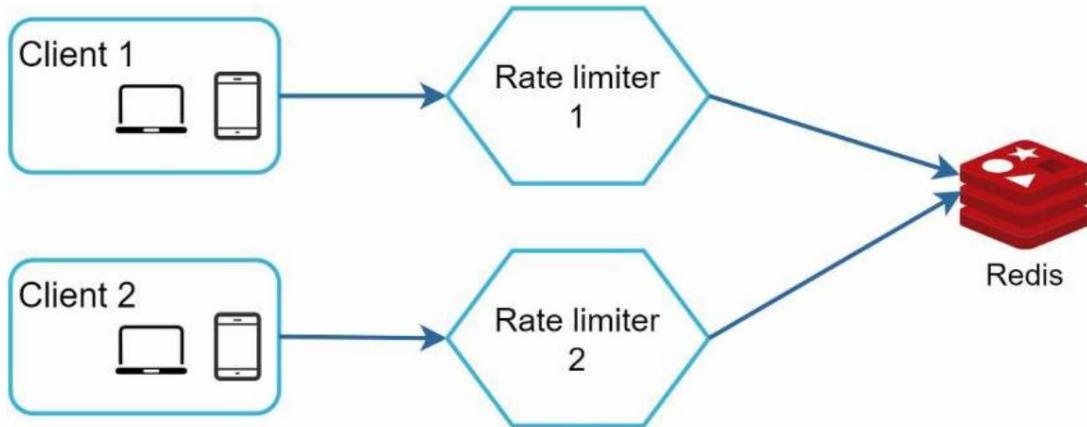


Figure 4-16

Tối ưu hóa hiệu suất

Hiệu suất là một chủ đề phổ biến trong các cuộc phỏng vấn thiết kế hệ thống. Chúng tôi sẽ đề cập đến hai lĩnh vực cần cải thiện.

Đầu tiên, thiết lập nhiều trung tâm dữ liệu là rất quan trọng đối với bộ giới hạn tốc độ vì độ trễ cao đối với người dùng ở xa trung tâm dữ liệu. Hầu hết các nhà cung cấp dịch vụ đám mây đều xây dựng nhiều vị trí máy chủ bên trên toàn thế giới. Ví dụ, tính đến ngày 20/5/2020, Cloudflare có 194 máy chủ bên phân bố theo địa lý [14]. Lưu lượng được tự động định tuyến đến máy chủ bên gần nhất để giảm độ trễ.



Figure 4-17 (source: [10])

Thứ hai, đồng bộ hóa dữ liệu với mô hình nhất quán cuối cùng. Nếu bạn không rõ về mô hình nhất quán cuối cùng, hãy tham khảo phần “Tính nhất quán” trong “Chương 6: Thiết kế kho lưu trữ khóa giá trị”.

Giám sát Sau khi

bộ giới hạn tốc độ được đưa vào sử dụng, điều quan trọng là phải thu thập dữ liệu phân tích để kiểm tra xem bộ giới hạn tốc độ có hiệu quả hay không. Trước hết, chúng tôi muốn đảm bảo:

- Thuật toán giới hạn tốc độ có hiệu quả. • Các quy tắc giới hạn tốc độ có hiệu quả.

Ví dụ, nếu quy tắc giới hạn tỷ lệ quá nghiêm ngặt, nhiều yêu cầu hợp lệ sẽ bị loại bỏ. Trong trường hợp này, chúng tôi muốn nới lỏng các quy tắc một chút. Trong một ví dụ khác, chúng tôi nhận thấy bộ giới hạn tỷ lệ của mình trở nên không hiệu quả khi có sự gia tăng đột ngột về lưu lượng truy cập như bán hàng chớp nhoáng. Trong trường hợp này, chúng tôi có thể thay thế thuật toán để hỗ trợ lưu lượng truy cập bùng nổ. Token bucket phù hợp ở đây.

Bài 4 - Kết thúc Trong

chương này, chúng ta đã thảo luận về các thuật toán giới hạn tốc độ khác nhau và ưu/nhược điểm của chúng.

Các thuật toán được thảo luận bao

gồm: • Token bucket

• Xô bị rò rỉ • Cửa

số cố định

• Cửa số trự ợt log • Cửa

số trự ợt counter

Sau đó, chúng tôi thảo luận về kiến trúc hệ thống, bộ giới hạn tốc độ trong môi trường phân tán, tối ưu hóa hiệu suất và giám sát. Tương tự như bất kỳ câu hỏi phòng vấn thiết kế hệ thống nào, có những điểm thảo luận bổ sung mà bạn có thể đề cập nếu thời gian cho phép:

- Giới hạn tốc độ cứng so với mềm.

Cứng: Số lượng yêu cầu không được vượt quá ngưỡng. • Mềm: Yêu cầu có thể vượt quá ngưỡng trong một thời gian ngắn.

- Giới hạn tốc độ ở các cấp độ khác nhau. Trong chương này, chúng tôi chỉ nói về giới hạn tốc độ ở cấp độ ứng dụng (HTTP: lớp 7). Có thể áp dụng giới hạn tốc độ ở các lớp khác. Ví dụ, bạn có thể áp dụng giới hạn tốc độ theo địa chỉ IP bằng cách sử dụng Iptables [15] (IP: lớp 3).

Lưu ý: Mô hình kết nối hệ thống mở (mô hình OSI) có 7 lớp [16]: Lớp 1: Lớp vật lý, Lớp 2: Lớp liên kết dữ liệu, Lớp 3: Lớp mạng, Lớp 4: Lớp vận chuyển, Lớp 5: Lớp phiên, Lớp 6: Lớp trình bày, Lớp 7: Lớp ứng dụng.

- Tránh bị giới hạn tốc độ. Thiết kế máy khách của bạn với các biện pháp tốt nhất:

Sử dụng bộ đệm máy khách để tránh thực hiện các cuộc gọi API thường xuyên.

Hiểu giới hạn và không gửi quá nhiều yêu cầu trong một khung thời gian ngắn.

- Bao gồm mã để bắt các ngoại lệ hoặc lỗi để máy khách của bạn có thể phục hồi bình thường sau các ngoại lệ.
- Thêm đủ thời gian lùi lại để thử lại logic.

Xin chúc mừng vì đã đi đến đây! Đây giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

[1] Chiến lược và kỹ thuật giới hạn tỷ lệ: <https://cloud.google.com/solutions/rate-limiting-strategic-techniques>

[2] Giới hạn tỷ lệ Twitter: <https://developer.twitter.com/en/docs/basics/rate-limits>

[3] Giới hạn sử dụng Google docs: <https://developers.google.com/docs/api/limits>

[4] Các dịch vụ siêu nhỏ của IBM: <https://www.ibm.com/cloud/learn/microservices>

[5] Yêu cầu API điều chỉnh để có thông lượng tốt hơn:

<https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html>

[6] Bộ giới hạn tốc độ Stripe: <https://stripe.com/blog/rate-limiters>

[7] Giới hạn tỷ lệ API quản trị REST của Shopify: <https://help.shopify.com/en/api/reference/rest-admin-api-rate-limits>

[8] Giới hạn tỷ lệ tốt hơn với Redis Sorted Sets:

<https://engineering.classdojo.com/blog/2015/02/06/rolling-rate-limiter/>

[9] Thiết kế hệ thống – Bộ giới hạn tốc độ và Mô hình dữ liệu:

<https://medium.com/@saisandeepmopuri/system-design-rate-limiter-and-data-modelling-9304b0d18250>

[10] Cách chúng tôi xây dựng khả năng giới hạn tốc độ có thể mở rộng tới hàng triệu miền:

<https://blog.cloudflare.com/counting-things-a-lot-of-different-things/>

[11] Trang web Redis: <https://redis.io/>

[12] Giới hạn giá cước Lyft: <https://github.com/lyft/ratelimit>

[13] Mở rộng API của bạn bằng bộ giới hạn tốc độ:

<https://gist.github.com/ptarjan/e38f45f2dfe601419ca3af937fff574d#request-rate-limiter>

[14] Điện toán biên là gì : <https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing/>

[15] Yêu cầu giới hạn tốc độ với Iptables: <https://blog.programster.org/rate-limit-requests-with-iptables>

[16] Mô hình OSI: https://en.wikipedia.org/wiki/OSI_model#Layer_architecture

CHƯƠNG 5: THIẾT KẾ BASHING NHẤT QUÁN

Để đạt được khả năng mở rộng theo chiều ngang, điều quan trọng là phân phối các yêu cầu/dữ liệu một cách hiệu quả và đồng đều trên các máy chủ. Băm nhất quán là một kỹ thuật thường được sử dụng để đạt được mục tiêu này. Như ng trước tiên, chúng ta hãy xem xét sâu hơn vấn đề này.

Vấn đề lắp lại

Nếu bạn có n máy chủ bộ nhớ đệm, một cách phổ biến để cân bằng tải là sử dụng phương pháp băm sau:

```
serverIndex = hash(key) % N, trong đó N là kích thước của nhóm máy chủ.
```

Chúng ta hãy sử dụng một ví dụ để minh họa cách thức hoạt động. Như được hiển thị trong Bảng 5-1, chúng ta có 4 máy chủ và 8 khóa chuỗi với hàm băm của chúng.

key	hash	hash % 4
key0	18358617	1
key1	26143584	0
key2	18131146	2
key3	35863496	0
key4	34085809	1
key5	27581703	3
key6	38164978	2
key7	22530351	3

Table 5-1

Để lấy máy chủ lưu trữ khóa, chúng ta thực hiện thao tác mô-đun $f(key) \% 4$. Ví dụ, $hash(key0) \% 4 = 1$ có nghĩa là máy khách phải liên hệ với máy chủ 1 để lấy dữ liệu được lưu trong bộ nhớ đệm.

Hình 5-1 cho thấy sự phân bổ khóa dựa trên Bảng 5-1.

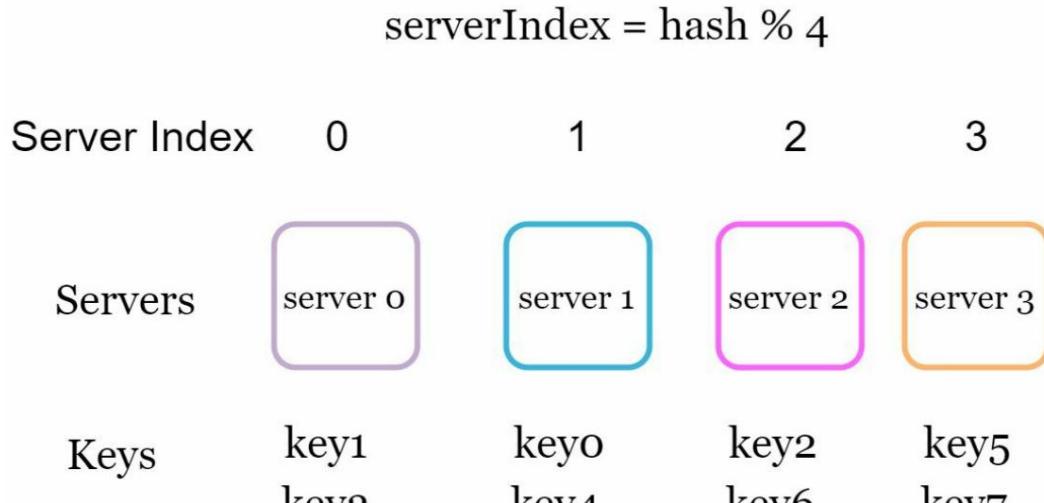


Figure 5-1

Cách tiếp cận này hoạt động tốt khi kích thước của nhóm máy chủ được cố định và phân phối dữ liệu là đồng đều. Tuy nhiên, vấn đề phát sinh khi thêm máy chủ mới hoặc xóa máy chủ hiện có. Ví dụ, nếu máy chủ 1 ngoại tuyến, kích thước của nhóm máy chủ trở thành 3. Sử dụng cùng một hàm băm, chúng ta có cùng giá trị băm cho một khóa. Nhưng áp dụng phép toán modulo cho chúng ta các chỉ mục máy chủ khác nhau vì số lượng máy chủ giảm đi 1. Chúng ta có được kết quả như thể hiện trong Bảng 5-2 bằng cách áp dụng hàm băm % 3:

key	Hash	hash % 3
key0	18358617	0
key1	26143584	0
key2	18131146	1
key3	35863496	2
key4	34085809	1
key5	27581703	0
key6	38164978	1
key7	22530351	0

Table 5-2

Hình 5-2 hiển thị phân phối khóa mới dựa trên Bảng 5-2.

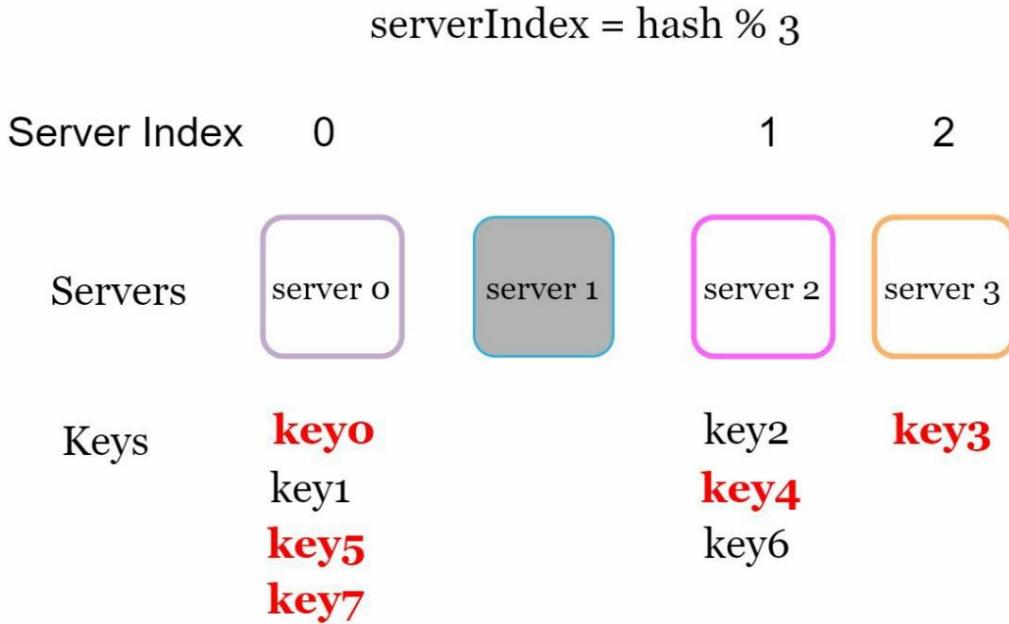


Figure 5-2

Như thể hiện trong Hình 5-2, hầu hết các khóa được phân phối lại, không chỉ các khóa ban đầu được lưu trữ trong máy chủ ngoại tuyến (máy chủ 1). Điều này có nghĩa là khi máy chủ 1 ngoại tuyến, hầu hết các máy khách bộ nhớ đệm sẽ kết nối đến các máy chủ sai để lấy dữ liệu. Điều này gây ra một cơ n bão lỗi bộ nhớ đệm. Bao nhất quán là một kỹ thuật hiệu quả để giảm thiểu vấn đề này.

Băm nhất quán Trích

dẫn từ Wikipedia: "Băm nhất quán là một loại băm đặc biệt sao cho khi một bảng băm được thay đổi kích thước và sử dụng băm nhất quán, chỉ cần ánh xạ lại trung bình k/n khóa, trong đó k là số khóa và n là số khe. Ngược lại, trong hầu hết các bảng băm truyền thống, việc thay đổi số khe mang khiến gần như tất cả các khóa được ánh xạ lại [1]".

Không gian băm và vòng băm Bay

giờ chúng ta đã hiểu định nghĩa của băm nhất quán, hãy cùng tìm hiểu cách thức hoạt động của nó. Giả sử SHA-1 được sử dụng làm hàm băm f và phạm vi đầu ra của hàm băm là: $x_0, x_1, x_2, \dots, x_n$. Trong mật mã học, không gian băm của SHA-1 chạy từ 0 đến $2^{160} - 1$. Điều đó có nghĩa là x_0 tương ứng với 0, x_n tương ứng với $2^{160} - 1$ và tất cả các giá trị băm khác ở giữa nằm trong khoảng từ 0 đến $2^{160} - 1$. Hình 5-3 cho thấy không gian băm.

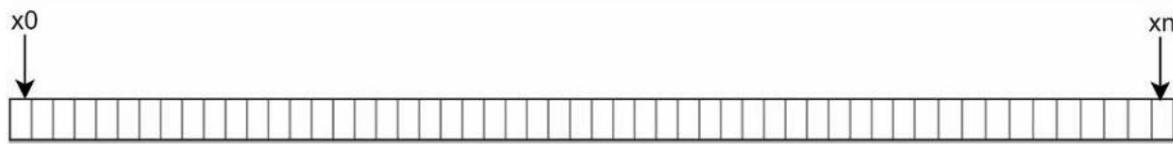


Figure 5-3

Bằng cách thu thập cả hai đầu, chúng ta có được một vòng băm như thể hiện trong Hình 5-4:

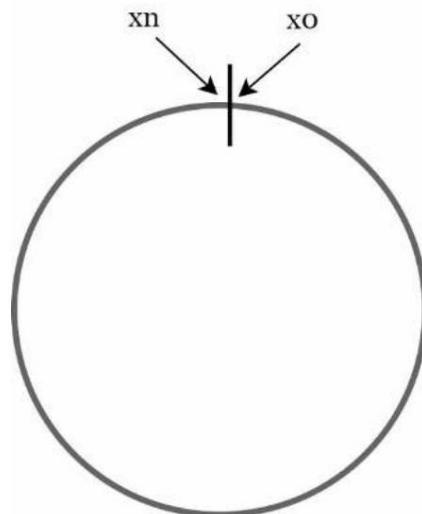


Figure 5-4

Máy chủ băm

Sử dụng cùng một hàm băm f, chúng tôi ánh xạ các máy chủ dựa trên IP hoặc tên máy chủ vào vòng. Hình 5-5 cho thấy 4 máy chủ được ánh xạ trên vòng băm.

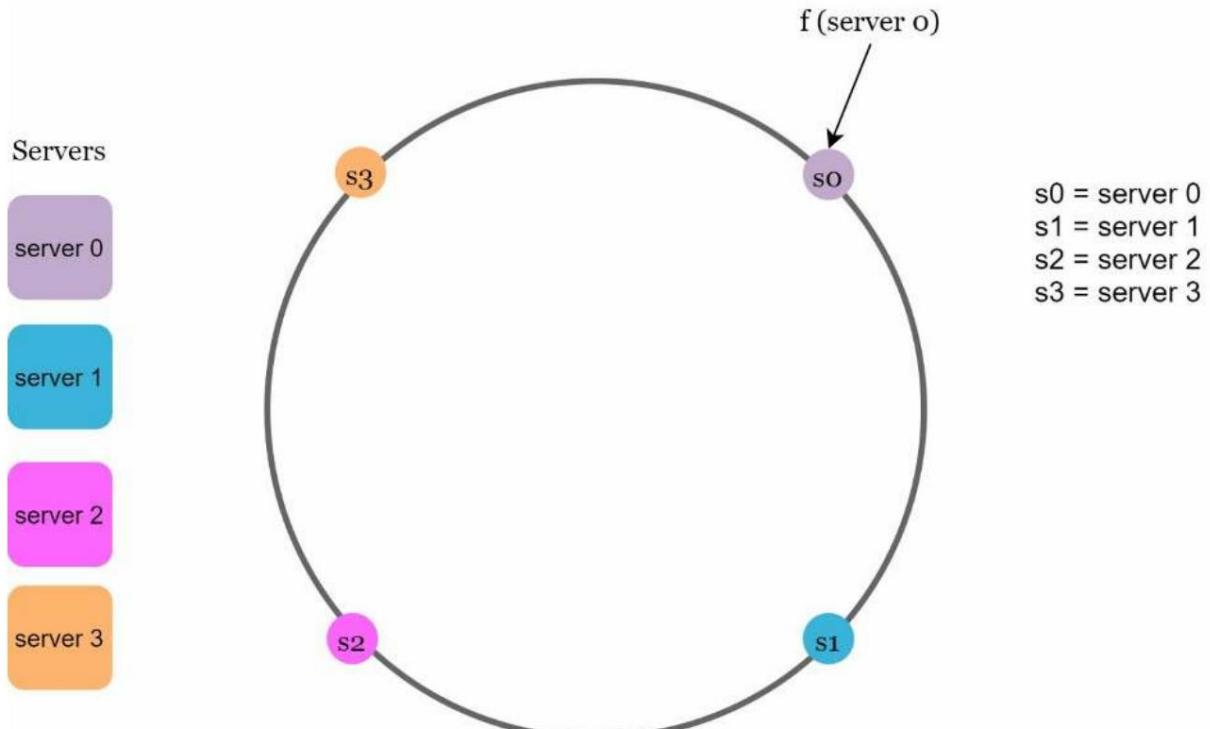


Figure 5-5

Khóa băm

Một điều đáng nói đến là hàm băm được sử dụng ở đây khác với hàm băm trong "ván đè bäm lại" và không có hoạt động mô-đun. Như thể hiện trong Hình 5-6, 4 khóa bộ nhớ đệm (key_0 , key_1 , key_2 và key_3) được băm vào vòng băm

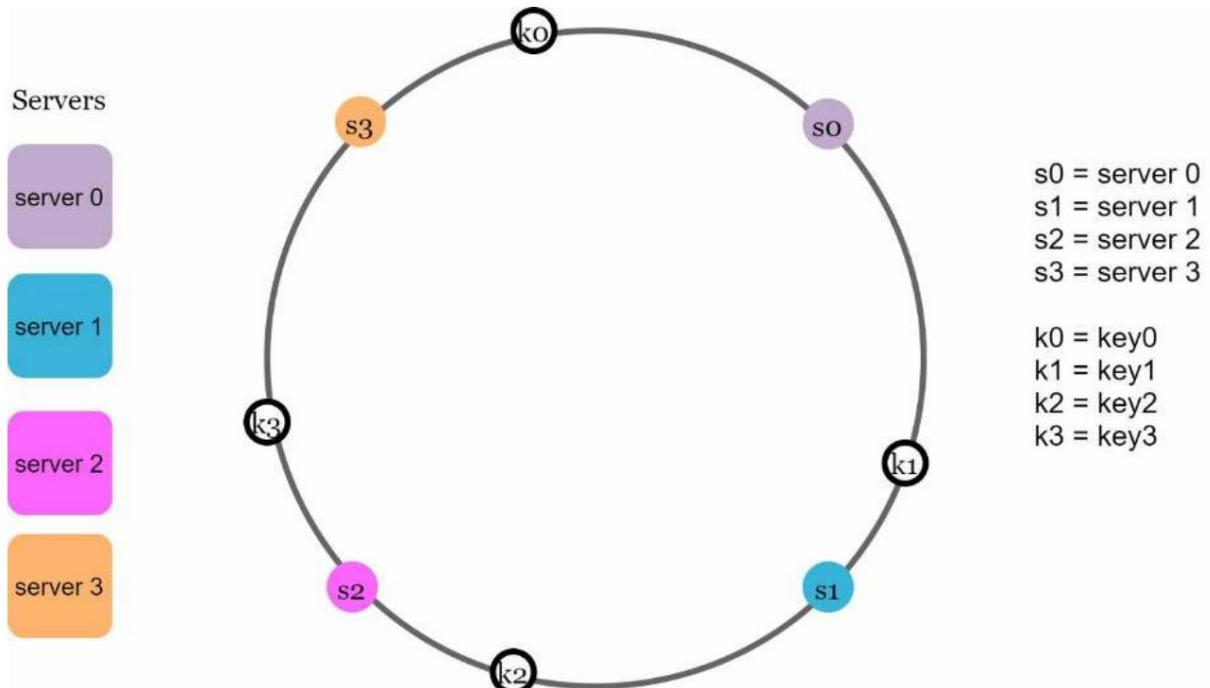


Figure 5-6

Tra cứu máy chủ

Để xác định khóa được lưu trữ trên máy chủ nào, chúng ta đi theo chiều kim đồng hồ từ vị trí khóa trên vòng cho đến khi tìm thấy máy chủ. Hình 5-7 giải thích quá trình này. Đi theo chiều kim đồng hồ, key0 được lưu trữ trên máy chủ 0; key1 được lưu trữ trên máy chủ 1; key2 được lưu trữ trên máy chủ 2 và key3 được lưu trữ trên máy chủ 3.

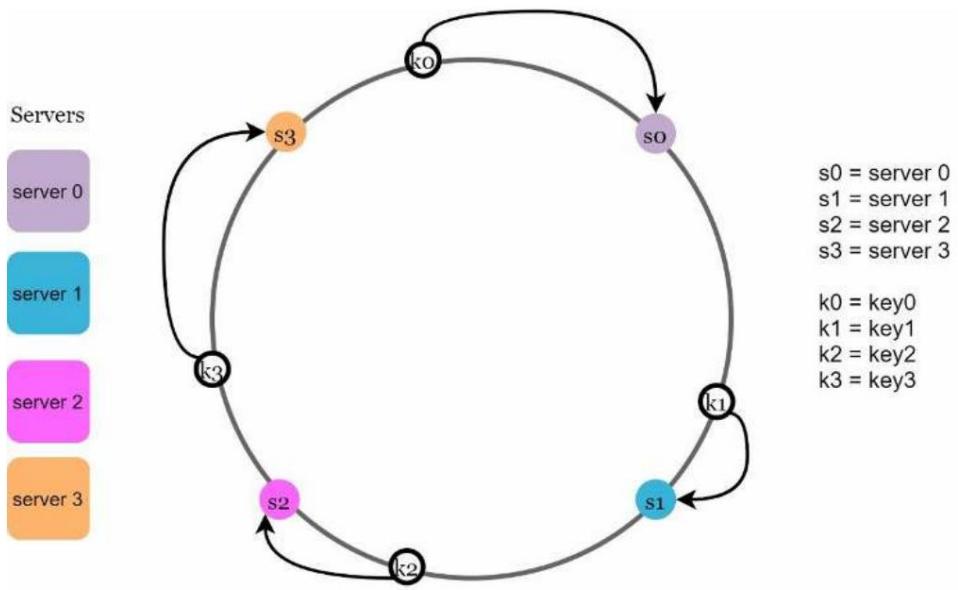


Figure 5-7

Thêm một máy chủ

Sử dụng logic được mô tả ở trên, việc thêm máy chủ mới sẽ chỉ yêu cầu phân phối lại một phần nhỏ khóa.

Trong Hình 5-8, sau khi thêm máy chủ 4 mới, chỉ cần phân phối lại key0 . k1, k2 và k3 vẫn nằm trên cùng một máy chủ. Chúng ta hãy xem xét kỹ logic. Trước khi thêm máy chủ 4 , key0 được lưu trữ trên máy chủ 0. Bây giờ, key0 sẽ được lưu trữ trên máy chủ 4 vì máy chủ 4 là máy chủ đầu tiên mà nó gặp phải khi đi theo chiều kim đồng hồ từ vị trí của key0 trên vòng. Các khóa khác không được phân phối lại dựa trên thuật toán băm nhất quán.

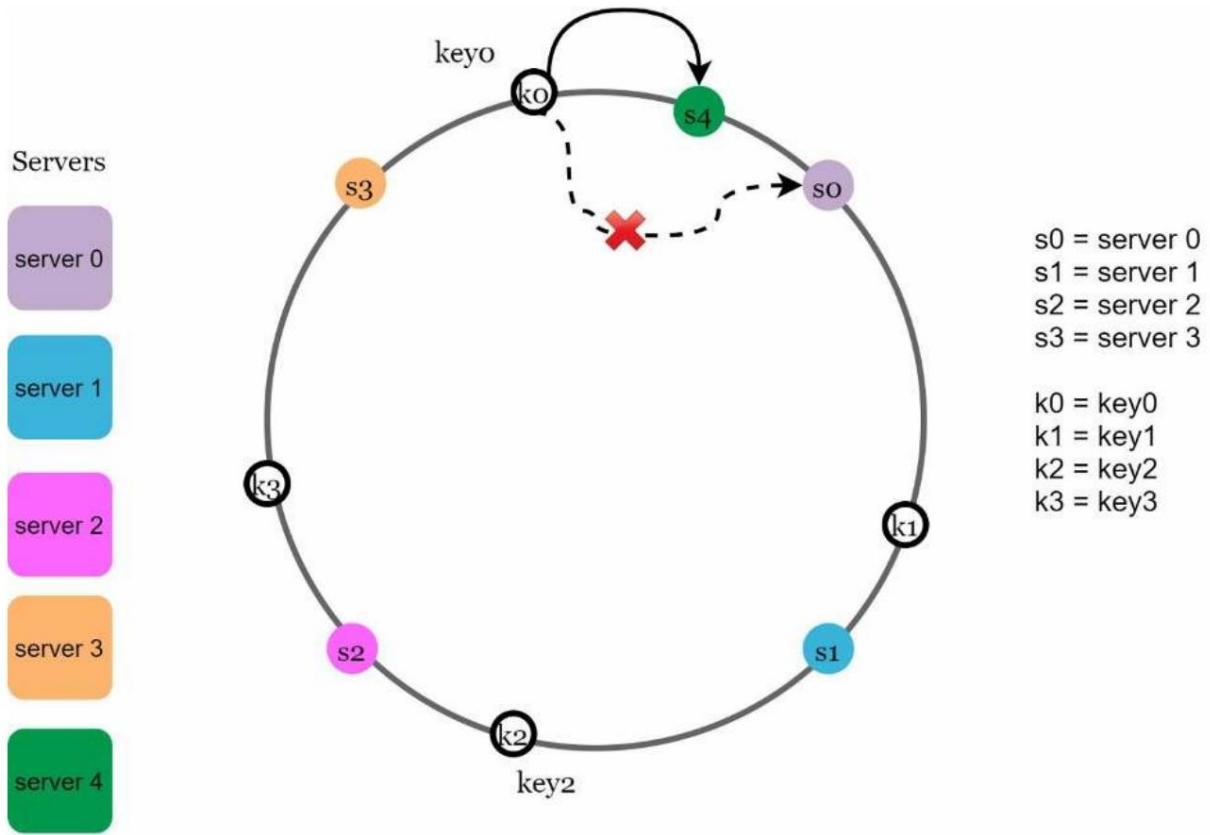


Figure 5-8

Xóa máy chủ

Khi máy chủ bị xóa, chỉ một phần nhỏ khóa cần được phân phối lại với băm nhất quán. Trong Hình 5-9, khi máy chủ 1 bị xóa, chỉ có khóa 1 phải được ánh xạ lại thành máy chủ 2.

Các phím còn lại không bị ảnh hưởng.

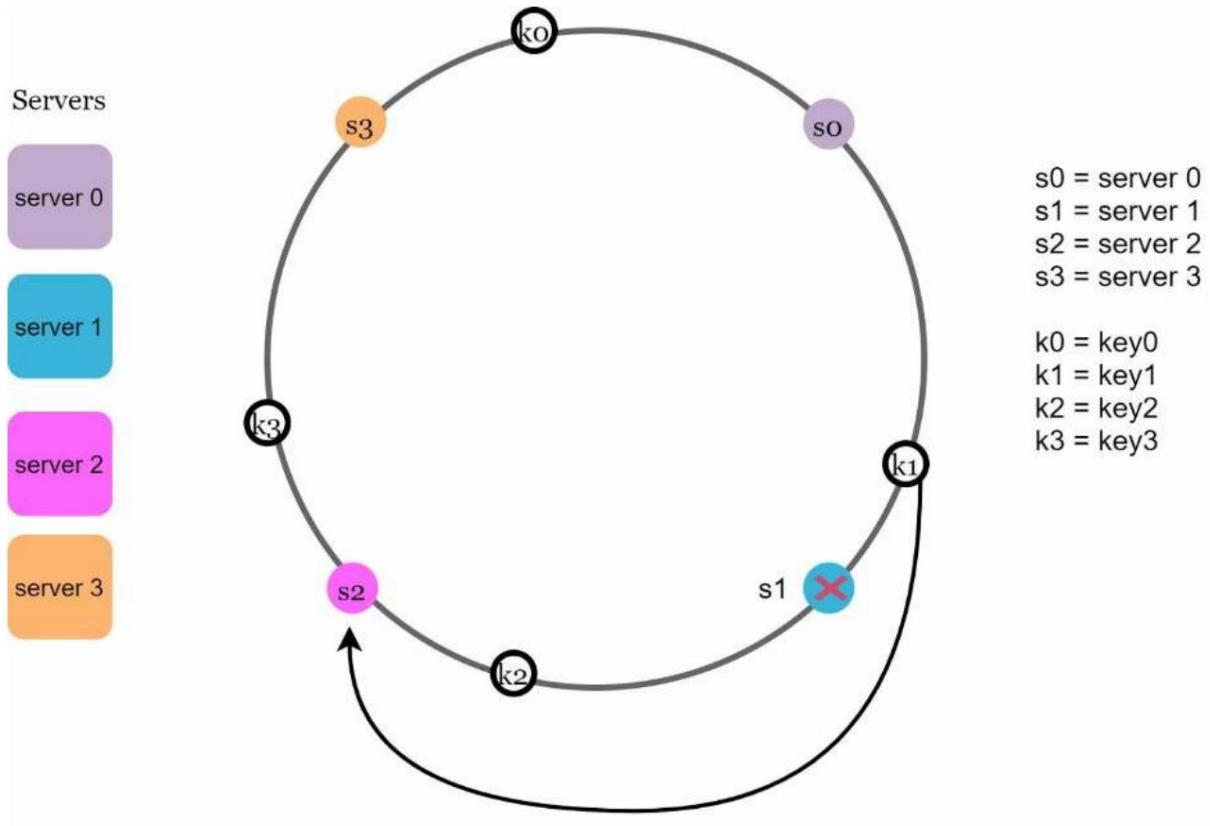


Figure 5-9

Hai vấn đề trong cách tiếp cận cơ bản

Thuật toán băm nhất quán được giới thiệu bởi Karger và cộng sự tại MIT [1]. Các bước cơ bản là:

- Ánh xạ máy chủ và khóa vào vòng bằng hàm băm phân phối đồng đều. • Để tìm ra máy chủ nào được ánh xạ khóa, hãy đi theo chiều kim đồng hồ từ vị trí khóa cho đến khi tìm thấy máy chủ đầu tiên trên vòng.

Có hai vấn đề được xác định với cách tiếp cận này. Đầu tiên, không thể giữ nguyên kích thước phân vùng trên vòng cho tất cả các máy chủ khi xem xét một máy chủ có thể được thêm vào hoặc xóa. Phân vùng là không gian băm giữa các máy chủ liền kề. Có thể kích thước của các phân vùng trên vòng được chỉ định cho mỗi máy chủ là rất nhỏ hoặc khá lớn. Trong Hình 5-10, nếu s_1 bị xóa, phân vùng của s_2 (được tô sáng bằng các mũi tên hai chiều) sẽ lớn gấp đôi phân vùng của s_0 và s_3 .

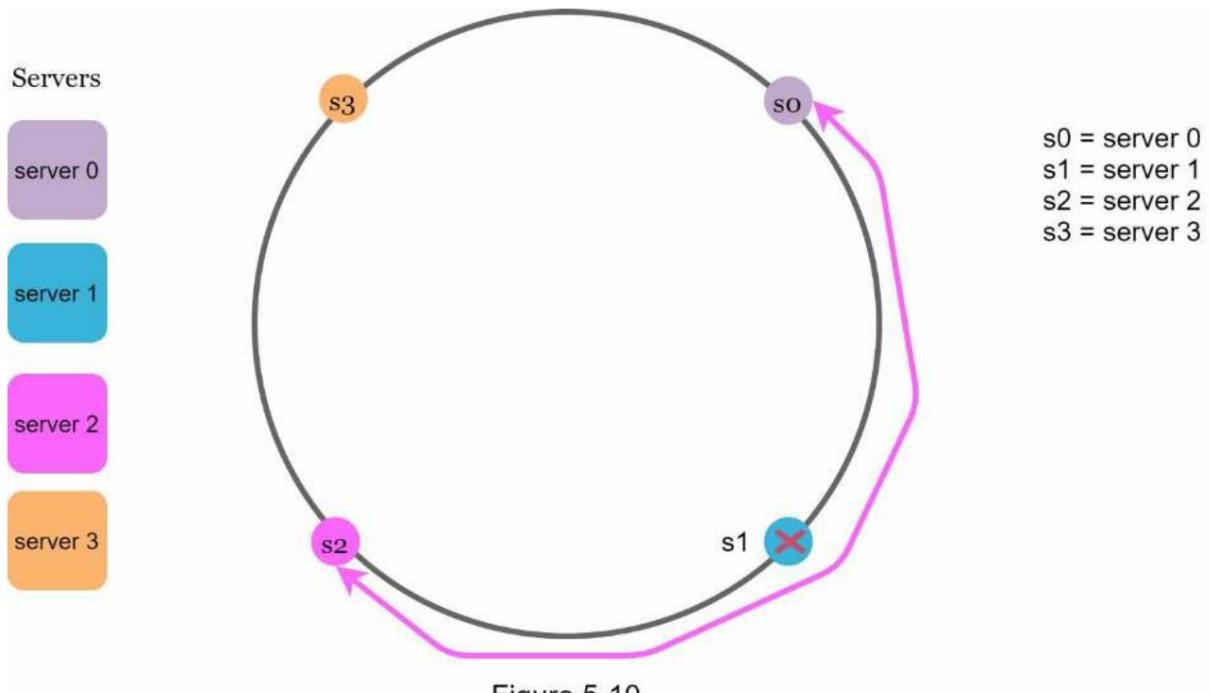


Figure 5-10

Thứ hai, có thể có sự phân phối khóa không đồng nhất trên vòng. Ví dụ, nếu máy chủ đư ợc ánh xạ tới các vị trí được liệt kê trong Hình 5-11, hầu hết các khóa đư ợc lưu trữ trên máy chủ 2. Tuy nhiên, máy chủ 1 và máy chủ 3 không có dữ liệu.

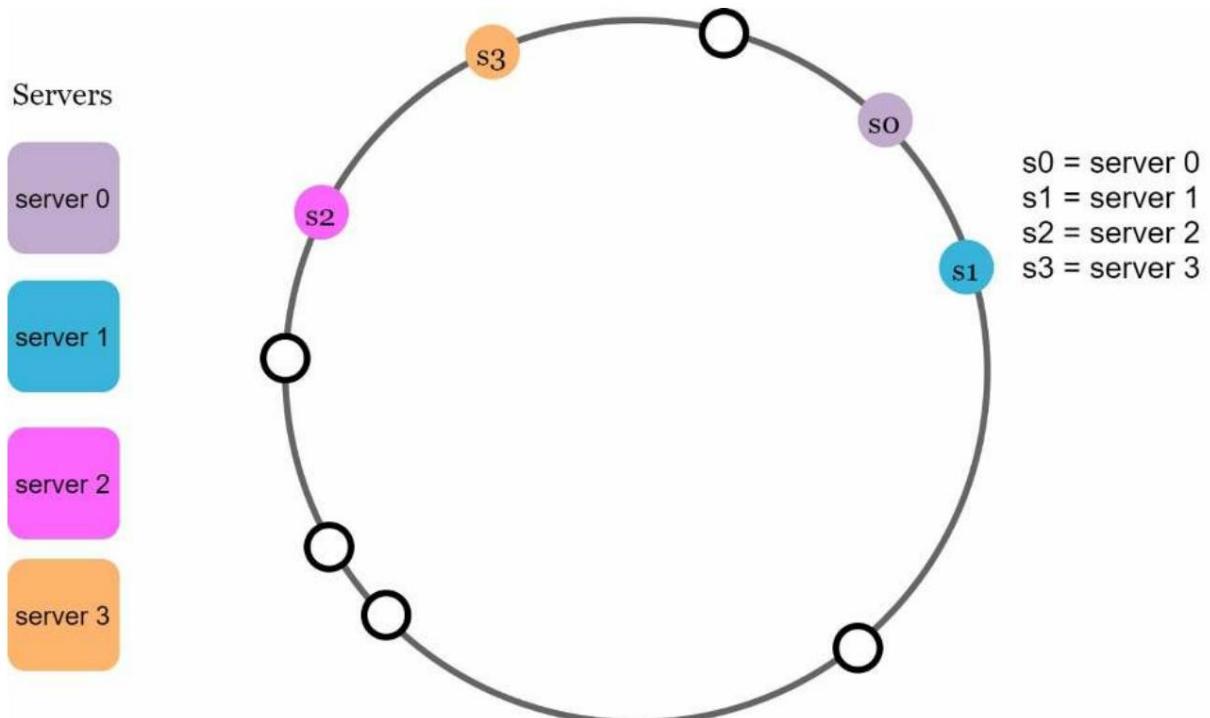


Figure 5-11

Một kỹ thuật gọi là nút ảo hoặc bản sao đư ợc sử dụng để giải quyết những vấn đề này.

Các nút ảo

Một nút ảo đê cập đến nút thực và mỗi máy chủ đư ợc biểu diễn bằng nhiều nút ảo trên vòng. Trong Hình 5-12, cả máy chủ 0 và máy chủ 1 đều có 3 nút ảo. 3 là

đư ợc lựa chọn tùy ý; và trong các hệ thống thực tế, số lượng nút ảo lớn hơn nhiều. Thay vì sử dụng s_0 , chúng ta có $s0_0$, $s0_1$ và $s0_2$ để biểu diễn máy chủ 0 trên vòng. Tương tự, $s1_0$, $s1_1$ và $s1_2$ biểu diễn máy chủ 1 trên vòng. Với các nút ảo, mỗi máy chủ chịu trách nhiệm cho nhiều phân vùng. Các phân vùng (cạnh) có nhãn $s0$ được quản lý bởi máy chủ 0. Mặt khác, các phân vùng có nhãn $s1$ được quản lý bởi máy chủ 1.

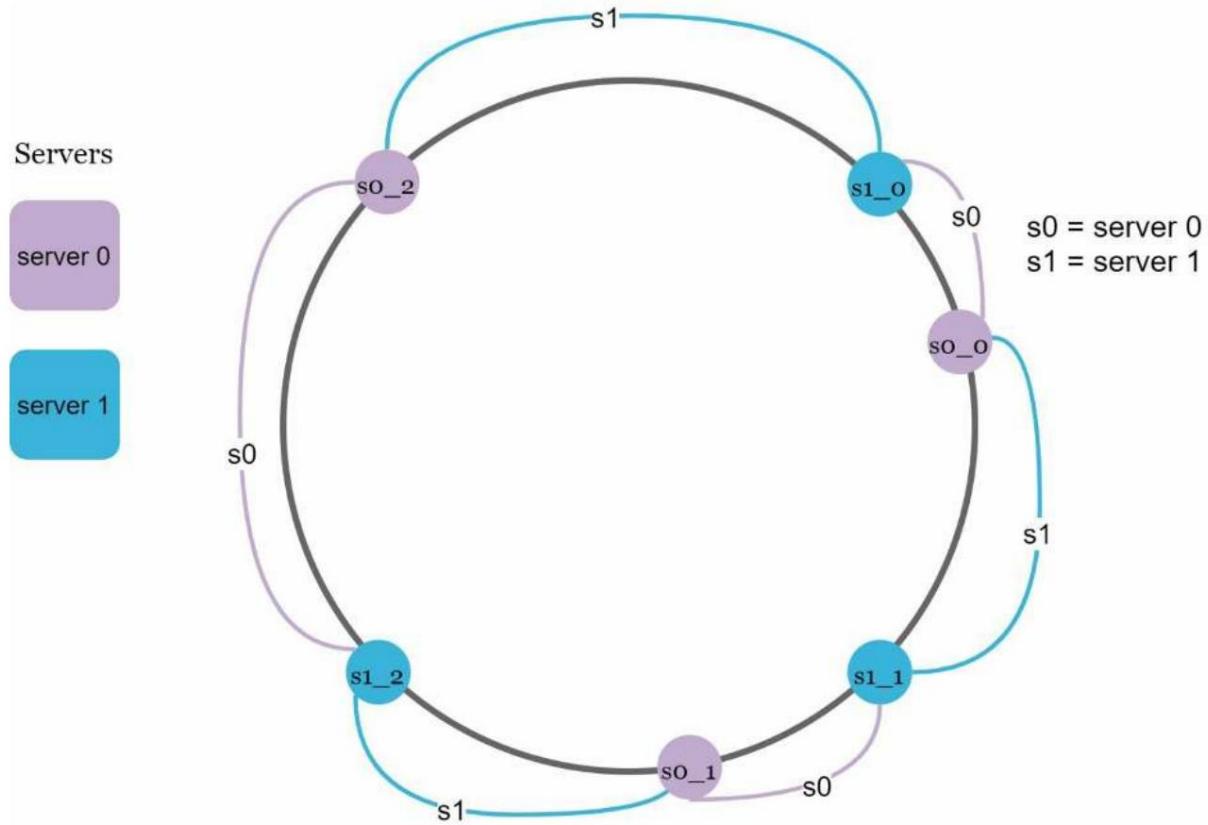


Figure 5-12

Để tìm máy chủ lưu trữ khóa, chúng ta đi theo chiều kim đồng hồ từ vị trí khóa và tìm nút ảo đầu tiên gặp phải trên vòng. Trong Hình 5-13, để tìm máy chủ lưu trữ k_0 , chúng ta đi theo chiều kim đồng hồ từ vị trí k_0 và tìm nút ảo $s1_1$, tham chiếu đến máy chủ 1.

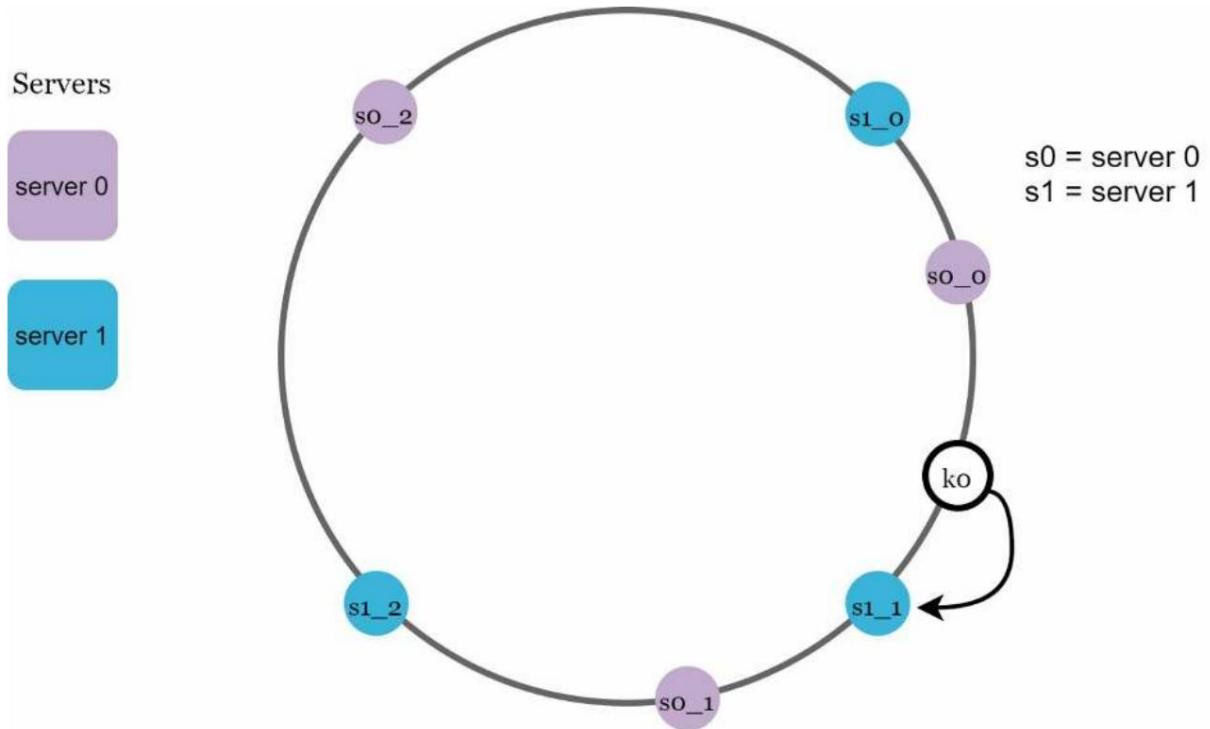


Figure 5-13

Khi số lượng nút ảo tăng lên, việc phân phối khóa sẽ trở nên cân bằng hơn.

Điều này là do độ lệch chuẩn trở nên nhỏ hơn với nhiều nút ảo hơn, dẫn đến phân phối dữ liệu cân bằng.

Độ lệch chuẩn đo lường cách dữ liệu được phân tán ra sao. Kết quả của một thí nghiệm do nghiên cứu trực tuyến [2] thực hiện cho thấy với một hoặc hai trăm nút ảo, độ lệch chuẩn nằm trong khoảng từ 5% (200 nút ảo) đến 10% (100 nút ảo) của giá trị trung bình. Độ lệch chuẩn sẽ nhỏ hơn khi chúng ta tăng số lượng nút ảo. Tuy nhiên, cần nhiều không gian hơn để lưu trữ dữ liệu về các nút ảo.

Đây là một sự đánh đổi và chúng ta có thể điều chỉnh số lượng nút ảo để phù hợp với yêu cầu hệ thống của mình.

Tìm khóa bị ảnh hưởng Khi

thêm hoặc xóa máy chủ, một phần dữ liệu cần được phân phối lại. Làm thế nào chúng ta có thể tìm phạm vi bị ảnh hưởng để phân phối lại khóa?

Trong Hình 5-14, máy chủ 4 được thêm vào vòng. Phạm vi bị ảnh hưởng bắt đầu từ s4 (nút mới được thêm) và di chuyển ngược chiều kim đồng hồ quanh vòng cho đến khi tìm thấy máy chủ (s3). Do đó, các khóa nằm giữa s3 và s4 cần được phân phối lại cho s4.

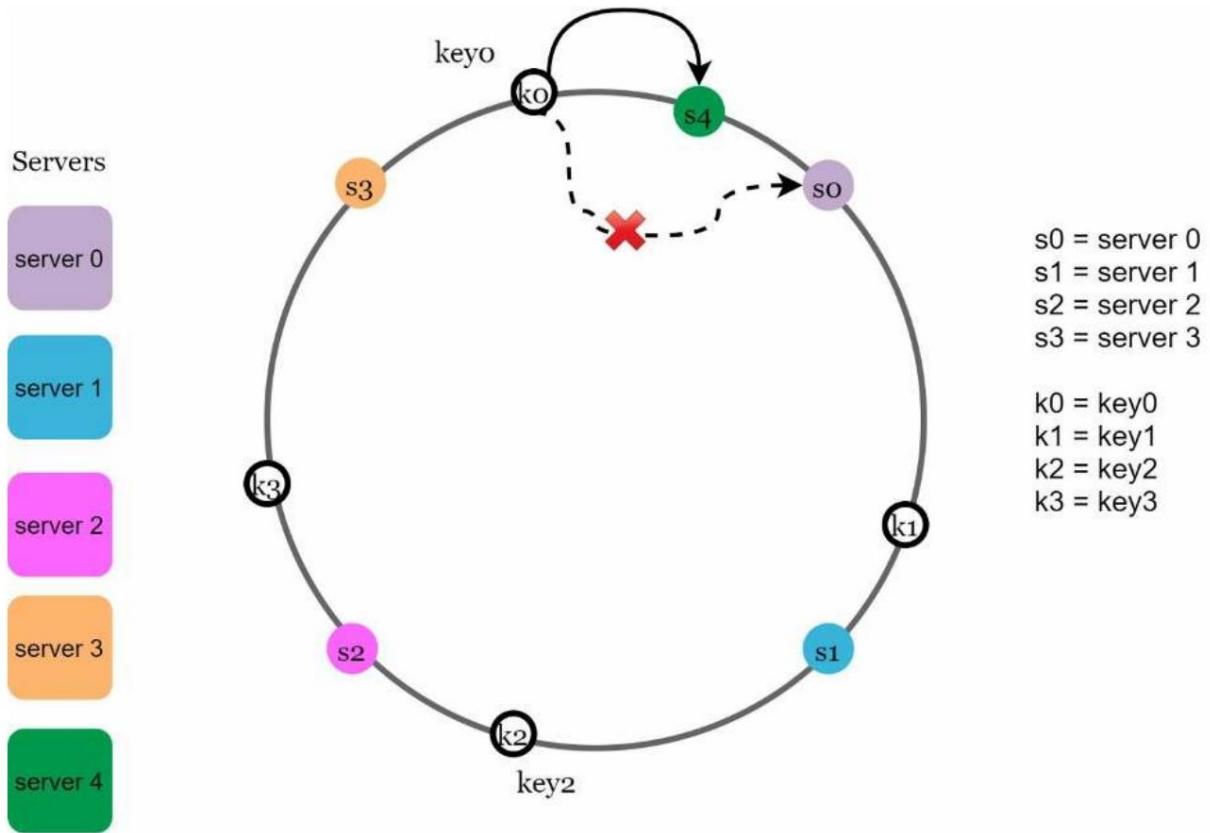
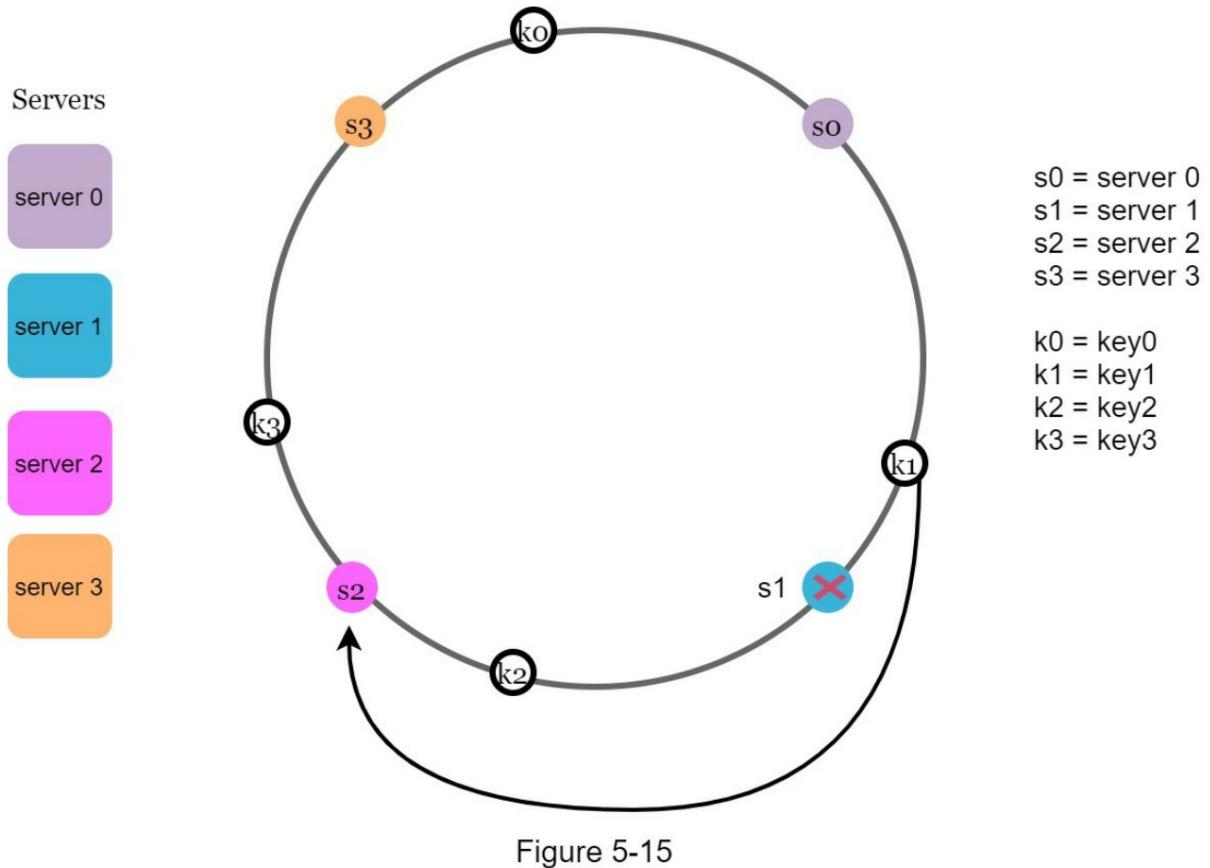


Figure 5-14

Khi một máy chủ (s₁) bị xóa như thể hiện trong Hình 5-15, phạm vi bị ảnh hưởng bắt đầu từ s₁ (nút bị xóa) và di chuyển ngược chiều kim đồng hồ quanh vòng cho đến khi tìm thấy máy chủ (s₀). Do đó, các khóa nằm giữa s₀ và s₁ phải được phân phối lại cho s₂.



Tóm tắt

Trong chương này, chúng ta đã thảo luận sâu về hàm băm nhất quán, bao gồm lý do tại sao cần có hàm băm nhất quán và cách thức hoạt động của nó. Các lợi ích của hàm băm nhất quán bao gồm:

- Các khóa thu nhỏ được phân phối lại khi máy chủ được thêm vào hoặc xóa. • Để dàng mở rộng theo chiều ngang vì dữ liệu được phân phối đều hơn. • Giảm thiểu vấn đề khóa điểm truy cập. Truy cập quá mức vào một phân đoạn cụ thể có thể gây quá tải máy chủ. Hãy thử ẩn tư ợng dữ liệu của Katy Perry, Justin Bieber và Lady Gaga đều nằm trên cùng một phân đoạn. Băm nhất quán giúp giảm thiểu vấn đề bằng cách phân phối dữ liệu đồng đều hơn.

Băm nhất quán được sử dụng rộng rãi trong các hệ thống thực tế, bao gồm một số hệ thống đáng chú ý:

- Phân vùng thành phần của cơ sở dữ liệu Dynamo của Amazon [3] • Phân vùng dữ liệu trên toàn bộ cụm trong Apache Cassandra [4] • Ứng dụng trò chuyện Discord [5] • Mạng phân phối nội dung Akamai [6] • Bộ cân bằng tải mạng Maglev [7]

Xin chúc mừng vì đã đi được đến đây! Bây giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

[1] Bấm nháy quán: https://en.wikipedia.org/wiki/Consistent_hashing [2]

Bấm nháy quán: <https://tom-e-white.com/2007/11/consistent-hashing.html> [3]

Dynamo: Kho lưu trữ khóa-giá trị có sẵn cao của Amazon:

<https://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf> [4]

Cassandra - Hệ thống lưu trữ có cấu trúc phi tập trung:

<http://www.cs.cornell.edu/Projects/ladis2009/papers/Lakshman-ladis2009.PDF> [5]

Discord đã mở rộng Elixir lên 5.000.000 người dùng đồng thời

nó như thế nào: <https://blog.discord.com/scaling-elixir-f9b8e1e7c29b> [6]

CS168: Bài giảng số 1 về Hộp công cụ thuật toán hiện đại: Giới thiệu và Bấm nháy quán: <http://theory.stanford.edu/~tim/s16/l11.pdf> [7]

Maglev: Bộ cân bằng tải mạng phần mềm nhanh và đáng tin cậy: <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/44824.pdf>

CHƯƠNG 6: THIẾT KẾ MỘT CỦA HÀNG CHÍ NH-GIÁ TRỊ

Kho lưu trữ khóa-giá trị, còn được gọi là cơ sở dữ liệu khóa-giá trị, là cơ sở dữ liệu phi quan hệ. Mỗi mã định danh duy nhất được lưu trữ dưới dạng khóa với giá trị liên quan. Cặp dữ liệu này được gọi là cặp "khóa-giá trị".

Trong một cặp khóa-giá trị, khóa phải là duy nhất và giá trị liên kết với khóa có thể được truy cập thông qua khóa. Khóa có thể là văn bản thuần túy hoặc giá trị băm. Vì lý do hiệu suất, khóa ngắn hoạt động tốt hơn. Khóa trông như thế nào? Sau đây là một số ví dụ:

- Khóa văn bản thuần túy:
"last_logged_in_at" • Khóa băm: 253DDEC4

Giá trị trong cặp khóa-giá trị có thể là chuỗi, danh sách, đối tượng, v.v. Giá trị thường được coi là đối tượng mờ trong các kho lưu trữ khóa-giá trị, chẳng hạn như Amazon dynamo [1], Memcached [2], Redis [3], v.v.

Sau đây là một đoạn dữ liệu trong kho lưu trữ khóa-giá trị:

Key	value
145	john
147	bob
160	Julia

Table 6-1

Trong chương này, bạn được yêu cầu thiết kế một kho lưu trữ khóa-giá trị hỗ trợ các hoạt động sau:

- put(key, value) // chèn "value" liên kết với "key"
- get(key) // lấy "giá trị" liên kết với "key"

Hiểu vấn đề và thiết lập phạm vi thiết kế Không có thiết kế hoàn hảo. Mỗi thiết kế đạt được sự cân bằng cụ thể liên quan đến sự đánh đổi giữa việc sử dụng đọc, ghi và bộ nhớ. Một sự đánh đổi khác phải được thực hiện là giữa tính nhất quán và tính khả dụng. Trong chương này, chúng tôi thiết kế một kho lưu trữ khóa-giá trị bao gồm các đặc điểm sau:

- Kích thước của cặp khóa-giá trị nhỏ: nhỏ hơn 10 KB.
- Khả năng lưu trữ dữ liệu lớn.
- Tính khả dụng cao: Hệ thống phản hồi nhanh, ngay cả khi xảy ra lỗi.
- Khả năng mở rộng cao: Hệ thống có thể được mở rộng để hỗ trợ bộ dữ liệu lớn.
- Tự động mở rộng: Việc thêm/xóa máy chủ phải tự động dựa trên lưu lượng truy cập.
- Tính nhất quán có thể điều chỉnh.
- Độ trễ thấp.

Lưu trữ khóa-giá trị máy chủ đơn lẻ

Việc phát triển một kho lưu trữ khóa-giá trị nằm trong một máy chủ đơn lẻ rất dễ dàng. Một cách tiếp cận trực quan là lưu trữ các cặp khóa-giá trị trong một bảng băm, bảng này lưu giữ mọi thứ trong bộ nhớ.

Mặc dù truy cập bộ nhớ nhanh, nhưng việc đưa mọi thứ vào bộ nhớ có thể là không thể do hạn chế về không gian. Có thể thực hiện hai tối ưu hóa để đưa nhiều dữ liệu hơn vào một máy chủ đơn lẻ:

- Nén dữ liệu • Chỉ

lưu trữ dữ liệu thư ờng dùng trong bộ nhớ và phần còn lại trên đĩa

Ngay cả với những tối ưu hóa này, một máy chủ duy nhất có thể đạt đến công suất của nó rất nhanh. Một kho lưu trữ khóa-giá trị phân tán là cần thiết để hỗ trợ dữ liệu lớn.

Lưu trữ khóa-giá trị phân tán

Lưu trữ khóa-giá trị phân tán còn được gọi là bảng băm phân tán, phân phối các cặp khóa-giá trị trên nhiều máy chủ. Khi thiết kế hệ thống phân tán, điều quan trọng là phải hiểu định lý CAP (Tính nhất quán, Tính khả dụng, Dung sai phân vùng).

Định lý CAP

Định lý CAP nêu rằng một hệ thống phân tán không thể đồng thời cung cấp nhiều hơn hai trong ba đảm bảo sau: tính nhất quán, tính khả dụng và khả năng chịu đựng phân vùng. Chúng ta hãy thiết lập một vài định nghĩa.

Tính nhất quán: tính nhất quán có nghĩa là tất cả máy khách đều thấy cùng một dữ liệu tại cùng một thời điểm bất kể họ kết nối tới nút nào.

Tính khả dụng: tính khả dụng có nghĩa là bất kỳ máy khách nào yêu cầu dữ liệu đều nhận được phản hồi ngay cả khi một số nút ngừng hoạt động.

Dung sai phân vùng: phân vùng chỉ ra sự gián đoạn giao tiếp giữa hai nút.

Khả năng chịu đựng phân vùng có nghĩa là hệ thống vẫn tiếp tục hoạt động bất chấp các phân vùng mạng.

Định lý CAP phát biểu rằng một trong ba tính chất phải bị hy sinh để hỗ trợ 2 trong 3 tính chất như thể hiện trong Hình 6-1.

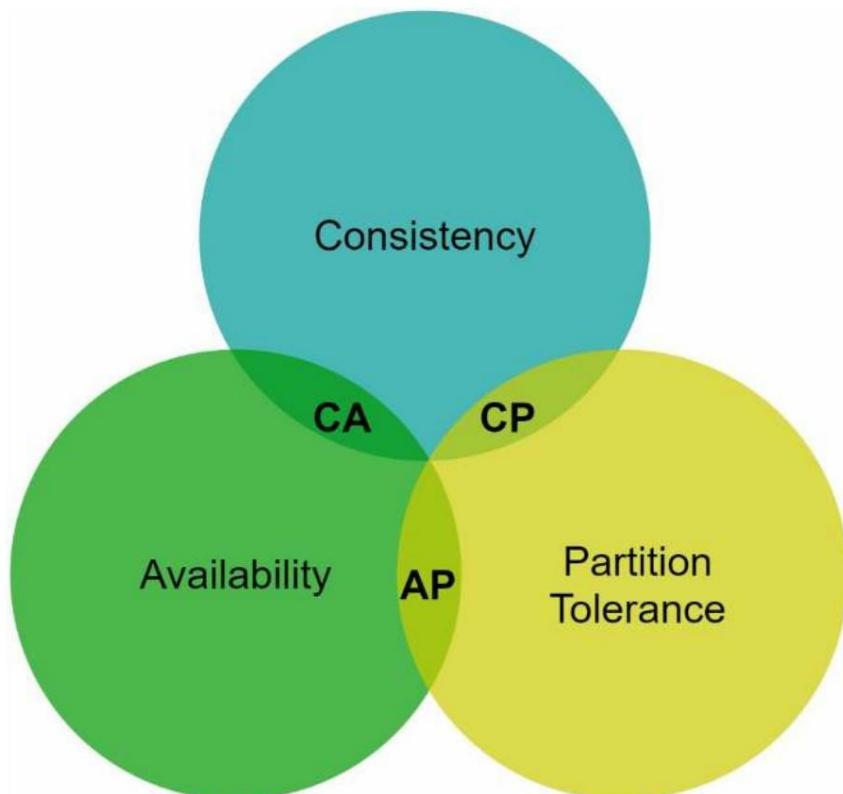


Figure 6-1

Ngày nay, các kho lưu trữ khóa-giá trị được phân loại dựa trên hai đặc điểm CAP mà chúng hỗ trợ:

Hệ thống CP (tính nhất quán và dung sai phân vùng): kho lưu trữ khóa-giá trị CP hỗ trợ tính nhất quán và dung sai phân vùng trong khi vẫn đảm bảo tính khả dụng.

Hệ thống AP (khả dụng và dung sai phân vùng): kho lưu trữ khóa-giá trị AP hỗ trợ khả dụng và dung sai phân vùng trong khi vẫn đảm bảo tính nhất quán.

Hệ thống CA (tính nhất quán và khả dụng): kho lưu trữ giá trị khóa CA hỗ trợ tính nhất quán và

tính khả dụng trong khi hy sinh khả năng chịu đựng phân vùng. Vì lõi mạng là không thể tránh khỏi, nên hệ thống phân tán phải chịu đựng được phân vùng mạng. Do đó, hệ thống CA không thể tồn tại trong các ứng dụng thực tế.

Những gì bạn đọc ở trên chủ yếu là phần định nghĩa. Để dễ hiểu hơn, chúng ta hãy xem một số ví dụ cụ thể. Trong các hệ thống phân tán, dữ liệu thường được sao chép nhiều lần. Giả sử dữ liệu được sao chép trên ba nút bắn sao, n1, n2 và n3 như thể hiện trong Hình 6-2.

Tình huống lý thuyết

Trong thế giới lý thuyết, phân vùng mạng không bao giờ xảy ra. Dữ liệu được ghi vào n1 sẽ tự động được sao chép vào n2 và n3. Cả tính nhất quán và tính khả dụng đều đạt được.

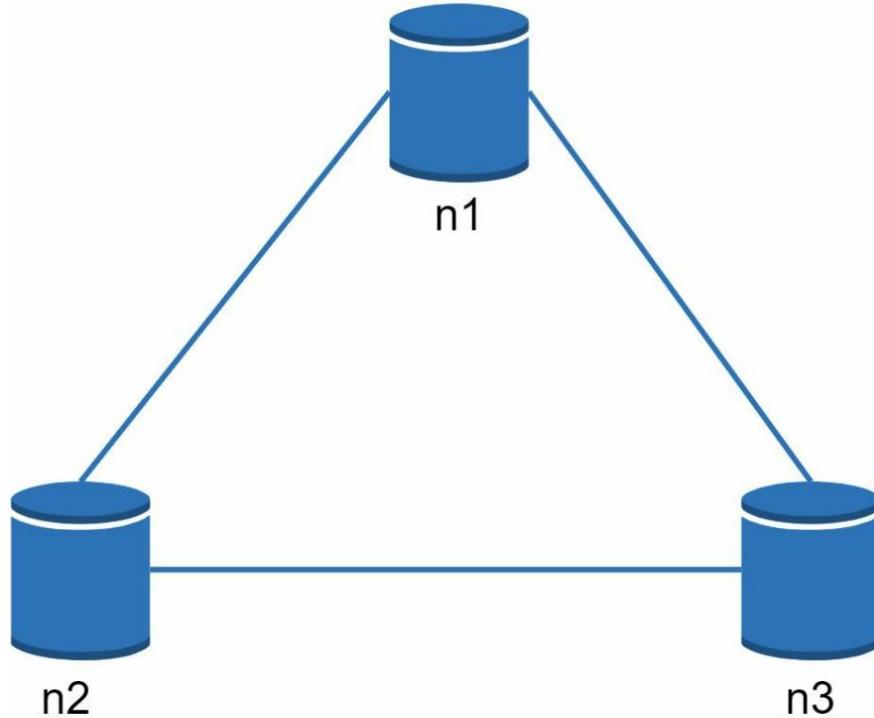


Figure 6-2

Hệ thống phân tán trong thế giới thực

Trong một hệ thống phân tán, không thể tránh khỏi việc phân vùng và khi xảy ra phân vùng, chúng ta phải lựa chọn giữa tính nhất quán và tính khả dụng. Trong Hình 6-3, n3 ngừng hoạt động và không thể giao tiếp với n1 và n2. Nếu máy khách ghi dữ liệu vào n1 hoặc n2, dữ liệu không thể được truyền đến n3. Nếu dữ liệu được ghi vào n3 như ng chia sẻ dữ liệu truyền đến n1 và n2, n1 và n2 sẽ có dữ liệu cũ.

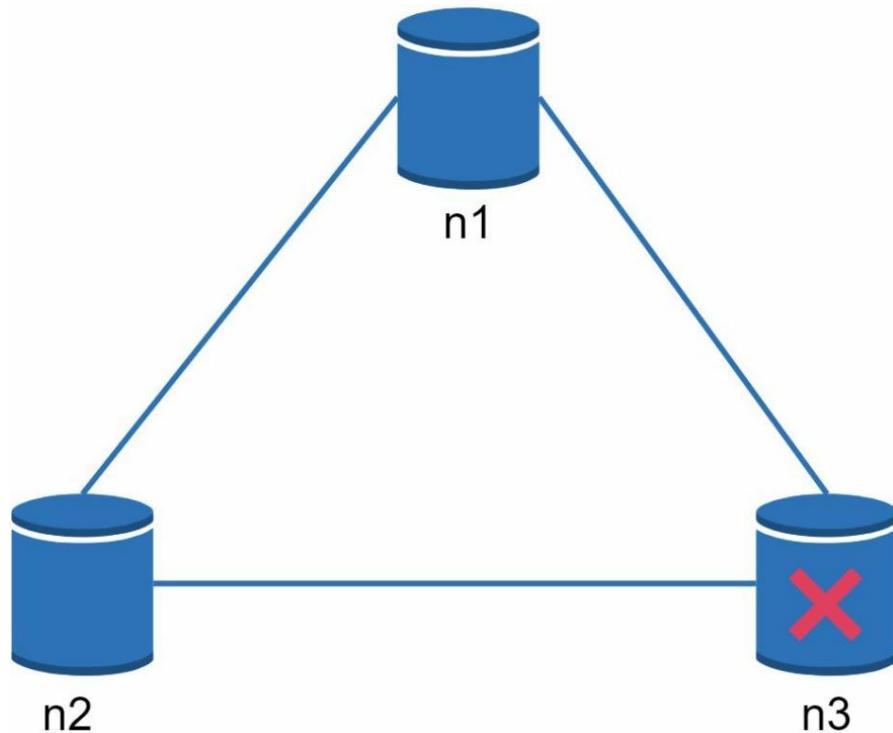


Figure 6-3

Nếu chúng ta chọn tính nhất quán thay vì tính khả dụng (hệ thống CP), chúng ta phải chặn tất cả các hoạt động ghi vào n1 và n2 để tránh sự không nhất quán dữ liệu giữa ba máy chủ này, khiến hệ thống không khả dụng. Các hệ thống ngân hàng thường có các yêu cầu nhất quán cực kỳ cao. Ví dụ, điều quan trọng đối với một hệ thống ngân hàng là hiển thị thông tin số dư mới nhất. Nếu sự không nhất quán xảy ra do phân vùng mạng, hệ thống ngân hàng sẽ trả về lỗi truer khi sự không nhất quán được giải quyết.

Tuy nhiên, nếu chúng ta chọn tính khả dụng thay vì tính nhất quán (hệ thống AP), hệ thống sẽ tiếp tục chấp nhận đọc, mặc dù nó có thể trả về dữ liệu cũ. Đối với ghi, n1 và n2 sẽ tiếp tục chấp nhận ghi và dữ liệu sẽ được đồng bộ hóa với n3 khi phân vùng mạng được giải quyết.

Việc lựa chọn đúng CAP đảm bảo phù hợp với nhu cầu sử dụng của bạn là một bước quan trọng trong việc xây dựng kho lưu trữ khóa-giá trị phân tán. Bạn có thể thảo luận điều này với người phỏng vấn và thiết kế hệ thống theo đó.

Các thành phần hệ thống

Trong phần này, chúng ta sẽ thảo luận về các thành phần cốt lõi và kỹ thuật sau đây sử dụng để xây dựng kho lưu trữ khóa-giá

- Phân vùng
- Sao chép
- Tính
- Giải quyết sự
- Xử lý lỗi
- Sơ đồ kiến trúc hệ thống
- Đa ờng dẫn
- Ghi
- Đọc

Nội dung bên dưới phần lớn dựa trên ba hệ thống lưu trữ khóa-giá trị phổ biến: Dynamo [4], Cassandra [5] và BigTable [6].

Phân vùng dữ liệu Đôi

với các ứng dụng lớn, không thể đưa toàn bộ tập dữ liệu vào một máy chủ duy nhất. Cách đơn giản nhất để thực hiện điều này là chia dữ liệu thành các phân vùng nhỏ hơn và lưu trữ chúng trên nhiều máy chủ. Có hai cách thức khi phân vùng dữ liệu:

- Phân phối dữ liệu đồng đều trên nhiều máy chủ. • Giảm thiểu việc di chuyển dữ liệu khi thêm hoặc xóa các nút.

Băm nhất quán được thảo luận trong Chương 5 là một kỹ thuật tuyệt vời để giải quyết những vấn đề này. Chúng ta hãy xem lại cách băm nhất quán hoạt động ở cấp độ cao. • Đầu

tiên, các máy chủ được đặt trên vòng băm. Trong Hình 6-4, tám máy chủ, được biểu diễn bởi s_0, s_1, \dots, s_7 , được đặt trên vòng băm. • Tiếp theo, một

khóa được băm vào cùng một vòng và được lưu trữ trên máy chủ đầu tiên gặp phải khi di chuyển theo chiều kim đồng hồ. Ví dụ, key_0 được lưu trữ trong s_1 bằng cách sử dụng logic này.

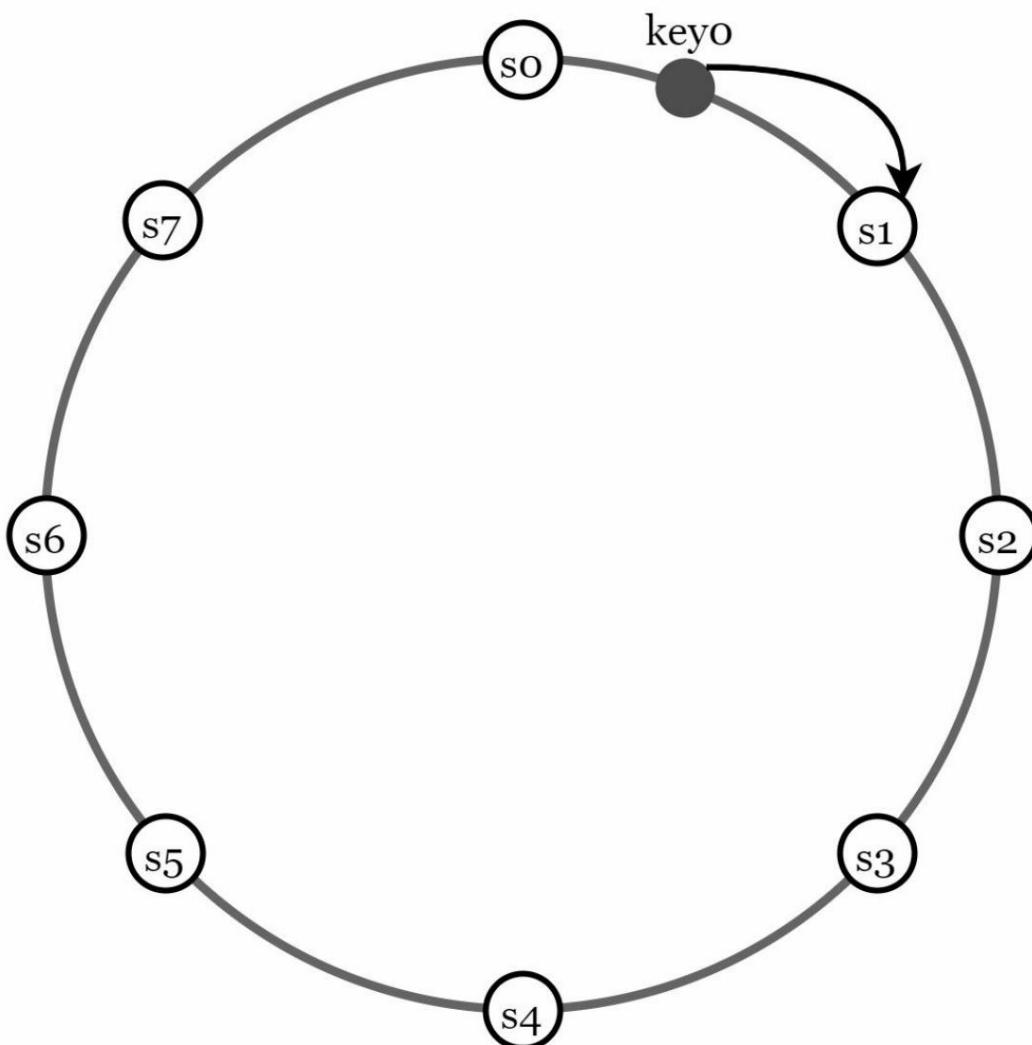


Figure 6-4

Sử dụng băm nhất quán để phân vùng dữ liệu có những ưu điểm sau:

Tự động mở rộng quy mô: máy chủ có thể được thêm vào và xóa tự động tùy thuộc vào

trọng tài.

Tính không đồng nhất: số lưu ợng nút ảo cho một máy chủ tỷ lệ thuận với dung lư ợng của máy chủ.

Ví dụ, các máy chủ có dung lư ợng cao hơn sẽ được chỉ định nhiều nút ảo hơn.

Sao chép dữ liệu Để

đạt được tính khả dụng và độ tin cậy cao, dữ liệu phải được sao chép không đồng bộ trên N máy chủ, trong đó N là tham số có thể cấu hình. N máy chủ này được chọn bằng logic sau: sau khi khóa được ánh xạ đến một vị trí trên vòng băm, hãy đi theo chiều kim đồng hồ từ vị trí đó và chọn N máy chủ đầu tiên trên vòng để lưu trữ các bản sao dữ liệu. Trong Hình 6-5 ($N = 3$), key0 được sao chép tại s1, s2 và s3.

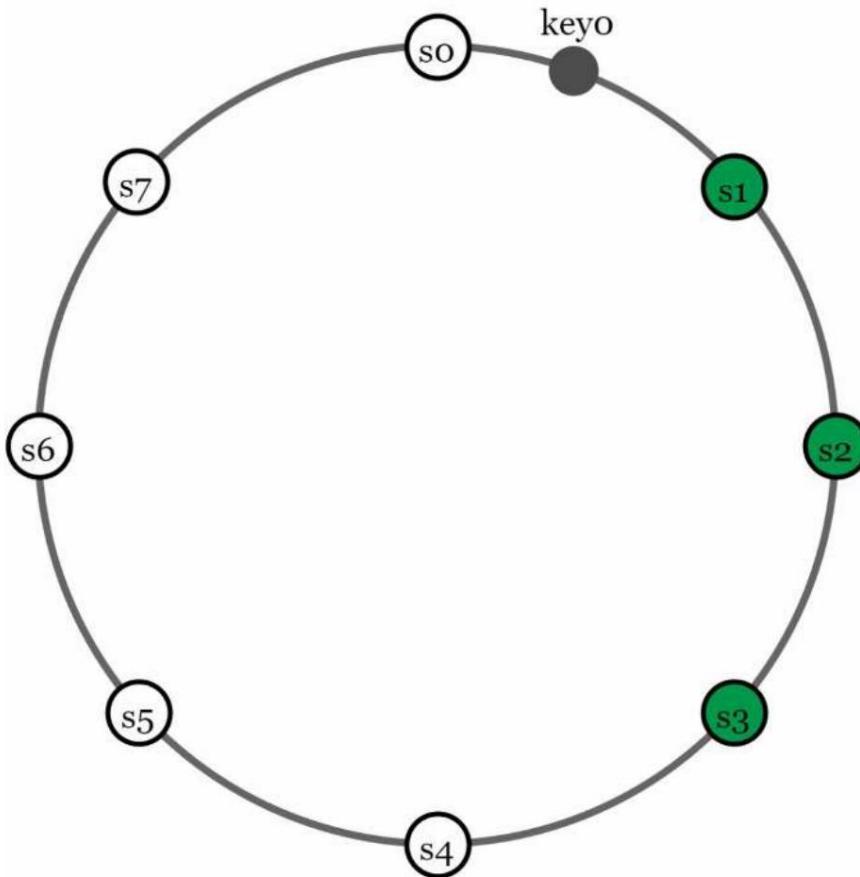


Figure 6-5

Với các nút ảo, N nút đầu tiên trên vòng có thể thuộc sở hữu của ít hơn N máy chủ vật lý. Để tránh vấn đề này, chúng tôi chỉ chọn các máy chủ duy nhất khi thực hiện logic đi theo chiều kim đồng hồ.

Các nút trong cùng một trung tâm dữ liệu thường bị lỗi cùng lúc do mất điện, sự cố mạng, thiên tai, v.v. Để có độ tin cậy tốt hơn, các bản sao được đặt trong các trung tâm dữ liệu riêng biệt và các trung tâm dữ liệu được kết nối thông qua mạng tốc độ cao.

Tính nhất quán Vì

dữ liệu được sao chép tại nhiều nút, nên nó phải được đồng bộ hóa trên các bản sao. Sự đồng thuận của Quorum có thể đảm bảo tính nhất quán cho cả hoạt động đọc và ghi. Trước tiên, chúng ta hãy thiết lập một vài định nghĩa.

N = Số lưu lượng bản sao

W = Số lưu lượng ghi tối thiểu có kích thước W . Để hoạt động ghi được coi là thành công, hoạt động ghi phải được xác nhận từ W bản sao.

R = Số lưu lượng đọc tối thiểu có kích thước R . Để một hoạt động đọc được coi là thành công, hoạt động đọc đó phải chờ phản hồi từ ít nhất R bản sao.

Hãy xem xét ví dụ sau được hiển thị trong Hình 6-6 với $N = 3$.

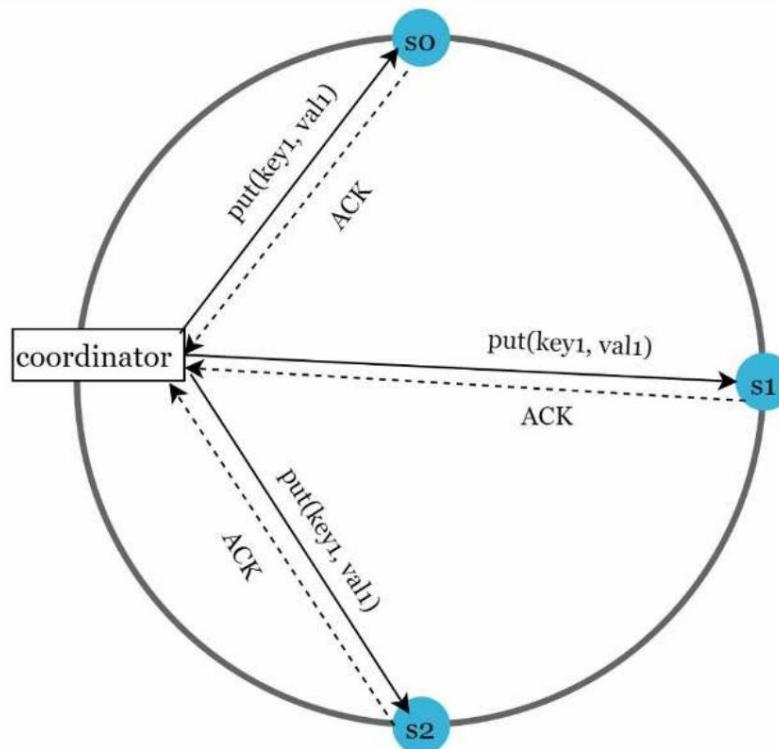


Figure 6-6 (ACK = acknowledgement)

$W = 1$ không có nghĩa là dữ liệu được ghi trên một máy chủ. Ví dụ, với cấu hình trong Hình 6-6, dữ liệu được sao chép tại s0, s1 và s2. $W = 1$ có nghĩa là bộ điều phối phải nhận được ít nhất một xác nhận trước khi hoạt động ghi được coi là thành công. Ví dụ, nếu chúng ta nhận được xác nhận từ s1, chúng ta không cần phải chờ xác nhận từ s0 và s2 nữa. Bộ điều phối hoạt động như một proxy giữa máy khách và các nút.

Cấu hình của W , R và N là sự đánh đổi diễn hình giữa độ trễ và tính nhất quán. Nếu $W = 1$ hoặc $R = 1$, một hoạt động được trả về nhanh chóng vì bộ điều phối chỉ cần đợi phản hồi từ bất kỳ bản sao nào. Nếu W hoặc $R > 1$, hệ thống cung cấp tính nhất quán tốt hơn; tuy nhiên, truy vấn sẽ chậm hơn vì bộ điều phối phải đợi phản hồi từ bản sao chậm nhất.

Nếu $W + R > N$, tính nhất quán mạnh được đảm bảo vì phải có ít nhất một

nút chòng lần có dữ liệu mới nhất để đảm bảo tính nhất quán.

Làm thế nào để cấu hình N, W và R để phù hợp với các trường hợp sử dụng của chúng ta? Sau đây là một số thiết lập có thể: Nếu $R = 1$ và $W = N$, hệ thống được tối ưu hóa để đọc nhanh.

Nếu $W = 1$ và $R = N$, hệ thống được tối ưu hóa để ghi nhanh.

Nếu $W + R > N$, tính nhất quán mạnh được đảm bảo (thường là $N = 3$, $W = R = 2$).

Nếu $W + R \leq N$, tính nhất quán mạnh không được đảm bảo.

Tùy theo yêu cầu, chúng ta có thể điều chỉnh các giá trị W, R, N để đạt được mức độ nhất quán mong muốn.

Mô hình nhất quán Mô hình

Nhất quán là một yếu tố quan trọng khác cần xem xét khi thiết kế kho lưu trữ khóa-giá trị. Mô hình nhất quán xác định mức độ nhất quán của dữ liệu và có nhiều mô hình nhất quán khả thi:

- **Nhất quán mạnh:** bất kỳ hoạt động đọc nào cũng trả về giá trị đúng.

Ung với kết quả của mục dữ liệu ghi được cập nhật nhất. Máy khách không bao giờ nhìn thấy dữ liệu lỗi thời.

- **Nhất quán yếu:** các hoạt động đọc tiếp theo có thể không nhìn thấy giá trị được cập nhật nhất.
- **Nhất quán cuối cùng:** đây là một dạng nhất quán yếu cụ thể. Khi có đủ thời gian, tất cả các bản cập nhật đều được truyền bá và tất cả các bản sao đều nhất quán.

Tính nhất quán mạnh thường đạt được bằng cách buộc một bản sao không chấp nhận các lần đọc/ghi mới cho đến khi mọi bản sao đồng ý về lần ghi hiện tại. Cách tiếp cận này không lý tưởng cho các hệ thống có tính khả dụng cao vì nó có thể chặn các hoạt động mới. Dynamo và Cassandra áp dụng tính nhất quán cuối cùng, đây là mô hình nhất quán được chúng tôi đề xuất cho kho lưu trữ khóa-giá trị của mình. Từ các lần ghi đồng thời, tính nhất quán cuối cùng cho phép các giá trị không nhất quán nhập vào hệ thống và buộc máy khách phải đọc các giá trị để đối chiếu. Phần tiếp theo giải thích cách đối chiếu hoạt động với phiên bản.

Giải quyết sự không nhất quán: phiên bản Sao

chép cung cấp tính khả dụng cao như ng gây ra sự không nhất quán giữa các bản sao. Phiên bản và khóa vectơ được sử dụng để giải quyết các vấn đề không nhất quán. Phiên bản có nghĩa là xử lý mỗi sửa đổi dữ liệu như một phiên bản dữ liệu mới không thể thay đổi. Trước khi nói về phiên bản, chúng ta hãy sử dụng một ví dụ để giải thích cách xảy ra sự không nhất quán:

Như thể hiện trong Hình 6-7, cả hai nút bản sao n1 và n2 đều có cùng giá trị. Chúng ta hãy gọi giá trị này là giá trị gốc. Máy chủ 1 và máy chủ 2 nhận được cùng một giá trị cho hoạt động `get("name")`.

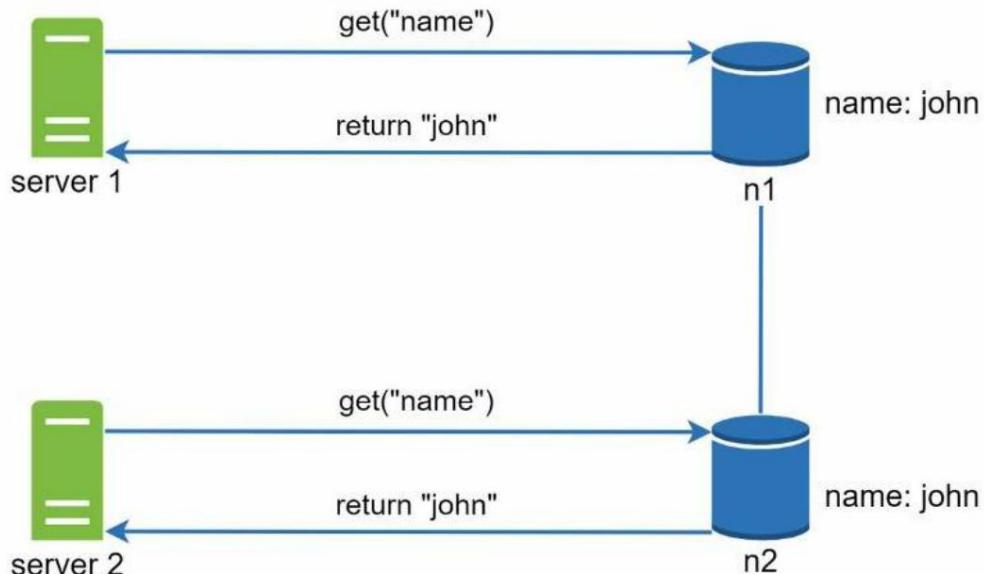


Figure 6-7

Tiếp theo, máy chủ 1 đổi tên thành "johnSanFrancisco", và máy chủ 2 đổi tên thành "johnNewYork" như thể hiện trong Hình 6-8. Hai thay đổi này được thực hiện đồng thời.

Bây giờ, chúng ta có các giá trị xung đột, được gọi là phiên bản v1 và v2.

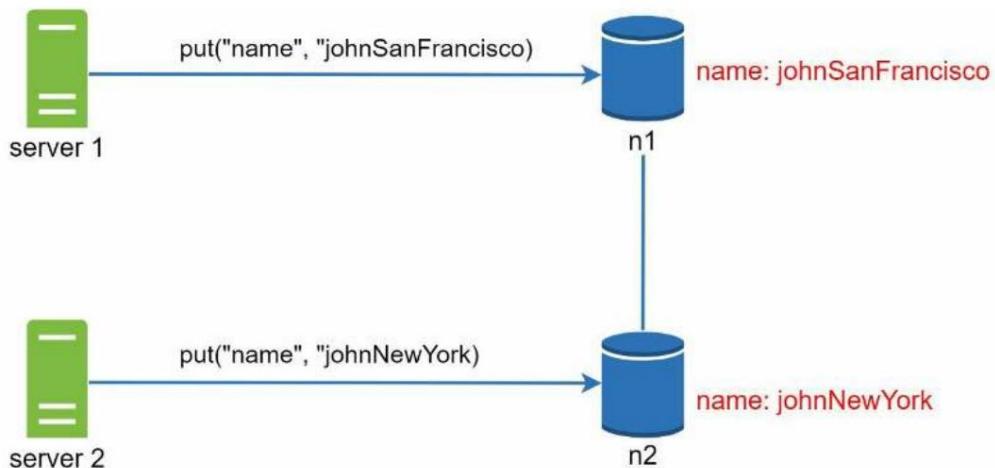


Figure 6-8

Trong ví dụ này, giá trị gốc có thể bị bỏ qua vì các sửa đổi dựa trên giá trị gốc. Tuy nhiên, không có cách rõ ràng nào để giải quyết xung đột của hai phiên bản cuối cùng. Để giải quyết vấn đề này, chúng ta cần một hệ thống quản lý phiên bản có thể phát hiện xung đột và điều hòa xung đột. Đồng hồ vector là một kỹ thuật phổ biến để giải quyết vấn đề này. Chúng ta hãy xem xét cách đồng hồ vector hoạt động.

Đồng hồ vector là cặp [máy chủ, phiên bản] liên kết với một mục dữ liệu. Nó có thể được sử dụng để kiểm tra xem một phiên bản có đi trước, thành công hay xung đột với các phiên bản khác không.

Giả sử đồng hồ vector được biểu diễn bởi $D([S_1, v_1], [S_2, v_2], \dots, [S_n, v_n])$, trong đó D là một mục dữ liệu, v_1 là bộ đếm phiên bản và S_1 là số máy chủ, v.v. Nếu mục dữ liệu D được ghi vào máy chủ S_i , hệ thống phải thực hiện một trong các tác vụ sau.

- Tăng vi nếu $[S_i, v_i]$ tồn tại.
- Nếu không, tạo một mục mới $[S_i, 1]$.

Logic trừu tư ợng trên được giải thích bằng ví dụ cụ thể như trong Hình 6-9.

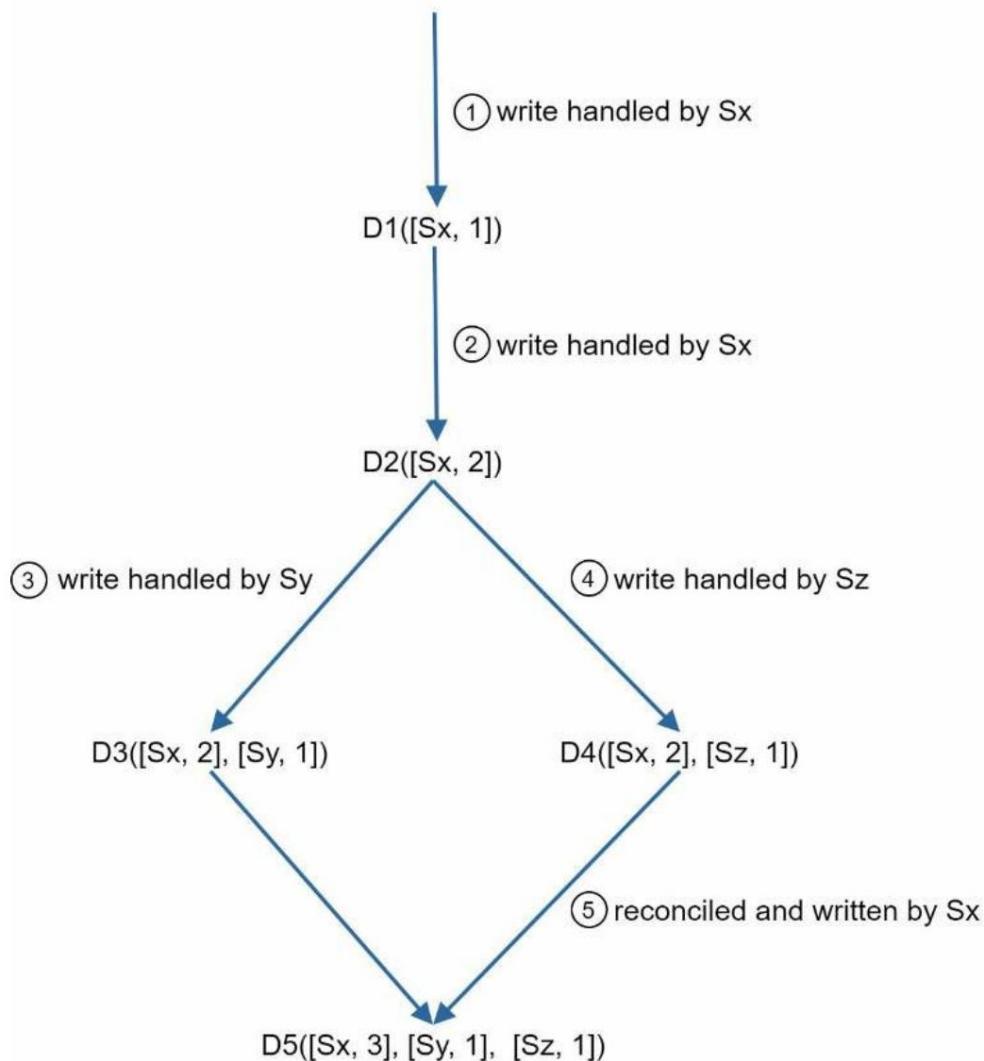


Figure 6-9 (source: [4])

1. Một máy khách ghi một mục dữ liệu $D1$ vào hệ thống và việc ghi này được xử lý bởi máy chủ S_x , hiện có đồng hồ vectơ $D1([S_x, 1])$.
2. Một máy khách khác đọc $D1$ mới nhất, cập nhật nó vào $D2$ và ghi lại. $D2$ đi xuống từ $D1$ nên nó ghi đè lên $D1$. Giả sử việc ghi được xử lý bởi cùng một máy chủ S_x , hiện có đồng hồ vectơ $D2([S_x, 2])$.
3. Một máy khách khác đọc $D2$ mới nhất, cập nhật thành $D3$ và ghi lại. Giả sử việc ghi được xử lý bởi máy chủ S_y , hiện có đồng hồ vectơ $D3([S_x, 2], [S_y, 1])$.
4. Một máy khách khác đọc $D2$ mới nhất, cập nhật nó thành $D4$ và ghi lại. Giả sử lệnh ghi

đư ợc xử lý bởi máy chủ Sz, hiện có D4([Sx, 2], [Sz, 1])).

5. Khi một máy khách khác đọc D3 và D4, nó phát hiện ra xung đột, do mục dữ liệu D2 bị cả Sy và Sz sửa đổi. Xung đột đư ợc máy khách giải quyết và dữ liệu cập nhật đư ợc gửi đến máy chủ. Giả sử lệnh ghi đư ợc xử lý bởi Sx, hiện có D5([Sx, 3], [Sy, 1], [Sz, 1]). Chúng tôi sẽ giải thích cách phát hiện xung đột ngay sau đây.

Khi sử dụng đồng hồ vectơ, có thể dễ dàng nhận biết phiên bản X là tổ tiên (tức là không có xung đột) của phiên bản Y nếu bộ đếm phiên bản cho mỗi người tham gia trong đồng hồ vectơ của Y lớn hơn hoặc bằng bộ đếm trong phiên bản X. Ví dụ, đồng hồ vectơ D([s0, 1], [s1, 1]) là tổ tiên của D([s0, 1], [s1, 2]). Do đó, không có xung đột nào đư ợc ghi lại.

Tuy nhiên, bạn có thể biết rằng phiên bản X là phiên bản anh chị em (tức là có xung đột) của Y nếu có bất kỳ người tham gia nào trong đồng hồ vectơ của Y có bộ đếm nhỏ hơn bộ đếm tương ứng của nó trong X. Ví dụ, hai đồng hồ vectơ sau đây chỉ ra rằng có xung đột: D([s0, 1], [s1, 2]) và D([s0, 2], [s1, 1]).

Mặc dù đồng hồ vector có thể giải quyết xung đột, nhưng có hai như ợc điểm đáng chú ý. Đầu tiên, đồng hồ vector làm tăng thêm độ phức tạp cho máy khách vì nó cần triển khai logic giải quyết xung đột.

Thứ hai, các cặp [server: version] trong đồng hồ vector có thể tăng nhanh. Để khắc phục vấn đề này, chúng tôi đặt người ờng cho độ dài và nếu vư ợt quá giới hạn, các cặp cũ nhất sẽ bị xóa. Điều này có thể dẫn đến tình trạng không hiệu quả trong việc đối chiếu vì không thể xác định chính xác mối quan hệ con cháu. Tuy nhiên, dựa trên bài báo Dynamo [4], Amazon vẫn chưa gặp phải vấn đề này trong quá trình sản xuất; do đó, đây có thể là giải pháp chấp nhận đư ợc đối với hầu hết các công ty.

Xử lý lỗi Cũng như bất kỳ

hệ thống lớn nào ở quy mô lớn, lỗi không chỉ là điều không thể tránh khỏi mà còn là lỗi thường gặp. Xử lý các tình huống lỗi là rất quan trọng. Trong phần này, trước tiên chúng tôi giới thiệu các kỹ thuật để phát hiện lỗi. Sau đó, chúng tôi sẽ xem xét các chiến lược giải quyết lỗi phổ biến.

Phát hiện lỗi

Trong một hệ thống phân tán, không đủ để tin rằng một máy chủ ngừng hoạt động vì một máy chủ khác nói như vậy. Thông thường, cần ít nhất hai nguồn thông tin độc lập để đánh dấu một máy chủ ngừng hoạt động.

Như thể hiện trong Hình 6-10, đa hưng tất cả đến tất cả là một giải pháp đơn giản. Tuy nhiên, giải pháp này không hiệu quả khi có nhiều máy chủ trong hệ thống.

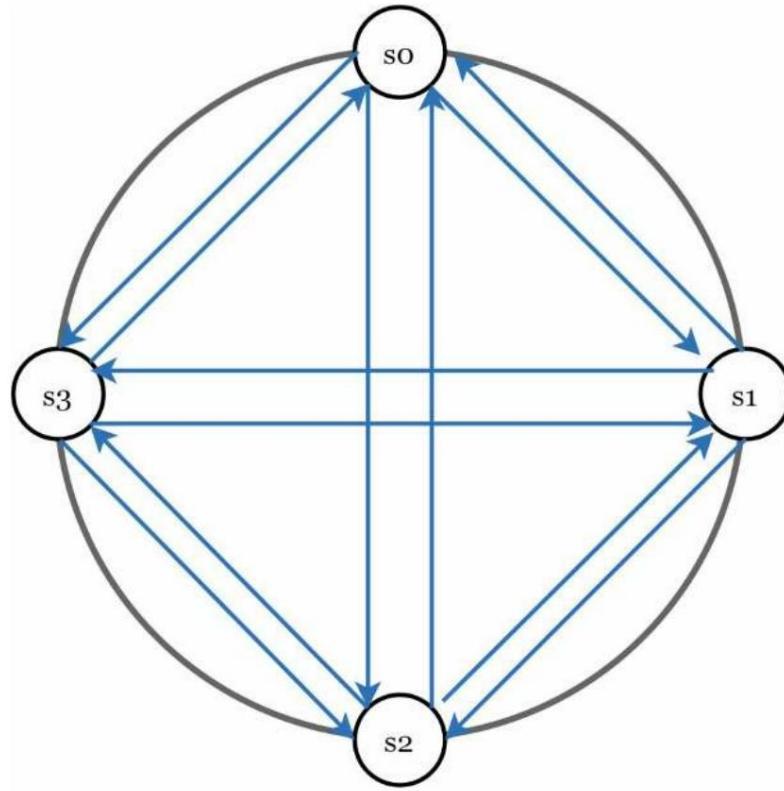


Figure 6-10

Một giải pháp tốt hơn là sử dụng các phương pháp phát hiện lỗi phi tập trung như giao thức gossip. Giao thức Gossip hoạt động như sau:

Mỗi nút duy trì một danh sách thành viên nút, trong đó có ID thành viên và nhịp tim quầy hàng.

- Mỗi nút định kỳ tăng bộ đếm nhịp tim của nó.
- Mỗi nút định kỳ gửi nhịp tim đến một nhóm các nút ngẫu nhiên, sau đó lan truyền đến một nhóm các nút khác.
- Khi các nút nhận được nhịp tim, danh sách thành viên sẽ được cập nhật thông tin mới nhất.
- Nếu nhịp tim không tăng trong hơn các khoảng thời gian được xác định trước, thành viên sẽ được coi là ngoại tuyến.

s0's membership list		
Member ID	Heartbeat counter	Time
0	10232	12:00:01
1	10224	12:00:10
2	9908	11:58:02
3	10237	12:00:20
4	10234	12:00:34

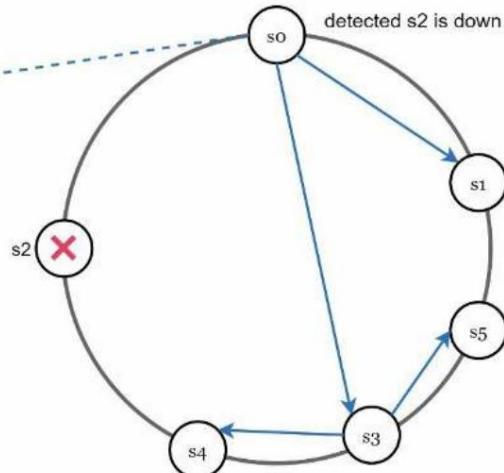


Figure 6-11

Như thể hiện trong Hình 6-11:

- Nút s0 duy trì danh sách thành viên nút được hiển thị ở bên trái.
- Nút s0 nhận thấy rằng bộ đếm nhịp tim của nút s2 (ID thành viên = 2) đã không tăng trong một thời gian dài.
- Nút s0 gửi nhịp tim bao gồm thông tin của s2 đến một tập hợp các nút ngẫu nhiên. Khi các nút khác xác nhận rằng bộ đếm nhịp tim của s2 đã không cập nhật trong một thời gian dài, nút s2 sẽ được đánh dấu xuống và thông tin này được truyền đến các nút khác.

Xử lý lỗi tạm thời Sau khi lỗi được

phát hiện thông qua giao thức gossip, hệ thống cần triển khai một số cơ chế nhất định để đảm bảo tính khả dụng. Trong phương pháp tiếp cận số lượng nghiêm ngặt, các hoạt động đọc và ghi có thể bị chặn như minh họa trong phần đồng thuận số lượng.

Một kỹ thuật được gọi là "slippery quorum" [4] được sử dụng để cải thiện tính khả dụng. Thay vì thực thi yêu cầu về quorum, hệ thống chọn W máy chủ khỏe mạnh đầu tiên để ghi và R máy chủ khỏe mạnh đầu tiên để đọc trên vòng băm. Máy chủ ngoại tuyến bị bỏ qua.

Nếu một máy chủ không khả dụng do lỗi mạng hoặc lỗi máy chủ, một máy chủ khác sẽ xử lý các yêu cầu tạm thời. Khi máy chủ ngừng hoạt động trở lại, các thay đổi sẽ được đẩy lùi để đạt được sự nhất quán của dữ liệu. Quá trình này được gọi là chuyển giao được gợi ý. Vì s2 không khả dụng trong Hình 6-12, các thao tác đọc và ghi sẽ được s3 xử lý tạm thời. Khi s2 trực tuyến trở lại, s3 sẽ chuyển dữ liệu trở lại cho s2.

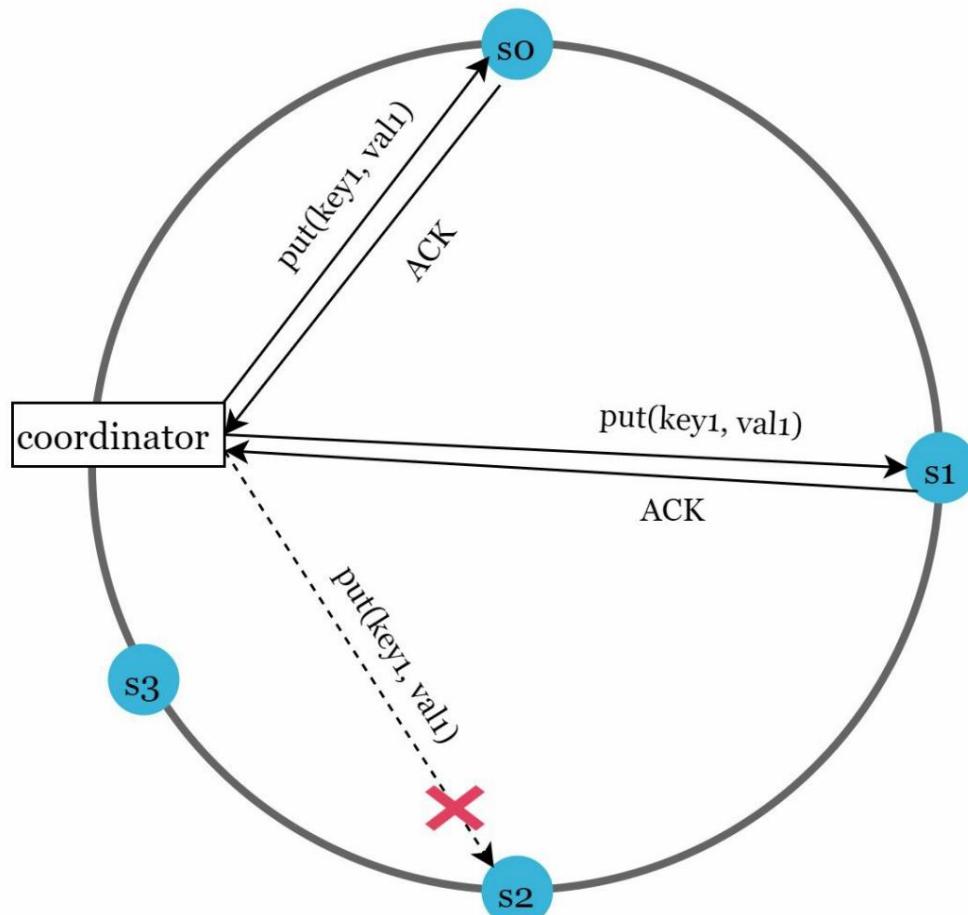


Figure 6-12

Xử lý lỗi vĩnh viễn

Hinted handoff được sử dụng để xử lý các lỗi tạm thời. Nếu một bản sao không khả dụng vĩnh viễn thì sao? Để xử lý tình huống như vậy, chúng tôi triển khai giao thức chống entropy để giữ cho các bản sao đồng bộ. Chống entropy bao gồm việc so sánh từng phần dữ liệu trên các bản sao và cập nhật từng bản sao lên phiên bản mới nhất. Cây Merkle được sử dụng để phát hiện sự không nhất quán và giảm thiểu lạm dụng dữ liệu được truyền.

Trích từ Wikipedia [7]: "Cây băm hoặc cây Merkle là một cây trong đó mọi nút không phải lá được gắn nhãn bằng băm của các nhãn hoặc giá trị (trong trường hợp lá) của các nút con của nó.

Cây băm cho phép xác minh hiệu quả và an toàn nội dung của các cấu trúc dữ liệu lớn".

Giả sử không gian khóa là từ 1 đến 12, các bước sau đây chỉ cách xây dựng cây Merkle.

Các hộp được tô sáng cho thấy sự không nhất quán.

Bước 1: Chia không gian khóa thành các bucket (4 trong ví dụ của chúng tôi) như thể hiện trong Hình 6-13. Một bucket được sử dụng làm nút cấp gốc để duy trì độ sâu giới hạn của cây.

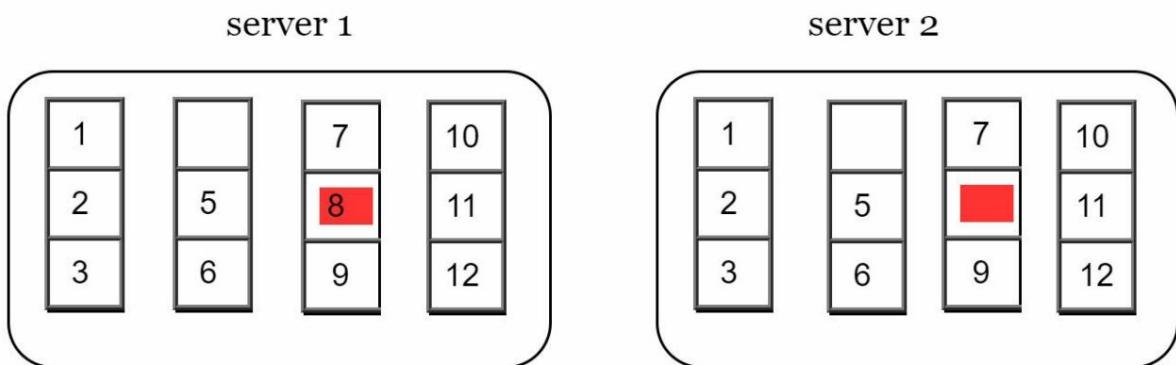


Figure 6-13

Bước 2: Sau khi các thùng được tạo, hãy băm từng khóa trong thùng bằng phư ơng pháp băm thống nhất (Hình 6-14).

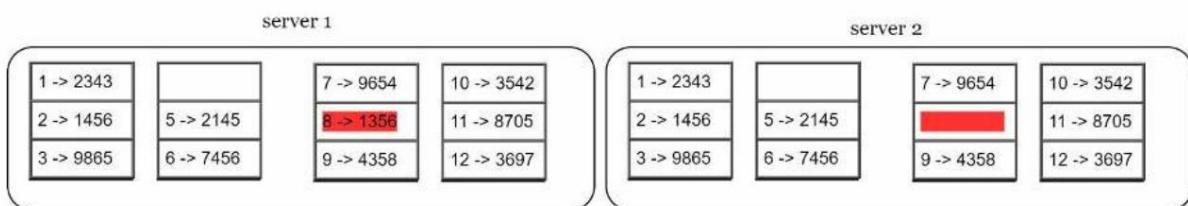


Figure 6-14

Bước 3: Tạo một nút băm duy nhất cho mỗi thùng (Hình 6-15).

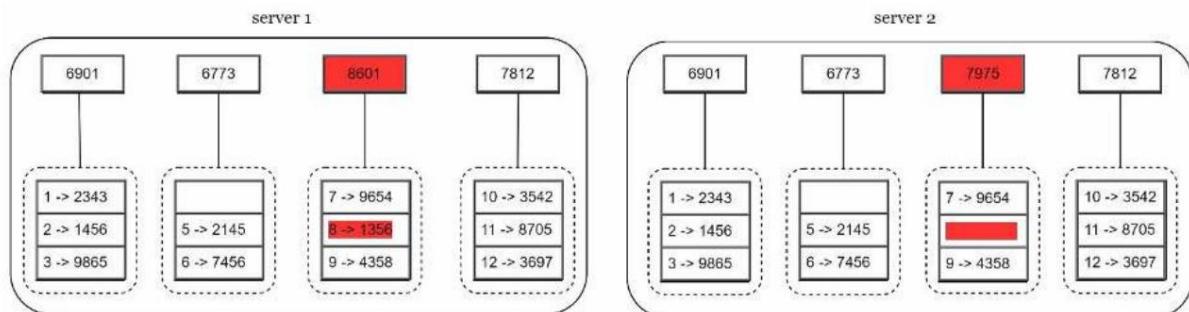


Figure 6-15

Bước 4: Xây dựng cây theo hướng từ gốc đến ngọn bằng cách tính toán các giá trị băm của các nút con (Hình 6-16).

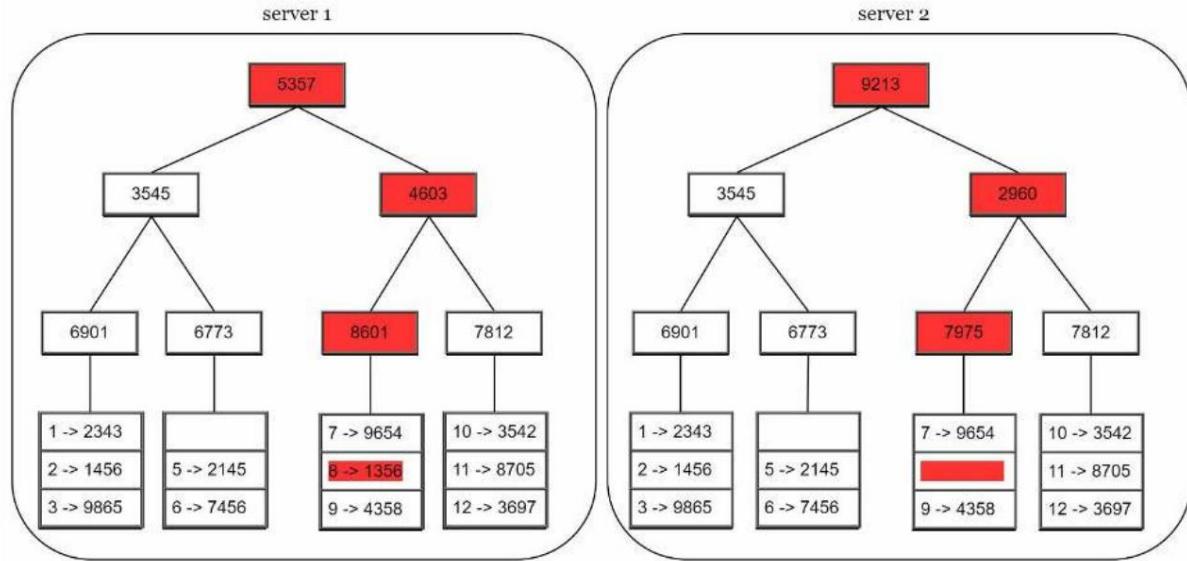


Figure 6-16

Để so sánh hai cây Merkle, hãy bắt đầu bằng cách so sánh các băm gốc. Nếu băm gốc khớp nhau, cả hai máy chủ đều có cùng dữ liệu. Nếu băm gốc không đồng nhất, thì băm con bên trái được so sánh tiếp theo là băm con bên phải. Bạn có thể duyệt cây để tìm các bucket nào không được đồng bộ hóa và chỉ đồng bộ hóa các bucket đó.

Sử dụng cây Merkle, lượng dữ liệu cần đồng bộ hóa tỷ lệ thuận với sự khác biệt giữa hai bản sao, chứ không phải lượng dữ liệu chúng chứa. Trong các hệ thống thực tế, kích thước bucket khá lớn. Ví dụ, một cấu hình khả thi là một triệu bucket cho một tỷ khóa, vì vậy mỗi bucket chỉ chứa 1000 khóa.

Xử lý sự cố mất điện của trung tâm dữ liệu

tâm dữ liệu Sự cố mất điện của trung tâm dữ liệu có thể xảy ra do mất điện, mất mạng, thiên tai, v.v. Để xây dựng một hệ thống có khả năng xử lý sự cố mất điện của trung tâm dữ liệu, điều quan trọng là phải sao chép dữ liệu trên nhiều trung tâm dữ liệu. Ngay cả khi một trung tâm dữ liệu hoàn toàn ngoại tuyến, người dùng vẫn có thể truy cập dữ liệu thông qua các trung tâm dữ liệu khác.

Sơ đồ kiến trúc hệ thống Bây giờ

chúng ta đã thảo luận về các cân nhắc kỹ thuật khác nhau khi thiết kế kho lưu trữ khóa-giá trị, chúng ta có thể chuyển trọng tâm vào sơ đồ kiến trúc, được hiển thị trong Hình 6-17.

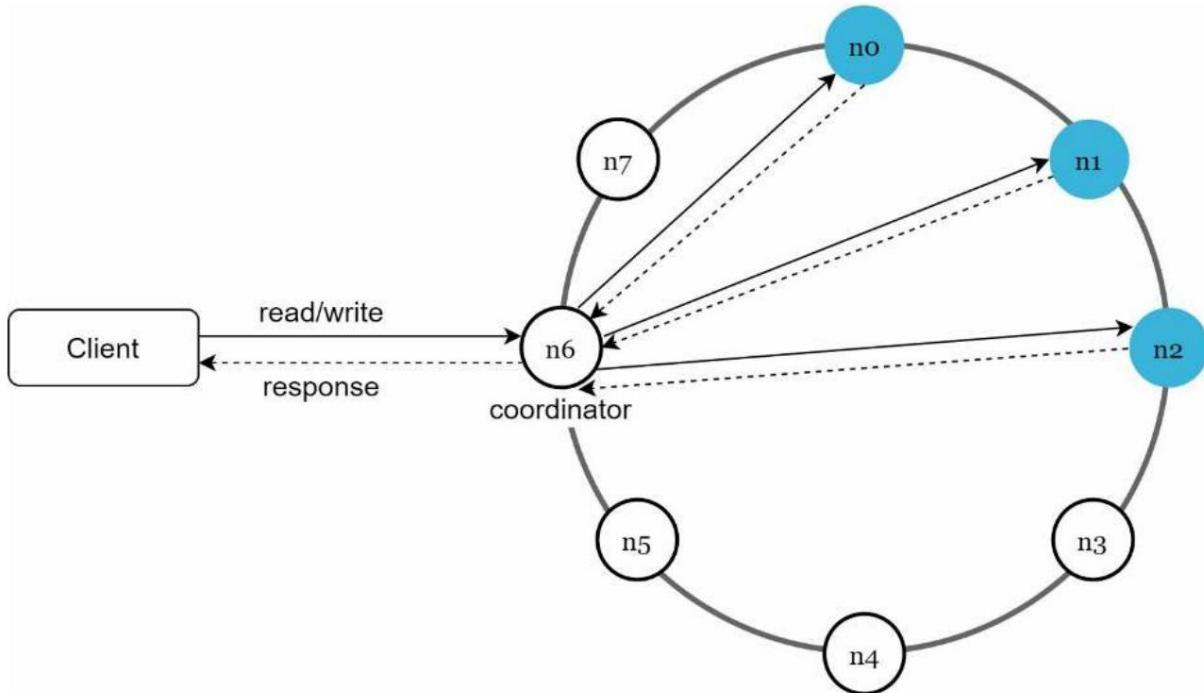


Figure 6-17

Các đặc điểm chính của kiến trúc được liệt kê như sau:

- Khách hàng giao tiếp với kho lưu trữ khóa-giá trị thông qua các API đơn giản: get(key) và put(key, value).
- Bộ điều phối là một nút hoạt động như một proxy giữa khách hàng và kho lưu trữ khóa-giá trị.

Các nút được phân phối trên một vòng sử dụng băm nhất quán. • Hệ thống hoàn toàn phi tập trung nên việc thêm và di chuyển các nút có thể tự động. • Dữ liệu được sao chép tại nhiều nút. • Không có điểm lỗi duy nhất vì mọi nút đều có cùng một tập hợp trách nhiệm.

Vì thiết kế được phân cấp nên mỗi nút thực hiện nhiều nhiệm vụ như được trình bày trong Hình 6-18.

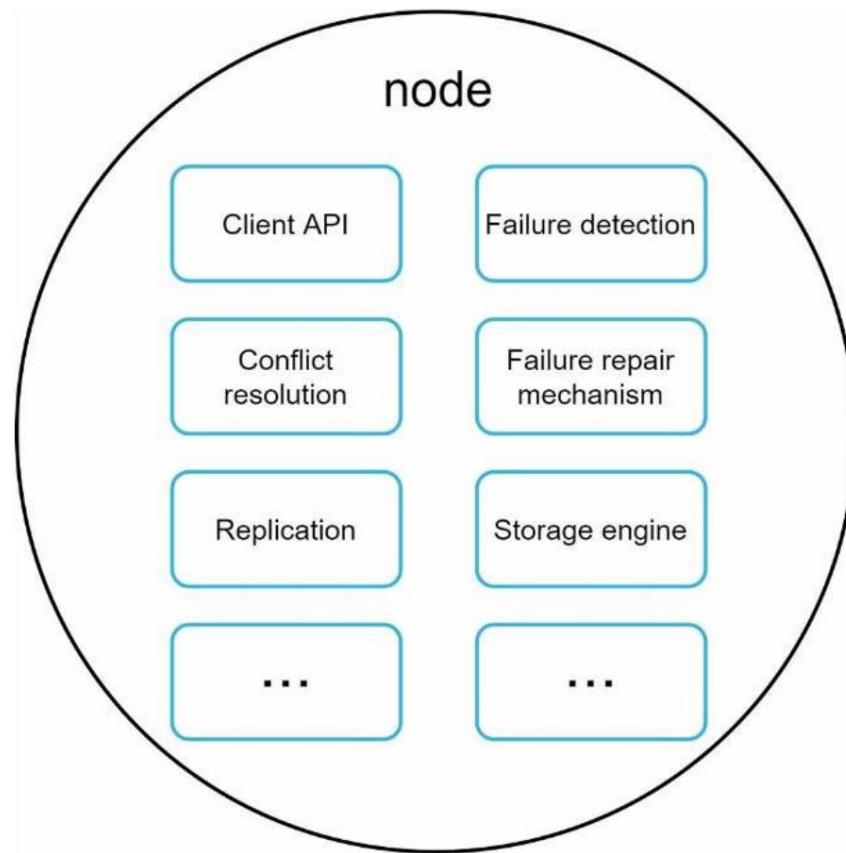


Figure 6-18

Đưa ứng dụng ghi

Hình 6-19 giải thích những gì xảy ra sau khi yêu cầu ghi được chuyển hướng đến một nút cụ thể. Xin lưu ý rằng các thiết kế được đề xuất cho đưa ứng dụng ghi/đọc chủ yếu dựa trên kiến trúc của Cassandra [8].

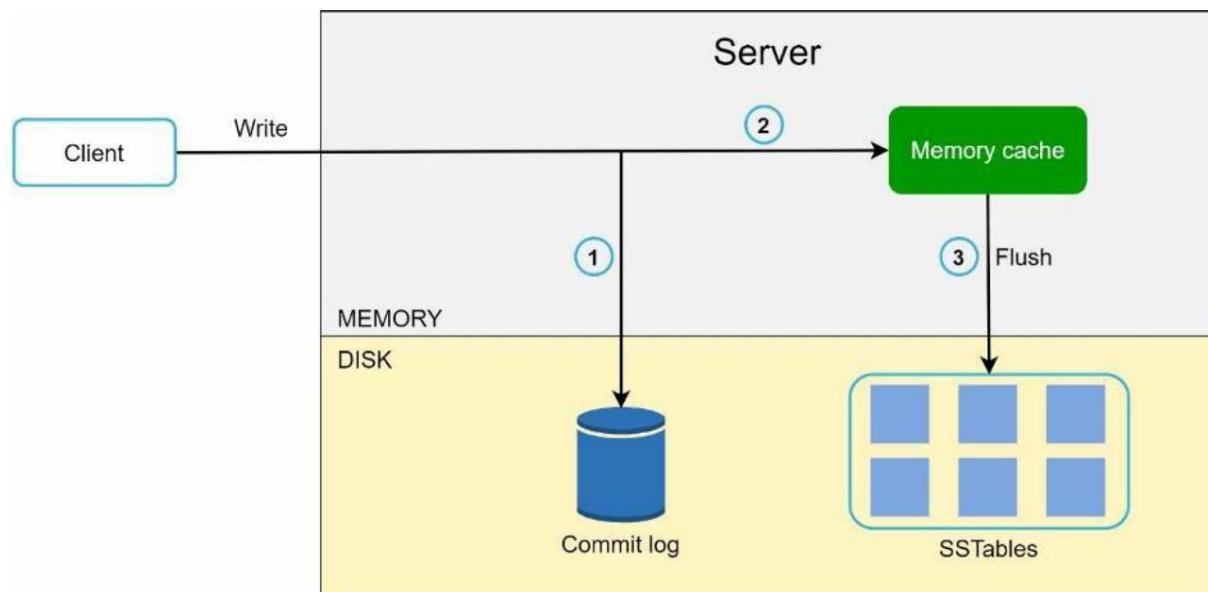


Figure 6-19

1. Yêu cầu ghi được lưu lại trong tệp nhật ký xác nhận.

2. Dữ liệu được lưu trong bộ nhớ đệm.

3. Khi bộ nhớ đệm đầy hoặc đạt đến ngưỡng định trước, dữ liệu sẽ được chuyển đến SSTable [9] trên đĩa. Lưu ý: Bảng chuỗi được sắp xếp (SSTable) là danh sách được sắp xếp của các cặp <khóa, giá trị>. Đối với đặc điểm quan tâm đến việc tìm hiểu thêm về SSTable, hãy tham khảo tài liệu tham khảo [9].

Để ờng dẫn đọc

Sau khi yêu cầu đọc được chuyển hướng đến một nút cụ thể, trước tiên nó sẽ kiểm tra xem dữ liệu có trong bộ nhớ đệm hay không. Nếu có, dữ liệu được trả về cho khách hàng như thể hiện trong Hình 6-20.

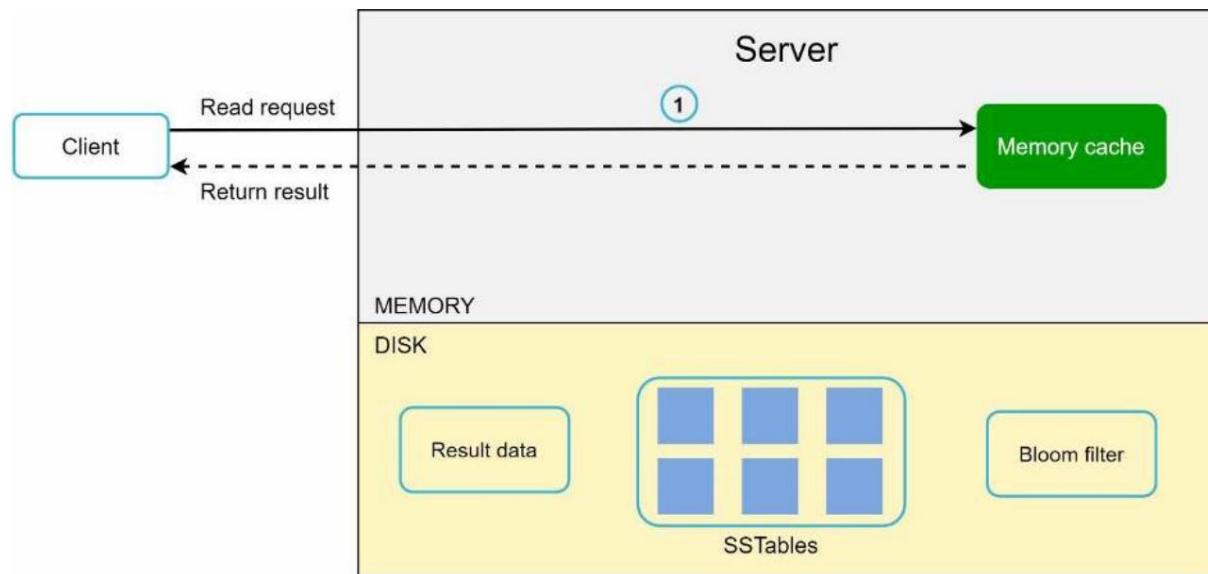


Figure 6-20

Nếu dữ liệu không có trong bộ nhớ, dữ liệu sẽ được lấy từ đĩa thay thế. Chúng ta cần một cách hiệu quả để tìm ra SSTable nào chứa khóa. Bộ lọc Bloom [10] thường được sử dụng để giải quyết vấn đề này.

Để ờng dẫn đọc được hiển thị ở Hình 6-21 khi dữ liệu không có trong bộ nhớ.

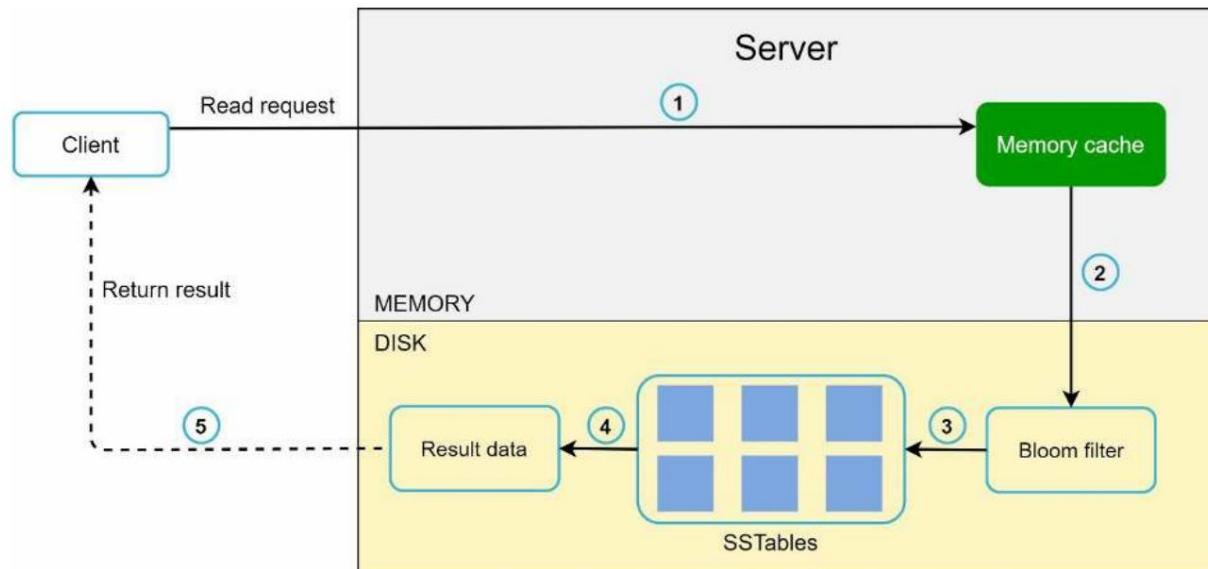


Figure 6-21

1. Đầu tiên, hệ thống kiểm tra xem dữ liệu có trong bộ nhớ không. Nếu không, hãy chuyển sang bước 2.

2. Nếu dữ liệu không có trong bộ nhớ, hệ thống sẽ kiểm tra bộ lọc Bloom.

3. Bộ lọc Bloom được sử dụng để tìm ra SSTable nào có thẻ chứa khóa.
4. SSTables trả về kết quả của tập dữ liệu.
5. Kết quả của tập dữ liệu được trả về cho máy khách.

Tóm tắt

Chư ơ ng này đề cập đến nhiều khái niệm và kỹ thuật. Để làm mới trí nhớ của bạn, bảng sau đây tóm tắt các tính năng và kỹ thuật tương ứng được sử dụng cho khóa-giá trị phân tán cửa hàng.

Goal/Problems	Technique
Ability to store big data	Use consistent hashing to spread the load across servers
High availability reads	Data replication Multi-data center setup
Highly available writes	Versioning and conflict resolution with vector clocks
Dataset partition	Consistent Hashing
Incremental scalability	Consistent Hashing
Heterogeneity	Consistent Hashing
Tunable consistency	Quorum consensus
Handling temporary failures	Sloppy quorum and hinted handoff
Handling permanent failures	Merkle tree
Handling data center outage	Cross-data center replication

Table 6-2

Tài liệu tham khảo

[1] Amazon DynamoDB: <https://aws.amazon.com/dynamodb/> [2] _____
memcached: <https://memcached.org/> [3] _____
Redis: <https://redis.io/> _____
[4] Dynamo: Kho lưu trữ khóa-giá trị có sẵn cao của Amazon: <https://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf> [5] Cassandra: <https://cassandra.apache.org/> [6] Bigtable: Hệ thống lưu trữ phân tán dữ liệu có cấu trúc: <https://static.googleusercontent.com/media/research.google.com/en/archive/bigtable-osdi06.pdf> [7] Cây Merkle: https://en.wikipedia.org/wiki/Merkle_tree [8] Kiến trúc Cassandra: <https://cassandra.apache.org/doc/latest/architecture/> [9] SSTable: <https://www.igvita.com/2012/02/06/sstable-and-log-structured-storage-leveledb/> [10] Bộ lọc Bloom https://en.wikipedia.org/wiki/Bloom_filter

CHƯƠNG 7: THIẾT KẾ MỘT TRÌNH TẠO ID DUY NHẤT TRONG HỆ THỐNG PHÂN TÁN

Trong chương này, bạn được yêu cầu thiết kế một trình tạo ID duy nhất trong các hệ thống phân tán. Ý nghĩ đầu tiên của bạn có thể là sử dụng khóa chính với thuộc tính auto_increment trong cơ sở dữ liệu truyền thống. Tuy nhiên, auto_increment không hoạt động trong môi trường phân tán vì một máy chủ cơ sở dữ liệu duy nhất không đủ lớn và việc tạo ID duy nhất trên nhiều cơ sở dữ liệu với độ trễ tối thiểu là một thách thức.

Sau đây là một số ví dụ về ID duy nhất:

user_id
1227238262110117894
1241107244890099715
1243643959492173824
1247686501489692673
1567981766075453440

Figure 7-1

Bư ớc 1 - Hiểu rõ vấn đề và xác định phạm vi thiết kế Đặt câu hỏi làm rõ là bư ớc đầu tiên để giải quyết bất kỳ câu hỏi phỏng vấn thiết kế hệ thống nào.

Sau đây là một ví dụ về tư ơng tác giữa ứng viên và người phỏng

vấn: Ứng viên: Đặc điểm của ID duy nhất là gì?

Người phỏng vấn: ID phải là duy nhất và có thể sắp xếp được.

Ứng viên: Với mỗi bản ghi mới, ID có tăng thêm 1 không?

Người phỏng vấn: ID tăng dần theo thời gian như ng không nhất thiết chỉ tăng 1. ID được tạo vào buổi tối lớn hơn ID được tạo vào buổi sáng cùng ngày.

Ứng viên: ID chỉ chứa giá trị số phải không?

Người phỏng vấn: Vâng, đúng vậy.

Ứng viên: Yêu cầu về độ dài ID là bao nhiêu?

Người phỏng vấn: ID phải phù hợp với chuẩn 64-bit.

Ứng viên: Quy mô của hệ thống như thế nào?

Người phỏng vấn: Hệ thống phải có khả năng tạo ra 10.000 ID mỗi giây.

Trên đây là một số câu hỏi mẫu mà bạn có thể hỏi người phỏng vấn. Điều quan trọng là phải hiểu các yêu cầu và làm rõ những điều mơ hồ. Đối với câu hỏi phỏng vấn này, các yêu cầu được liệt kê như sau:

- ID phải là duy nhất.
- ID chỉ là giá trị số.

- ID phù hợp với 64-bit.
- ID được sắp xếp theo ngày.
- Khả năng tạo ra hơn 10.000 ID duy nhất mỗi giây.

Bưu ốc 2 - Đề xuất thiết kế cấp cao và nhận được sự đồng thuận Nhiều tùy

chọn có thể được sử dụng để tạo ID duy nhất trong các hệ thống phân tán. Các tùy chọn chúng tôi xem xét là:

- Sao chép nhiều máy chủ • Mã định danh duy nhất toàn cầu (UUID) • Máy chủ vé
- Phư ơng pháp bông tuyết Twitter

Chúng ta hãy cùng xem xét từng phư ơng án, cách chúng hoạt động và ưu/nhược điểm của từng lựa chọn.

Sao chép nhiều máy chủ Như

thể hiện trong Hình 7-2, cách tiếp cận đầu tiên là sao chép nhiều máy chủ.

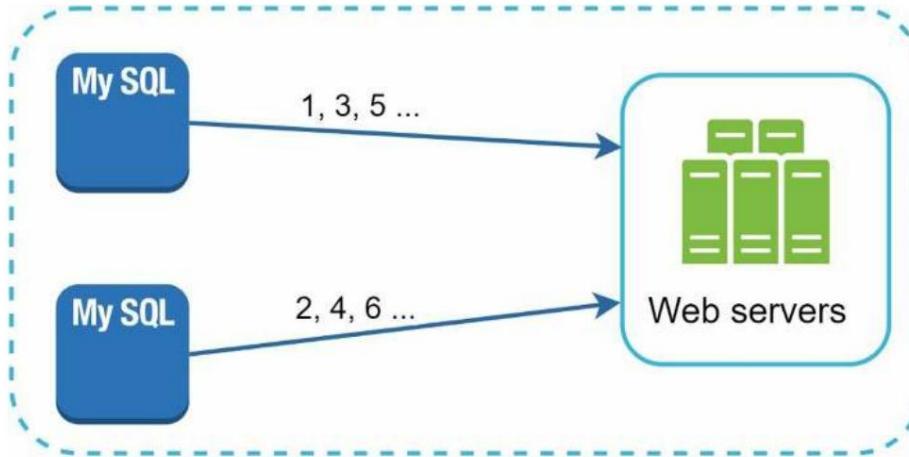


Figure 7-2

Phư ơng pháp này sử dụng tính năng auto_increment của cơ sở dữ liệu . Thay vì tăng ID tiếp theo thêm 1, chúng tôi tăng nó thêm k, trong đó k là số máy chủ cơ sở dữ liệu đang sử dụng. Như minh họa trong Hình 7-2, ID tiếp theo được tạo bằng ID trước đó trong cùng một máy chủ cộng với 2. Điều này giải quyết một số vấn đề về khả năng mở rộng vì ID có thể mở rộng theo số lượng máy chủ cơ sở dữ liệu.

Tuy nhiên, chi phí lưu trữ này có một số nhược điểm lớn:

- Khó mở rộng quy mô với nhiều trung tâm dữ liệu • ID không tăng theo thời gian trên nhiều máy chủ. • Không mở rộng quy mô tốt khi thêm hoặc xóa máy chủ.

Mã UUID

UUID là một cách dễ dàng khác để có được ID duy nhất. UUID là số 128 bit được sử dụng để xác định thông tin trong hệ thống máy tính. UUID có khả năng thông đồng rất thấp.

Trích dẫn từ Wikipedia, “sau khi tạo ra 1 tỷ UUID mỗi giây trong khoảng 100 năm, khả năng tạo ra một bản sao duy nhất sẽ đạt 50%” [1].

Sau đây là ví dụ về UUID: 09c93e62-50b4-468d-bf8a-c07e1040fb2. UUID có thể được tạo độc lập mà không cần sự phối hợp giữa các máy chủ. Hình 7-3 trình bày thiết kế UUID.

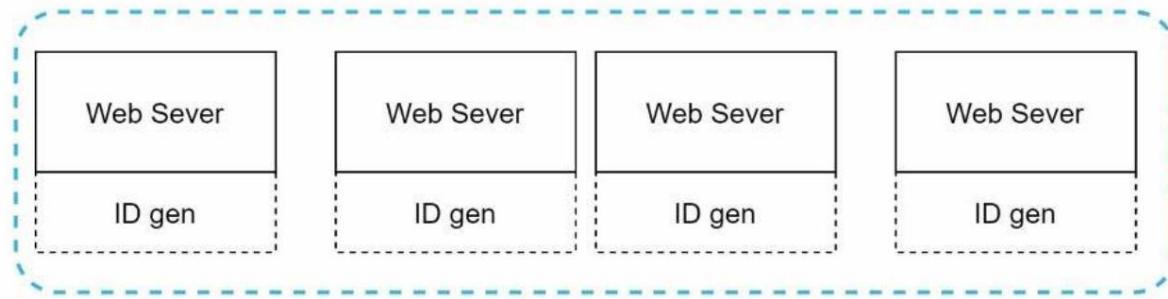


Figure 7-3

Trong thiết kế này, mỗi máy chủ web đều chứa một trình tạo ID và mỗi máy chủ web có trách nhiệm tạo ID một cách độc lập.

Ưu điểm:

- Việc tạo UUID rất đơn giản. Không cần sự phối hợp giữa các máy chủ nên sẽ không có bất kỳ vấn đề đồng bộ hóa nào.
- Hệ thống dễ dàng mở rộng quy mô vì mỗi máy chủ web chịu trách nhiệm tạo ID mà chúng sử dụng. Trình tạo ID có thể dễ dàng mở rộng quy mô với các máy chủ web.

Nhược điểm:

- ID dài 128 bit, nhưng yêu cầu của chúng tôi là 64 bit.
- ID có thể không tăng theo thời gian.
- ID có thể không phải là số.

Máy chủ vé

Máy chủ ticket là một cách thú vị khác để tạo ID duy nhất. Flicker đã phát triển máy chủ ticket để tạo khóa chính phân tán [2]. Cần đề cập đến cách thức hoạt động của hệ thống.

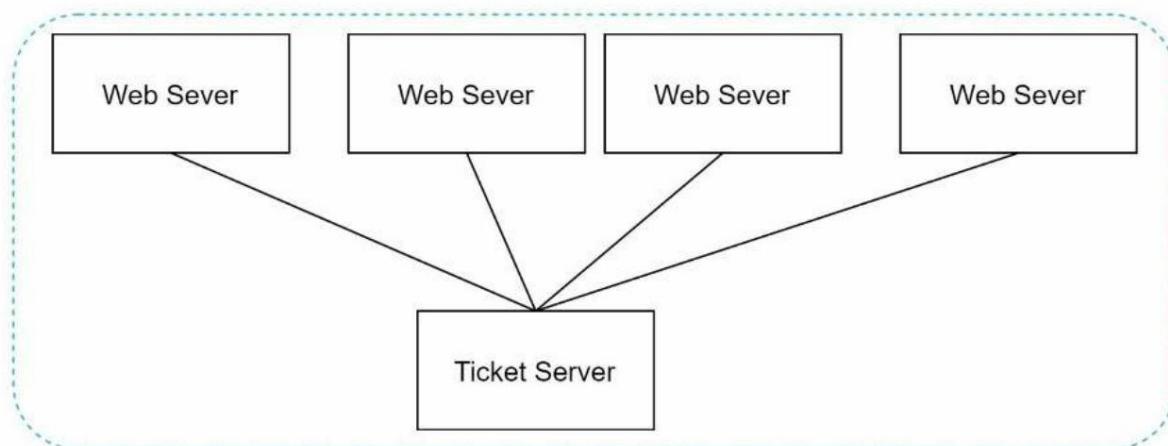


Figure 7-4

Ý tưởng là sử dụng tính năng auto_increment tập trung trong một máy chủ cơ sở dữ liệu duy nhất (Ticket Server). Để tìm hiểu thêm về điều này, hãy tham khảo bài viết trên blog kỹ thuật của flicker [2].

Ưu điểm:

- ID số.
- Dễ triển khai và phù hợp với các ứng dụng vừa và nhỏ.

Nhược điểm:

- Điểm lỗi đơn. Máy chủ ticket đơn có nghĩa là nếu máy chủ ticket ngừng hoạt động, tất cả các hệ thống phụ thuộc vào nó sẽ gặp sự cố. Để tránh điểm lỗi đơn, chúng ta có thể thiết lập nhiều máy chủ ticket. Tuy nhiên, điều này sẽ tạo ra những thách thức mới như đồng bộ hóa dữ liệu.

Phương pháp bông tuyết Twitter Các phương pháp

phương pháp được đề cập ở trên cung cấp cho chúng ta một số ý tưởng về cách thức hoạt động của các hệ thống tạo ID khác nhau. Tuy nhiên, không có phương pháp nào trong số chúng đáp ứng được các yêu cầu cụ thể của chúng tôi; do đó, chúng tôi cần một phương pháp khác. Hệ thống tạo ID đặc đáo của Twitter có tên là "bông tuyết" [3] rất truyền cảm hứng và có thể đáp ứng được các yêu cầu của chúng tôi.

Chia sẻ trị là bạn của chúng ta. Thay vì tạo ID trực tiếp, chúng ta chia ID thành các phần khác nhau. Hình 7-5 cho thấy bố cục của ID 64 bit.

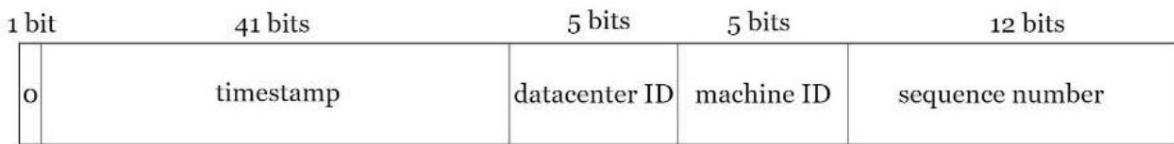


Figure 7-5

Mỗi phần được giải thích bên dưới. • Bit

dấu: 1 bit. Nó sẽ luôn là 0. Điều này được dành riêng cho những mục đích sử dụng trong tương lai. Nó có khả năng được sử dụng để phân biệt giữa số có dấu và không dấu. • Đầu thời gian: 41

bit. Mili giây kể từ kỷ nguyên hoặc kỷ nguyên tùy chỉnh. Chúng tôi sử dụng kỷ nguyên mặc định của Twitter snowflake là 1288834974657, tương đương với ngày 04 tháng 11 năm 2010, 01:42:54 UTC. • ID trung tâm dữ liệu: 5

bit, cung cấp cho chúng tôi $2^5 = 32$ trung tâm dữ liệu. • ID máy: 5 bit, cung cấp cho chúng tôi $2^5 = 32$ máy trên mỗi trung tâm dữ liệu. • Số thứ tự: 12 bit. Đối với mỗi ID được tạo trên máy/quy trình đó, số thứ tự sẽ tăng thêm 1. Số này được đặt lại thành 0 sau mỗi mili giây.

Bước 3 - Thiết kế sâu Trong thiết kế

cấp cao, chúng ta đã thảo luận về các tùy chọn khác nhau để thiết kế một trình tạo ID duy nhất trong các hệ thống phân tán. Chúng ta quyết định một phưƠng pháp dựa trên trình tạo ID bÔng tuyêt Twitter. Chúng ta hãy đi sâu vào thiết kế. Để làm mới trí nhớ của chúng ta, sơ đồ thiết kế được liệt kê lại bên dưới.

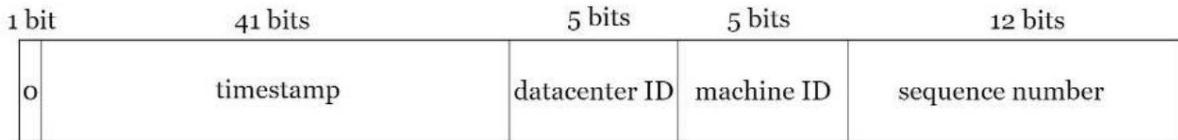


Figure 7-6

ID trung tâm dữ liệu và ID máy được chọn tại thời điểm khởi động, thường được sửa sau khi hệ thống chạy. Bất kỳ thay đổi nào trong ID trung tâm dữ liệu và ID máy đều cần được xem xét cẩn thận vì việc vô tình thay đổi các giá trị đó có thể dẫn đến xung đột ID. Dấu thời gian và số thứ tự được tạo khi trình tạo ID đang chạy.

Dấu thời gian

41 bit quan trọng nhất tạo nên phần dấu thời gian. Khi dấu thời gian tăng theo thời gian, ID có thể sắp xếp theo thời gian. Hình 7-7 cho thấy ví dụ về cách biểu diễn nhị phân được chuyển đổi thành UTC. Bạn cũng có thể chuyển đổi UTC trở lại biểu diễn nhị phân bằng phưƠng pháp tương tự.

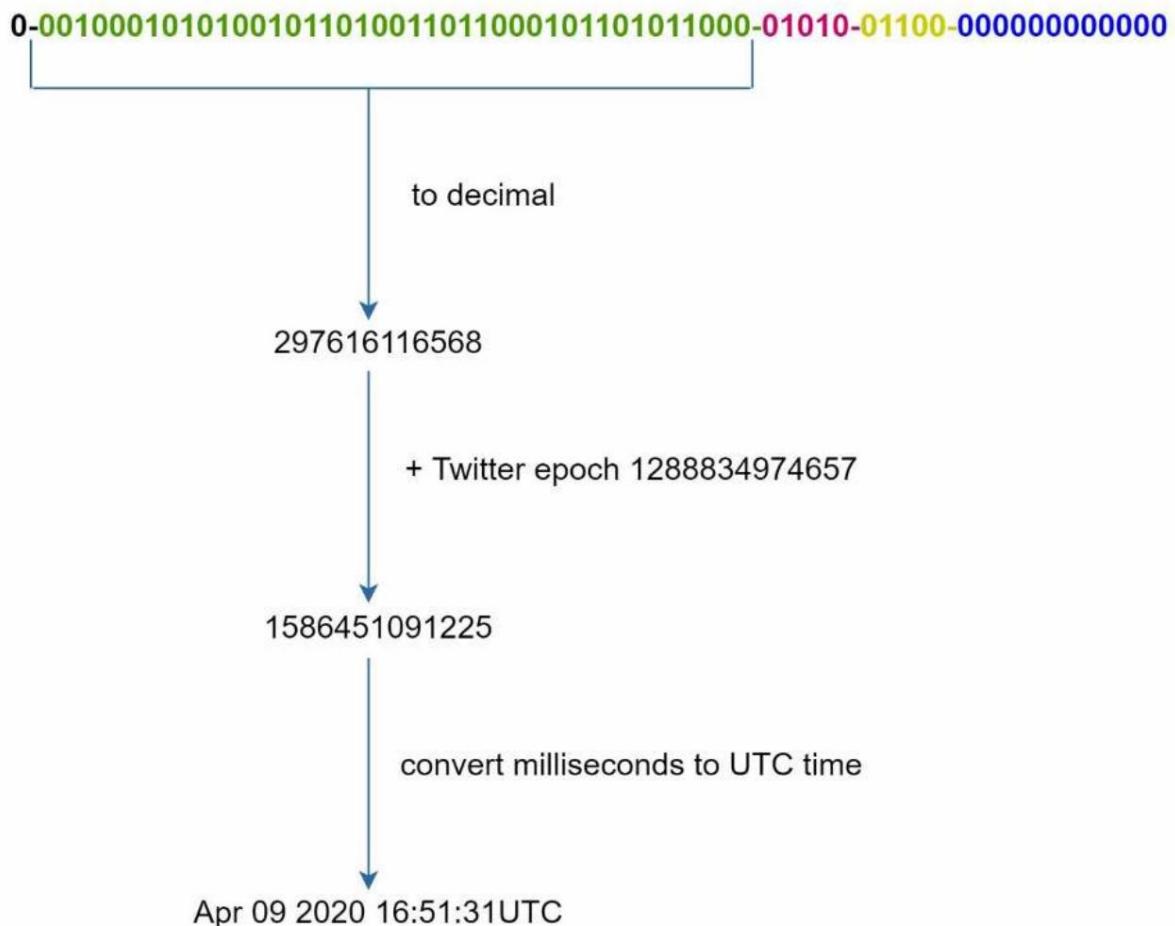


Figure 7-7

Dấu thời gian tối đa có thể được biểu diễn trong 41 bit là

$2^{41} - 1 = 2199023255551$ mili giây (ms), cho chúng ta: ~ 69 năm =

2199023255551 ms / 1000 giây / 365 ngày / 24 giờ / 3600 giây. Điều này có nghĩa là trình tạo ID sẽ hoạt động trong 69 năm và việc có thời gian kỷ nguyên tùy chỉnh gần với ngày hôm nay sẽ trì hoãn thời gian tràn. Sau 69 năm, chúng ta sẽ cần thời gian kỷ nguyên mới hoặc áp dụng các kỹ thuật khác để di chuyển ID.

Số thứ tự Số thứ tự

là 12 bit, cung cấp cho chúng ta $2^{12} = 4096$ kết hợp. Trừ ờng này là 0 trừ khi có nhiều hơn một ID được tạo trong một mili giây trên cùng một máy chủ. Về lý thuyết, một máy có thể hỗ trợ tối đa 4096 ID mới mỗi mili giây.

Bút ớc 4 - Tổng kết Trong

chư ơng này, chúng ta đã thảo luận về các cách tiếp cận khác nhau để thiết kế trình tạo ID duy nhất: sao chép địa chủ, UUID, máy chủ ticket và trình tạo ID duy nhất giống như Twitter Snowflake. Chúng tôi quyết định sử dụng Snowflake vì nó hỗ trợ tất cả các trường hợp sử dụng của chúng tôi và có thể mở rộng trong môi trường phân tán.

Nếu còn thời gian vào cuối buổi phòng vấn, sau đây là một số điểm cần thảo luận thêm:

- Đồng bộ hóa đồng hồ. Trong thiết kế của chúng tôi, chúng tôi giả định các máy chủ tạo ID có cùng đồng hồ. Giả định này có thể không đúng khi máy chủ chạy trên nhiều lỗi. Thách thức tương tự tồn tại trong các tình huống nhiều máy. Các giải pháp đồng bộ hóa đồng hồ nằm ngoài phạm vi của cuốn sách này; tuy nhiên, điều quan trọng là phải hiểu vấn đề tồn tại.

Giao thức thời gian mạng là giải pháp phổ biến nhất cho vấn đề này. Đối với đặc điểm quan trọng, hãy tham khảo tài liệu tham khảo [4]. • Điều chỉnh độ

dài phần. Ví dụ, ít số thứ tự hơn như ng nhiều bit dấu thời gian hơn có hiệu quả đối với các ứng dụng đồng thời thấp và dài hạn. • Tính khả dụng cao. Vì trình tạo ID là hệ thống quan trọng đối với nhiệm vụ nên nó phải có tính khả dụng cao.

Xin chúc mừng vì đã đi đư ợc đến đây! Bây giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

[1] Mã định danh duy nhất toàn cầu: https://en.wikipedia.org/wiki/Universally_unique_identifier

[2] Máy chủ vé: Phân phối khóa chính duy nhất với giá rẻ:

<https://code.flickr.net/2010/02/08/ticket-servers-distributed-unique-primary-keys-on-the-cheap/>

[3] Công bố Snowflake: https://blog.twitter.com/engineering/en_us/a/2010/announcing-snowflake.html

[4] Giao thức thời gian mạng: https://en.wikipedia.org/wiki/Network_Time_Protocol

CHÚ Ó NG 8: THIẾT KẾ URL RÚT GỌN

Trong chương này, chúng ta sẽ giải quyết một câu hỏi phỏng vấn thiết kế hệ thống thú vị và kinh điển: thiết kế dịch vụ rút gọn URL như tinyurl.

Bút ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế Các câu hỏi phỏng vấn thiết kế hệ thống đư ợc cô ý để mở. Để thiết kế một hệ thống đư ợc thiết kế tốt, điều quan trọng là phải đặt câu hỏi làm rõ.

Ứng viên: Bạn có thể đưa ra ví dụ về cách thức hoạt động của trình rút gọn URL không?

Người phỏng vấn: Giả sử URL

<https://www.systeminterview.com/q=chatsystem&c=loggedin&v=v3&l=long> là URL gốc. Dịch vụ của bạn tạo một bí danh có độ dài ngắn hơn: <https://tinyurl.com/y7keocwj>. Nếu bạn nhấp vào bí danh, nó sẽ chuyển hướng bạn đến URL gốc.

Ứng viên: Lượng giao thông là bao nhiêu?

Người phỏng vấn: 100 triệu URL đư ợc tạo ra mỗi ngày.

Ứng viên: URL rút gọn dài bao nhiêu?

Người phỏng vấn: Càng ngắn càng tốt.

Ứng viên: Những ký tự nào đư ợc phép có trong URL rút gọn?

Người phỏng vấn: URL rút gọn có thể là sự kết hợp của các số (0-9) và ký tự (az, AZ).

Ứng viên: Có thể xóa hoặc cập nhật các URL rút gọn không?

Người phỏng vấn: Để đơn giản, chúng ta hãy giả sử rằng các URL rút gọn không thể bị xóa hoặc cập nhật.

Sau đây là các trường hợp sử dụng cơ bản:

1. Rút ngắn URL: cung cấp một URL dài => trả về một URL ngắn hơn nhiều 2. Chuyển hướng URL: cung cấp một URL ngắn hơn => chuyển hướng đến URL gốc 3. Cảnh báo về tính khả dụng cao, khả năng mở rộng và khả năng chịu lỗi

Ước tính sơ bộ

- Hoạt động ghi: 100 triệu URL đư ợc tạo ra mỗi ngày. • Hoạt động ghi mỗi giây: $100 \text{ triệu} / 24 / 3600 = 1160$ • Hoạt động đọc: Giả sử tỷ lệ hoạt động đọc so với hoạt động ghi là 10:1, hoạt động đọc mỗi giây: $1160 * 10 = 11.600$ • Giả sử dịch vụ rút gọn URL sẽ chạy trong 10 năm, điều này có nghĩa là chúng ta phải hỗ trợ $100 \text{ triệu} * 365 * 10 = 365 \text{ tỷ bản ghi}$.

- Giả sử độ dài URL trung bình là 100.
- Yêu cầu lưu trữ trong 10 năm: $365 \text{ tỷ} * 100 \text{ byte} * 10 \text{ năm} = 365 \text{ TB}$

Điều quan trọng là bạn phải thảo luận các giả định và tính toán với người phỏng vấn để cả hai bên đều hiểu rõ.

BƯỚC 2 - ĐỀ XUẤT THIẾT KẾ CẤP CAO VÀ NHẬN ĐƯỢC SỰ CHẤP THUẬN

Trong phần này, chúng tôi thảo luận về các điểm cuối API, chuyển hướng URL và luồng rút ngắn URL.

Điểm cuối API Điểm

điểm cuối API tạo điều kiện thuận lợi cho việc giao tiếp giữa máy khách và máy chủ. Chúng tôi sẽ thiết kế API theo kiểu REST. Nếu bạn không quen với API RESTful, bạn có thể tham khảo tài liệu bên ngoài, chẳng hạn như tài liệu trong tài liệu tham khảo [1]. Một trình rút gọn URL chính cần hai điểm cuối API.

1. Rút gọn URL. Để tạo một URL ngắn mới, máy khách gửi yêu cầu POST, trong đó có một tham số: URL dài ban đầu. API trông như thế này: POST api/v1/data/shorten • tham số yêu cầu: {longUrl}:

longURLString) • trả về shortURL

2. Chuyển hướng URL. Để chuyển hướng một URL ngắn đến URL dài ứng, máy khách sẽ gửi yêu cầu GET. API trông như thế này:

NHẬN api/v1/shortUrl

- Trả về longURL để chuyển hướng HTTP

Chuyển hướng URL Hình

8-1 cho thấy điều gì xảy ra khi bạn nhập tinyurl vào trình duyệt. Khi máy chủ nhận được yêu cầu tinyurl, nó sẽ thay đổi URL ngắn thành URL dài bằng chuyển hướng 301.

```

Request URL: https://tinyurl.com/qtj5opu
Request Method: GET
Status Code: 🌐 301
Remote Address: [2606:4700:10::6814:391e]:443
Referrer Policy: no-referrer-when-downgrade

▼ Response Headers
alt-svc: h3-27=:443; ma=86400, h3-25=:443; ma=86400, h3-24=:443; ma=86400, h3-23=:443; ma=86400
cache-control: max-age=0, no-cache, private
cf-cache-status: DYNAMIC
cf-ray: 581fb8ac986ed33-SJC
content-type: text/html; charset=UTF-8
date: Fri, 10 Apr 2020 22:00:23 GMT
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
location: https://www.amazon.com/dp/B017V4NTFA?pLink=63eaef76-979c-4d&ref=adblp13nvvxx_0_2_im

```

Figure 8-1

Giao tiếp chi tiết giữa máy khách và máy chủ được thể hiện trong Hình 8-2.

short URL: <https://tinyurl.com/q7j5opu>
 long URL: https://www.amazon.com/dp/B017V4NTFA?pLink=63eaef76-979c-4d&ref=adblp13nvvxx_0_2_im

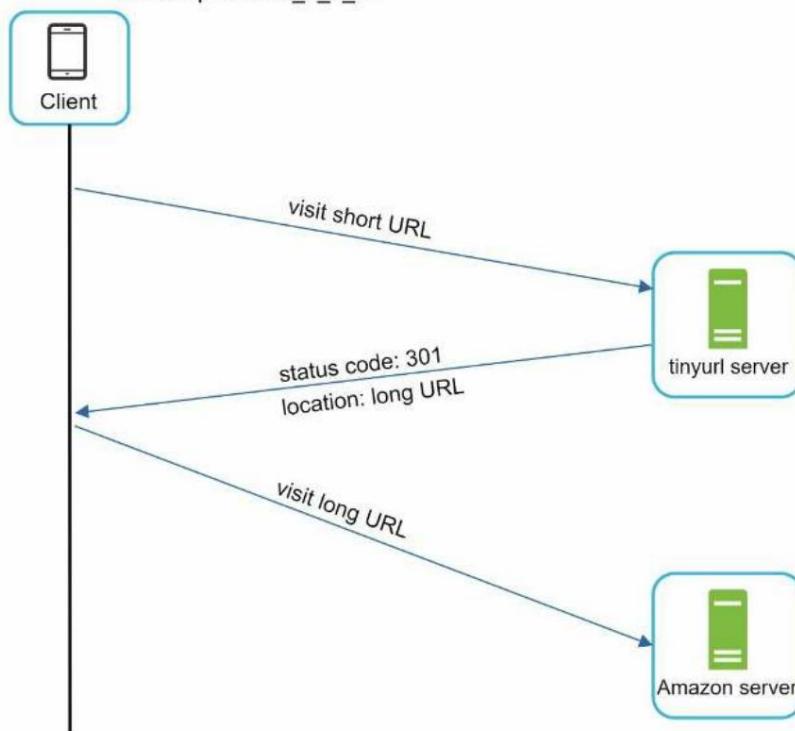


Figure 8-2

Một điều đáng thảo luận ở đây là chuyển hướng 301 so với chuyển hướng

302. Chuyển hướng 301. Chuyển hướng 301 cho thấy URL được yêu cầu được chuyển "vĩnh viễn" đến URL dài.

Vì nó được chuyển hướng vĩnh viễn, trình duyệt sẽ lưu trữ phản hồi và các yêu cầu tiếp theo cho cùng một URL sẽ không được gửi đến dịch vụ rút gọn URL.

Thay vào đó, các yêu cầu được chuyển hướng trực tiếp đến máy chủ URL dài.

Chuyển hướng 302. Chuyển hướng 302 có nghĩa là URL được "tạm thời" chuyển đến URL dài, nghĩa là các yêu cầu tiếp theo cho cùng một URL sẽ được gửi đến dịch vụ rút gọn URL trước. Sau đó, chúng được chuyển hướng đến máy chủ URL dài.

Mỗi phương pháp chuyển hướng đều có ưu và nhược điểm riêng. Nếu ưu tiên là giảm tải cho máy chủ, sử dụng chuyển hướng 301 là hợp lý vì chỉ có yêu cầu đầu tiên của cùng một URL được gửi đến máy chủ rút gọn URL. Tuy nhiên, nếu phân tích là quan trọng, chuyển hướng 302 là lựa chọn tốt hơn vì nó có thể theo dõi tỷ lệ nhập và nguồn nhập dễ dàng hơn.

Cách trực quan nhất để triển khai chuyển hướng URL là sử dụng bảng băm. Giả sử bảng băm lưu trữ các cặp <shortURL, longURL>, chuyển hướng URL có thể được triển khai theo cách sau:

- Lấy longURL: longURL = hashTable.get(shortURL)
- Sau khi bạn lấy được longURL, hãy thực hiện chuyển hướng URL.

Rút gọn URL Giả sử URL

ngắn trông như thế này: www.tinyurl.com/{hashValue}. Để hỗ trợ trang hợp sử dụng rút gọn URL, chúng ta phải tìm một hàm băm fx ánh xạ một URL dài tới hashValue, như thể hiện trong Hình 8-3.

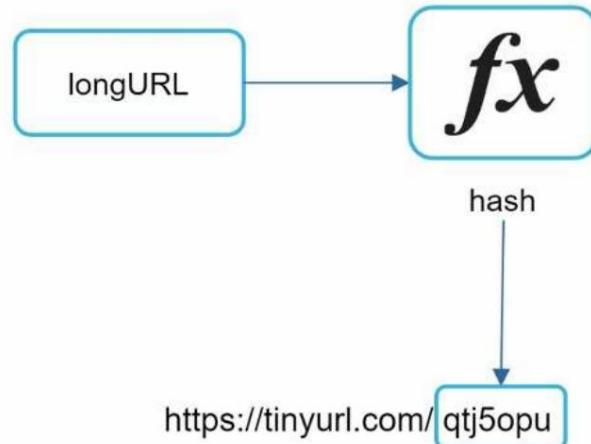


Figure 8-3

Hàm băm phải đáp ứng các yêu cầu sau:

- Mỗi longURL phải được băm thành một hashValue.
- Mỗi hashValue có thể được ánh xạ trở lại longURL.

Thiết kế chi tiết cho hàm băm được thảo luận sâu hơn.

Bài 3 - Thiết kế chuyên sâu

Cho đến bây giờ, chúng ta đã thảo luận về thiết kế cấp cao của việc rút ngắn URL và chuyển hướng URL. Trong phần này, chúng ta sẽ đi sâu vào những nội dung sau: mô hình dữ liệu, hàm băm, rút ngắn URL và chuyển hướng URL.

Mô hình dữ liệu

Trong thiết kế cấp cao, mọi thứ đều được lưu trữ trong một bảng băm. Đây là một điểm khởi đầu tốt; tuy nhiên, cách tiếp cận này không khả thi đối với các hệ thống trong thế giới thực vì tài nguyên bộ nhớ bị hạn chế và tối kén. Một lựa chọn tốt hơn là lưu trữ ánh xạ `<shortURL, longURL>` trong cơ sở dữ liệu quan hệ. Hình 8-4 cho thấy một thiết kế bảng cơ sở dữ liệu đơn giản. Phiên bản đơn giản hóa của bảng chứa 3 cột: `id`, `shortURL`, `longURL`.

url Table	
PK	<u>id (auto increment)</u>
	shortURL
	longURL

Figure 8-4

Hàm băm

Hàm băm được sử dụng để băm một URL dài thành một URL ngắn, còn được gọi là `hashValue`.

Độ dài giá trị băm

`HashValue` bao gồm các ký tự từ `[0-9, az, AZ]`, chứa $10 + 26 + 26 = 62$ ký tự có thể. Để tính độ dài của `hashValue`, hãy tìm n nhỏ nhất sao cho $62^n \geq 365$ tỷ. Hệ thống phải hỗ trợ tối đa 365 tỷ URL dựa trên ước tính mặt sau của phong bì. Bảng 8-1 hiển thị độ dài của `hashValue` và số lượng URL tối đa tương ứng mà nó có thể hỗ trợ.

N	Maximal number of URLs
1	$62^1 = 62$
2	$62^2 = 3,844$
3	$62^3 = 238,328$
4	$62^4 = 14,776,336$
5	$62^5 = 916,132,832$
6	$62^6 = 56,800,235,584$
7	$62^7 = 3,521,614,606,208 = \sim 3.5 \text{ trillion}$

Table 8-1

Khi $n = 7$, $62^n = \sim 3,5$ nghìn tỷ, $3,5$ nghìn tỷ là quá đủ để chứa 365 tỷ URL, do đó độ dài của hashValue là 7 .

Chúng ta sẽ khám phá hai loại hàm băm cho một trình rút gọn URL. Loại đầu tiên là "hash + giải quyết và chèm", và loại thứ hai là "chuyển đổi cơ sở 62 ". Chúng ta hãy xem xét từng loại một.

Giải quyết và chèm + băm

Để rút ngắn một URL dài, chúng ta nên triển khai một hàm băm băm một URL dài thành một chuỗi 7 ký tự. Một giải pháp đơn giản là sử dụng các hàm băm nổi tiếng như CRC32, MD5 hoặc SHA-1. Bảng sau đây so sánh kết quả băm sau khi áp dụng các hàm băm khác nhau trên URL này: https://en.wikipedia.org/wiki/Systems_design.

Hash function	Hash value (Hexadecimal)
CRC32	5cb54054
MD5	5a62509a84df9ee03fe1230b9df8b84e
SHA-1	0eeae7916c06853901d9ccbefbfcaf4de57ed85b

Table 8-2

Như thể hiện trong Bảng 8-2, ngay cả giá trị băm ngắn nhất (từ CRC32) cũng quá dài (hơn 7 ký tự). Làm thế nào chúng ta có thể làm cho nó ngắn hơn?

Cách tiếp cận đầu tiên là thu thập 7 ký tự đầu tiên của giá trị băm; tuy nhiên, phương pháp này có thể dẫn đến xung đột băm. Để giải quyết xung đột băm, chúng ta có thể thêm một chuỗi mới được xác định trước cho đến khi không còn xung đột nào được phát hiện. Quá trình này được giải thích trong Hình 8-

5.

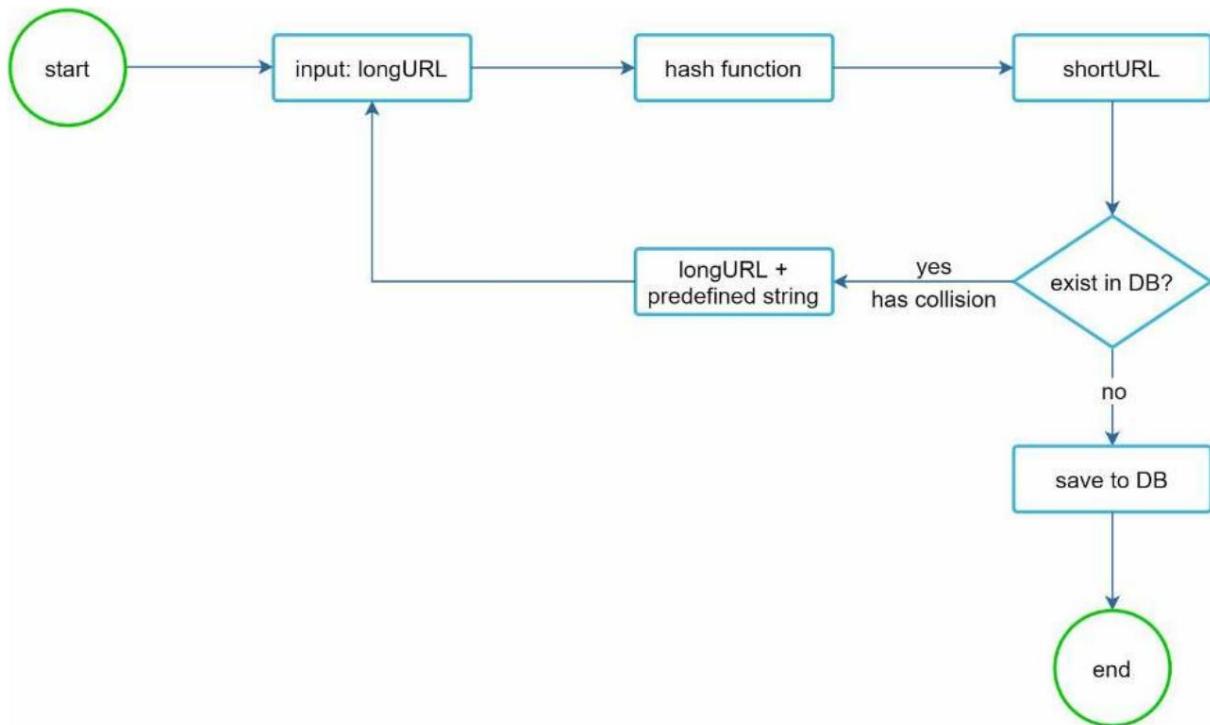


Figure 8-5

Phương pháp này có thể loại bỏ xung đột; tuy nhiên, việc truy vấn cơ sở dữ liệu để kiểm tra xem shortURL có tồn tại cho mọi yêu cầu hay không rất tốn kém. Một kỹ thuật gọi là bộ lọc bloom [2] có thể cải thiện hiệu suất. Bộ lọc bloom là một kỹ thuật xác suất tiết kiệm không gian để kiểm tra xem một phần tử có phải là thành viên của một tập hợp hay không. Tham khảo tài liệu tham khảo [2] để biết thêm chi tiết.

Chuyển đổi cơ số 62

Chuyển đổi cơ sở là một cách tiếp cận khác thường được sử dụng cho các trình rút gọn URL. Chuyển đổi cơ sở giúp chuyển đổi cùng một số giữa các hệ thống biểu diễn số khác nhau của nó. Chuyển đổi cơ sở 62 được sử dụng vì có 62 ký tự có thể có cho hashCode. Chúng ta hãy sử dụng một ví dụ để giải thích cách chuyển đổi hoạt động: chuyển đổi 1115710 sang biểu diễn cơ sở 62

(1115710 biểu thị 11157 trong hệ cơ số 10).

- Theo tên của nó, cơ số 62 là một cách sử dụng 62 ký tự để mã hóa. Các ánh xạ là: 0-0, ..., 9-9, 10-a, 11-b, ..., 35-z, 36-A, ..., 61-Z, trong đó 'a' biểu thị 10, 'Z' biểu thị 61,

vân vân.

- $1115710 = 2 \times 62^2 + 55 \times 62^1 + 59 \times 62^0 = [2, 55, 59] \rightarrow [2, T, X]$ trong cơ số 62

Biểu diễn. Hình 8-6 cho thấy quá trình hội thoại.

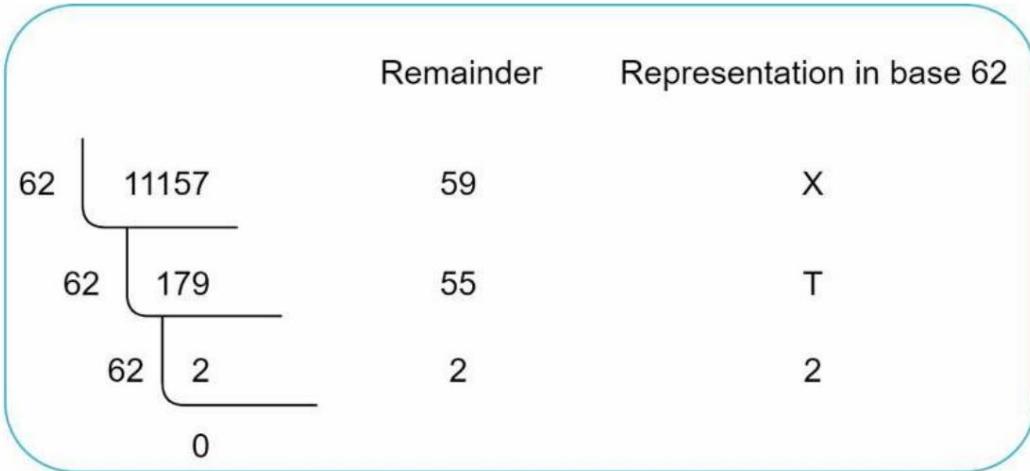


Figure 8-6

- Do đó, URL ngắn là <https://tinyurl.com/2TX> So sánh hai cách tiếp cận Bảng 8-3 cho thấy sự khác biệt giữa hai cách tiếp cận.

Hash + collision resolution	Base 62 conversion
Fixed short URL length.	The short URL length is not fixed. It goes up with the ID.
It does not need a unique ID generator.	This option depends on a unique ID generator.
Collision is possible and must be resolved.	Collision is impossible because ID is unique.
It is impossible to figure out the next available short URL because it does not depend on ID.	It is easy to figure out the next available short URL if ID increments by 1 for a new entry. This can be a security concern.

Table 8-3

Đi sâu vào rút gọn URL Là một trong những thành phần cốt lõi của hệ thống, chúng tôi muốn luồng rút gọn URL phải đơn giản về mặt logic và có chức năng. Chuyển đổi cơ sở 62 được sử dụng trong thiết kế của chúng tôi. Chúng tôi xây dựng sơ đồ sau (Hình 8-7) để chứng minh luồng.

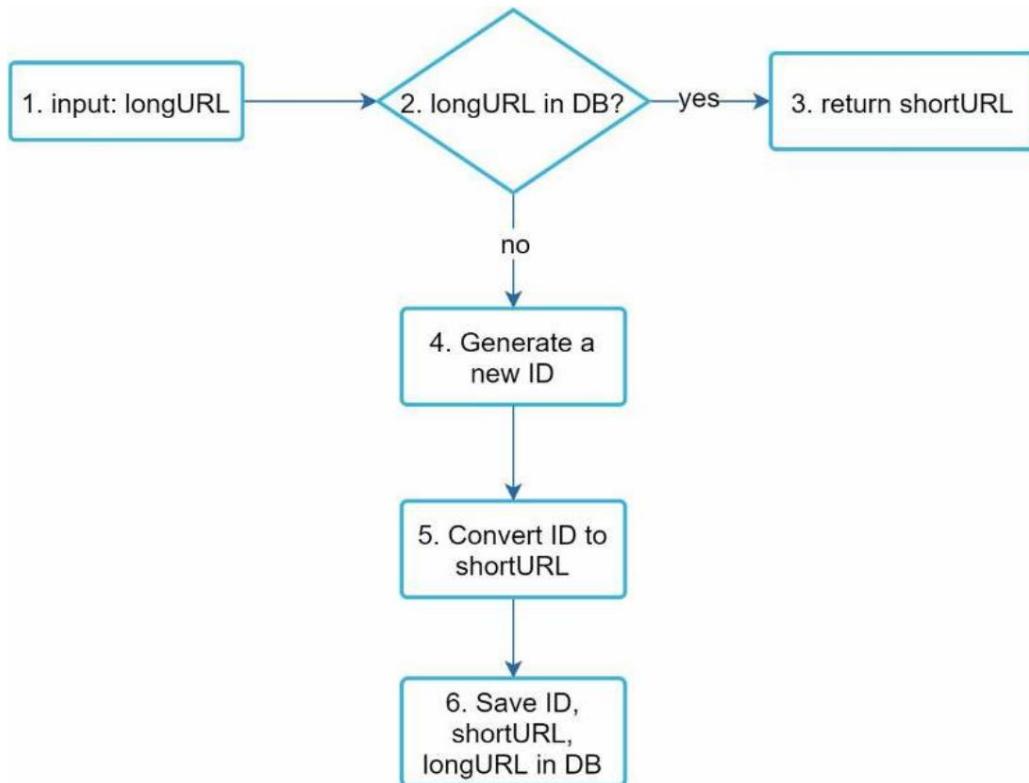


Figure 8-7

1. longURL là đầu vào.
 2. Hệ thống kiểm tra xem longURL có trong cơ sở dữ liệu hay không.
 3. Nếu có, điều đó có nghĩa là longURL đã được chuyển đổi thành shortURL trước đó. Trong trường hợp này, hãy lấy shortURL từ cơ sở dữ liệu và trả về cho máy khách.
 4. Nếu không, longURL là mới. Một ID duy nhất mới (khóa chính) được tạo bởi trình tạo ID duy nhất.
 5. Chuyển đổi ID thành shortURL bằng cách chuyển đổi cơ số 62.
 6. Tạo một hàng cơ sở dữ liệu mới với ID, shortURL và longURL.
- Để hiểu rõ hơn về luồng này, chúng ta hãy xem một ví dụ cụ thể.
- Giả sử đầu vào longURL là: https://en.wikipedia.org/wiki/Systems_design • Trình tạo ID duy nhất trả về ID: 2009215674938. • Chuyển đổi ID thành shortURL bằng cách sử dụng phép chuyển đổi cơ số 62. ID (2009215674938) được chuyển đổi thành "zn9edcu".
 - Lưu ID, shortURL và longURL vào cơ sở dữ liệu như thể hiện trong Bảng 8-4.

id	shortURL	longURL
2009215674938	zn9edcu	https://en.wikipedia.org/wiki/Systems_design

Table 8-4

Trình tạo ID duy nhất phân tán đáng được đề cập đến. Chức năng chính của nó là tạo ra các ID duy nhất toàn cầu, được sử dụng để tạo shortURL. Trong một

môi trường, việc triển khai một trình tạo ID duy nhất là một thách thức. May mắn thay, chúng tôi đã thảo luận một số giải pháp trong "Chương 7: Thiết kế Trình tạo ID duy nhất trong Hệ thống phân tán". Bạn có thể tham khảo lại để làm mới trí nhớ của mình.

Chuyển hướng URL sâu hơn Hình 8-8

cho thấy thiết kế chi tiết của chuyển hướng URL. Vì có nhiều lần đọc hơn ghi, nên ánh xạ <shortURL, longURL> được lưu trữ trong bộ nhớ đệm để cải thiện hiệu suất.

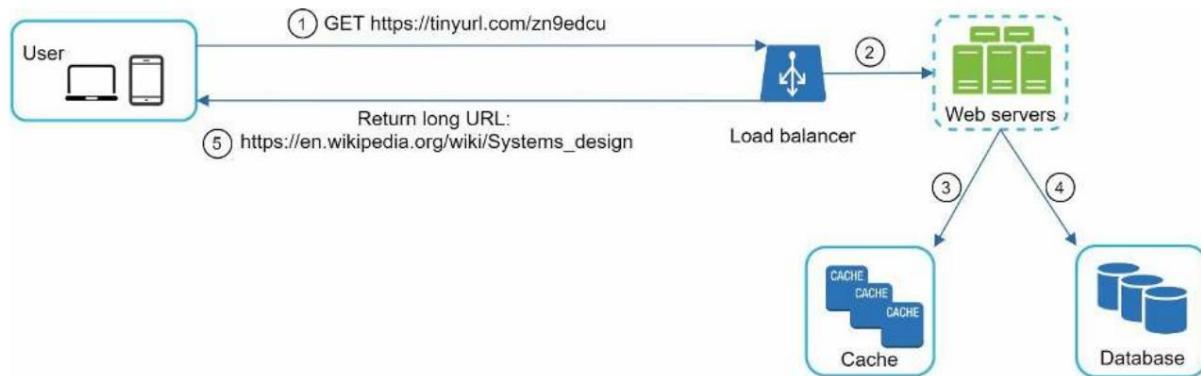


Figure 8-8

Luồng chuyển hướng URL được tóm tắt như sau:

1. Người dùng nhập vào liên kết URL ngắn: <https://tinyurl.com/zn9edcu>
2. Bộ cân bằng tải chuyển tiếp yêu cầu đến máy chủ web.
3. Nếu shortURL đã có trong bộ nhớ đệm, hãy trả về longURL trực tiếp.
4. Nếu shortURL không có trong bộ nhớ đệm, hãy lấy longURL từ cơ sở dữ liệu. Nếu không có trong cơ sở dữ liệu, có khả năng người dùng đã nhập shortURL không hợp lệ.
5. longURL được trả về cho người dùng.

Bài 4 - Tóm tắt

Trong chương này, chúng ta đã nói về thiết kế API, mô hình dữ liệu, hàm băm, rút ngắn URL và chuyển hướng URL.

Nếu còn thời gian vào cuối buổi phỏng vấn, sau đây là một vài điểm cần nói thêm.

- Bộ giới hạn tốc độ: Một vấn đề bảo mật tiềm ẩn mà chúng ta có thể gặp phải là người dùng có mục đích xấu gửi một số lượng lớn các yêu cầu rút ngắn URL. Bộ giới hạn tốc độ giúp lọc các yêu cầu dựa trên địa chỉ IP hoặc các quy tắc lọc khác. Nếu bạn muốn làm mới lại trí nhớ của mình về giới hạn tốc độ, hãy tham khảo "Chương 4: Thiết kế bộ giới hạn tốc độ". • Mở rộng

quy mô máy chủ web: Vì tầng web không có trạng thái nên có thể dễ dàng mở rộng tầng web bằng cách thêm hoặc xóa máy chủ web. • Mở

rộng quy mô cơ sở dữ liệu: Sao chép và phân mảnh cơ sở dữ liệu là các kỹ thuật phổ biến.

- Phân tích: Dữ liệu ngày càng quan trọng đối với thành công của doanh nghiệp. Việc tích hợp giải pháp phân tích vào bộ rút ngắn URL có thể giúp trả lời các câu hỏi quan trọng như có bao nhiêu người truy cập vào liên kết? Họ nhấp vào liên kết khi nào? v.v. •

Tính khả dụng, tính nhất quán và độ tin cậy. Các khái niệm này là cốt lõi của sự thành công của bất kỳ hệ thống lớn nào. Chúng tôi đã thảo luận chi tiết về chúng trong Chương 1, vui lòng làm mới trí nhớ của bạn về các chủ đề này.

Xin chúc mừng vì đã đi được đến đây! Hãy giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

- [1] Hướng dẫn RESTful: <https://www.restapitutorial.com/index.html>
- [2] Bộ lọc Bloom: https://en.wikipedia.org/wiki/Bloom_filter

CHƯƠNG 9: THIẾT KẾ WEB CRAWLER

Trong chương này, chúng ta tập trung vào thiết kế trình thu thập dữ liệu web: một câu hỏi phỏng vấn thiết kế hệ thống thú vị và có điểm.

Web crawler được gọi là robot hoặc spider. Nó được các công cụ tìm kiếm sử dụng rộng rãi để khám phá nội dung mới hoặc nội dung đã được cập nhật trên web. Nội dung có thể là một trang web, hình ảnh, video, tệp PDF, v.v. Web crawler bắt đầu bằng cách thu thập một vài trang web và sau đó theo dõi các liên kết trên các trang đó để thu thập nội dung mới. Hình 9-1 cho thấy một ví dụ trực quan về quá trình thu thập.

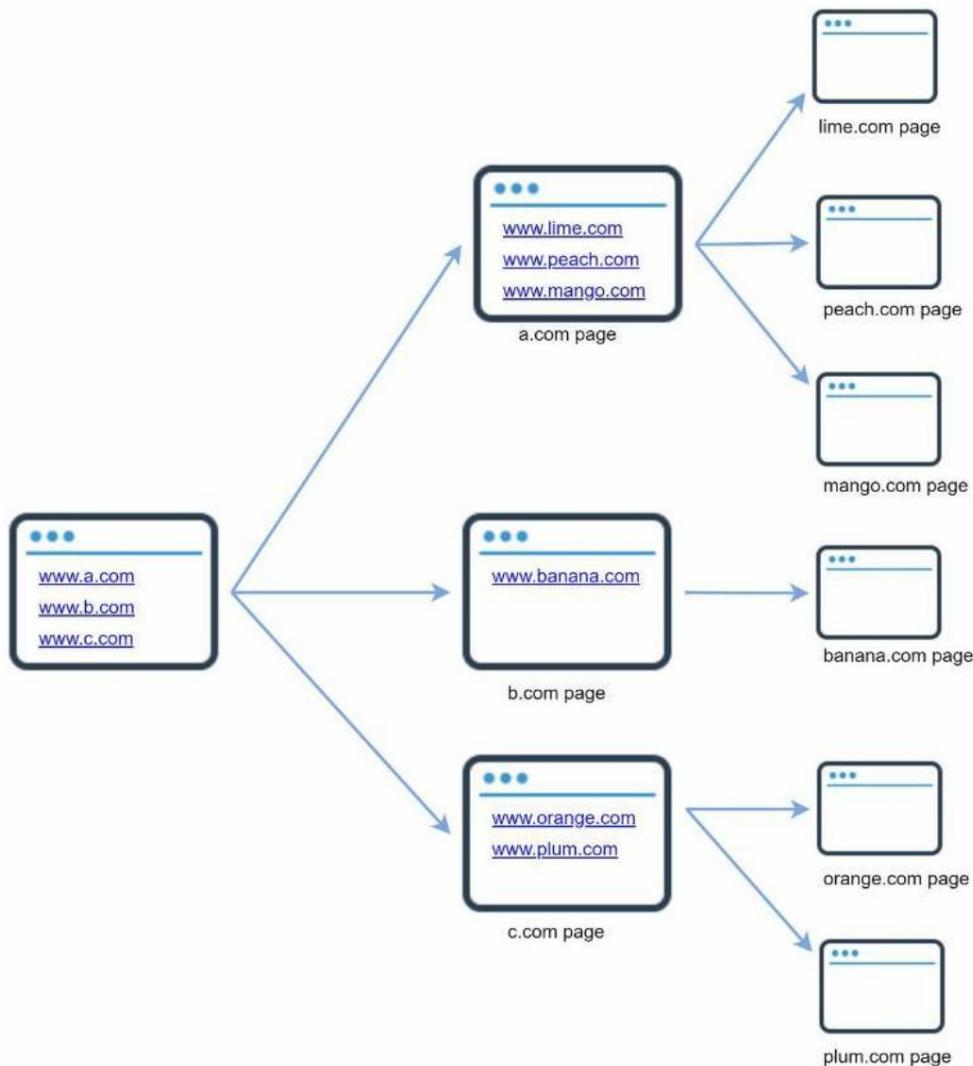


Figure 9-1

Trình thu thập thông tin được sử dụng cho nhiều mục đích:

- Lập chỉ mục công cụ tìm kiếm: Đây là truy cập hợp sử dụng phổ biến nhất. Một trình thu thập thông tin thu thập các trang web để tạo chỉ mục cục bộ cho các công cụ tìm kiếm. Ví dụ: Googlebot là trình thu thập thông tin web đầu tiên sau công cụ tìm kiếm Google.
- Lưu trữ web: Đây là quá trình thu thập thông tin từ web để bảo quản dữ liệu cho mục đích sử dụng trong tương lai. Ví dụ, nhiều thư viện quốc gia chạy trình thu thập thông tin để lưu trữ các trang web. Các ví dụ đáng chú ý là Thư viện Quốc hội Hoa Kỳ [1] và kho lưu trữ web EU [2].
- Khai thác web: Sự phát triển bùng nổ của web mang đến một cơ hội chưa từng có cho

khai thác dữ liệu. Khai thác web giúp khám phá kiến thức hữu ích từ internet. Ví dụ, các công ty tài chính hàng đầu sử dụng trình thu thập thông tin để tải xuống các cuộc họp cổ đông và báo cáo thư ờng niên để tìm hiểu các sáng kiến quan trọng của

công ty. • Giám sát web. Trình thu thập thông tin giúp giám sát các hành vi vi phạm bản quyền và nhãn hiệu trên Internet. Ví dụ, Digimarc [3] sử dụng trình thu thập thông tin để khám phá các tác phẩm và báo cáo vi phạm bản quyền.

Mức độ phức tạp của việc phát triển trình thu thập dữ liệu web phụ thuộc vào quy mô mà chúng tôi dự định hỗ trợ. Có thể là một dự án trư ờng học nhỏ, chỉ mất vài giờ để hoàn thành hoặc một dự án khổng lồ đòi hỏi sự cải tiến liên tục từ một nhóm kỹ sư chuyên dụng. Do đó, chúng tôi sẽ khám phá quy mô và các tính năng cần hỗ trợ bên dưới.

Bút ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế

Thuật toán cơ bản của trình thu thập dữ liệu web rất đơn giản:

1. Cho một tập hợp các URL, hãy tải xuống tất cả các trang web có địa chỉ được giải quyết bằng các URL đó.

2. Trích xuất URL từ các trang web này 3. Thêm URL

mới vào danh sách URL cần tải xuống. Lặp lại 3 bút ớc này.

Liệu một trình thu thập dữ liệu web có thực sự hoạt động đơn giản như thuật toán cơ bản này không? Không hẳn vậy.

Thiết kế một trình thu thập dữ liệu web có khả năng mở rộng quy mô lớn là một nhiệm vụ cực kỳ phức tạp. Không có khả năng bất kỳ ai có thể thiết kế một trình thu thập dữ liệu web lớn trong thời gian phỏng vấn. Trước khi bắt đầu thiết kế, chúng ta phải đặt câu hỏi để hiểu các yêu cầu và thiết lập phạm vi thiết kế: Ứng viên: Mục đích chính

của trình thu thập dữ liệu là gì? Nó được sử dụng để lập chỉ mục công cụ tìm kiếm, khai thác dữ liệu hay mục đích nào khác?

Người phỏng vấn: Lập chỉ mục công cụ tìm kiếm.

Ứng viên: Công cụ thu thập dữ liệu web thu thập bao nhiêu trang web mỗi tháng?

Người phỏng vấn: 1 tỷ trang.

Ứng viên: Những loại nội dung nào được bao gồm? Chỉ có HTML hay các loại nội dung khác như PDF và hình ảnh?

Người phỏng vấn: Chỉ có HTML.

Ứng viên: Chúng ta có nên xem xét các trang web mới được thêm hoặc chỉnh sửa không?

Người phỏng vấn: Vâng, chúng ta nên xem xét các trang web mới được thêm vào hoặc chỉnh sửa.

Ứng viên: Chúng ta có cần lưu trữ các trang HTML được thu thập từ web không?

Người phỏng vấn: Có, lên đến 5 năm

Ứng viên: Chúng ta xử lý các trang web có nội dung trùng lặp như thế nào?

Người phỏng vấn: Các trang có nội dung trùng lặp nên bị bỏ qua.

Trên đây là một số câu hỏi mẫu mà bạn có thể hỏi người phỏng vấn. Điều quan trọng là phải hiểu các yêu cầu và làm rõ những điều mơ hồ. Ngay cả khi bạn được yêu cầu thiết kế một sản phẩm đơn giản như trình thu thập dữ liệu web, bạn và người phỏng vấn có thể không có cùng giả định.

Bên cạnh các chức năng cần làm rõ với người phỏng vấn, điều quan trọng là phải lưu ý các đặc điểm sau của một trình thu thập dữ liệu web tốt:

- Khả năng mở rộng: Web

rất lớn. Có hàng tỷ trang web ngoài kia. Thu thập dữ liệu web phải cực kỳ hiệu quả khi sử dụng song song hóa.

- Độ bền: Web đầy rẫy những cạm bẫy. HTML xấu, máy chủ không phản hồi, sự cố, liên kết độc hại, v.v. đều là những lỗi thường gặp. Trình thu thập dữ liệu phải xử lý tất cả các trường hợp ngoại lệ đó.
- Sự lịch sự: Trình thu thập dữ liệu không nên thực hiện quá nhiều yêu cầu đến một trang web trong một khoảng thời gian ngắn.

- Khả năng mở rộng: Hệ thống linh hoạt nên chỉ cần thay đổi tối thiểu để hỗ trợ các loại nội dung mới. Ví dụ, nếu chúng ta muốn thu thập các tệp hình ảnh trong tương lai, chúng ta không cần phải thiết kế lại toàn bộ hệ thống.

Ước tính sơ bộ Các ước tính sau

đây dựa trên nhiều giả định và điều quan trọng là phải trao đổi với người phỏng vấn để có cùng quan điểm.

- Giả sử có 1 tỷ trang web được tải xuống mỗi tháng.

- QPS: $1.000.000.000 / 30 \text{ ngày} / 24 \text{ giờ} / 3600 \text{ giây} = \sim 400 \text{ trang mỗi giây.}$ • QPS
định = $2 * \text{QPS} = 800$ • Giả sử
kích thước trang web trung bình là 500k. •
1 tỷ trang x 500k = 500 TB lưu trữ mỗi tháng. Nếu bạn không rõ về các đơn vị lưu
trữ kỹ thuật số, hãy xem lại phần "Lũy thừa của 2" trong Chương
2. • Giả sử dữ liệu được lưu trữ trong năm năm, $500 \text{ TB} * 12 \text{ tháng} * 5 \text{ năm} = 30 \text{ PB.}$ Cần có
bộ lưu trữ 30 PB để lưu trữ nội dung trong năm năm.

Bứ ớc 2 - Đè xuất thiết kế cấp cao và nhận đư ợc sự đồng thuận Khi các yêu cầu đã rõ ràng, chúng tôi chuyển sang thiết kế cấp cao. Lấy cảm hứng từ các nghiên cứu trước đây về thu thập dữ liệu web [4] [5], chúng tôi đè xuất một thiết kế cấp cao như thể hiện trong Hình 9-2.

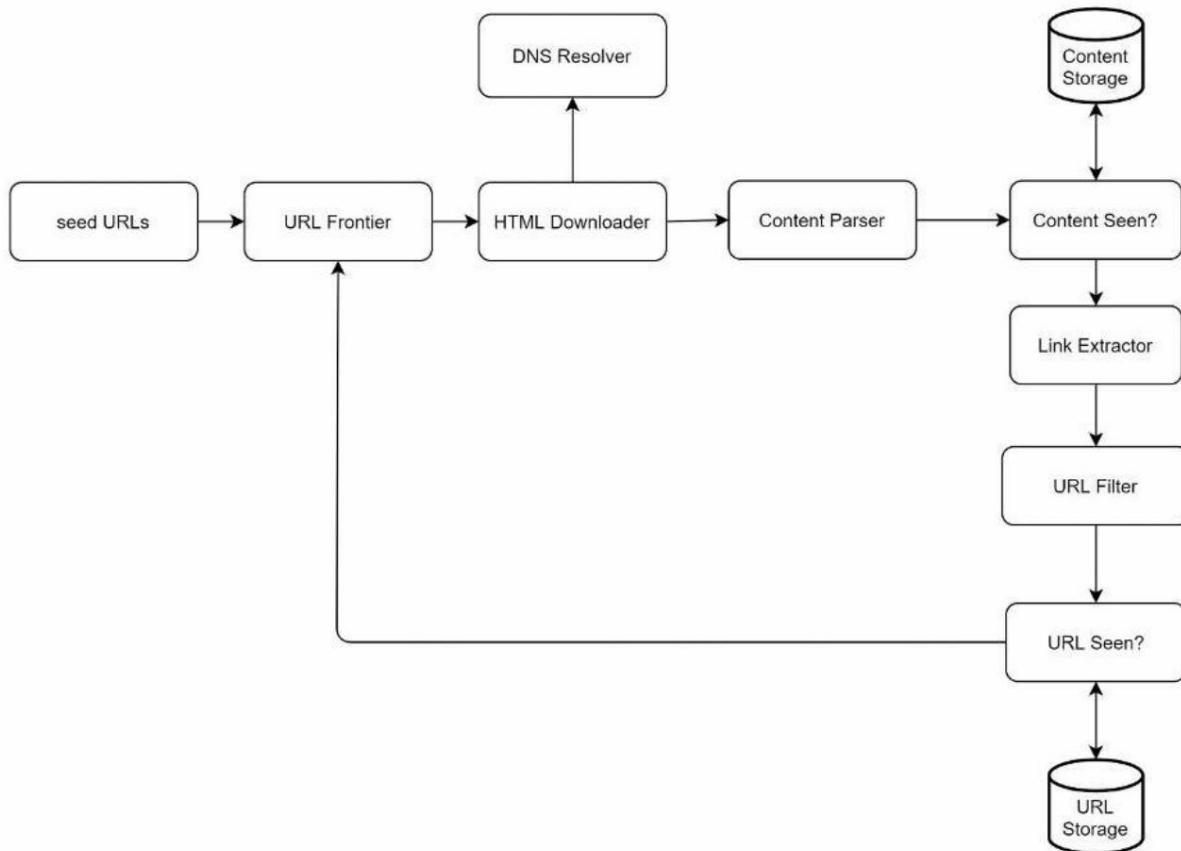


Figure 9-2

Đầu tiên, chúng tôi khám phá từng thành phần thiết kế để hiểu chức năng của chúng. Sau đó, chúng tôi kiểm tra quy trình làm việc của trình thu thập thông tin từng bứ ớc.

URL hạt giống

Trình thu thập dữ liệu web sử dụng URL hạt giống làm điểm khởi đầu cho quá trình thu thập dữ liệu. Ví dụ, để thu thập dữ liệu tất cả các trang web từ trang web của trường đại học, một cách trực quan để chọn URL hạt giống là sử dụng tên miền của trường đại học.

Để thu thập toàn bộ trang web, chúng ta cần sáng tạo trong việc lựa chọn URL hạt giống. Một URL hạt giống tốt đóng vai trò là điểm khởi đầu tốt mà trình thu thập có thể sử dụng để duyệt qua càng nhiều liên kết càng tốt.

Chiến lược chung là chia toàn bộ không gian URL thành các không gian nhỏ hơn. Phương pháp tiếp cận đầu tiên được đề xuất dựa trên địa phư ơng vì các quốc gia khác nhau có thể có các trang web phổ biến khác nhau.

Một cách khác là chọn URL hạt giống dựa trên chủ đề; ví dụ, chúng ta có thể chia không gian URL thành mua sắm, thể thao, chăm sóc sức khỏe, v.v. Việc lựa chọn URL hạt giống là một câu hỏi mở. Bạn không đư ợc mong đợi đưa ra câu trả lời hoàn hảo. Chỉ cần suy nghĩ thành tiếng.

Biên giới URL

Hầu hết các trình thu thập dữ liệu web hiện đại chia trạng thái thu thập dữ liệu thành hai: đang đư ợc tải xuống và đã đư ợc tải xuống. Thành phần lưu trữ URL cần tải xuống đư ợc gọi là URL Frontier.

Bạn có thể gọi đây là hàng đợi First-in-First-out (FIFO). Để biết thông tin chi tiết về URL Frontier, hãy tham khảo phần phân tích sâu.

Trình tải xuống HTML

Trình tải xuống HTML tải xuống các trang web từ internet. Các URL đó được cung cấp bởi URL Frontier.

Bộ giải quyết DNS

Để tải xuống một trang web, URL phải được dịch thành địa chỉ IP. HTML Downloader gọi DNS Resolver để lấy địa chỉ IP tương ứng cho URL. Ví dụ, URL www.wikipedia.org được chuyển đổi thành địa chỉ IP 198.35.26.96 tính đến ngày 3/5/2019.

Trình phân tích nội dung

Sau khi tải xuống một trang web, trang web đó phải được phân tích cú pháp và xác thực vì các trang web bị lỗi có thể gây ra sự cố và lãng phí không gian lưu trữ. Việc triển khai trình phân tích cú pháp nội dung trong máy chủ thu thập dữ liệu sẽ làm chậm quá trình thu thập dữ liệu. Do đó, trình phân tích cú pháp nội dung là một thành phần riêng biệt.

Đã xem nội dung?

Nghiên cứu trực tuyến [6] cho thấy 29% các trang web là nội dung trùng lặp, điều này có thể khiến cùng một nội dung được lưu trữ nhiều lần. Chúng tôi giới thiệu cấu trúc dữ liệu "Content Seen?" để loại bỏ sự trùng lặp dữ liệu và rút ngắn thời gian xử lý. Nó giúp phát hiện nội dung mới đã được lưu trữ trước đó trong hệ thống. Để so sánh hai tài liệu HTML, chúng ta có thể so sánh chúng theo từng ký tự. Tuy nhiên, phương pháp này chậm và tốn thời gian, đặc biệt là khi liên quan đến hàng tỷ trang web. Một cách hiệu quả để thực hiện nhiệm vụ này là so sánh các giá trị băm của hai trang web [7].

Lưu trữ nội dung

Đây là hệ thống lưu trữ để lưu trữ nội dung HTML. Việc lựa chọn hệ thống lưu trữ phụ thuộc vào các yếu tố như kiểu dữ liệu, kích thước dữ liệu, tần suất truy cập, tuổi thọ, v.v. Cả đĩa và bộ nhớ đều được sử dụng.

- Phần lớn nội dung được lưu trữ trên đĩa vì bộ dữ liệu quá lớn không thể chứa trong bộ nhớ.
- Nội dung phổ biến được lưu trong bộ nhớ để giảm độ trễ.

Trình trích xuất URL

URL Extractor phân tích cú pháp và trích xuất liên kết từ các trang HTML. Hình 9-3 cho thấy một ví dụ về quy trình trích xuất liên kết. Đường dẫn tương đối được chuyển đổi thành URL tuyệt đối bằng cách thêm tiền tố "<https://en.wikipedia.org>".

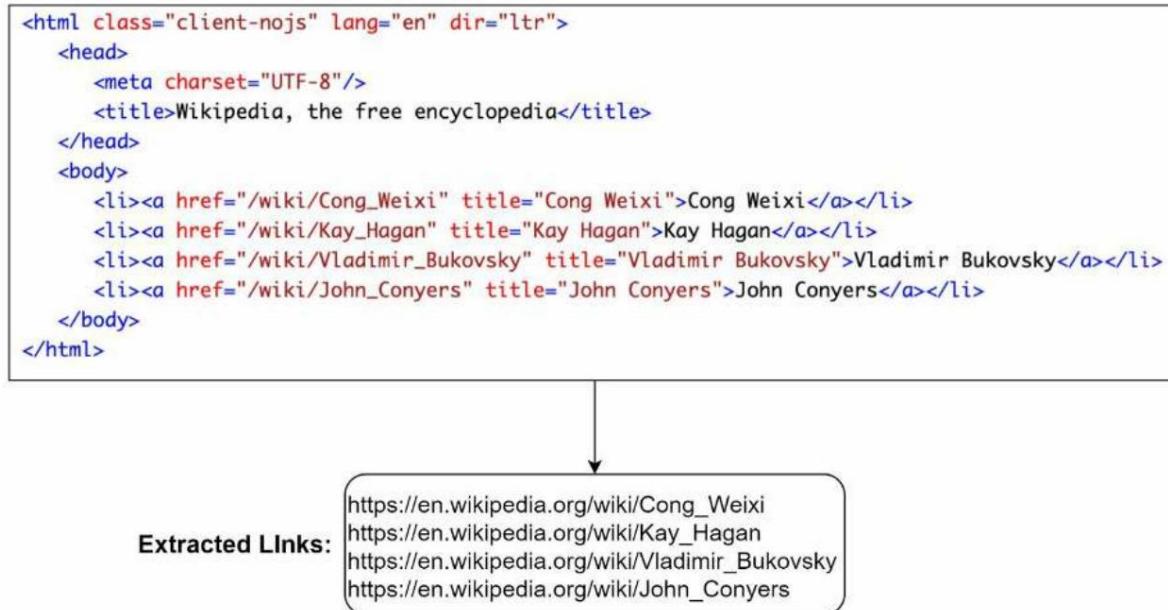


Figure 9-3

Bộ lọc URL

Bộ lọc URL loại trừ một số loại nội dung, phần mở rộng tệp, liên kết lõi và URL trong các trang web “đưa vào danh sách đen”.

Đã thấy URL?

“URL Seen?” là một cấu trúc dữ liệu theo dõi các URL đã được truy cập trước hoặc đã có trong Frontier. “URL Seen?” giúp tránh việc thêm cùng một URL nhiều lần vì điều này có thể làm tăng tải máy chủ và gây ra vòng lặp vô hạn tiềm ẩn.

Bộ lọc Bloom và bảng băm là các kỹ thuật phổ biến để triển khai thành phần “URL Seen?”. Chúng tôi sẽ không đề cập đến việc triển khai chi tiết bộ lọc Bloom và bảng băm ở đây. Để biết thêm thông tin, hãy tham khảo tài liệu tham khảo [4] [8].

Lưu trữ URL Lưu trữ

lưu trữ URL lưu trữ các URL đã truy cập.

Cho đến nay, chúng ta đã thảo luận về mọi thành phần của hệ thống. Tiếp theo, chúng ta sẽ ghép chúng lại với nhau để giải thích quy trình làm việc.

Quy trình làm việc của trình thu thập dữ liệu web

Để giải thích rõ hơn từng bước quy trình làm việc, số thứ tự đưa ra thêm vào sơ đồ thiết kế như thể hiện trong Hình 9-4.

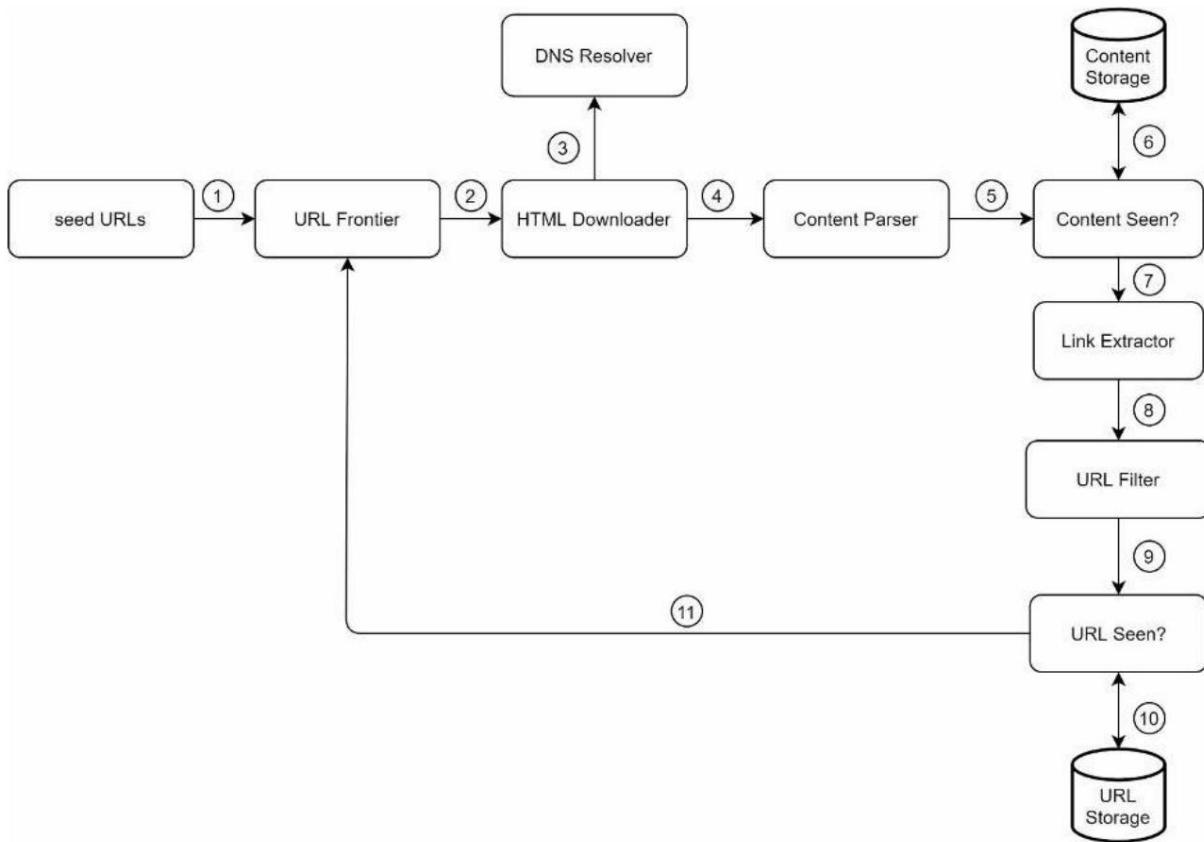


Figure 9-4

Bước 1: Thêm URL hạt giống vào URL Frontier

Bước 2: HTML Downloader sẽ lấy danh sách URL từ URL Frontier.

Bước 3: Trình tải xuống HTML lấy địa chỉ IP của URL từ trình phân giải DNS và bắt đầu tải xuống.

Bước 4: Content Parser phân tích các trang HTML và kiểm tra xem các trang có bị lỗi định dạng hay không.

Bước 5: Sau khi nội dung được phân tích và xác thực, nó sẽ được chuyển đến thành phần "Nội dung đã xem?"

Bước 6: Thành phần "Content Seen" kiểm tra xem trang HTML đã có trong bộ lưu trữ hay chưa.

- Nếu nó nằm trong bộ lưu trữ, điều này có nghĩa là cùng một nội dung trong một URL khác đã được xử lý. Trong trường hợp này, trang HTML bị loại bỏ.
- Nếu nó không nằm trong bộ lưu trữ, hệ thống chưa xử lý cùng một nội dung trước đó. Nội dung được chuyển đến Link Extractor.

Bước 7: Trình trích xuất liên kết sẽ trích xuất các liên kết từ các trang HTML.

Bước 8: Các liên kết được trích xuất sẽ được chuyển đến bộ lọc URL.

Bước 9: Sau khi các liên kết được lọc, chúng sẽ được chuyển đến thành phần "URL đã xem?"

Bước 10: Thành phần "URL Seen" kiểm tra xem URL đã có trong bộ lưu trữ hay chưa, nếu có, URL đã được xử lý trước đó và không cần phải làm gì thêm.

Bước 11: Nếu URL chưa được xử lý trước đó, nó sẽ được thêm vào URL Frontier.

Bài 9c - Thiết kế chuyên sâu

Cho đến bây giờ, chúng ta đã thảo luận về thiết kế cấp cao. Tiếp theo, chúng ta sẽ thảo luận sâu hơn về các thành phần và kỹ thuật xây dựng quan trọng nhất:

- Tìm kiếm theo chiều sâu (DFS) so với tìm kiếm theo chiều rộng (BFS)
- Biên giới URL
- Trình tải xuống HTML
- Độ bền
- Khả năng mở rộng • Phát hiện và tránh nội dung có vấn đề

DFS so với BFS

Bạn có thể nghĩ về web như một đồ thị có hướng, trong đó các trang web đóng vai trò là các nút và các siêu liên kết (URL) đóng vai trò là các cạnh. Quá trình thu thập dữ liệu có thể được coi là việc duyệt qua một đồ thị có hướng từ một trang web đến các trang web khác. Hai thuật toán duyệt đồ thị phổ biến là DFS và BFS. Tuy nhiên, DFS thường không phải là lựa chọn tốt vì độ sâu của DFS có thể rất sâu.

BFS thường được các trình thu thập dữ liệu web sử dụng và được triển khai theo hàng đợi vào trước ra trước (FIFO). Trong hàng đợi FIFO, các URL được đưa ra khỏi hàng đợi theo thứ tự chúng được đưa vào hàng đợi. Tuy nhiên, triển khai này có hai vấn

- đề:
 - Hầu hết các liên kết từ cùng một trang web đều được liên kết trở lại cùng một máy chủ. Trong Hình 9-5, tất cả các liên kết trong wikipedia.com đều là các liên kết nội bộ, khiến trình thu thập dữ liệu bận rộn xử lý các URL từ cùng một máy chủ (wikipedia.com). Khi trình thu thập dữ liệu cố gắng tải xuống các trang web song song, các máy chủ Wikipedia sẽ bị quá tải các yêu cầu. Điều này được coi là "thiếu lịch sự".

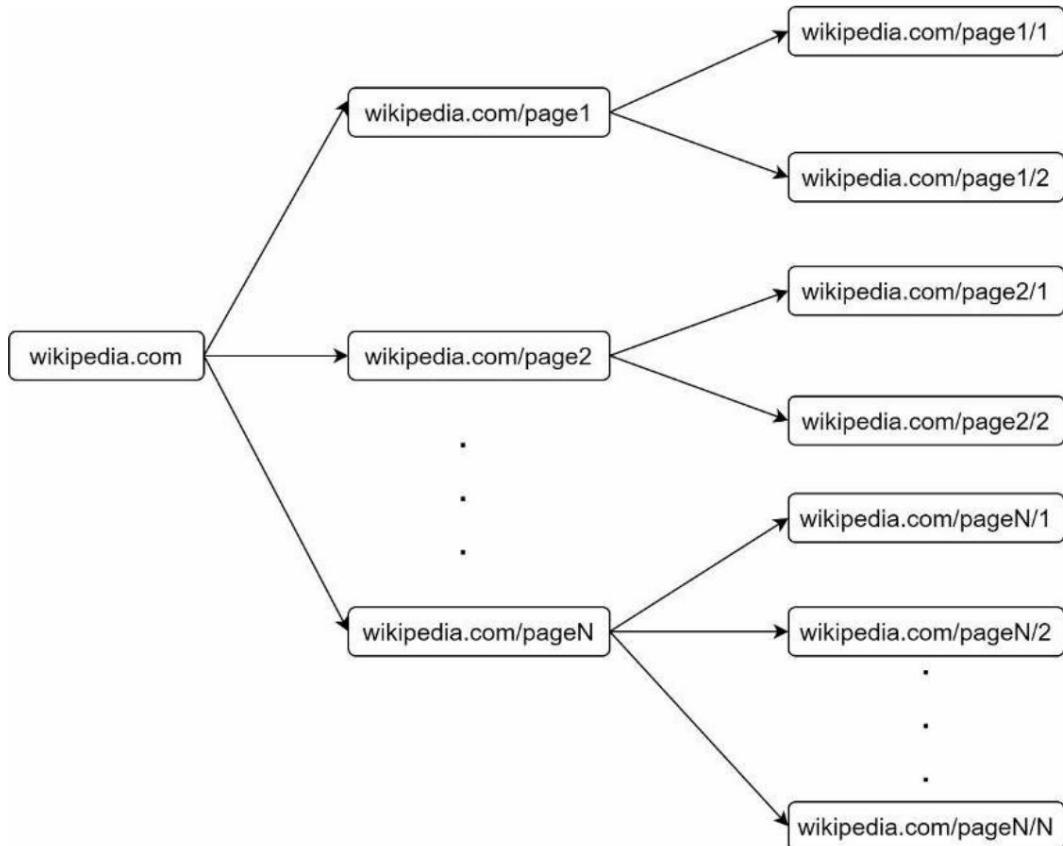


Figure 9-5

- BFS chuẩn không tính đến mức độ ưu tiên của URL. Web rất lớn và không phải mọi trang đều có cùng mức chất lượng và tầm quan trọng. Do đó, chúng ta có thể muốn ưu tiên URL theo thứ hạng trang, lưu lượn truy cập web, tần suất cập nhật, v.v.

Biên giới URL

Biên giới URL giúp giải quyết những vấn đề này. Biên giới URL là một cấu trúc dữ liệu lưu trữ các URL cần tải xuống. Biên giới URL là một thành phần quan trọng để đảm bảo tính lịch sự, ưu tiên URL và tính mới mẻ. Một số bài báo đáng chú ý về biên giới URL được đề cập trong các tài liệu tham khảo [5] [9]. Những phát hiện từ các bài báo này như sau:

Sự lịch sự

Nhìn chung, một trình thu thập dữ liệu web nên tránh gửi quá nhiều yêu cầu đến cùng một máy chủ lưu trữ trong một thời gian ngắn. Gửi quá nhiều yêu cầu được coi là "bất lịch sự" hoặc thậm chí được coi là tấn công từ chối dịch vụ (DOS). Ví dụ, không có bất kỳ ràng buộc nào, trình thu thập dữ liệu có thể gửi hàng nghìn yêu cầu mỗi giây đến cùng một trang web. Điều này có thể làm quá tải trang web máy chủ.

Ý tưởng chung về việc thực thi tính lịch sự là tải xuống từng trang một từ cùng một máy chủ. Có thể thêm độ trễ giữa hai tác vụ tải xuống. Ràng buộc lịch sự được triển khai bằng cách duy trì ánh xạ từ tên máy chủ trang web đến luồng tải xuống (công nhân).

Mỗi luồng tải xuống có một hàng đợi FIFO riêng và chỉ tải xuống các URL lấy được từ hàng đợi đó. Hình 9-6 cho thấy thiết kế quản lý sự lịch sự.

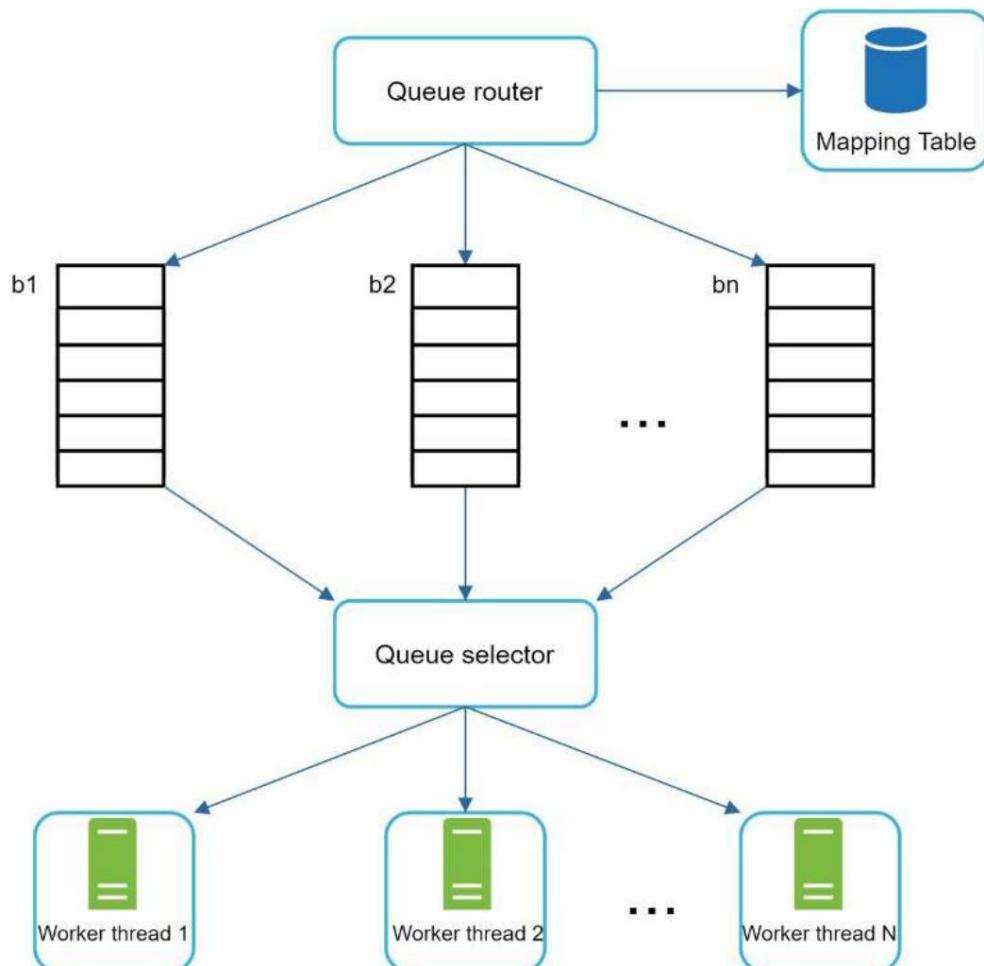


Figure 9-6

- Bộ định tuyến hàng đợi: Đảm bảo rằng mỗi hàng đợi (b_1, b_2, \dots, b_n) chỉ chứa các URL từ cùng một máy chủ.
- Bảng ánh xạ: Bảng này ánh xạ mỗi máy chủ vào một hàng đợi.

Host	Queue
wikipedia.com	b_1
apple.com	b_2
...	...
nike.com	b_n

Table 9-1

- Hàng đợi FIFO b_1, b_2 đến b_n : Mỗi hàng đợi chứa URL từ cùng một máy chủ.
- Bộ chọn hàng đợi: Mỗi luồng công nhân được ánh xạ đến một hàng đợi FIFO và chỉ tải xuống URL từ hàng đợi đó. Logic lựa chọn hàng đợi được thực hiện bởi bộ chọn hàng đợi.
- Luồng công nhân 1 đến N. Một luồng công nhân tải xuống từng trang web từ cùng một máy chủ. Có thể thêm độ trễ giữa hai tác vụ tải xuống.

Ưu tiên

Một bài đăng ngẫu nhiên từ diễn đàn thảo luận về các sản phẩm của Apple có trọng số rất khác so với các bài đăng trên trang chủ của Apple. Mặc dù cả hai đều có từ khóa "Apple", nhưng việc trình thu thập thông tin thu thập thông tin trang chủ của Apple trước là hợp lý.

Chúng tôi ưu tiên các URL dựa trên tính hữu ích, có thể được đo bằng PageRank [10], lưu lượng truy cập trang web, tần suất cập nhật, v.v. "Bộ ưu tiên" là thành phần xử lý việc ưu tiên URL.

Tham khảo tài liệu tham khảo [5] [10] để biết thông tin chi tiết về khái niệm này.

Hình 9-7 hiển thị thiết kế quản lý mức độ ưu tiên của URL.

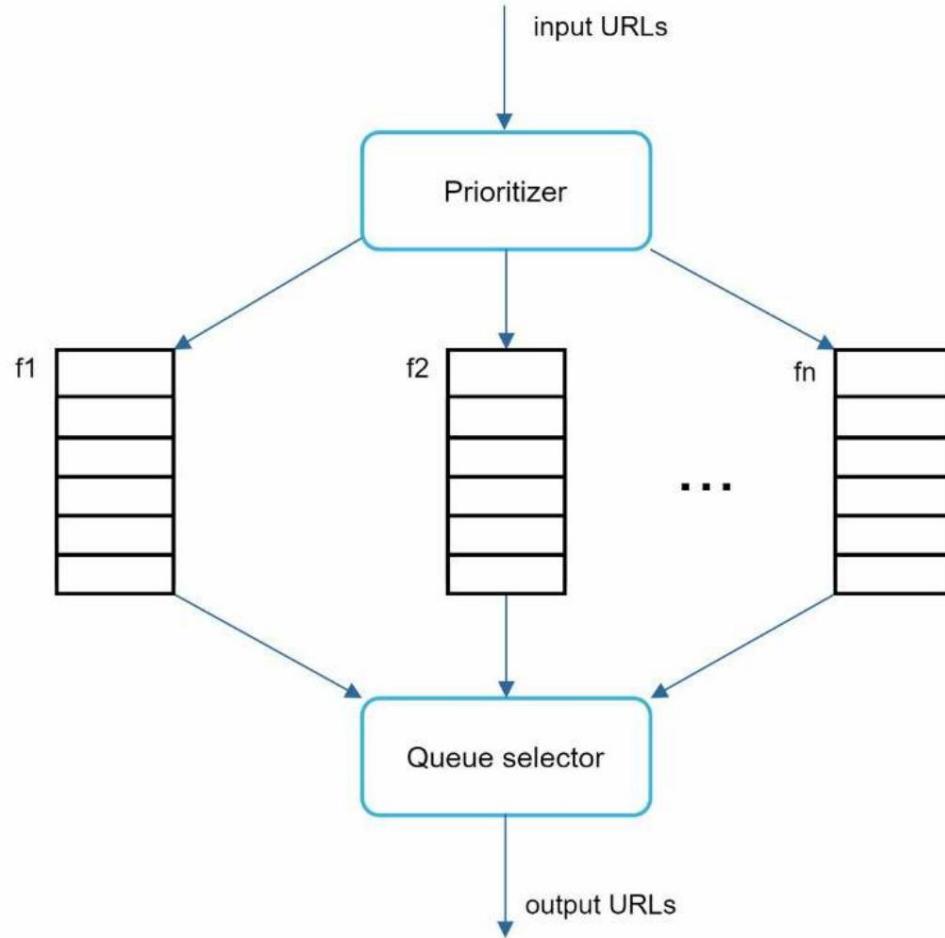


Figure 9-7

- Bộ ưu tiên: Lấy URL làm đầu vào và tính toán các mức ưu tiên. • Hàng đợi từ f1 đến fn: Mỗi hàng đợi có một mức ưu tiên được chỉ định. Các hàng đợi có mức ưu tiên cao được chọn với xác suất cao hơn.
- Bộ chọn hàng đợi: Chọn ngẫu nhiên một hàng đợi có xu hướng thiên về các hàng đợi có mức ưu tiên cao hơn.

Hình 9-8 trình bày thiết kế biên giới URL và bao gồm hai mô-đun:

- Hàng đợi phía trước: quản lý việc ưu tiên
- Hàng đợi phía sau: quản lý sự lịch sự

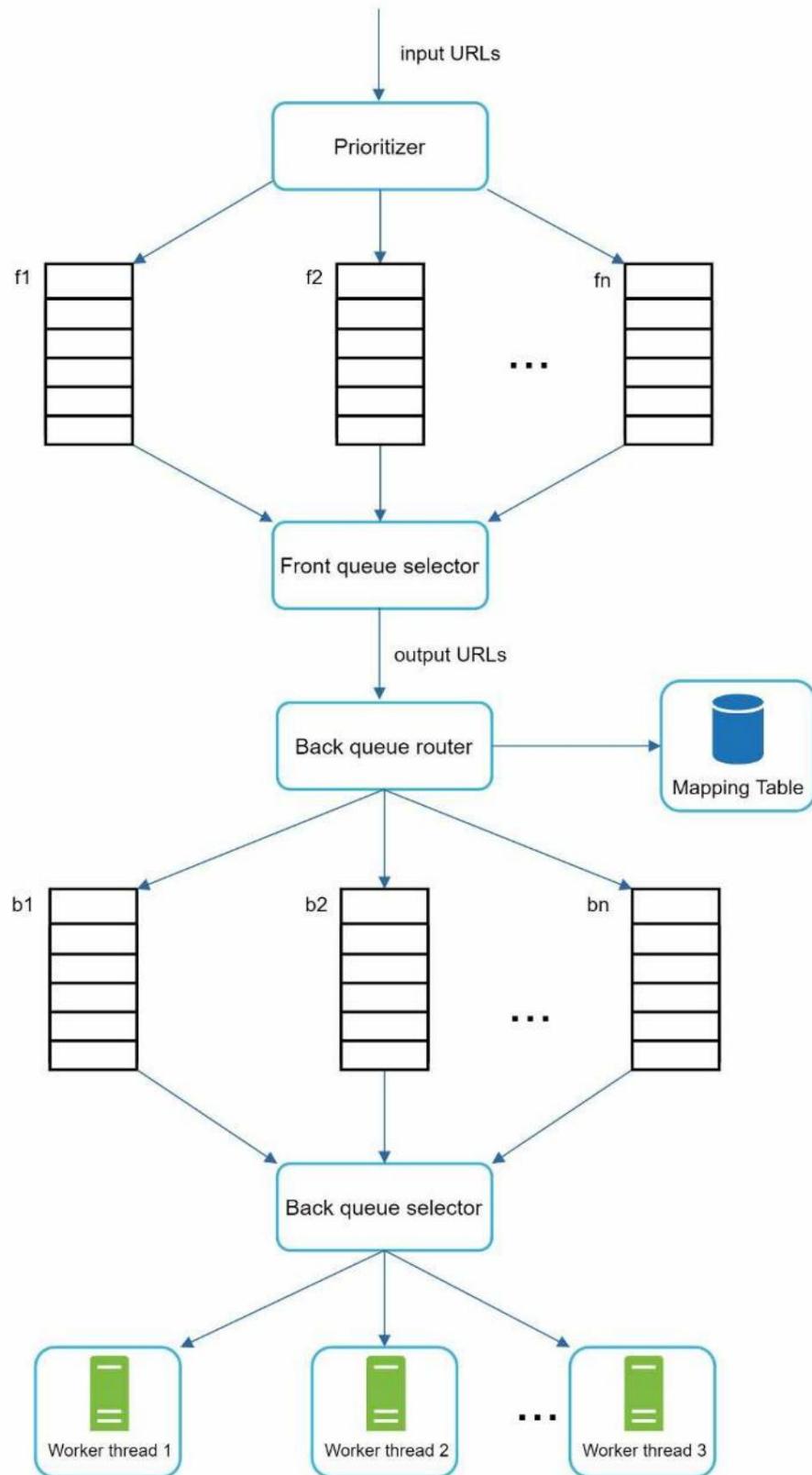


Figure 9-8

Độ ưu tiên

Các trang web liên tục được thêm vào, xóa và chỉnh sửa. Một trình thu thập dữ liệu web phải định kỳ thu thập lại các trang đã tải xuống để giữ cho bộ dữ liệu của chúng tôi luôn mới. Thu thập lại tất cả các URL tồn tại nhiều thời gian và tài nguyên. Một số chiến lược để tối ưu hóa độ mới được liệt kê sau:

- Thu thập lại dữ liệu dựa trên lịch sử cập nhật của các trang web.
- Ưu tiên các URL và thu thập lại dữ liệu các trang quan trọng trước và thường xuyên hơn.

Lưu trữ cho URL Frontier Trong

quá trình thu thập dữ liệu thực tế cho các công cụ tìm kiếm, số lượng URL trong frontier có thể lên tới hàng trăm triệu [4]. Việc đưa mọi thứ vào bộ nhớ không bền vững cũng như không thể mở rộng. Việc giữ mọi thứ trong đĩa cũng không được mong muốn vì đĩa chậm; và nó có thể dễ dàng trở thành nút thắt cổ chai cho quá trình thu thập dữ liệu.

Chúng tôi áp dụng phương pháp tiếp cận kết hợp. Phần lớn các URL được lưu trữ trên đĩa, do đó không gian lưu trữ không phải là vấn đề. Để giảm chi phí đọc từ đĩa và ghi vào đĩa, chúng tôi duy trì bộ đệm trong bộ nhớ cho các hoạt động enqueue/dequeue. Dữ liệu trong bộ đệm được ghi định kỳ vào đĩa.

Trình tải xuống HTML

Trình tải xuống HTML tải xuống các trang web từ internet bằng giao thức HTTP.

Trước khi thảo luận về HTML Downloader, chúng ta hãy cùng tìm hiểu về Giao thức loại trừ robot trước.

Robots.txt

Robots.txt, còn gọi là Giao thức loại trừ robot, là một tiêu chuẩn được các trang web sử dụng để giao tiếp với trình thu thập thông tin. Giao thức này chỉ định những trang nào trình thu thập thông tin được phép tải xuống. Trước khi cố gắng thu thập thông tin một trang web, trình thu thập thông tin phải kiểm tra robots.txt tương ứng trước và tuân theo các quy tắc của nó.

Để tránh việc tải xuống tệp robots.txt nhiều lần, chúng tôi lưu trữ kết quả của tệp. Tệp được tải xuống và lưu vào bộ nhớ đệm định kỳ. Đây là một phần của tệp robots.txt được lấy từ <https://www.amazon.com/robots.txt>. Một số thư mục như creatorhub không được phép đối với bot Google.

Tác nhân người dùng: Googlebot

Không cho phép: /creatorhub/*

Không cho phép: /rss/people/*/reviews

Không cho phép: /gp/pdp/rss/*/reviews

Không cho phép: /gp/cdp/member-reviews/

Không cho phép: /gp/aw/cr/

Bên cạnh robots.txt, tối ưu hóa hiệu suất là một khái niệm quan trọng khác mà chúng ta sẽ đề cập đến đối với trình tải xuống HTML.

Tối ưu hóa hiệu suất Dưới đây là

danh sách các tối ưu hóa hiệu suất cho trình tải xuống HTML.

1. Thu thập dữ liệu phân tán

Để đạt được hiệu suất cao, các tác vụ thu thập dữ liệu được phân phối vào nhiều máy chủ và mỗi máy chủ chạy nhiều luồng. Không gian URL được phân vùng thành các phần nhỏ hơn; do đó, mỗi trình tải xuống chịu trách nhiệm cho một tập hợp con các URL. Hình 9-9 cho thấy một ví dụ về thu thập dữ liệu phân tán.

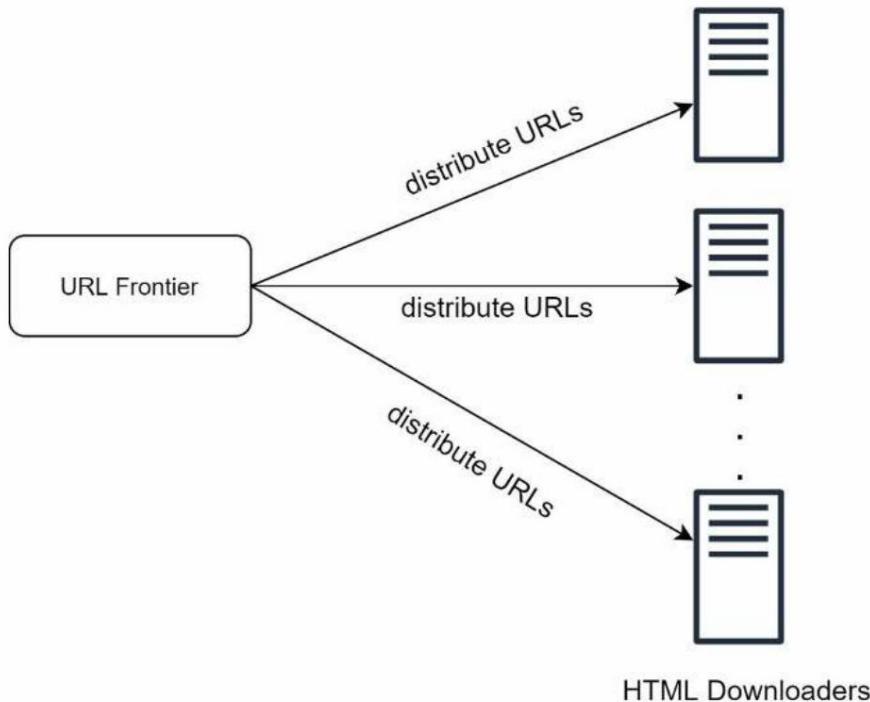


Figure 9-9

2. Bộ nhớ đệm giải quyết DNS

DNS Resolver là nút thắt cổ chai đối với các trình thu thập thông tin vì các yêu cầu DNS có thể mất thời gian do bản chất đồng bộ của nhiều giao diện DNS. Thời gian phản hồi của DNS dao động từ 10ms đến 200ms. Khi một yêu cầu tới DNS được thực hiện bởi một luồng trình thu thập thông tin, các luồng khác sẽ bị chặn cho đến khi yêu cầu đầu tiên được hoàn tất. Duy trì bộ đệm DNS của chúng tôi để tránh gọi DNS thường xuyên là một kỹ thuật hiệu quả để tối ưu hóa tốc độ. Bộ đệm DNS của chúng tôi giữ nguyên ánh xạ tên miền với địa chỉ IP và được cập nhật định kỳ bởi các tác vụ cron.

3. Vị trí Phân

phối máy chủ thu thập dữ liệu theo địa lý. Khi máy chủ thu thập dữ liệu gần hơn với máy chủ lưu trữ trang web, trình thu thập dữ liệu sẽ có thời gian tải xuống nhanh hơn. Vị trí thiết kế áp dụng cho hầu hết các thành phần hệ thống: máy chủ thu thập dữ liệu, bộ nhớ đệm, hàng đợi, lưu trữ, v.v.

4. Thời gian chờ ngắn

Một số máy chủ web phản hồi chậm hoặc có thể không phản hồi. Để tránh thời gian chờ lâu, thời gian chờ tối đa được chỉ định. Nếu máy chủ không phản hồi trong thời gian được xác định trước, trình thu thập dữ liệu sẽ dừng công việc và thu thập một số trang khác.

Sự mạnh mẽ

Bên cạnh việc tối ưu hóa hiệu suất, tính mạnh mẽ cũng là một cân nhắc quan trọng. Chúng tôi trình bày một số cách tiếp cận để cải thiện tính mạnh mẽ của hệ thống:

- **Băm nhất quán:** Điều này giúp phân phối tài nguyên giữa các trình tải xuống. Có thể thêm hoặc xóa máy chủ tải xuống mới bằng cách sử dụng băm nhất quán. Tham khảo Chương 5: Thiết kế băm nhất quán để biết thêm chi tiết.
- **Lưu trạng thái thu thập dữ liệu và dữ liệu:** Để bảo vệ chống lại lỗi, trạng thái thu thập dữ liệu và dữ liệu được ghi vào hệ thống lưu trữ. Có thể dễ dàng khởi động lại quá trình thu thập dữ liệu bị gián đoạn bằng cách tải trạng thái và dữ liệu đã lưu.
- **Xử lý ngoại lệ:** Lỗi là điều không thể tránh khỏi và phổ biến trong một hệ thống quy mô lớn.

trình thu thập dữ liệu phải xử lý các ngoại lệ một cách nhẹ nhàng mà không làm hệ thống bị sập. • Xác thực dữ liệu: Đây là biện pháp quan trọng để ngăn ngừa lỗi hệ thống.

Khả năng mở rộng

Khi hầu hết mọi hệ thống đều phát triển, một trong những mục tiêu thiết kế là làm cho hệ thống đủ linh hoạt để hỗ trợ các loại nội dung mới. Trình thu thập thông tin có thể được mở rộng bằng cách cắm các mô-đun mới. Hình 9-10 cho thấy cách thêm các mô-đun mới.

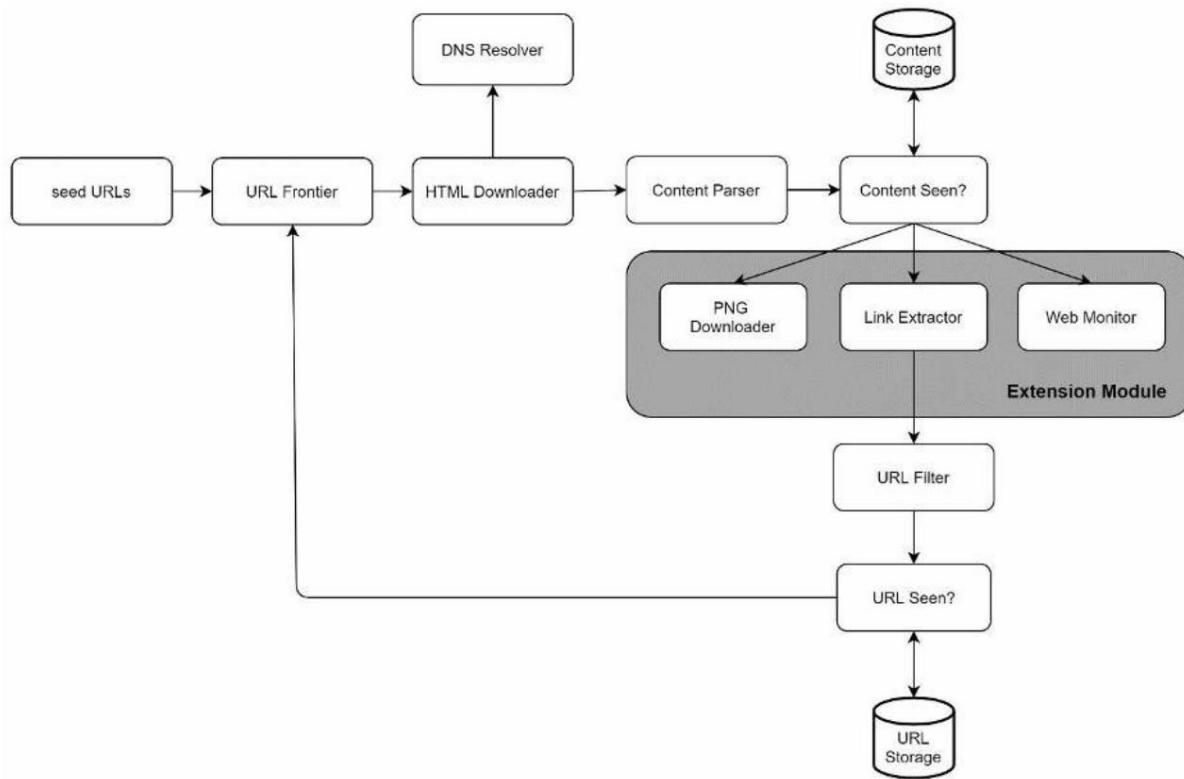


Figure 9-10

- Mô-đun PNG Downloader được cắm vào để tải xuống các tệp PNG. • Mô-đun Web Monitor được thêm vào để giám sát web và ngăn chặn vi phạm bản quyền và nhẫn hiệu.

Phát hiện và tránh nội dung có vấn đề

Phần này thảo luận về việc phát hiện và ngăn ngừa các thông tin thừa, vô nghĩa hoặc có hại nội dung.

1. Nội dung trùng lặp

Như đã thảo luận trước đó, gần 30% các trang web là trùng lặp. Băm hoặc tổng kiểm tra giúp phát hiện trùng lặp [11].

2. Bẫy nhện Bẫy

Nhện là một trang web khiến trình thu thập dữ liệu rơi vào vòng lặp vô hạn. Ví dụ, một cấu trúc thư mục sâu vô hạn được liệt kê như sau: www.spidertrapexample.com/foo/bar/foo/bar/foo/bar/.

Có thể tránh được những bẫy nhện như vậy bằng cách đặt độ dài tối đa cho URL. Tuy nhiên, không có giải pháp nào phù hợp với tất cả mọi người để phát hiện ra bẫy nhện. Các trang web có bẫy nhện rất dễ nhận dạng do số lượng trang web được phát hiện trên các trang web như vậy là rất lớn. Thật khó để phát triển các thuật toán tự động để tránh bẫy nhện; tuy nhiên, người dùng có thể thực hiện thủ công

xác minh và xác định bẫy nhện, sau đó loại trừ các trang web đó khỏi trình thu thập thông tin hoặc áp dụng một số bộ lọc URL tùy chỉnh.

3. Tiếng ồn dữ liệu

Một số nội dung có ít hoặc không có giá trị, chẳng hạn như quảng cáo, đoạn mã, URL spam, v.v. Những nội dung đó không hữu ích cho trình thu thập thông tin và nên loại trừ nếu có thể.

Bứ ớc 4 - Tǒng kêt Trong

chư ơ ng này, trư ớc ti ên chung ta thảo luận về các đặc điểm của một crawler tốt: khả năng mở rộng, tính lịch sự, khả năng mở rộng và tính mạnh mẽ. Sau đó, chúng ta sẽ xuất một thiết kế và thảo luận về các thành phần chính. Xây dựng một trình thu thập dữ liệu web có khả năng mở rộng không phải là một nhiệm vụ đơn giản vì web cực kỳ lớn và đầy rẫy cạm bẫy. Mặc dù chúng tôi đã đề cập đến nhiều chủ đề, chúng tôi vẫn bỏ lỡ nhiều điểm thảo luận có liên

quan: • Kết xuất phía máy chủ: Nhiều trang web sử dụng các tập lệnh như JavaScript, AJAX, v.v. để tạo liên kết khi đang chạy. Nếu chúng tôi tải xuống và phân tích cú pháp các trang web trực tiếp, chúng tôi sẽ không thể truy xuất các liên kết được tạo động. Để giải quyết vấn đề này, trư ớc ti ên chúng tôi thực hiện kết xuất phía máy chủ (còn gọi là kết xuất động) trước khi phân tích cú pháp một trang [12]. • Lọc các trang không mong muốn: Với dung lượng lưu trữ và tài nguyên thu thập dữ liệu hữu hạn, một thành phần chống thư rác có lợi trong việc lọc các trang chất lượng thấp và thư rác [13] [14]. • Sao chép và phân mảnh cơ sở dữ liệu: Các kỹ thuật như sao chép và phân mảnh được sử dụng để cải thiện tính khả dụng, khả năng mở rộng và độ tin cậy của lớp dữ liệu.

• Mở rộng theo chiều ngang: Đối với quá trình thu thập dữ liệu quy mô lớn, cần hàng trăm hoặc thậm chí hàng nghìn máy chủ để thực hiện các tác vụ tải xuống. Chìa khóa là giữ cho các máy chủ không có trạng thái. • Tính khả dụng, tính nhất quán và độ tin cậy: Các khái niệm này là cốt lõi của thành công của bất kỳ hệ thống nào. Chúng tôi đã thảo luận chi tiết về các khái niệm này trong Chương 1. Làm mới lại trí nhớ của bạn về các chủ đề này. • Phân tích: Thu thập và phân tích dữ liệu là những phần quan trọng của bất kỳ hệ thống nào vì dữ liệu là thành phần chính để tinh chỉnh.

Xin chúc mừng vì đã đi đư ợc đến đây! Bây giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

[1] Thư viện Quốc hội Hoa Kỳ: <https://www.loc.gov/websites/> [2]

Lưu trữ web EU: <http://data.europa.eu/webarchive> [3] Digimarc:

<https://www.digimarc.com/products/digimarc-services/piracy-intelligence> [4] Heydon A., Najork M.

Mercator: Một trình thu thập dữ liệu web có thể mở rộng, có thể mở rộng World Wide Web, 2 (4) (1999), trang 219-229 [5]

Christopher Olston, Marc Najork: Thu thập dữ liệu web. http://infolab.stanford.edu/~olston/publications/crawling_survey.pdf [6] 29% các

trang web phải đối mặt với các vấn đề về nội dung trùng lặp: <https://tinyurl.com/y6tmh55y>

[7] Rabin MO, et al. Lấy dấu vân tay bằng đa thức ngẫu nhiên Trung tâm nghiên cứu công nghệ máy tính, Phòng thí nghiệm tính toán Aiken, Đại học. (1981)

[8] BH Bloom, "Sự đánh đổi không gian/thời gian trong mã hóa băm với các lỗi cho phép," Thông tin liên lạc của ACM, tập 13, số 7, trang 422-426, 1970.

[9] Donald J. Patterson, Web Crawling:

<https://www.ics.uci.edu/~lopes/teaching/cs221W12/slides/Lecture05.pdf> [10] L. Page,

S. Brin, R. Motwani và T. Winograd, "Xếp hạng trích dẫn PageRank: Mang lại trật tự cho web," Báo cáo kỹ thuật, Đại học Stanford, 1998.

[11] Burton Bloom. Sự đánh đổi không gian/thời gian trong mã hóa băm với các lỗi cho phép. Thông tin liên lạc của ACM, 13(7), trang 422--426, tháng 7 năm 1970.

[12] Google Dynamic Rendering: <https://developers.google.com/search/docs/guides/dynamic-rendering> [13] T. Urvoy, T.

Lavergne và P. Filoche, "Theo dõi thư rác trên web với sự tươn đồng về phong cách ẩn", trong Biên bản Hội thảo quốc tế lần thứ 2 về Truy xuất thông tin đối nghịch trên Web, 2006.

[14] H.-T. Lee, D. Leonard, X. Wang và D. Loguinov, "IRLbot: Mở rộng quy mô lên 6 tỷ trang và hơn thế nữa," trong Biên bản Hội nghị quốc tế lần thứ 17 về mạng lưới toàn cầu, 2008.

CHÚ Ó NG 10: THIẾT KẾ HỆ THỐNG THÔNG BÁO

Hệ thống thông báo đã trở thành một tính năng rất phổ biến cho nhiều ứng dụng trong những năm gần đây.

Thông báo cảnh báo người dùng về thông tin quan trọng như tin tức mới nhất, cập nhật sản phẩm, sự kiện, ưu đãi, v.v. Nó đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của chúng ta.

Trong chương này, bạn được yêu cầu thiết kế một hệ thống thông báo.

Thông báo không chỉ là thông báo đầy trên thiết bị di động. Có ba loại định dạng thông báo: thông báo đầy trên thiết bị di động, tin nhắn SMS và Email. Hình 10-1 cho thấy ví dụ về từng loại thông báo này.

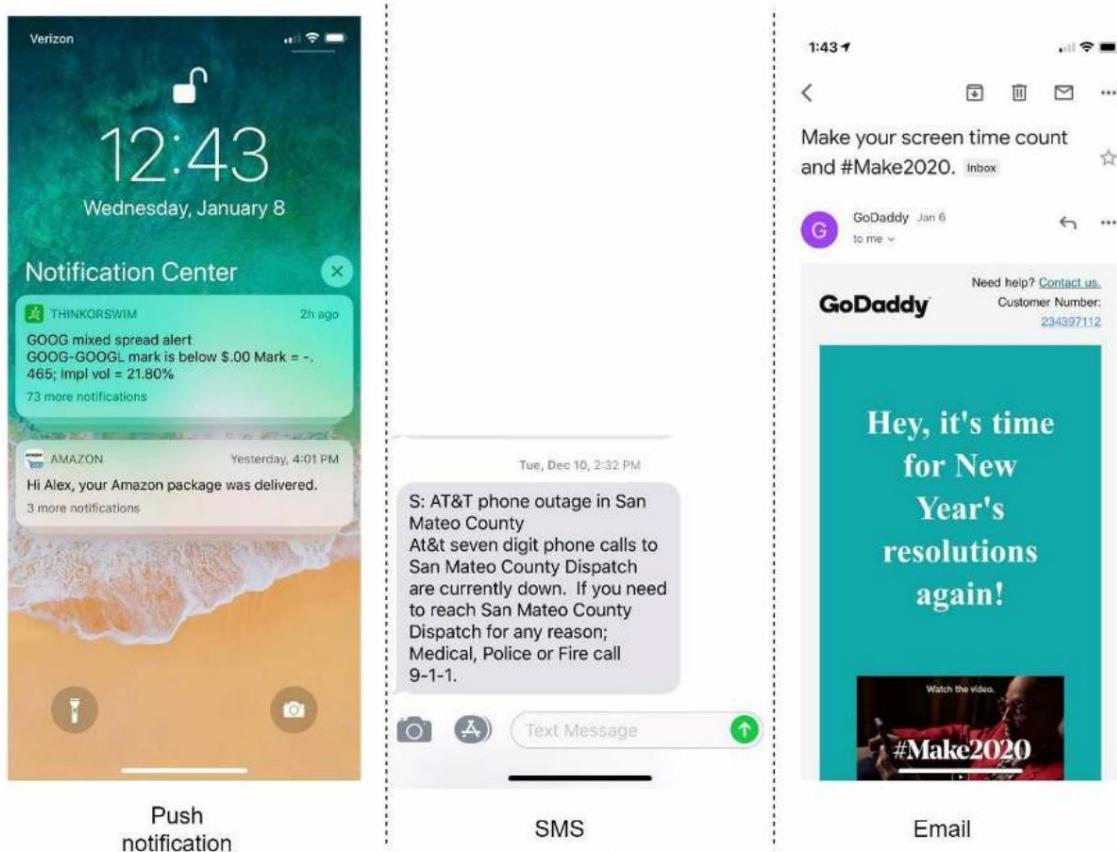


Figure 10-1

Bư ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế Xây dựng một hệ thống có khả năng mở rộng, gửi hàng triệu thông báo mỗi ngày không phải là một nhiệm vụ dễ dàng. Nó đòi hỏi sự hiểu biết sâu sắc về hệ sinh thái thông báo. Câu hỏi phỏng vấn được thiết kế có chủ đích để mở và mở rộng, và bạn có trách nhiệm đặt câu hỏi để làm rõ các yêu cầu.

Ứng viên: Hệ thống hỗ trợ những loại thông báo nào?

Người phỏng vấn: Thông báo đầy, tin nhắn SMS và email.

Ứng viên: Đây có phải là hệ thống thời gian thực không?

Người phỏng vấn: Giả sử đây là hệ thống thời gian thực mềm. Chúng tôi muốn người dùng nhận được thông báo sớm nhất có thể. Tuy nhiên, nếu hệ thống đang chịu khó lượng công việc lớn, thì có thể chấp nhận được độ trễ nhỏ.

Ứng viên: Thiết bị nào được hỗ trợ?

Người phỏng vấn: Thiết bị iOS, thiết bị Android và máy tính xách tay/máy tính để bàn.

Ứng viên: Cái gì kích hoạt thông báo?

Người phỏng vấn: Thông báo có thể được kích hoạt bởi các ứng dụng khách hàng. Chúng cũng có thể được gửi lên lịch trên phía máy chủ.

Ứng viên: Người dùng có thể chọn không tham gia không?

Người phỏng vấn: Có, những người dùng chọn không tham gia sẽ không còn nhận được thông báo nữa.

Ứng viên: Mỗi ngày có bao nhiêu thông báo được gửi đi?

Người phỏng vấn: 10 triệu thông báo đầy trên thiết bị di động, 1 triệu tin nhắn SMS và 5 triệu email.

Bài Ớc 2 - Đề xuất thiết kế cấp cao và nhận đư ợc sự đồng thuận

Phần này hiển thị thiết kế cấp cao hỗ trợ nhiều loại thông báo khác nhau: thông báo đẩy iOS, thông báo đẩy Android, tin nhắn SMS và Email. Nó đư ợc cấu trúc như sau:

- Các loại thông báo khác nhau • Luồng

thu thập thông tin liên hệ • Luồng

gửi/nhận thông báo

Các loại thông báo khác nhau Chúng ta bắt

đầu bằng cách xem xét cách thức hoạt động của từng loại thông báo ở cấp độ cao.

Thông báo đẩy iOS

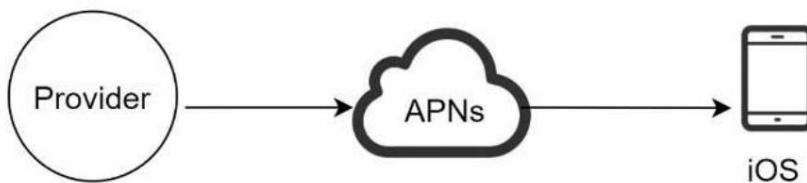


Figure 10-2

về cơ bản, chúng ta cần ba thành phần để gửi thông báo đẩy iOS:

- Nhà cung cấp. Nhà cung cấp xây dựng và gửi yêu cầu thông báo đến Apple Push Notification Service (APNS). Để xây dựng thông báo đẩy, nhà cung cấp cung cấp dữ liệu sau:

- Mã thông báo thiết bị: Đây là mã định danh duy nhất đư ợc sử dụng để gửi thông báo đẩy. •

Tải trọng: Đây là từ điển JSON chứa tải trọng của thông báo. Sau đây là một ví dụ:

```
{
  "aps": {
    "alert": {
      "title": "Game Request",
      "body": "Bob wants to play chess",
      "action-loc-key": "PLAY"
    },
    "badge": 5
  }
}
```

- APNS: Đây là dịch vụ từ xa do Apple cung cấp để truyền thông báo đẩy đến các thiết bị iOS.

- Thiết bị iOS: Đây là máy khách cuối cùng nhận đư ợc thông báo đẩy.

Thông báo đẩy Android

Android áp dụng luồng thông báo tự ng tự. Thay vì sử dụng APN, Firebase Cloud Messaging (FCM) thường đư ợc sử dụng để gửi thông báo đẩy đến các thiết bị Android.

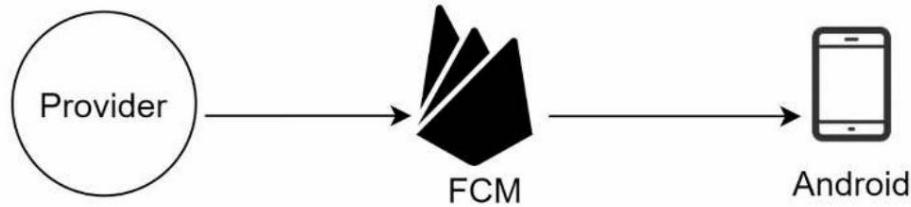


Figure 10-3

Tin nhắn SMS

Đối với tin nhắn SMS, các dịch vụ SMS của bên thứ ba như Twilio [1], Nexmo [2] và nhiều dịch vụ khác thường được sử dụng. Hầu hết chúng là dịch vụ thư điện tử.

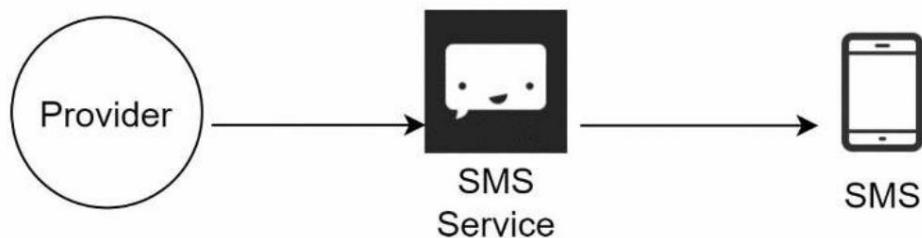


Figure 10-4

E-mail

Mặc dù các công ty có thể thiết lập máy chủ email của riêng mình, nhiều công ty trong số họ lựa chọn dịch vụ email thương mại. Sendgrid [3] và Mailchimp [4] là một trong những dịch vụ email phổ biến nhất, cung cấp tỷ lệ phân phối và phân tích dữ liệu tốt hơn.

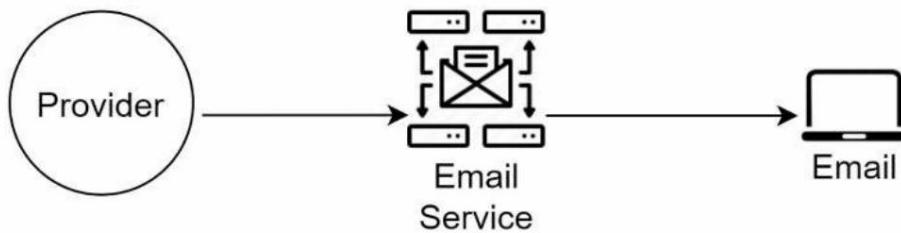


Figure 10-5

Hình 10-6 hiển thị thiết kế sau khi bao gồm tất cả các dịch vụ của bên thứ ba.

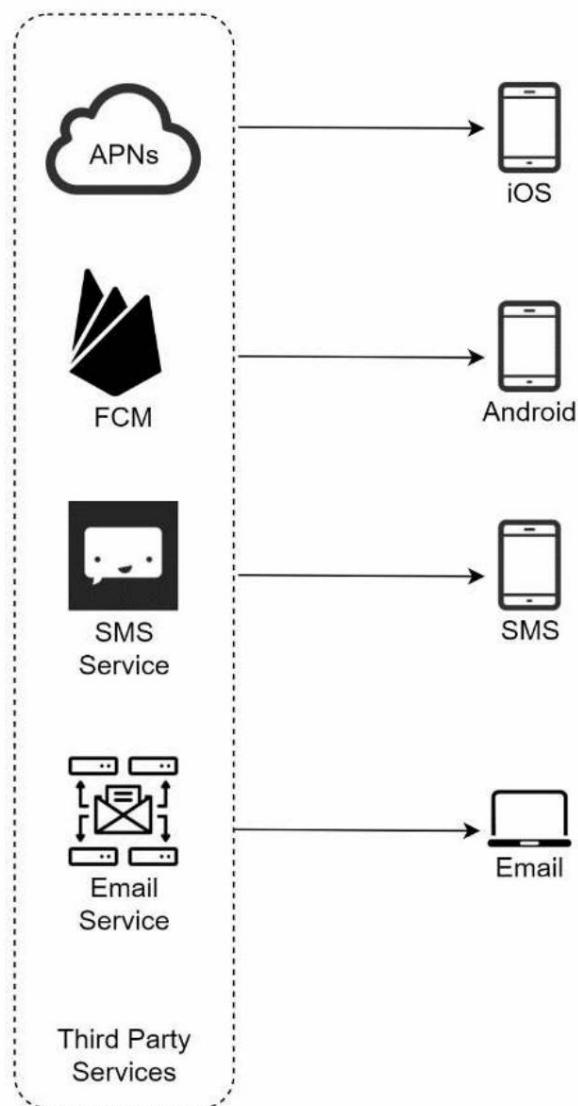


Figure 10-6

Luồng thu thập thông tin liên hệ Đè gửi

thông báo, chúng ta cần thu thập mã thông báo thiết bị di động, số điện thoại hoặc địa chỉ email. Như thể hiện trong Hình 10-7, khi người dùng cài đặt ứng dụng của chúng tôi hoặc đăng ký lần đầu tiên, máy chủ API sẽ thu thập thông tin liên hệ của người dùng và lưu trữ trong cơ sở dữ liệu.

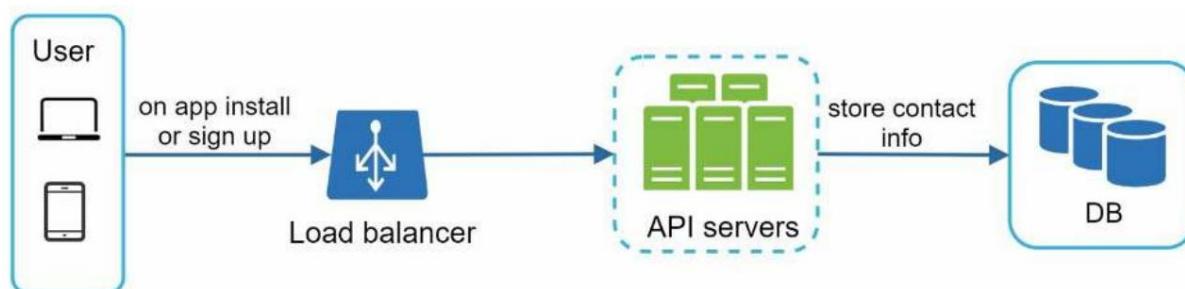


Figure 10-7

Hình 10-8 hiển thị các bảng cơ sở dữ liệu được đơn giản hóa để lưu trữ thông tin liên lạc. Địa chỉ email và số điện thoại được lưu trữ trong bảng người dùng, trong khi mã thông báo thiết bị được lưu trữ trong bảng thiết bị.

người dùng có thể có nhiều thiết bị, cho biết thông báo đầy có thể được gửi đến tất cả các thiết bị của người dùng.

user		device
user_id	bigint	1
email	varchar	
country_code	integer	
phone_number	integer	*
created_at	timestamp	

Figure 10-8

Luồng gửi/nhận thông báo Đầu tiên chúng tôi sẽ trình bày thiết kế ban đầu; sau đó, đề xuất một số tối ưu hóa.

Thiết kế cấp cao

Hình 10-9 cho thấy thiết kế và từng thành phần của hệ thống được giải thích bên dưới.

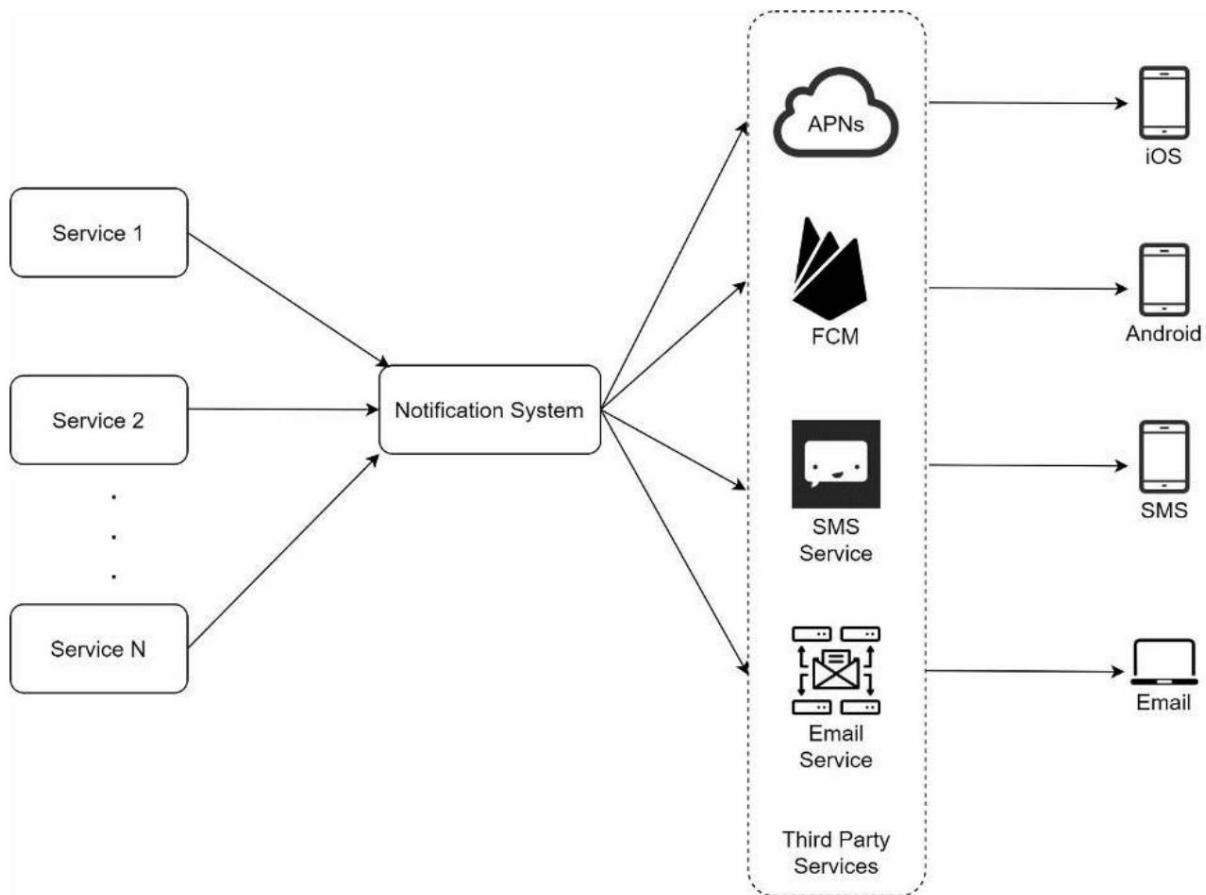


Figure 10-9

Dịch vụ 1 đến N: Một dịch vụ có thể là một dịch vụ vi mô, một công việc cron hoặc một hệ thống phân tán kích hoạt các sự kiện gửi thông báo. Ví dụ, một dịch vụ thanh toán gửi email để nhắc nhở khách hàng về khoản thanh toán đến hạn hoặc một trang web mua sắm thông báo với khách hàng rằng các gói hàng của họ sẽ được giao vào ngày mai qua tin nhắn SMS.

Hệ thống thông báo: Hệ thống thông báo là trung tâm của việc gửi/nhận thông báo. Bắt đầu với một cái gì đó đơn giản, chỉ có một máy chủ thông báo được sử dụng. Nó cung cấp API cho các dịch vụ từ 1 đến N và xây dựng các tài liệu thông báo cho các dịch vụ của bên thứ ba.

Dịch vụ của bên thứ ba: Dịch vụ của bên thứ ba chịu trách nhiệm gửi thông báo đến người dùng. Trong khi tích hợp với các dịch vụ của bên thứ ba, chúng ta cần chú ý nhiều hơn đến khả năng mở rộng. Khả năng mở rộng tốt có nghĩa là một hệ thống linh hoạt có thể dễ dàng cắm hoặc rút dịch vụ của bên thứ ba. Một cân nhắc quan trọng khác là dịch vụ của bên thứ ba có thể không khả dụng ở các thị trường mới hoặc trong tương lai. Ví dụ, FCM không khả dụng ở Trung Quốc. Do đó, các dịch vụ thay thế của bên thứ ba như Jpush, PushY, v.v. được sử dụng ở đó.

iOS, Android, SMS, Email: Người dùng nhận thông báo trên thiết bị của họ.

Có ba vấn đề được xác định trong thiết kế này:

- Điểm lỗi đơn (SPOF): Một máy chủ thông báo đơn có nghĩa là SPOF.
- Khó mở rộng: Hệ thống thông báo xử lý mọi thứ liên quan đến thông báo đầy trong một máy chủ. Thật khó để mở rộng cơ sở dữ liệu, bộ nhớ đệm và các thành phần xử lý thông báo khác nhau một cách độc lập.
- Nút thắt hiệu suất: Xử lý và gửi thông báo có thể tồn tại nguyên.

Ví dụ, việc xây dựng các trang HTML và chờ phản hồi từ các dịch vụ của bên thứ ba có thể mất thời gian.

Xử lý mọi thứ trong một hệ thống có thể dẫn đến quá tải hệ thống, đặc biệt là trong giờ cao điểm.

Thiết kế cấp cao (cải thiện)

Sau khi liệt kê những thách thức trong thiết kế ban đầu, chúng tôi cải thiện thiết kế như dưới đây:

- Di chuyển cơ sở dữ liệu và bộ nhớ đệm ra khỏi máy chủ thông báo.
- Thêm nhiều máy chủ thông báo hơn và thiết lập khả năng mở rộng theo chiều ngang tự động.
- Giới thiệu hàng đợi tin nhắn để tách biệt các thành phần hệ thống.

Hình 10-10 cho thấy thiết kế cấp cao được cải tiến.

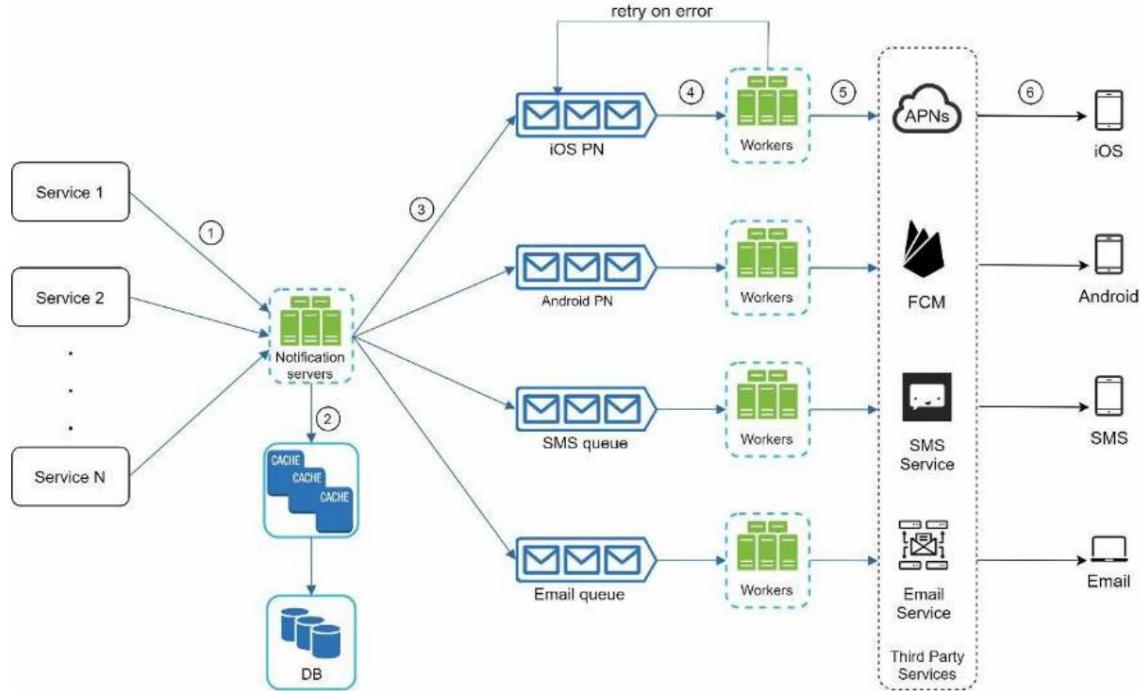


Figure 10-10

Cách tốt nhất để xem sơ đồ trên là từ trái sang phải:

Dịch vụ 1 đến N: Chúng đại diện cho các dịch vụ khác nhau gửi thông báo qua API do máy chủ thông báo cung cấp.

Máy chủ thông báo: Chúng cung cấp các chức năng sau:

- Cung cấp API cho các dịch vụ để gửi thông báo. Các API đó chỉ có thể truy cập nội bộ hoặc bởi các máy khách đã xác minh để ngăn chặn thư rác.
- Thực hiện các xác thực cơ bản để xác minh email, số điện thoại, v.v.
- Truy vấn cơ sở dữ liệu hoặc bộ nhớ đệm để lấy dữ liệu cần thiết để hiển thị thông báo.
- Đưa dữ liệu thông báo vào hàng đợi tin nhắn để xử lý song song.

Sau đây là một ví dụ về API để gửi email:

```
POST https://api.example.com/v/sms/send
Nội dung
yêu cầu
```

```
{
  "to": [
    {
      "user_id": 123456
    }
  ],
  "from": {
    "email": "from_address@example.com"
  },
  "subject": "Hello, World!",
  "content": [
    {
      "type": "text/plain",
      "value": "Hello, World!"
    }
  ]
}
```

Bộ nhớ đệm: Thông tin người dùng, thông tin thiết bị, mẫu thông báo được lưu vào bộ nhớ đệm.

DB: Lưu trữ dữ liệu về người dùng, thông báo, cài đặt, v.v.

Hàng đợi tin nhắn: Chúng loại bỏ sự phụ thuộc giữa các thành phần. Hàng đợi tin nhắn đóng vai trò như bộ đệm khi cần gửi khỏi lượng lớn thông báo. Mỗi loại thông báo được chỉ định một hàng đợi tin nhắn riêng biệt để sự cố ngừng hoạt động trong một dịch vụ của bên thứ ba sẽ không ảnh hưởng đến các loại thông báo khác.

Công nhân: Công nhân là danh sách các máy chủ lấy các sự kiện thông báo từ hàng đợi tin nhắn và gửi chúng đến các dịch vụ của bên thứ ba ứng.

Dịch vụ của bên thứ ba: Đã được giải thích trong thiết kế ban đầu.

iOS, Android, SMS, Email: Đã được giải thích trong thiết kế ban đầu.

Tiếp theo, chúng ta hãy xem xét cách từng thành phần hoạt động cùng nhau để gửi thông báo: 1. Một

dịch vụ gọi API do máy chủ thông báo cung cấp để gửi thông báo.

2. Máy chủ thông báo sẽ lấy siêu dữ liệu như thông tin người dùng, mã thông báo thiết bị và cài đặt thông báo từ bộ nhớ đệm hoặc cơ sở dữ liệu.

3. Một sự kiện thông báo được gửi đến hàng đợi tương ứng để xử lý. Ví dụ, một sự kiện thông báo đẩy iOS được gửi đến hàng đợi iOS PN.

4. Công nhân lấy các sự kiện thông báo từ hàng đợi tin nhắn.

5. Người lao động gửi thông báo đến các dịch vụ của bên thứ ba.

6. Dịch vụ của bên thứ ba gửi thông báo đến thiết bị của người dùng.

Bút ớc 3 - Thiết kế chuyên sâu Trong

thiết kế cấp cao, chúng tôi đã thảo luận về các loại thông báo khác nhau, luồng thu thập thông tin liên hệ và luồng gửi/nhận thông báo. Chúng tôi sẽ khám phá những điều sau đây trong phần chuyên sâu:

- Độ tin cậy.
- Thành phần bổ sung và các cản nhặc: mẫu thông báo, cài đặt thông báo, giới hạn tỷ lệ, cơ chế thử lại, bảo mật trong thông báo đầy, giám sát thông báo xếp hàng và theo dõi sự kiện.
- Thiết kế được cập nhật.

Độ tin cậy

Chúng ta phải trả lời một số câu hỏi quan trọng về độ tin cậy khi thiết kế hệ thống thông báo trong môi trường phân tán.

Làm thế nào để ngăn ngừa mất dữ liệu?

Một trong những yêu cầu quan trọng nhất trong hệ thống thông báo là không được làm mất dữ liệu.

Thông báo thường có thể bị trì hoãn hoặc sắp xếp lại, nhưng không bao giờ bị mất. Để đáp ứng yêu cầu này, hệ thống thông báo sẽ lưu trữ dữ liệu thông báo trong cơ sở dữ liệu và triển khai cơ chế thử lại. Cơ sở dữ liệu nhật ký thông báo được bao gồm để lưu trữ dữ liệu, như thể hiện trong Hình 10-11.

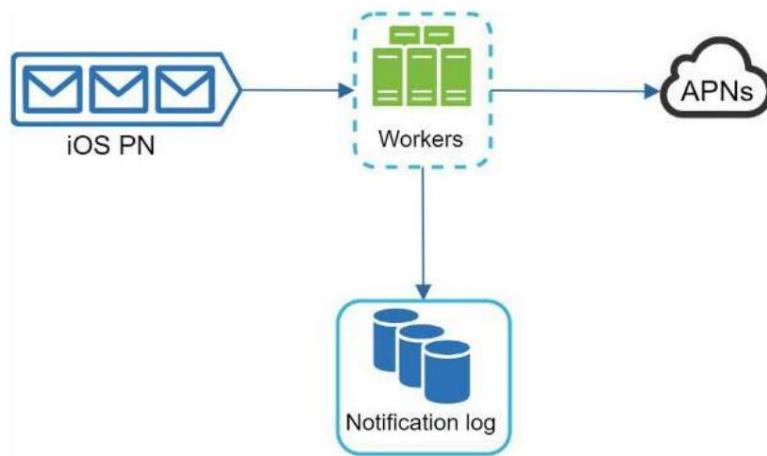


Figure 10-11

Người nhận có nhận được thông báo đúng một lần không?

Câu trả lời ngắn gọn là không. Mặc dù thông báo được gửi chính xác một lần hầu hết thời gian, bản chất phân tán có thể dẫn đến thông báo trùng lặp. Để giảm sự trùng lặp, chúng tôi giới thiệu cơ chế loại bỏ trùng lặp và xử lý cẩn thận từng trung hợp lỗi. Sau đây là logic loại bỏ trùng lặp đơn giản:

Khi sự kiện thông báo đầu

tiên đến, chúng tôi kiểm tra xem sự kiện đó đã được nhìn thấy trước đó hay chưa bằng cách kiểm tra ID sự kiện. Nếu đã thấy trước thì loại bỏ. Nếu không, chúng tôi sẽ gửi thông báo. Để độc giả quan tâm hiểu lý do tại sao chúng tôi không thể giao hàng đúng một lần, hãy tham khảo tài liệu tham khảo [5].

Các thành phần và cản nhặc bổ sung Chúng tôi đã thảo luận về

cách thu thập thông tin liên hệ của người dùng, gửi và nhận thông báo. Hệ thống thông báo còn hơn thế nữa. Ở đây chúng tôi thảo luận về các thành phần bổ sung bao gồm việc sử dụng lại mẫu, cài đặt thông báo, theo dõi sự kiện, giám sát hệ thống, giới hạn tỷ lệ, v.v.

Mẫu thông báo

Một hệ thống thông báo lớn gửi hàng triệu thông báo mỗi ngày và nhiều thông báo trong số này có định dạng tự do ng tự nhau. Các mẫu thông báo được giới thiệu để tránh phải xây dựng mọi thông báo từ đầu. Mẫu thông báo là thông báo được định dạng sẵn để tạo thông báo duy nhất của bạn bằng cách tùy chỉnh các tham số, kiểu dáng, liên kết theo dõi, v.v. Sau đây là một mẫu thông báo đầy ví dụ.

THÂN HÌNH:

Bạn đã mơ về nó. Chúng tôi đã thử. [TÊN MỤC] đã trở lại – chỉ đến [NGÀY].

Lời kêu gọi hành động:

Đặt hàng ngay. Hoặc, Lưu [TÊN MẶT HÀNG] của tôi

Lợi ích của việc sử dụng mẫu thông báo bao gồm duy trì định dạng nhất quán, giảm lỗi lè và tiết kiệm thời gian.

Cài đặt thông báo Người

dùng thư ờng nhận được quá nhiều thông báo mỗi ngày và họ có thể dễ dàng cảm thấy quá tải. Do đó, nhiều trang web và ứng dụng cung cấp cho người dùng quyền kiểm soát chi tiết đối với cài đặt thông báo. Thông tin này được lưu trữ trong bảng cài đặt thông báo, với các trường sau:

```
user_id bigint  
channel varchar # thông báo đầy, email hoặc SMS opt_in boolean  
# chọn tham gia để nhận thông báo Trước khi gửi bất kỳ  
thông báo nào tới người dùng, trước tiên chúng tôi kiểm tra xem người dùng có chọn tham gia nhận loại thông báo này hay không.
```

Giới hạn tốc độ

Để tránh làm người dùng quá tải với quá nhiều thông báo, chúng ta có thể giới hạn số lượng thông báo mà người dùng có thể nhận được. Điều này rất quan trọng vì người nhận có thể tắt hoàn toàn thông báo nếu chúng ta gửi quá thường xuyên.

Cơ chế thử lại Khi dịch

vụ của bên thứ ba không gửi được thông báo, thông báo sẽ được thêm vào hàng đợi tin nhắn để thử lại. Nếu sự cố vẫn tiếp diễn, cảnh báo sẽ được gửi đến các nhà phát triển.

Bảo mật trong thông báo đầy Đối với ứng

dụng iOS hoặc Android, appKey và appSecret được sử dụng để bảo mật API thông báo đầy [6]. Chỉ những khách hàng đã xác thực hoặc xác minh mới được phép gửi thông báo đầy bằng API của chúng tôi. Người dùng quan tâm nên tham khảo tài liệu tham khảo [6].

Giám sát thông báo xếp hàng Một số liệu

quan trọng cần giám sát là tổng số thông báo xếp hàng. Nếu số lượng lớn, các sự kiện thông báo không được xử lý đủ nhanh bởi các công nhân. Để tránh sự chậm trễ trong việc phân phối thông báo, cần có nhiều công nhân hơn.

Hình 10-12 (ghi công cho [7]) hiển thị một ví dụ về các thông báo xếp hàng cần được xử lý.

Queued messages (chart: last minute) (?)



Hình 10-12

Theo dõi sự kiện

Các số liệu thông báo, chẳng hạn như tỷ lệ mở, tỷ lệ nhấp và mức độ tương tác rất quan trọng trong việc hiểu hành vi của khách hàng. Dịch vụ phân tích triển khai theo dõi sự kiện. Thư ờng thì cần phải tích hợp giữa hệ thống thông báo và dịch vụ phân tích. Hình 10-13 cho thấy một ví dụ về các sự kiện có thể đư ợc theo dõi cho mục đích phân tích.

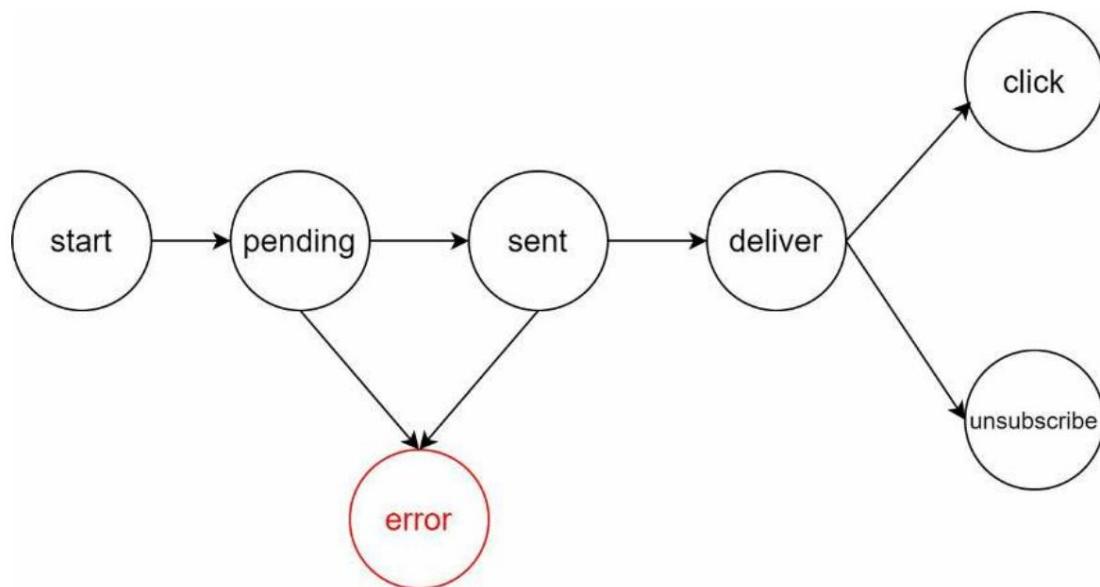


Figure 10-13

Thiết kế đư ợc cập

nhật Tổng hợp tất cả lại, Hình 10-14 cho thấy thiết kế hệ thống thông báo đư ợc cập nhật.

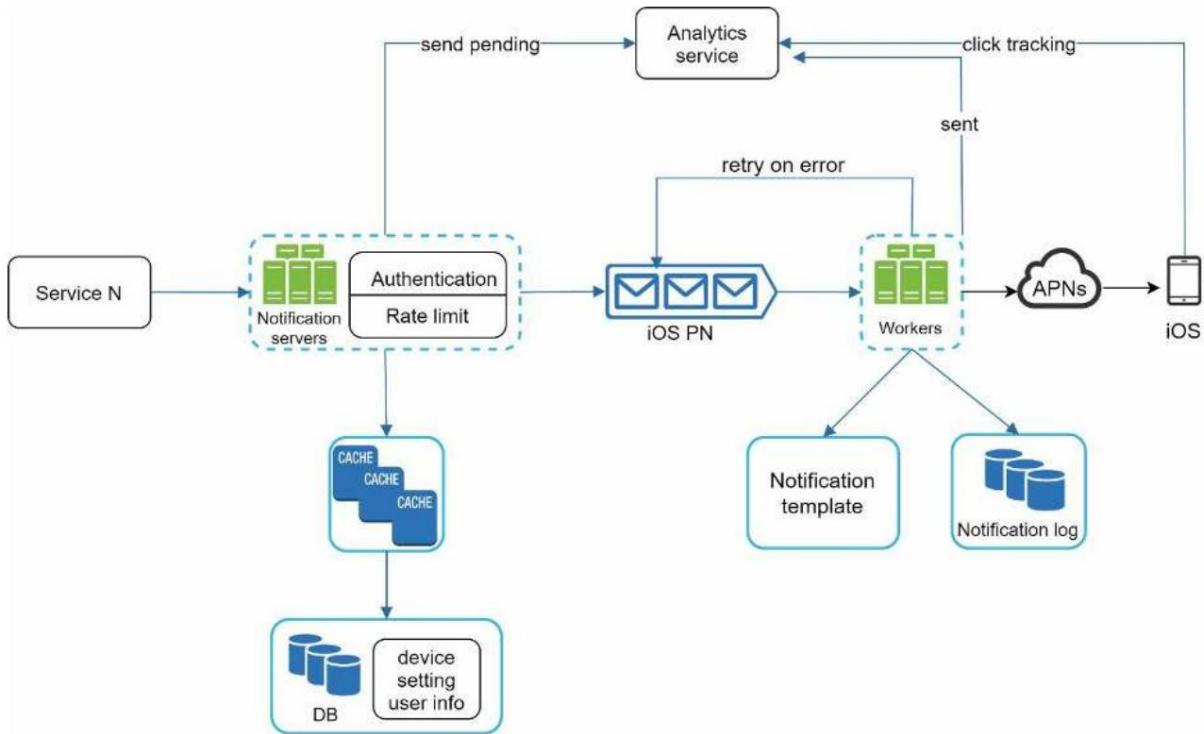


Figure 10-14

Trong thiết kế này, nhiều thành phần mới được thêm vào so với thiết kế trứ ớc đó.

- Máy chủ thông báo được trang bị thêm hai tính năng quan trọng nữa: xác thực và giới hạn tốc độ.

Chúng tôi cũng thêm cơ chế thử lại để xử lý lỗi thông báo. Nếu hệ thống không gửi được thông báo, thông báo sẽ được đưa trở lại hàng đợi tin nhắn và các nhân viên sẽ thử lại trong một số lần được xác định trước.

Hơn nữa, các mẫu thông báo cung cấp quy trình tạo thông báo nhất quán và hiệu quả.

Cuối cùng, các hệ thống giám sát và theo dõi được thêm vào để kiểm tra tình trạng hệ thống và cải tiến trong tương lai.

Bài Óc 4 - Tổng kết

Thông báo là điều không thể thiếu vì chúng giúp chúng ta cập nhật thông tin quan trọng. Đó có thể là thông báo đầy về bộ phim yêu thích của bạn trên Netflix, email về giảm giá sản phẩm mới hoặc tin nhắn xác nhận thanh toán mua sắm trực tuyến của bạn.

Trong chương này, chúng tôi mô tả thiết kế của một hệ thống thông báo có khả năng mở rộng hỗ trợ nhiều định dạng thông báo: thông báo đầy, tin nhắn SMS và email. Chúng tôi áp dụng hàng đợi tin nhắn để tách rời các thành phần hệ thống.

Bên cạnh thiết kế cấp cao, chúng tôi còn đào sâu hơn vào nhiều thành phần và tối ưu hóa hơn. • Độ tin cậy: Chúng tôi đề xuất một cơ chế thử lại mạnh mẽ để giảm thiểu tỷ lệ lỗi. • Bảo mật: Cặp AppKey/appSecret được sử dụng để đảm bảo chỉ những khách hàng đã xác minh mới có thể gửi thông báo.

- Theo dõi và giám sát: Những tính năng này được triển khai ở bất kỳ giai đoạn nào của luồng thông báo để nắm bắt các số liệu thống kê quan trọng. • Tôn trọng cài đặt của người dùng: Người dùng có thể chọn không nhận thông báo. Hệ thống của chúng tôi sẽ kiểm tra cài đặt của người dùng trước khi gửi thông báo. • Giới hạn tốc độ: Người dùng sẽ đánh giá cao việc giới hạn tần suất về số lượng thông báo họ nhận được.

Xin chúc mừng vì đã đi đư ợc đến đây! Đây giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

- [1] Twilio SMS: <https://www.twilio.com/sms>
- [2] Nexmo SMS: <https://www.nexmo.com/products/sms> [3]
- Sendgrid: <https://sendgrid.com/>
- [4] Mailchimp: <https://mailchimp.com/>
- [5] Bạn không thể có Giao hàng chính xác một lần: <https://bravenewgeek.com/you-cannot-have-exactly-once-delivery/>
- [6] Bảo mật trong Thông báo đầy: <https://cloud.ibm.com/docs/services/mobilepush?topic=mobile-pushnotification-security-in-push-notifications> [7]
- RabbitMQ: <https://bit.ly/2sotIa6>

CHƯƠNG 11: THIẾT KẾ HỆ THỐNG TIN TỨC

Trong chương này, bạn được yêu cầu thiết kế một hệ thống nguồn cấp tin tức. Nguồn cấp tin tức là gì? Theo trang trợ giúp của Facebook, "Nguồn cấp tin tức là danh sách các câu chuyện được cập nhật liên tục ở giữa trang chủ của bạn. Nguồn cấp tin tức bao gồm các cập nhật trạng thái, ảnh, video, liên kết, hoạt động ứng dụng và lượt thích từ những người, trang và nhóm mà bạn theo dõi trên Facebook" [1]. Đây là một câu hỏi phỏng vấn phổ biến. Các câu hỏi tương tự thường được hỏi là: thiết kế nguồn cấp tin tức Facebook, nguồn cấp tin tức Instagram, dòng thời gian Twitter, v.v.

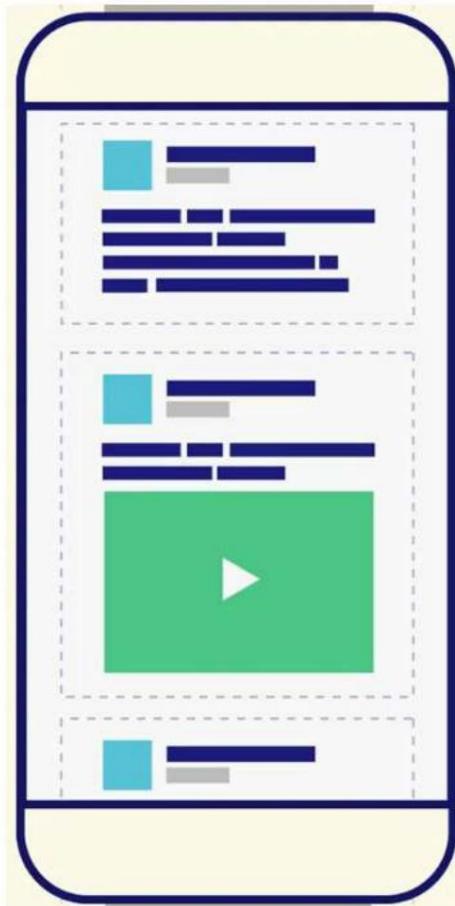


Figure 11-1 (source : <https://bit.ly/2Vv9JCm>)

Bút ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế Bộ câu hỏi làm rõ đầu tiên là để hiểu người phỏng vấn nghĩ gì khi cô ấy yêu cầu bạn thiết kế một hệ thống nguồn cấp tin tức. Ít nhất, bạn nên tìm ra những tính năng nào cần hỗ trợ. Sau đây là một ví dụ về tư duy tác giữa ứng viên và người phỏng vấn:

Ứng viên: Đây có phải là ứng dụng di động không? Hay ứng dụng web? Hoặc cả hai?

Người phỏng vấn: Cả hai

Ứng viên: Những đặc điểm quan trọng là gì?

Phỏng vấn: Người dùng có thể đăng bài viết và xem bài viết của bạn bè trên trang tin tức.

Ứng viên: Nguồn cấp tin tức được sắp xếp theo thứ tự thời gian người hay bất kỳ thứ tự cụ thể nào như điểm chủ đề? Ví dụ, bài đăng từ bạn bè thân thiết của bạn có điểm cao hơn.

Người phỏng vấn: Để đơn giản, chúng ta hãy giả sử nguồn cấp dữ liệu được sắp xếp theo thứ tự thời gian người lại.

Ứng viên: Một người dùng có thể có bao nhiêu bạn bè?

Người phỏng vấn: 5000

Ứng viên: Lượng giao thông là bao nhiêu?

Người phỏng vấn: 10 triệu DAU

Ứng viên: Nguồn cấp dữ liệu có thể chứa hình ảnh, video hoặc chỉ văn bản không?

Người phỏng vấn: Nó có thể chứa các tập tin phương tiện, bao gồm cả hình ảnh và video.

Bây giờ bạn đã thu thập được các yêu cầu, chúng tôi tập trung vào việc thiết kế hệ thống.

Bù ớc 2 - Đề xuất thiết kế cấp cao và nhận đư ợc sự đồng thuận Thiết kế
đư ợc chia thành hai luồng: xuất bản nguồn cấp dữ liệu và xây dựng nguồn cấp dữ liệu tin tức.

- Xuất bản nguồn cấp dữ liệu: khi ngư ời dùng xuất bản bài đăng, dữ liệu tương ứng đư ợc ghi vào bộ nhớ đệm và cơ sở dữ liệu. Bài đăng đư ợc đưa vào nguồn cấp dữ liệu tin tức của bạn bè. • Xây

dựng nguồn cấp dữ liệu tin tức: để đơn giản, chúng ta hãy giả sử nguồn cấp dữ liệu tin tức đư ợc xây dựng bằng cách tổng hợp các bài đăng của bạn bè theo thứ tự thời gian ngư ợc.

API nguồn cấp tin tức

API nguồn cấp tin tức là cách chính để khách hàng giao tiếp với máy chủ. Các API đó dựa trên HTTP cho phép khách hàng thực hiện các hành động, bao gồm đăng trạng thái, truy xuất nguồn cấp tin tức, thêm bạn bè, v.v. Chúng tôi thảo luận về hai API quan trọng nhất: API xuất bản nguồn cấp tin tức và API truy xuất nguồn cấp tin tức.

API xuất bản nguồn cấp dữ

liệu Để xuất bản bài đăng, yêu cầu HTTP POST sẽ đư ợc gửi đến máy chủ. API đư ợc hiển thị bên dưới: POST /v1/me/feed Tham số:

- content: nội dung là văn bản của bài đăng. •
- auth_token: đư ợc sử dụng để xác thực các yêu cầu API.

API truy xuất nguồn cấp tin tức

API để lấy nguồn cấp tin tức đư ợc hiển thị bên dưới:

GET /v1/me/feed Tham
số: •

auth_token: đư ợc sử dụng để xác thực các yêu cầu API.

Xuất bản nguồn cấp dữ

liệu Hình 11-2 cho thấy thiết kế cấp cao của luồng xuất bản nguồn cấp dữ liệu.

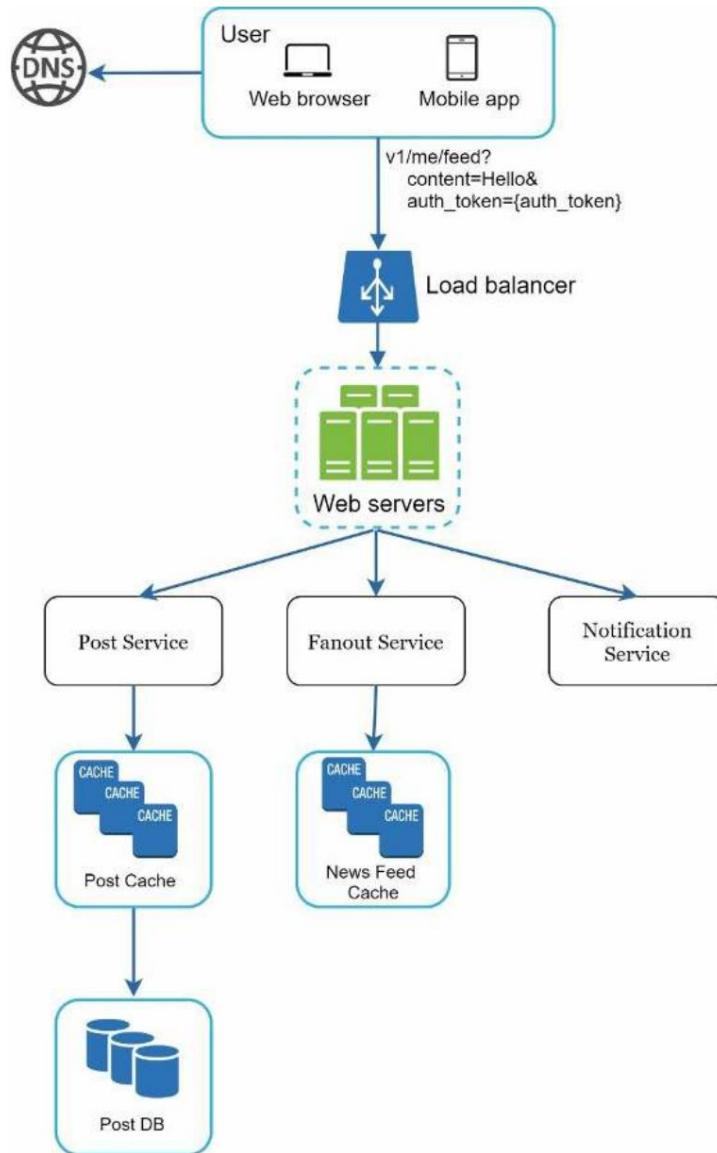


Figure 11-2

- Người dùng: người dùng có thể xem nguồn cấp tin tức trên trình duyệt hoặc ứng dụng di động. Người dùng tạo bài đăng có nội dung "Xin chào" thông qua

API: /v1/me/feed?content=Hello&auth_token={auth_token} • Bộ cân bằng tải:

phân phối lưu lượng đến máy chủ web.

- Máy chủ web: máy chủ web chuyển hướng lưu lượng truy cập đến các dịch vụ nội bộ khác nhau.
- Dịch vụ đăng bài: lưu bài đăng trong cơ sở dữ liệu và bộ nhớ đệm.
- Dịch vụ Fanout: đẩy nội dung mới vào nguồn cấp tin tức của bạn bè. Dữ liệu nguồn cấp tin tức được lưu trữ trong bộ nhớ đệm để truy xuất nhanh.
- Dịch vụ thông báo: thông báo cho bạn bè về nội dung mới và gửi thông báo đẩy.

Xây dựng nguồn cấp tin tức

Trong phần này, chúng tôi thảo luận về cách xây dựng nguồn cấp tin tức ở hậu trường. Hình 11-3 cho thấy thiết kế cấp cao:

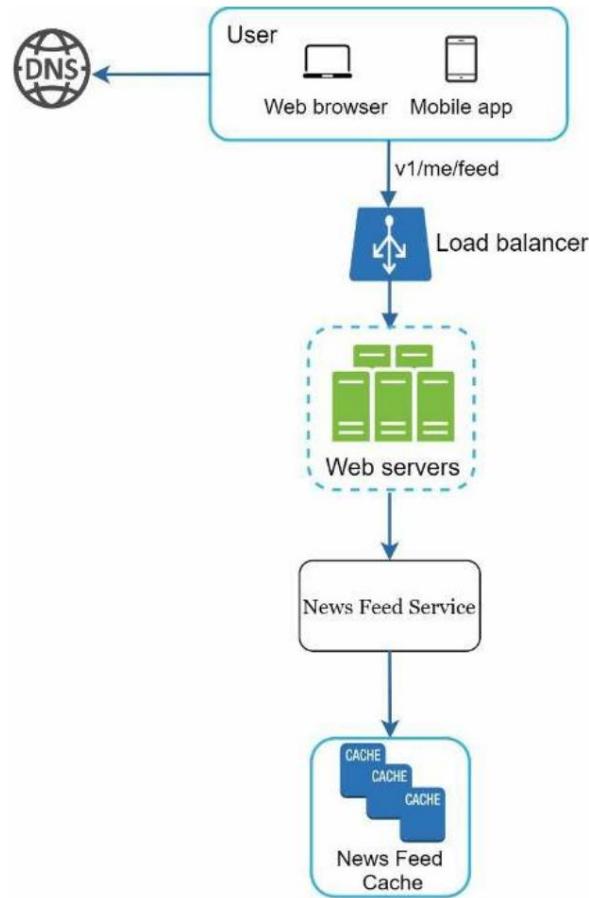


Figure 11-3

- Người dùng: người dùng gửi yêu cầu để lấy nguồn cấp tin tức của mình. Yêu cầu trông như thế này: / v1/me/feed.
- Bộ cân bằng tải: bộ cân bằng tải chuyển hướng lưu lượng truy cập đến máy chủ web.
- Máy chủ web: máy chủ web định tuyến các yêu cầu đến dịch vụ nguồn cấp tin tức. • Dịch vụ nguồn cấp tin tức: dịch vụ nguồn cấp tin tức lấy nguồn cấp tin tức từ bộ nhớ đệm.
- Bộ nhớ đệm nguồn cấp tin tức: lưu trữ ID nguồn cấp tin tức cần thiết để hiển thị nguồn cấp tin tức.

Bài 3 - Thiết kế chuyên sâu Thiết

kết cấu cao bao gồm tóm tắt hai luồng: xuất bản nguồn cấp dữ liệu và xây dựng nguồn cấp dữ liệu tin tức.

Ở đây, chúng ta sẽ thảo luận sâu hơn về những chủ đề đó.

Đi sâu vào việc xuất bản nguồn cấp

dữ liệu Hình 11-4 phác thảo thiết kế chi tiết cho việc xuất bản nguồn cấp dữ liệu. Chúng ta đã thảo luận hầu hết các thành phần trong thiết kế cấp cao và chúng ta sẽ tập trung vào hai thành phần: máy chủ web và dịch vụ fanout.

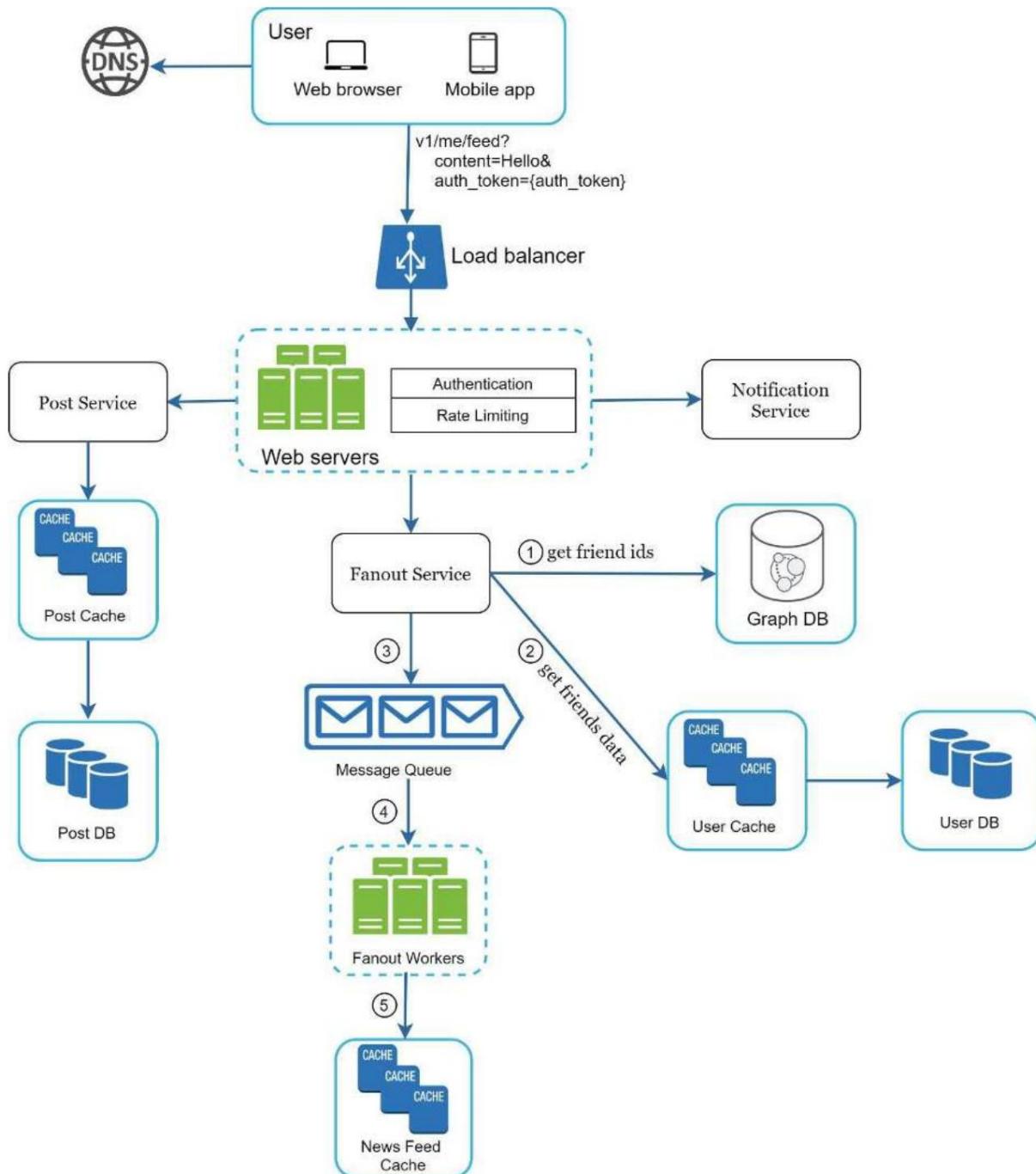


Figure 11-4

Máy chủ web

Bên cạnh việc giao tiếp với khách hàng, máy chủ web còn thực thi xác thực và giới hạn tốc độ.

Chỉ những người dùng đã đăng nhập bằng auth_token hợp lệ mới được phép đăng bài. Hệ thống giới hạn số bài đăng mà người dùng có thể đăng trong một khoảng thời gian nhất định, điều này rất quan trọng để ngăn chặn thư rác và lạm dụng nội dung.

Dịch vụ Fanout

Fanout là quá trình gửi bài đăng đến tất cả bạn bè. Có hai loại mô hình fanout: fanout khi ghi (còn gọi là mô hình đẩy) và fanout khi đọc (còn gọi là mô hình kéo). Cả hai mô hình đều có ưu và nhược điểm. Chúng tôi giải thích quy trình làm việc của chúng và khám phá cách tiếp cận tốt nhất để hỗ trợ hệ thống của chúng tôi.

Fanout khi viết. Với cách tiếp cận này, nguồn cấp tin tức được tính toán trước trong thời gian viết. Một bài đăng mới được gửi đến bộ nhớ đệm của bạn bè ngay sau khi được xuất bản.

Ưu điểm:

- Nguồn cấp tin tức được tạo theo thời gian thực và có thể được gửi đến bạn bè ngay lập tức.
- Việc lấy nguồn cấp tin tức nhanh chóng vì nguồn cấp tin tức được tính toán trước trong thời gian viết.

Nhược điểm:

- Nếu người dùng có nhiều bạn bè, việc lấy danh sách bạn bè và tạo nguồn cấp tin tức cho tất cả họ sẽ chậm và tốn thời gian. Nó được gọi là vấn đề phím nóng.
- Đối với những người dùng không hoạt động hoặc những người viết hiếm khi đăng nhập, việc tính toán trước nguồn cấp tin tức sẽ lãng phí thời gian tính toán tài nguyên.

Fanout khi đọc. Nguồn cấp tin tức được tạo ra trong thời gian đọc. Đây là mô hình theo yêu cầu.

Các bài đăng gần đây sẽ được kéo xuống khi người dùng tải trang chủ của mình.

Ưu điểm:

- Đối với người dùng không hoạt động hoặc những người viết hiếm khi đăng nhập, việc bật chế độ đọc sẽ hiệu quả hơn vì nó sẽ không lãng phí tài nguyên máy tính của họ.
- Dữ liệu không được đẩy cho bạn bè nên không có vấn đề về phím nóng.

Nhược điểm:

- Việc lấy nguồn cấp tin tức chậm vì nguồn cấp tin tức không được tính toán trước.

Chúng tôi áp dụng phương pháp kết hợp để tận dụng lợi ích của cả hai phương pháp và tránh những sai lầm trong cả hai phương pháp.

Vì việc tải tin tức nhanh chóng là rất quan trọng nên chúng tôi sử dụng mô hình đẩy cho phần lớn người dùng.

Đối với những người nói tiếng hoặc người dùng có nhiều bạn bè/người dùng theo dõi, chúng tôi cho phép người dùng theo dõi nội dung tin tức theo yêu cầu để tránh quá tải hệ thống. Bước nhất quán là một kỹ thuật hữu ích để giảm thiểu vấn đề phím nóng vì nó giúp phân phối các yêu cầu/dữ liệu đồng đều hơn.

Chúng ta hãy xem xét kỹ hơn dịch vụ fanout như thể hiện trong Hình 11-5.

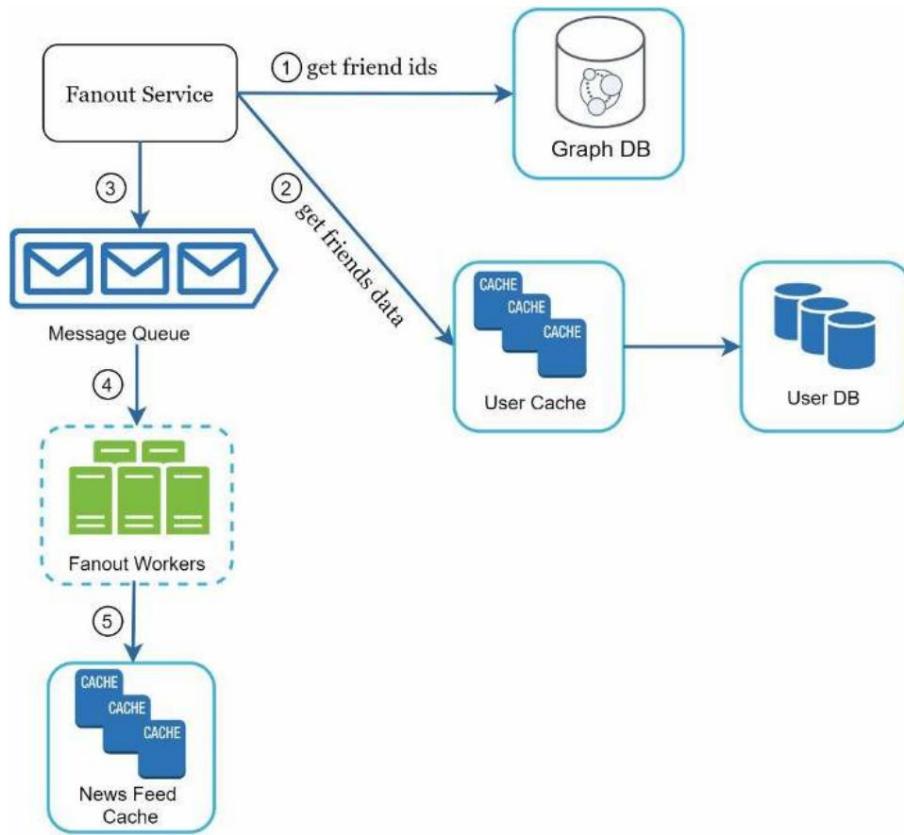


Figure 11-5

Dịch vụ fanout hoạt động như sau:

1. Lấy ID bạn bè từ cơ sở dữ liệu đồ thị. Cơ sở dữ liệu đồ thị phù hợp để quản lý mối quan hệ bạn bè và đề xuất bạn bè. Đặc già quan tâm muốn tìm hiểu thêm về khái niệm này nên tham khảo tài liệu tham khảo [2].
2. Lấy thông tin bạn bè từ bộ nhớ đệm của người dùng. Sau đó, hệ thống sẽ lọc bạn bè dựa trên cài đặt của người dùng. Ví dụ, nếu bạn tắt tiếng ai đó, bài đăng của người đó sẽ không hiển thị trên nguồn cấp tin tức của bạn mặc dù bạn vẫn là bạn bè. Một lý do khác khiến bài đăng không hiển thị là người dùng có thể chia sẻ thông tin một cách có chọn lọc với những người bạn cụ thể hoặc ẩn thông tin đó khỏi những người khác.
3. Gửi danh sách bạn bè và ID bài đăng mới vào hàng đợi tin nhắn.
4. Các công nhân Fanout lấy dữ liệu từ hàng đợi tin nhắn và lưu trữ dữ liệu nguồn cấp tin tức trong bộ nhớ nguồn cấp tin tức. Bạn có thể coi bộ nhớ nguồn cấp tin tức như một bảng ánh xạ $\langle \text{post_id}, \text{user_id} \rangle$.
Bất cứ khi nào một bài đăng mới được tạo, nó sẽ được thêm vào bảng nguồn cấp tin tức như thể hiện trong Hình 11-6. Mức tiêu thụ bộ nhớ có thể trở nên rất lớn nếu chúng ta lưu trữ toàn bộ người dùng và các đối tượng bài đăng trong bộ nhớ đệm. Do đó, chỉ có ID được lưu trữ. Để giữ cho kích thước bộ nhớ nhỏ, chúng tôi đặt một giới hạn có thể định cấu hình. Khả năng người dùng cuộn qua hàng nghìn bài đăng trong nguồn cấp tin tức là rất thấp. Hầu hết người dùng chỉ quan tâm đến nội dung mới nhất, do đó tỷ lệ bỏ lỡ bộ nhớ đệm là thấp.
5. Lưu trữ $\langle \text{post_id}, \text{user_id} \rangle$ trong bộ nhớ đệm nguồn cấp tin tức. Hình 11-6 cho thấy ví dụ về giao diện của nguồn cấp tin tức trong bộ nhớ đệm.

post_id	user_id

Figure 11-6

Phân tích sâu về việc truy xuất nguồn cấp

tin tức Hình 11-7 minh họa thiết kế chi tiết cho việc truy xuất nguồn cấp tin tức.

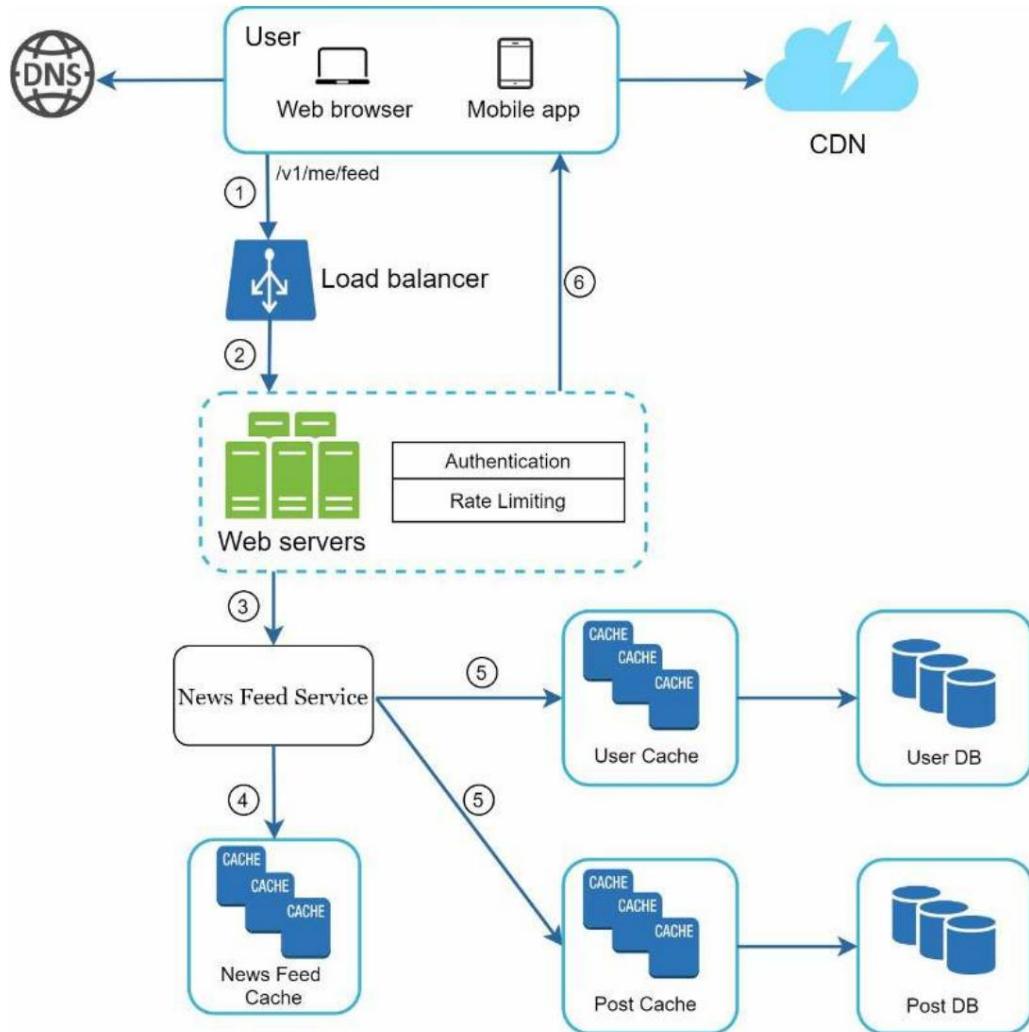


Figure 11-7

Như thể hiện trong Hình 11-7, nội dung phuơng tiện (hình ảnh, video, v.v.) được lưu trữ trong CDN để truy xuất nhanh. Chúng ta hãy xem cách máy khách truy xuất nguồn cấp tin tức.

1. Người dùng gửi yêu cầu để lấy nguồn cấp tin tức của mình. Yêu cầu trông như thế này: /v1/me/feed
2. Bộ cân bằng tải phân phối lại yêu cầu đến máy chủ web.
3. Máy chủ web gọi dịch vụ nguồn cấp tin tức để lấy nguồn cấp tin tức.
4. Dịch vụ nguồn cấp tin tức lấy danh sách ID bài đăng từ bộ đệm nguồn cấp tin tức.
5. Nguồn cấp tin tức của người dùng không chỉ là danh sách ID nguồn cấp tin tức. Nó chứa tên người dùng, ảnh đại diện, nội dung bài đăng, ảnh bài đăng, v.v. Do đó, dịch vụ nguồn cấp tin tức sẽ lấy toàn bộ đối tượng người dùng và bài đăng từ bộ nhớ đệm (bộ nhớ đệm người dùng và bộ nhớ đệm bài đăng) để xây dựng nguồn cấp tin tức đủ cung cấp đầy đủ.
6. Nguồn cấp tin tức đầy đủ sẽ được trả về dưới dạng JSON cho máy khách để hiển thị.

Kiến trúc bộ nhớ đệm Bộ nhớ đệm

cực kỳ quan trọng đối với hệ thống nguồn cấp tin tức. Chúng tôi chia tầng bộ nhớ đệm thành 5 lớp như thể hiện trong Hình 11-8.

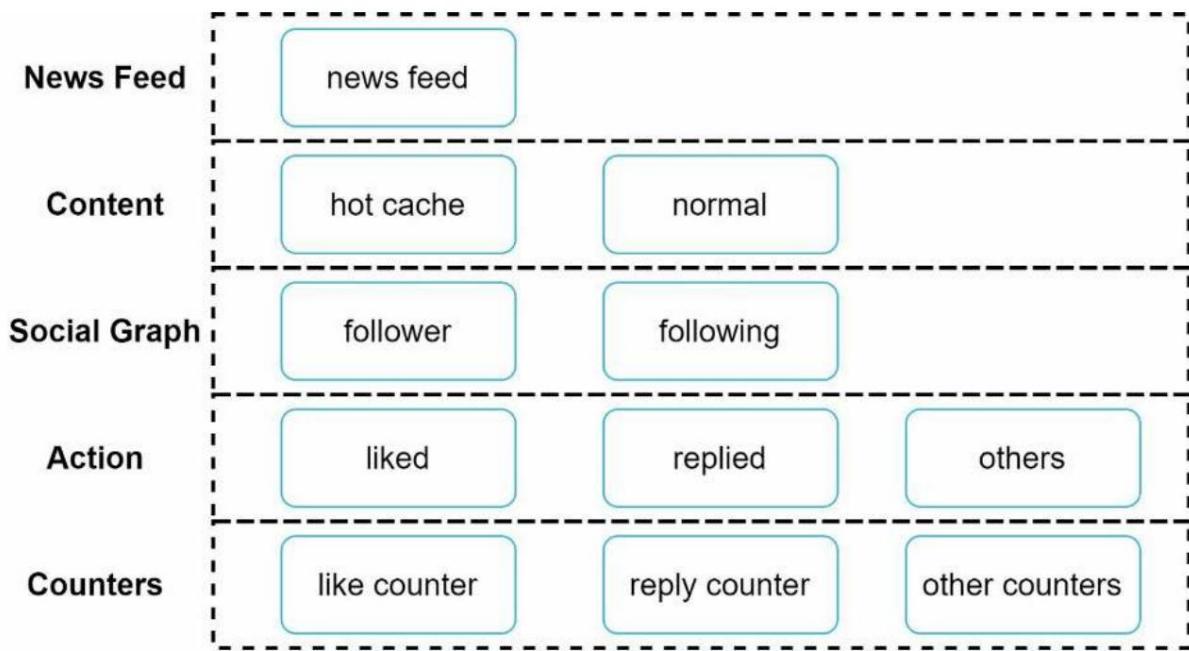


Figure 11-8

- Nguồn cấp tin tức: Lưu trữ ID của các nguồn cấp tin tức.
- Nội dung: Lưu trữ mọi dữ liệu bài đăng. Nội dung phổ biến được lưu trữ trong bộ nhớ đệm nóng. • Biểu đồ xã hội: Lưu trữ dữ liệu mối quan hệ của người dùng. • Hành động: Lưu trữ thông tin về việc người dùng thích bài đăng, trả lời bài đăng hay thực hiện các hành động khác trên bài đăng hay không. •
- Bộ đếm: Lưu trữ bộ đếm cho lượt thích, trả lời, người theo dõi, đang theo dõi, v.v.

Bài 8c 4 - Tổng kết Trong

chương này, chúng tôi đã thiết kế một hệ thống nguồn cấp tin tức. Thiết kế của chúng tôi bao gồm hai luồng: xuất bản nguồn cấp tin tức và truy xuất nguồn cấp tin tức.

Giống như bất kỳ câu hỏi phòng vấn thiết kế hệ thống nào, không có cách hoàn hảo nào để thiết kế một hệ thống.

Mỗi công ty đều có những hạn chế riêng và bạn phải thiết kế một hệ thống phù hợp với những hạn chế đó.

Hiểu được sự đánh đổi giữa thiết kế và lựa chọn công nghệ của bạn là điều quan trọng. Nếu còn vài phút nữa, bạn có thể nói về các vấn đề về khả năng mở rộng. Để tránh thảo luận trùng lặp, chỉ những điểm thảo luận cấp cao được liệt kê bên dưới.

Mở rộng cơ sở dữ liệu:

- Mở rộng theo chiều dọc so với mở rộng theo

chiều ngang • SQL so

với NoSQL • Sao chép chủ-tớ •

Bản sao đọc • Mô

hình nhất quán • Phân

mảnh cơ sở dữ liệu

Các điểm thảo luận khác:

- Giữ trạng thái không có

trạng thái của tầng web • Lưu trữ dữ

liệu đậm nhiều nhất có thể • Hỗ trợ

nhiều trung tâm dữ liệu • Mất một số thành phần với hàng

đợi tin nhắn • Theo dõi các số liệu chính. Ví dụ, QPS trong giờ cao điểm và độ trễ khi người dùng

làm mới nguồn cấp tin tức của họ là những thông tin thú vị để theo dõi.

Xin chúc mừng vì đã đi được đến đây! Bây giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

[1] Nguồn cấp tin tức hoạt

động như thế nào: <https://www.facebook.com/help/327131014036297/>

[2] Khuyến nghị của Friend of Friend Neo4j

và SQL Sever: <http://geekswithblogs.net/brendonpage/archive/2015/10/26/friend-of-friend-recommendations-with-neo4j.aspx>

CHƯƠNG 12: THIẾT KẾ HỆ THỐNG CHAT

Trong chương này, chúng ta sẽ khám phá thiết kế của một hệ thống trò chuyện. Hầu như mọi người đều sử dụng ứng dụng trò chuyện.

Hình 12-1 hiển thị một số ứng dụng phổ biến nhất trên thị trường.



Figure 12-1

Ứng dụng trò chuyện thực hiện các chức năng khác nhau cho những người khác nhau. Điều cực kỳ quan trọng là phải xác định chính xác các yêu cầu. Ví dụ, bạn không muốn thiết kế một hệ thống tập trung vào trò chuyện nhóm khi người dùng vẫn đang nghĩ đến trò chuyện một-một. Điều quan trọng là phải khám phá các yêu cầu về tính năng.

Bu ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế Điều quan trọng là phải thống nhất về loại ứng dụng trò chuyện để thiết kế. Trên thị trường, có các ứng dụng trò chuyện một-một như Facebook Messenger, WeChat và WhatsApp, các ứng dụng trò chuyện văn phòng tập trung vào trò chuyện nhóm như Slack hoặc các ứng dụng trò chuyện trò chơi, như Discord, tập trung vào tư ng tác nhóm lớn và độ trễ trò chuyện bằng giọng nói thấp.

Bộ câu hỏi làm rõ đầu tiên phải nêu rõ người phỏng vấn nghĩ gì khi yêu cầu bạn thiết kế hệ thống trò chuyện. Ít nhất, hãy tìm hiểu xem bạn nên tập trung vào ứng dụng trò chuyện một-một hay trò chuyện nhóm. Một số câu hỏi bạn có thể hỏi như sau:

Ứng viên: Chúng ta sẽ thiết kế loại ứng dụng trò chuyện nào? 1-1 hay theo nhóm?

Người phỏng vấn: Nó phải hỗ trợ cả trò chuyện 1-1 và trò chuyện nhóm.

Ứng viên: Đây có phải là ứng dụng di động không? Hay ứng dụng web? Hoặc cả hai?

Người phỏng vấn: Cả hai.

Ứng viên: Quy mô của ứng dụng này là bao nhiêu? Ứng dụng khởi nghiệp hay quy mô lớn?

Người phỏng vấn: Nó có thể hỗ trợ 50 triệu người dùng hoạt động hàng ngày (DAU).

Ứng viên: Đối với trò chuyện nhóm, giới hạn thành viên nhóm là bao nhiêu?

Người phỏng vấn: Tối đa 100 người

Ứng viên: Những tính năng nào là quan trọng đối với ứng dụng trò chuyện? Nó có hỗ trợ tệp đính kèm không?

Người phỏng vấn: Trò chuyện 1-1, trò chuyện nhóm, chỉ báo trực tuyến. Hệ thống chỉ hỗ trợ tin nhắn văn bản.

Ứng viên: Có giới hạn kích thước tin nhắn không?

Người phỏng vấn: Có, độ dài văn bản phải dưới 100.000 ký tự.

Ứng viên: Có yêu cầu mã hóa đầu cuối không?

Người phỏng vấn: Hiện tại thì không bắt buộc nhưng chúng ta sẽ thảo luận nếu có thời gian.

Ứng viên: Chúng tôi sẽ lưu trữ lịch sử trò chuyện trong bao lâu?

Người phỏng vấn: Mỗi mãi.

Trong chương này, chúng tôi tập trung vào việc thiết kế một ứng dụng trò chuyện như Facebook Messenger, nhấn mạnh vào các tính năng sau:

- Trò chuyện một-một với độ trễ gửi thấp
- Trò chuyện nhóm nhỏ (tối đa 100 người)
- Hiện diện trực tuyến
- Hỗ trợ nhiều thiết bị

Cùng một tài khoản có thể đăng nhập vào nhiều tài khoản cùng một lúc.

- Thông báo đầy

Việc thống nhất về quy mô thiết kế cũng rất quan trọng. Chúng tôi sẽ thiết kế một hệ thống hỗ trợ 50 triệu DAU.

Bút ớc 2 - Đề xuất thiết kế cấp cao và nhận đư ợc sự đồng thuận Đề phát

triển một thiết kế chất lư ợng cao, chúng ta cần có kiến thức cơ bản về cách máy khách và máy chủ giao tiếp. Trong hệ thống trò chuyện, máy khách có thể là ứng dụng di động hoặc ứng dụng web. Máy khách không giao tiếp trực tiếp với nhau. Thay vào đó, mỗi máy khách kết nối với một dịch vụ trò chuyện, hỗ trợ tất cả các tính năng đư ợc đề cập ở trên. Chúng ta hãy tập trung vào các hoạt động cơ bản. Dịch vụ trò chuyện phải hỗ trợ các chức năng sau:

- Nhận tin nhắn từ các máy khách khác. • Tìm đúng ngư ời nhận cho mỗi tin nhắn và chuyển tiếp tin nhắn đến ngư ời nhận. • Nếu ngư ời nhận không trực tuyến, hãy giữ tin nhắn cho ngư ời nhận đó trên máy chủ cho đến khi ngư ời đó trực tuyến.

Hình 12-2 hiển thị mối quan hệ giữa máy khách (ngư ời gửi và ngư ời nhận) và dịch vụ trò chuyện.

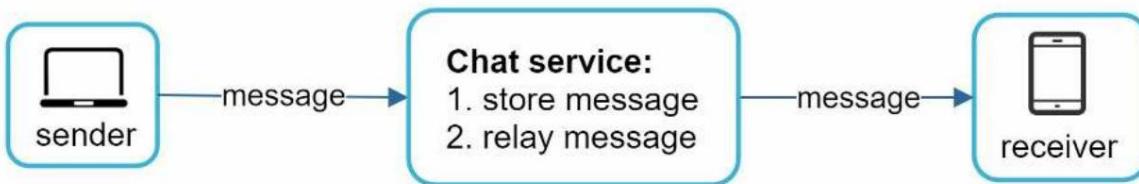


Figure 12-2

Khi khách hàng có ý định bắt đầu trò chuyện, họ sẽ kết nối dịch vụ trò chuyện bằng một hoặc nhiều giao thức mạng. Đối với dịch vụ trò chuyện, việc lựa chọn giao thức mạng là rất quan trọng. Chúng ta hãy thảo luận điều này với ngư ời phỏng vấn.

Yêu cầu đư ợc khởi tạo bởi máy khách đối với hầu hết các ứng dụng máy khách/máy chủ. Điều này cũng đúng đối với phía ngư ời gửi của ứng dụng trò chuyện. Trong Hình 12-2, khi ngư ời gửi gửi tin nhắn đến ngư ời nhận qua dịch vụ trò chuyện, nó sử dụng giao thức HTTP đã đư ợc kiểm tra theo thời gian, đây là giao thức web phổ biến nhất. Trong trường hợp này, máy khách mở kết nối HTTP với dịch vụ trò chuyện và gửi tin nhắn, thông báo cho dịch vụ gửi tin nhắn đến ngư ời nhận. Keep-alive hiệu quả cho việc này vì tiêu đề keep-alive cho phép máy khách duy trì kết nối liên tục với dịch vụ trò chuyện. Nó cũng làm giảm số lư ợng bắt tay TCP.

HTTP là một lựa chọn tốt ở phía ngư ời gửi và nhiều ứng dụng trò chuyện phổ biến như Facebook [1] ban đầu đã sử dụng HTTP để gửi tin nhắn.

Tuy nhiên, phía ngư ời nhận phức tạp hơn một chút. Vì HTTP đư ợc khởi tạo bởi máy khách, nên việc gửi tin nhắn từ máy chủ không phải là chuyện đơn giản. Trong nhiều năm qua, nhiều kỹ thuật đư ợc sử dụng để mô phỏng kết nối do máy chủ khởi tạo: polling, long polling và WebSocket. Đó là những kỹ thuật quan trọng đư ợc sử dụng rộng rãi trong các cuộc phỏng vấn thiết kế hệ thống, vì vậy chúng ta hãy cùng xem xét từng kỹ thuật.

Polling

Như thể hiện trong Hình 12-3, polling là một kỹ thuật mà máy khách định kỳ hỏi máy chủ xem có tin nhắn nào khả dụng không. Tùy thuộc vào tần suất polling, polling có thể tốn kém. Nó có thể tiêu tốn tài nguyên máy chủ quý giá để trả lời một câu hỏi mà hầu hết thời gian đều trả lời là không.

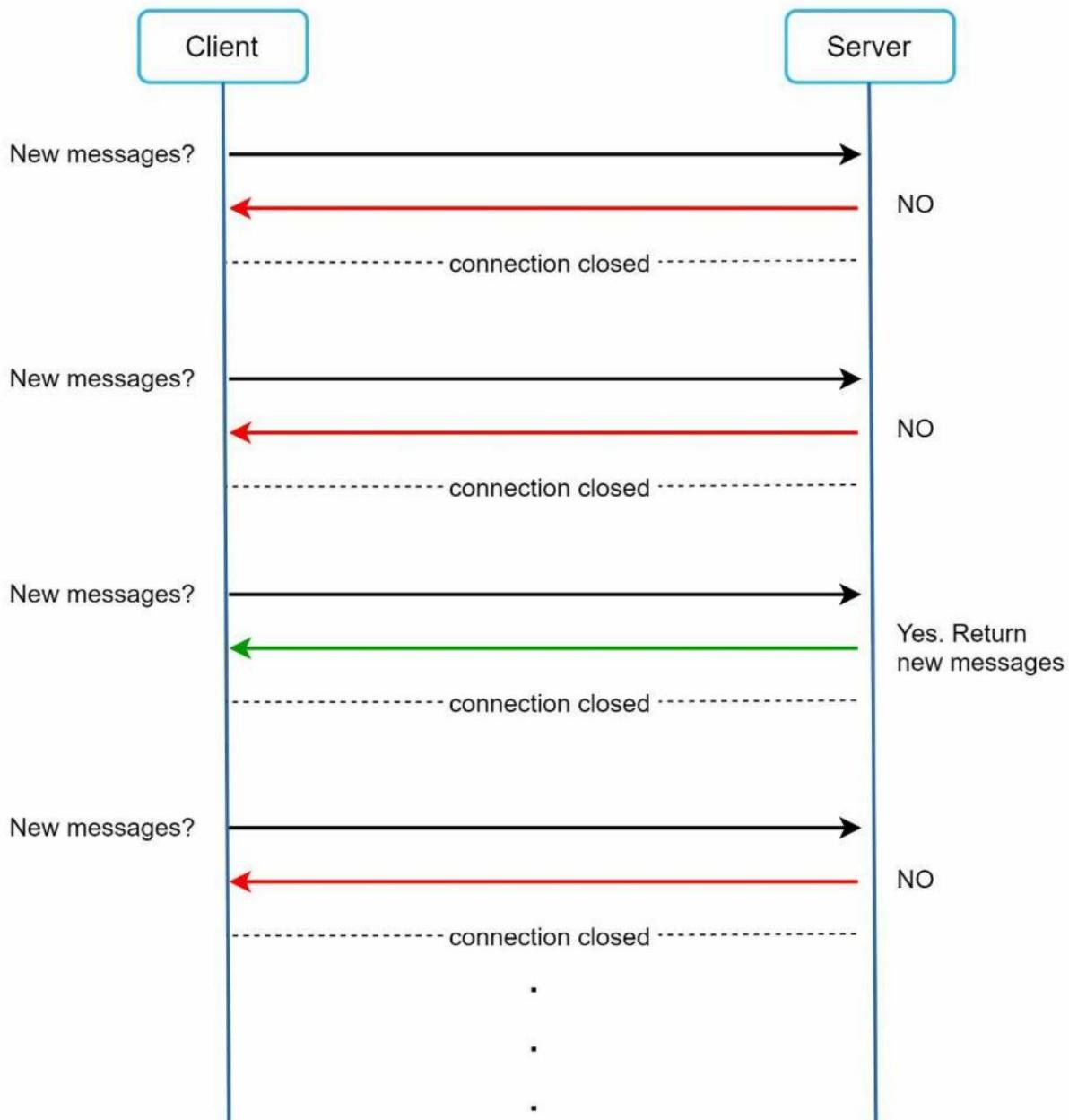


Figure 12-3

Kiểm tra dài

Vì kiểm tra có thể không hiệu quả nên tiến trình tiếp theo là kiểm tra dài (Hình 12-4).

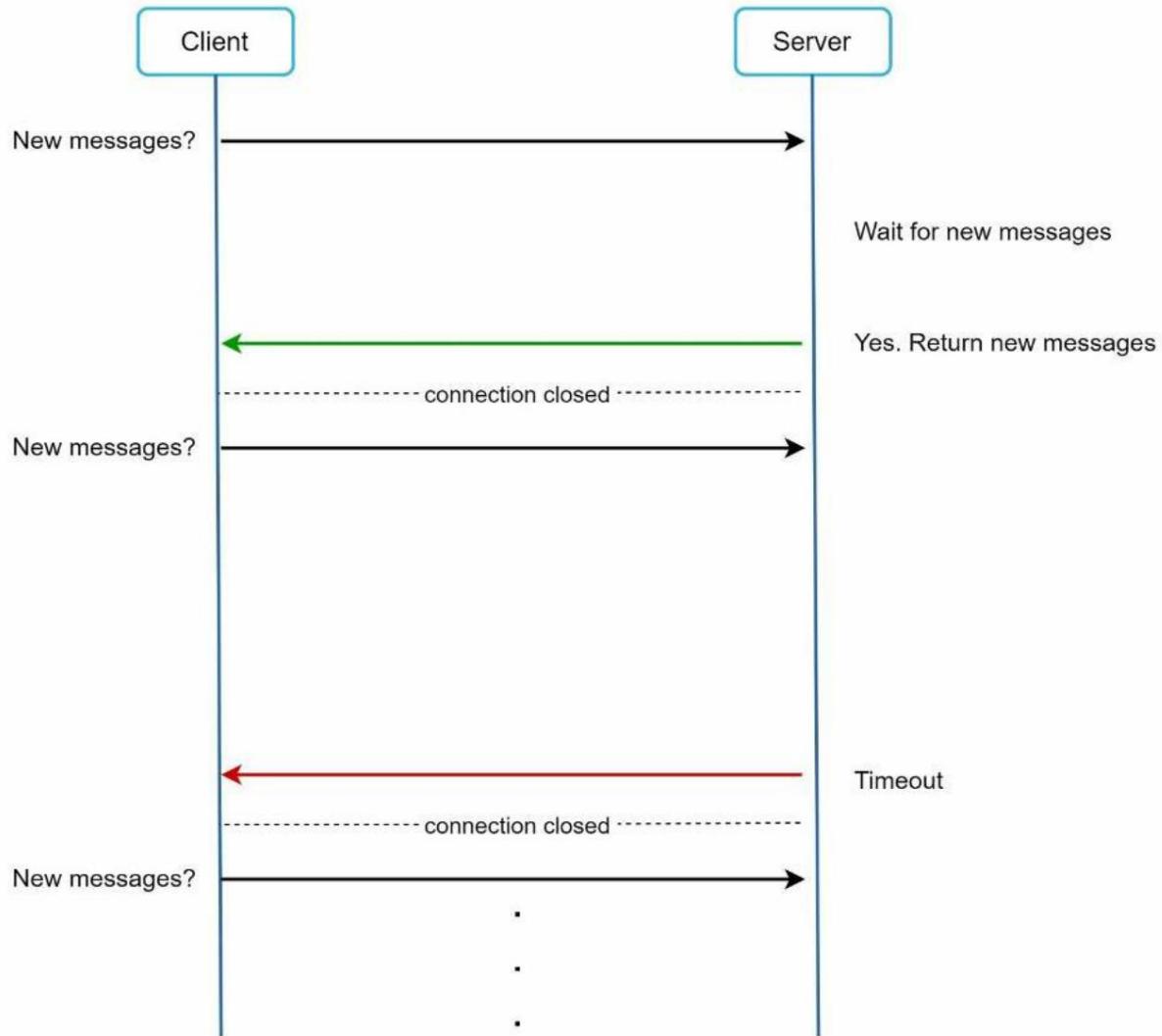


Figure 12-4

Trong thăm dò dài, máy khách giữ kết nối mở cho đến khi thực sự có tin nhắn mới hoặc đạt đến người thòi gian chờ. Khi máy khách nhận đư ợc tin nhắn mới, nó sẽ ngay lập tức gửi một yêu cầu khác đến máy chủ, khởi động lại quy trình. Thăm dò dài có một số nhú ợc điểm:

- Người gửi và người nhận có thể không kết nối đến cùng một máy chủ trò chuyện. Các máy chủ dựa trên HTTP thường không có trạng thái. Nếu bạn sử dụng vòng tròn để cân bằng tải, máy chủ nhận đư ợc tin nhắn có thể không có kết nối thăm dò dài với máy khách nhận đư ợc tin nhắn.
- Máy chủ không có cách nào tốt để biết máy khách có bị ngắt kết nối hay không.
- Không hiệu quả. Nếu người dùng không trò chuyện nhiều, thăm dò dài vẫn tạo kết nối định kỳ sau khi hết thời gian chờ.

WebSocket

WebSocket là giải pháp phổ biến nhất để gửi các bản cập nhật không đồng bộ từ máy chủ đến máy khách. Hình 12-5 cho thấy cách thức hoạt động của nó.

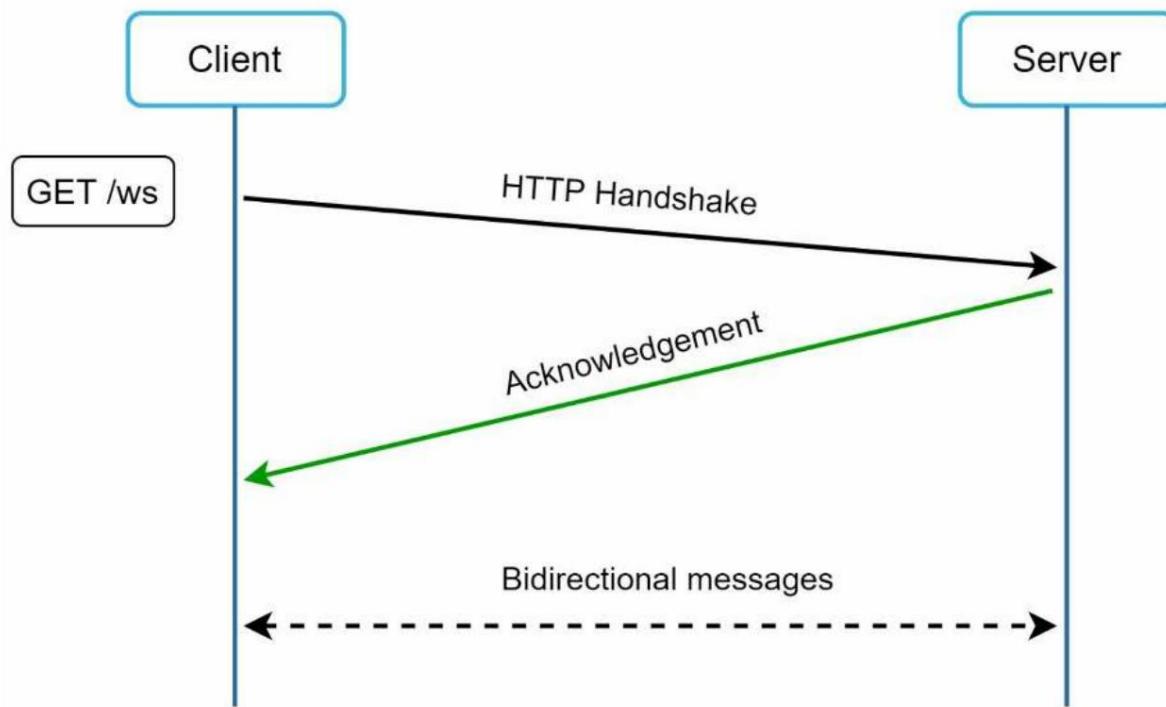


Figure 12-5

Kết nối WebSocket được khởi tạo bởi máy khách. Nó là kết nối hai chiều và liên tục. Nó bắt đầu cuộc sống của mình như một kết nối HTTP và có thể được "nâng cấp" thông qua một số bắt tay được xác định rõ ràng với kết nối WebSocket. Thông qua kết nối liên tục này, máy chủ có thể gửi các bản cập nhật cho máy khách. Các kết nối WebSocket thường hoạt động ngay cả khi có tia lửa. Điều này là do chúng sử dụng cổng 80 hoặc 443 cũng được sử dụng bởi các kết nối HTTP/HTTPS.

Trước đó chúng tôi đã nói rằng ở phía người gửi, HTTP là một giao thức tốt để sử dụng, nhưng vì WebSocket là giao thức song song nên không có lý do kỹ thuật mạnh mẽ nào để không sử dụng nó cho cả việc gửi. Hình 12-6 cho thấy cách WebSockets (ws) được sử dụng cho cả phía người gửi và phía người nhận.

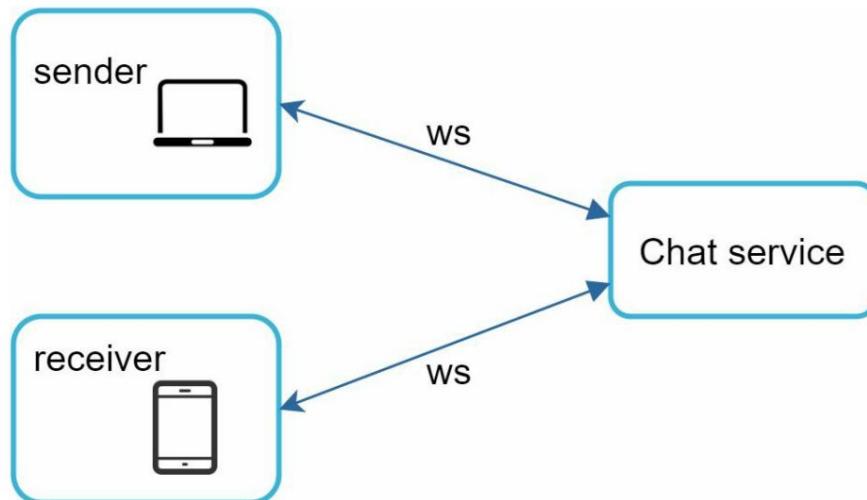


Figure 12-6

Bằng cách sử dụng WebSocket cho cả việc gửi và nhận, nó đơn giản hóa thiết kế và giúp việc triển khai trên cả máy khách và máy chủ trở nên dễ dàng hơn. Vì các kết nối WebSocket là liên tục, nên việc quản lý kết nối hiệu quả là rất quan trọng ở phía máy chủ.

Thiết kế cấp cao Vừa

rồi chúng ta đã đề cập rằng WebSocket được chọn làm giao thức truyền thông chính giữa máy khách và máy chủ cho giao tiếp hai chiều của nó, điều quan trọng cần lưu ý là mọi thứ khác không nhất thiết phải là WebSocket. Trên thực tế, hầu hết các tính năng (đăng ký, đăng nhập, hồ sơ người dùng, v.v.) của ứng dụng trò chuyện có thể sử dụng phư ơng thức yêu cầu/phản hồi truyền thông qua HTTP. Chúng ta hãy đi sâu hơn một chút và xem xét các thành phần cấp cao của hệ thống.

Như thể hiện trong Hình 12-7, hệ thống trò chuyện được chia thành ba loại chính: dịch vụ không trạng thái, dịch vụ có trạng thái và tích hợp của bên thứ ba.

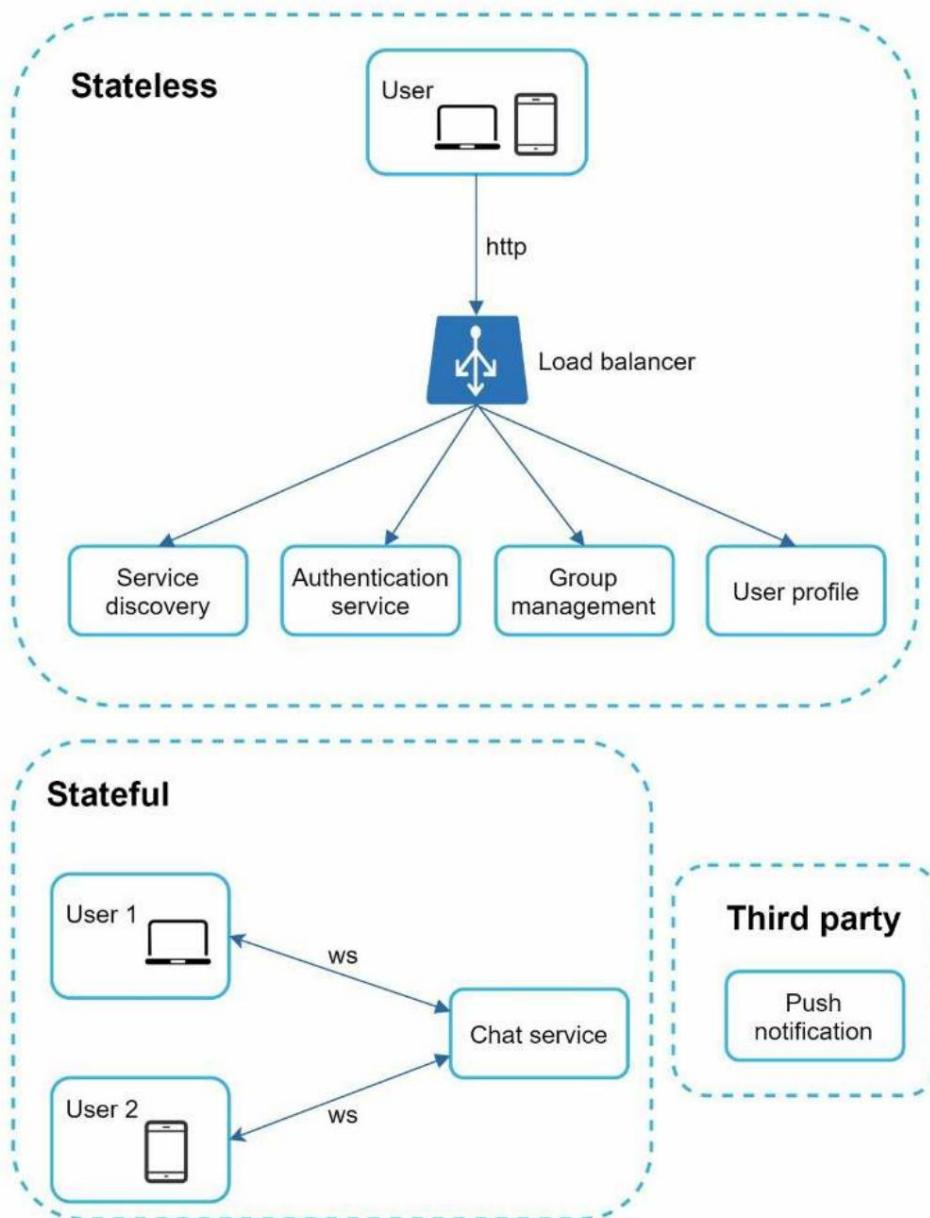


Figure 12-7

Dịch vụ không quốc tịch

Dịch vụ không trạng thái là dịch vụ yêu cầu/phản hồi công khai truyền thông, được sử dụng để quản lý thông tin đăng nhập, đăng ký, hồ sơ người dùng, v.v. Đây là những tính năng phổ biến trong nhiều trang web và ứng dụng.

Các dịch vụ không trạng thái nằm sau một bộ cân bằng tải có nhiệm vụ định tuyến các yêu cầu đến các dịch vụ chính xác dựa trên các đường dẫn yêu cầu. Các dịch vụ này có thể là đơn hoặc riêng lẻ

microservices. Chúng ta không cần phải tự xây dựng nhiều dịch vụ không trạng thái này vì có những dịch vụ trên thị trường có thể tích hợp dễ dàng. Một dịch vụ mà chúng ta sẽ thảo luận sâu hơn là khám phá dịch vụ. Nhiệm vụ chính của nó là cung cấp cho khách hàng danh sách tên máy chủ DNS của máy chủ trò chuyện mà khách hàng có thể kết nối.

Dịch vụ có trạng thái

Dịch vụ duy nhất có trạng thái là dịch vụ trò chuyện. Dịch vụ này có trạng thái vì mỗi máy khách duy trì kết nối mạng liên tục với máy chủ trò chuyện. Trong dịch vụ này, máy khách thường không chuyển sang máy chủ trò chuyện khác miễn là máy chủ vẫn khả dụng. Khám phá dịch vụ phối hợp chặt chẽ với dịch vụ trò chuyện để tránh quá tải máy chủ. Chúng ta sẽ đi sâu vào chi tiết.

Tích hợp bên thứ ba Đôi với ứng

dụng trò chuyện, thông báo đầy là tích hợp bên thứ ba quan trọng nhất. Đây là cách thông báo cho người dùng khi có tin nhắn mới, ngay cả khi ứng dụng không chạy. Tích hợp thông báo đầy đúng cách là rất quan trọng. Tham khảo Chương 10 Thiết kế hệ thống thông báo để biết thêm thông tin.

Khả năng mở

Ít ỏi ở quy mô nhỏ, tất cả các dịch vụ được liệt kê ở trên có thể phù hợp với một máy chủ. Ngay cả ở quy mô chúng tôi thiết kế, về mặt lý thuyết, có thể phù hợp với tất cả các kết nối của người dùng trong một máy chủ đám mây hiện đại. Số lượng kết nối đồng thời mà một máy chủ có thể xử lý rất có thể sẽ là yếu tố hạn chế. Trong kịch bản của chúng tôi, ở 1 triệu người dùng đồng thời, giả sử mỗi kết nối của người dùng cần 10K bộ nhớ trên máy chủ (đây là con số rất sơ bộ và phụ thuộc rất nhiều vào lựa chọn ngôn ngữ), thì chỉ cần khoảng 10GB bộ nhớ để chứa tất cả các kết nối trên một hộp.

Nếu chúng ta đề xuất một thiết kế mà mọi thứ đều nằm gọn trong một máy chủ, điều này có thể gây ra một dấu hiệu cảnh báo lớn trong tâm trí người phỏng vấn. Không một chuyên gia công nghệ nào sẽ thiết kế một quy mô như vậy trong một máy chủ duy nhất. Thiết kế máy chủ duy nhất là một sự phá vỡ thỏa thuận do nhiều yếu tố. Điểm lỗi duy nhất là lớn nhất trong số đó.

Tuy nhiên, hoàn toàn ổn khi bắt đầu với thiết kế máy chủ đơn. Chỉ cần đảm bảo người phỏng vấn biết đây là điểm khởi đầu. Tổng hợp mọi thứ chúng tôi đã đề cập, Hình 12-8 cho thấy thiết kế cấp cao đã điều chỉnh.

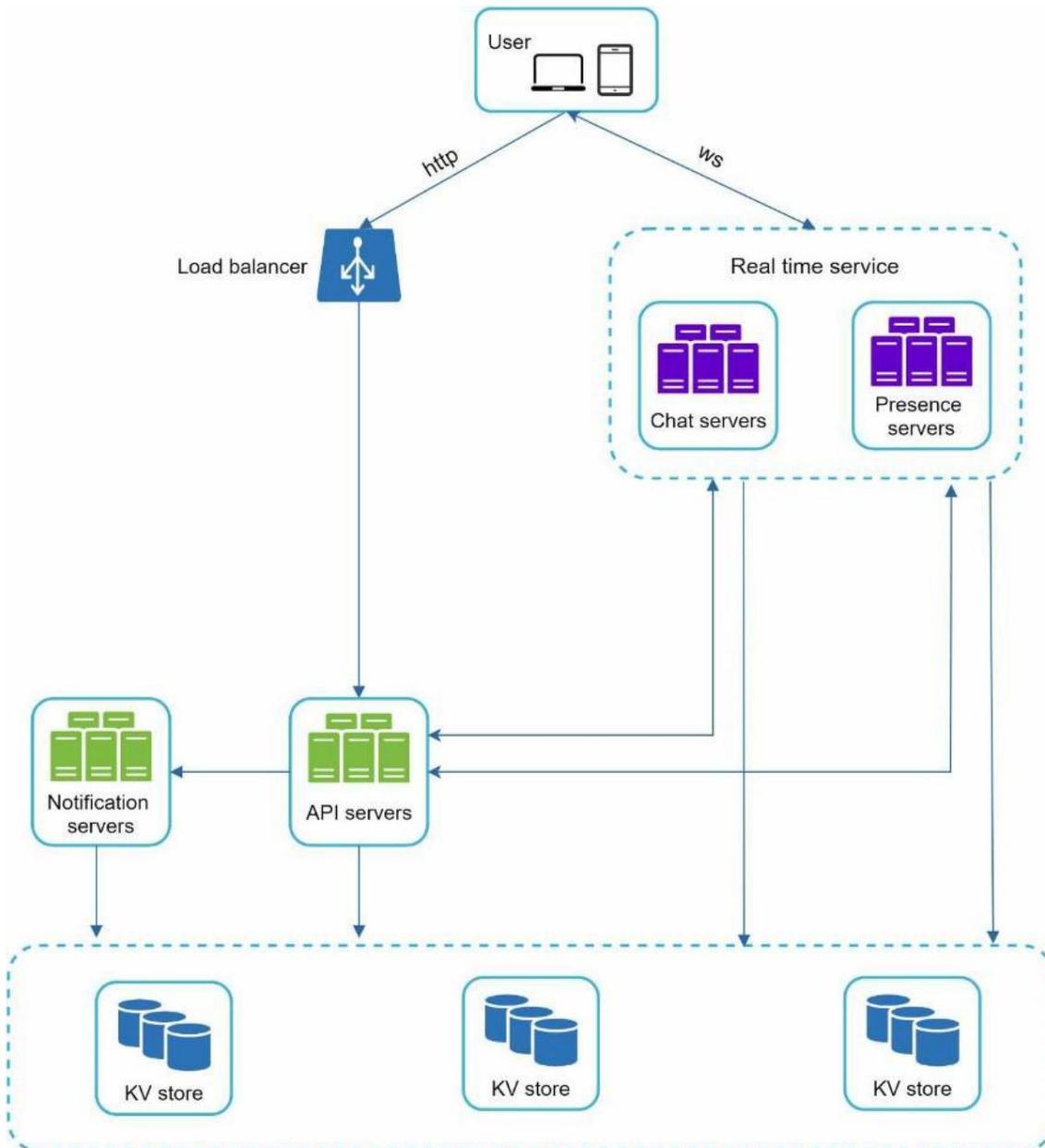


Figure 12-8

Trong Hình 12-8, máy khách duy trì kết nối WebSocket liên tục với máy chủ trò chuyện để nhắn tin theo thời gian thực. • Máy chủ trò

chuyện tạo điều kiện thuận lợi cho việc gửi/nhận tin nhắn. •

Máy chủ hiện diện quản lý trạng thái trực tuyến/ngoại

tuyến. • Máy chủ API xử lý mọi thứ bao gồm đăng nhập, đăng ký, thay đổi hồ sơ của người dùng, v.v. • Máy chủ thông báo gửi thông báo đầy.

• Cuối cùng, lưu trữ khóa-giá trị được sử dụng để lưu trữ lịch sử trò chuyện. Khi người dùng ngoại tuyến trực tuyến, cô ấy sẽ thấy tất cả lịch sử trò chuyện trước đó của mình.

Lưu trữ

Tại thời điểm này, chúng tôi đã có máy chủ sẵn sàng, các dịch vụ đang chạy và tích hợp của bên thứ ba đã hoàn tất. Sâu bên trong ngăn xếp kỹ thuật là lớp dữ liệu. Lớp dữ liệu thường đòi hỏi một số nỗ lực để làm cho nó chính xác. Một quyết định quan trọng mà chúng ta phải đưa ra là quyết định loại cơ sở dữ liệu phù hợp để sử dụng: cơ sở dữ liệu quan hệ hay cơ sở dữ liệu NoSQL? Để đưa ra quyết định sáng suốt, chúng tôi

sẽ kiểm tra các kiểu dữ liệu và các mẫu đọc/ghi.

Có hai loại dữ liệu trong một hệ thống trò chuyện thông thường. Loại đầu tiên là dữ liệu chung, chẳng hạn như hồ sơ người dùng, cài đặt, danh sách bạn bè của người dùng. Những dữ liệu này được lưu trữ trong cơ sở dữ liệu quan hệ mạnh mẽ và đáng tin cậy. Sao chép và phân mảnh là những kỹ thuật phổ biến để đáp ứng các yêu cầu về tính khả dụng và khả năng mở rộng.

Thứ hai là duy nhất đối với hệ thống trò chuyện: dữ liệu lịch sử trò chuyện. Điều quan trọng là phải hiểu mô hình đọc/ghi.

- Lượng dữ liệu rất lớn đối với các hệ thống trò chuyện. Một nghiên cứu trước đây [2] cho thấy Facebook Messenger và WhatsApp xử lý 60 tỷ tin nhắn mỗi ngày.
- Chỉ những cuộc trò chuyện gần đây mới được truy cập thường xuyên. Người dùng thường không tìm kiếm các cuộc trò chuyện cũ.
- Mặc dù lịch sử trò chuyện rất gần đây được xem trong hầu hết các truy cập hợp, người dùng có thể sử dụng các tính năng yêu cầu truy cập dữ liệu ngẫu nhiên, chẳng hạn như tìm kiếm, xem các đề cập của bạn, chuyển đến các tin nhắn cụ thể, v.v.
- Những truy cập này phải được lớp truy cập dữ liệu hỗ trợ.
- Tỷ lệ đọc để ghi là khoảng 1:1 đối với các ứng dụng trò chuyện 1 trên 1.

Việc lựa chọn hệ thống lưu trữ chính xác hỗ trợ tất cả các truy cập sử dụng của chúng tôi là rất quan trọng. Chúng tôi đề xuất các kho lưu trữ khóa-giá trị vì những lý do sau:

- Kho lưu trữ khóa-giá trị cho phép mở rộng theo chiều ngang dễ dàng.
 - Kho lưu trữ khóa-giá trị cung cấp độ trễ rất thấp để truy cập dữ liệu.
 - Cơ sở dữ liệu quan hệ không xử lý tốt phần đuôi dài [3] của dữ liệu. Khi các chỉ mục phát triển lớn, việc truy cập ngẫu nhiên sẽ tốn kém.
- Kho lưu trữ khóa-giá trị được các ứng dụng trò chuyện đáng tin cậy đã được chứng minh khác áp dụng. Ví dụ: cả Facebook Messenger và Discord đều sử dụng kho lưu trữ khóa-giá trị. Facebook Messenger sử dụng HBase [4] và Discord sử dụng Cassandra [5].

Mô hình dữ liệu

Vừa rồi, chúng ta đã nói về việc sử dụng kho lưu trữ khóa-giá trị làm lớp lưu trữ của chúng ta. Dữ liệu quan trọng nhất là dữ liệu tin nhắn. Chúng ta hãy cùng xem xét kỹ hơn.

Bảng tin nhắn cho trò chuyện 1-1 Hình

12-9 cho thấy bảng tin nhắn cho trò chuyện 1-1. Khóa chính là message_id, giúp quyết định trình tự tin nhắn. Chúng ta không thể dựa vào created_at để quyết định trình tự tin nhắn vì có thể tạo hai tin nhắn cùng một lúc.

message	
message_id	bigint
message_from	bigint
message_to	bitint
content	text
created_at	timestamp

Figure 12-9

Bảng tin nhắn cho trò chuyện

nhóm Hình 12-10 hiển thị bảng tin nhắn cho trò chuyện nhóm. Khóa chính tổng hợp là (channel_id, message_id). Kênh và nhóm biểu thị cùng một ý nghĩa ở đây. channel_id là khóa phân vùng vì tất cả các truy vấn trong trò chuyện nhóm đều hoạt động trong một kênh.

group_message	
channel_id	bigint
message_id	bigint
user_id	bigint
content	text
created_at	timestamp

Figure 12-10

ID tin nhắn

Cách tạo message_id là một chủ đề thú vị đáng để khám phá. Message_id chịu trách nhiệm đảm bảo thứ tự của các tin nhắn. Để xác định thứ tự của các tin nhắn, message_id phải đáp ứng hai yêu cầu sau:

- ID phải là duy nhất.
- ID phải có thể sắp xếp theo thời gian, nghĩa là các hàng mới có ID cao hơn các hàng cũ.

Làm sao chúng ta có thể đạt được hai điều đó? Ý tưởng đầu tiên xuất hiện trong đầu là từ khóa "auto_increment" trong MySql. Tuy nhiên, cơ sở dữ liệu NoSQL thường không cung cấp tính năng như vậy.

Cách tiếp cận thứ hai là sử dụng trình tạo số thứ tự toàn cầu 64 bit như Snowflake [6]. Vấn đề này được thảo luận trong "Chương 7: Thiết kế trình tạo ID duy nhất trong hệ thống phân tán". Cách tiếp cận cuối cùng là sử dụng trình tạo số thứ tự cục bộ. ID cục bộ có nghĩa là chỉ duy nhất trong một nhóm. Lý do tại sao ID cục bộ hoạt động là duy trì trình tự tin nhắn trong kênh một-một hoặc kênh nhóm là đủ. Cách tiếp cận này dễ triển khai hơn so với triển khai ID toàn cầu.

Bài 3 - Thiết kế chuyên sâu

Trong một cuộc phỏng vấn thiết kế hệ thống, thông thư ờng bạn được yêu cầu đào sâu vào một số thành phần trong thiết kế cấp cao. Đối với hệ thống trò chuyện, khám phá dịch vụ, luồng tin nhắn và các chỉ số trực tuyến/ngoại tuyến đáng để khám phá sâu hơn.

Khám phá dịch vụ Vai

trò chính của khám phá dịch vụ là đề xuất máy chủ trò chuyện tốt nhất cho máy khách dựa trên các tiêu chí như vị trí địa lý, dung lượng máy chủ, v.v. Apache Zookeeper [7] là giải pháp nguồn mở phổ biến để khám phá dịch vụ. Nó đăng ký tất cả các máy chủ trò chuyện khả dụng và chọn máy chủ trò chuyện tốt nhất cho máy khách dựa trên các tiêu chí được xác định trước.

Hình 12-11 cho thấy cách thức hoạt động của dịch vụ khám phá (Zookeeper).

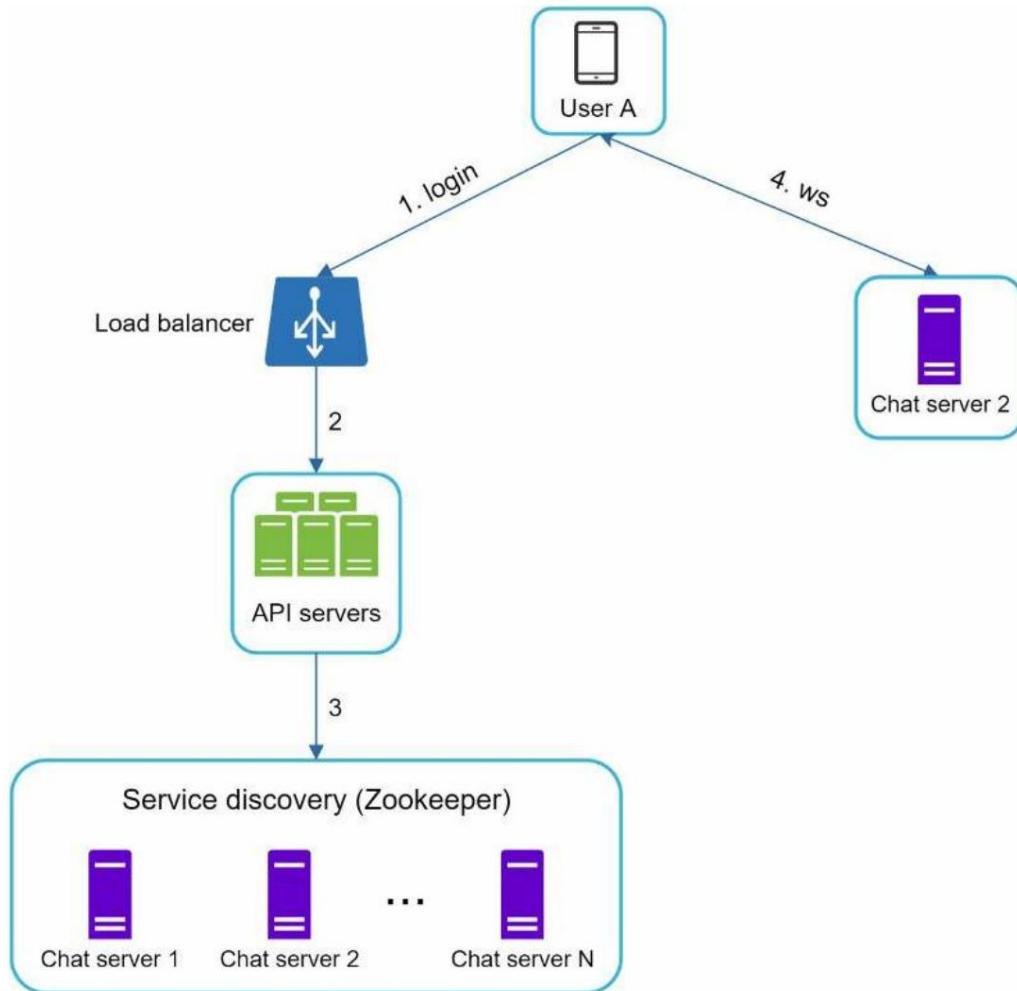


Figure 12-11

1. Người dùng A cố gắng đăng nhập vào ứng dụng.
2. Bộ cân bằng tải gửi yêu cầu đăng nhập đến máy chủ API.
3. Sau khi phần phụ trợ xác thực người dùng, dịch vụ khám phá sẽ tìm máy chủ trò chuyện tốt nhất cho Người dùng A. Trong ví dụ này, máy chủ 2 được chọn và thông tin máy chủ được trả về cho Người dùng A.
4. Người dùng A kết nối với máy chủ trò chuyện 2 thông qua WebSocket.

Luồng tin nhắn

Thật thú vị khi hiểu luồng đầu cuối của một hệ thống trò chuyện. Trong phần này, chúng ta sẽ

khám phá luồng trò chuyện 1-1, đồng bộ hóa tin nhắn trên nhiều thiết bị và luồng trò chuyện nhóm.

Luồng trò chuyện 1-1

Hình 12-12 giải thích điều gì xảy ra khi Người dùng A gửi tin nhắn cho Người dùng B.

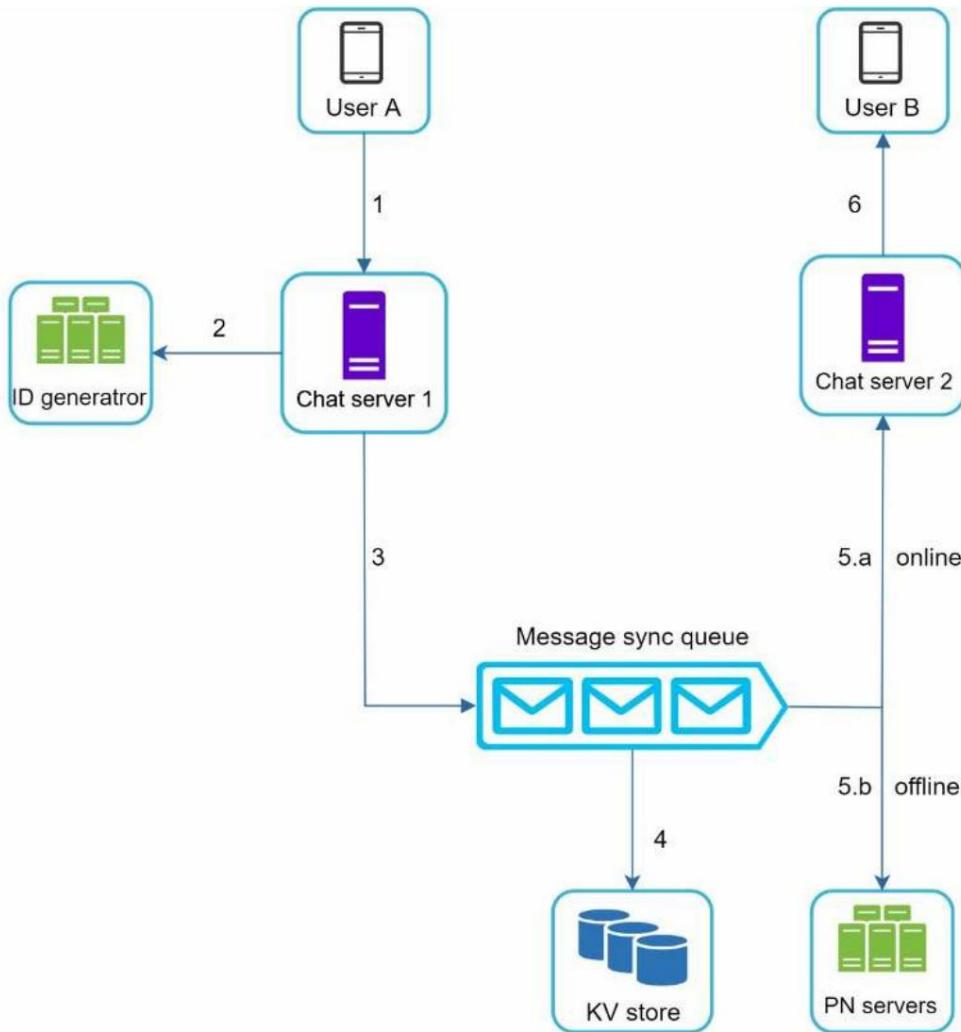


Figure 12-12

- Người dùng A gửi tin nhắn trò chuyện đến máy chủ trò chuyện 1.
- Máy chủ trò chuyện 1 lấy ID tin nhắn từ trình tạo ID.
- Máy chủ trò chuyện 1 gửi tin nhắn đến hàng đợi đồng bộ tin nhắn.
- Tin nhắn được lưu trữ trong kho lưu trữ khóa-giá trị.
- a. Nếu Người dùng B trực tuyến, tin nhắn sẽ được chuyển tiếp đến Máy chủ trò chuyện 2 nơi Người dùng B được kết nối.
- b. Nếu Người dùng B ngoại tuyến, thông báo đầy sẽ được gửi từ máy chủ thông báo đầy (PN).
- Máy chủ trò chuyện 2 chuyển tiếp tin nhắn đến Người dùng B. Có một kết nối WebSocket liên tục giữa Người dùng B và máy chủ trò chuyện 2.

Đồng bộ hóa tin nhắn trên nhiều thiết bị Nhiều người dùng có nhiều

thiết bị. Chúng tôi sẽ giải thích cách đồng bộ hóa tin nhắn trên nhiều thiết bị. Hình 12-13 cho thấy một ví dụ về đồng bộ hóa tin nhắn.

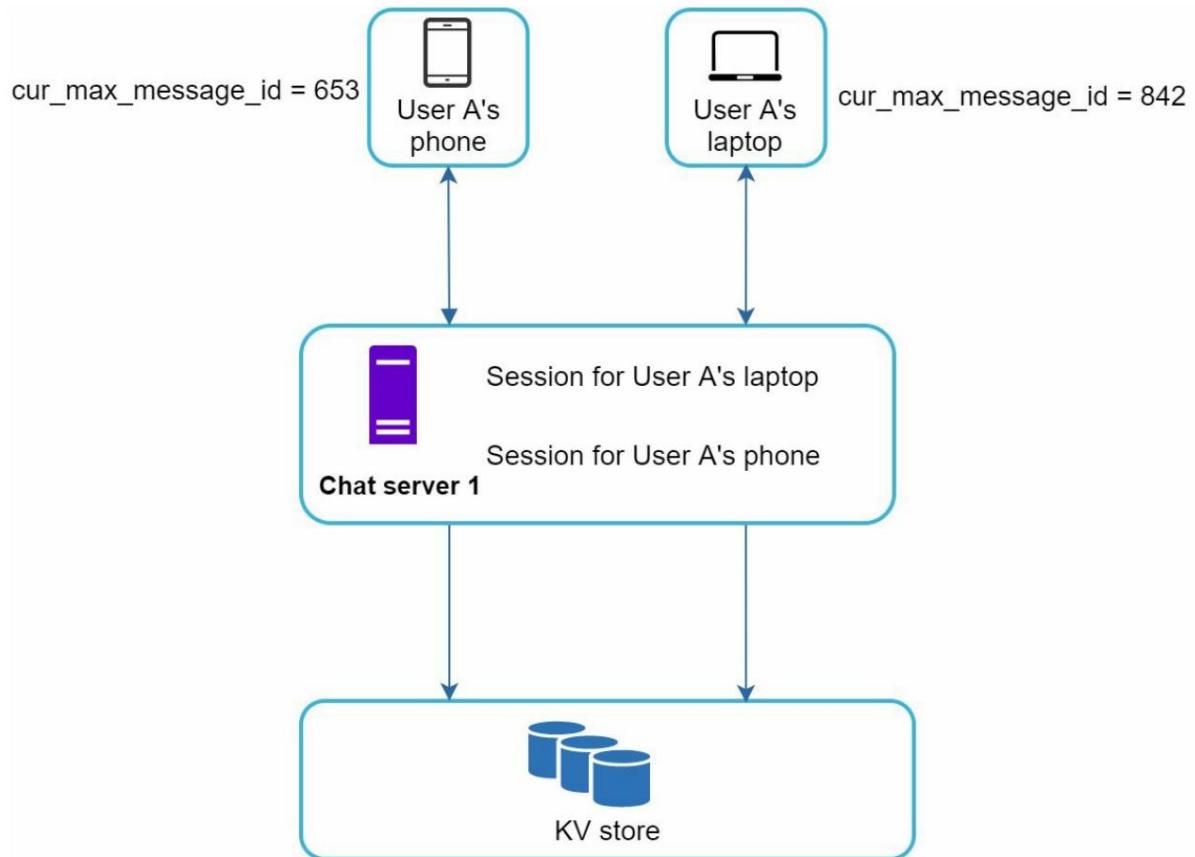


Figure 12-13

Trong Hình 12-13, người dùng A có hai thiết bị: điện thoại và máy tính xách tay. Khi Người dùng A đăng nhập vào ứng dụng trò chuyện bằng điện thoại của mình, nó sẽ thiết lập kết nối WebSocket với máy chủ Trò chuyện 1. Tương tự, có một kết nối giữa máy tính xách tay và máy chủ Trò chuyện 1.

Mỗi thiết bị duy trì một biến gọi là `cur_max_message_id`, biến này theo dõi ID tin nhắn mới nhất trên thiết bị. Các tin nhắn đáp ứng hai điều kiện sau được coi là tin nhắn tin tức:

- ID người nhận bằng ID người dùng hiện đang đăng nhập.
- ID tin nhắn trong kho lưu trữ khóa-giá trị lớn hơn `cur_max_message_id`.

Với `cur_max_message_id` riêng biệt trên mỗi thiết bị, việc đồng bộ hóa tin nhắn trở nên dễ dàng vì mỗi thiết bị có thể nhận được tin nhắn mới từ kho lưu trữ KV.

Luồng trò chuyện nhóm nhỏ

So với trò chuyện một-một, logic của trò chuyện nhóm phức tạp hơn. Hình 12-14 và 12-15 giải thích luồng trò chuyện.

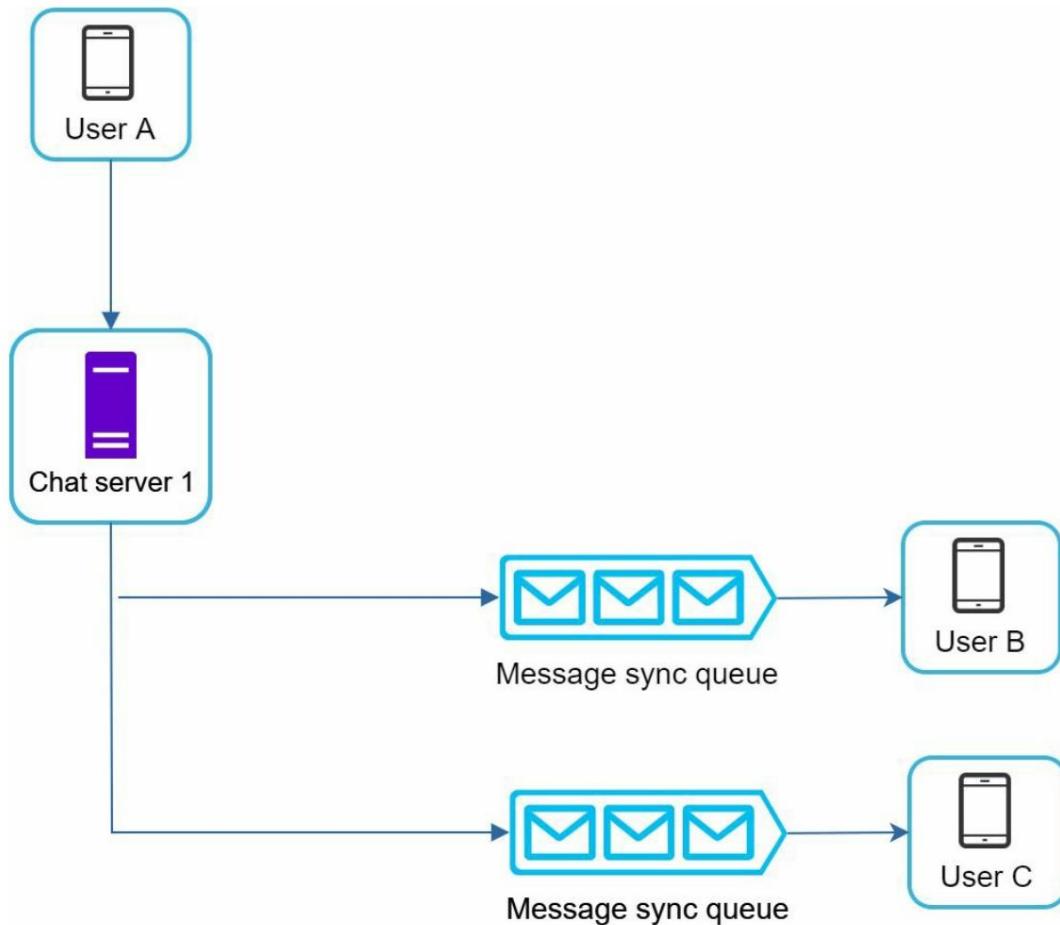


Figure 12-14

Hình 12-14 giải thích những gì xảy ra khi Người dùng A gửi tin nhắn trong trò chuyện nhóm. Giả sử có 3 thành viên trong nhóm (Người dùng A, Người dùng B và người dùng C). Đầu tiên, tin nhắn từ Người dùng A được sao chép vào hàng đợi đồng bộ tin nhắn của từng thành viên trong nhóm: một cho Người dùng B và một cho Người dùng C. Bạn có thể coi hàng đợi đồng bộ tin nhắn như hộp thư đến của người nhận. Lựa chọn thiết kế này phù hợp với trò chuyện nhóm nhỏ vì:

- đơn giản hóa luồng đồng bộ tin nhắn vì mỗi máy khách chỉ cần kiểm tra hộp thư đến của mình để nhận tin nhắn mới.
- khi số lượng nhóm nhỏ, việc lưu trữ một bản sao trong hộp thư đến của mỗi người nhận không quá tốn kém.

WeChat sử dụng một cách tiếp cận tự động và giới hạn một nhóm ở mức 500 thành viên [8]. Tuy nhiên, đối với các nhóm có nhiều người dùng, việc lưu trữ một bản sao tin nhắn cho mỗi thành viên là không thể chấp nhận được.

Về phía người nhận, người nhận có thể nhận được tin nhắn từ nhiều người dùng. Mỗi người nhận có một hộp thư đến (hàng đợi đồng bộ tin nhắn) chứa tin nhắn từ những người gửi khác nhau. Hình 12-15 minh họa thiết kế.

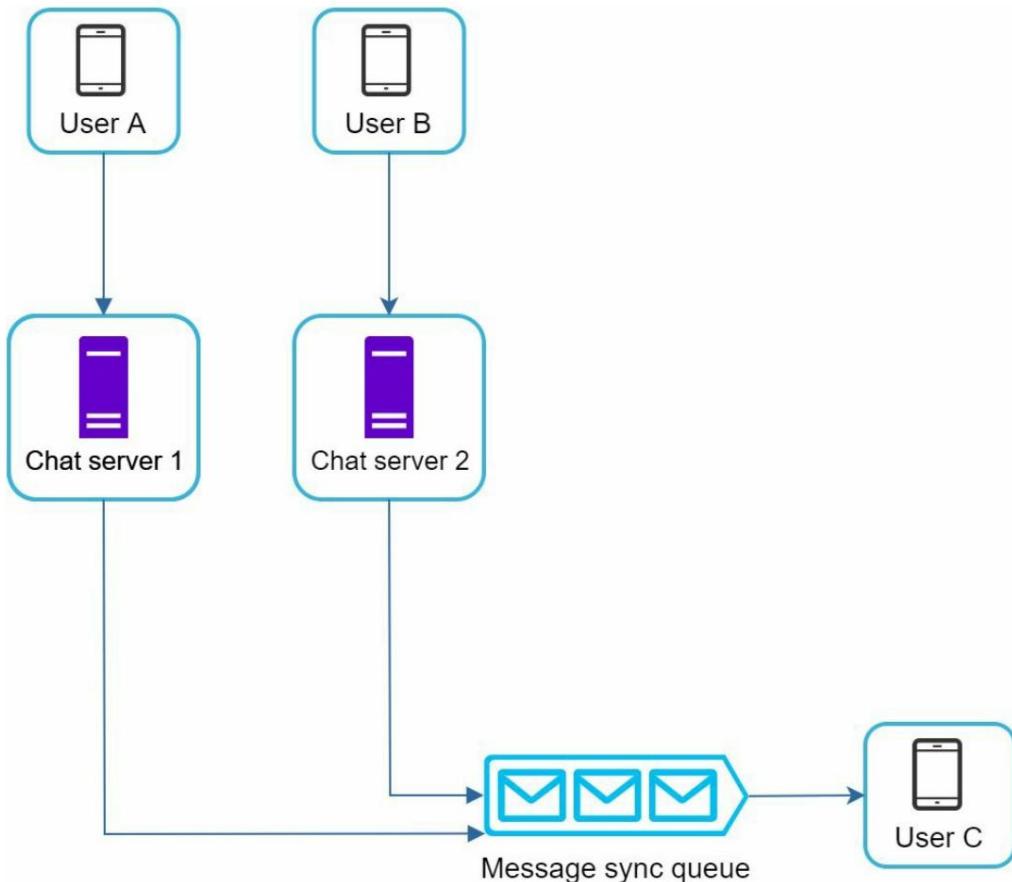


Figure 12-15

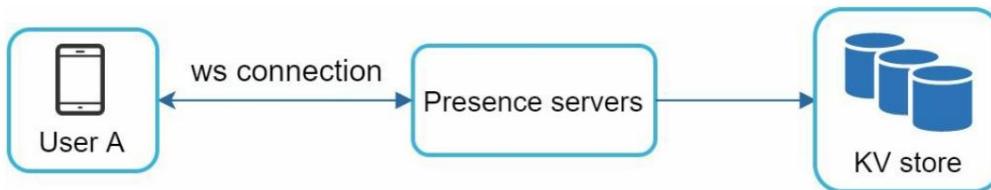
Hiện diện trực tuyến

Chỉ báo hiện diện trực tuyến là một tính năng thiết yếu của nhiều ứng dụng trò chuyện. Thông thường, bạn có thể thấy một chấm màu xanh lá cây bên cạnh ảnh hồ sơ hoặc tên người dùng của người dùng. Phần này giải thích những gì xảy ra ngay sau hậu trường.

Trong thiết kế cấp cao, máy chủ hiện diện chịu trách nhiệm quản lý trạng thái trực tuyến và giao tiếp với khách hàng thông qua WebSocket. Có một số luồng sẽ kích hoạt thay đổi trạng thái trực tuyến. Chúng ta hãy xem xét từng luồng.

Đăng nhập

người dùng Luồng đăng nhập người dùng được giải thích trong phần “Khám phá dịch vụ”. Sau khi kết nối WebSocket được xây dựng giữa máy khách và dịch vụ thời gian thực, trạng thái trực tuyến và dấu thời gian last_active_at của người dùng A được lưu trữ trong kho lưu trữ KV. Chỉ báo trạng thái hiện diện cho biết người dùng đang trực tuyến sau khi cô ấy đăng nhập.



User A: {status: online, last_active_at: timestamp}

Figure 12-16

Đăng xuất ngư ời dùng

dùng Khi ngư ời dùng đăng xuất, ngư ời dùng sẽ trải qua luồng đăng xuất ngư ời dùng như thể hiện trong Hình 12-17.

Trạng thái trực tuyến đư ợc thay đổi thành ngoại tuyến trong kho KV. Chỉ báo hiện diện cho biết ngư ời dùng đang ngoại tuyến.

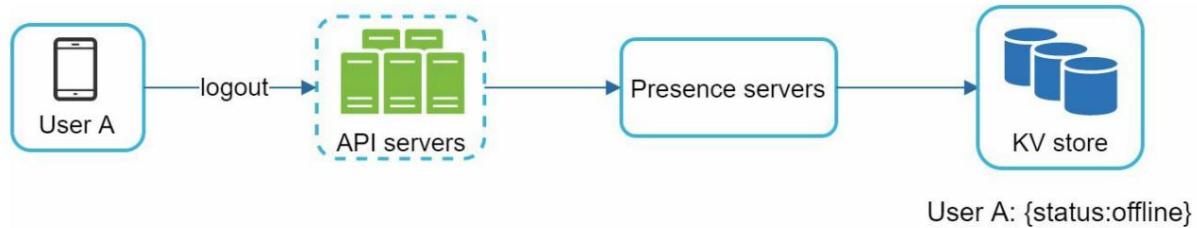


Figure 12-17

Ngắt kết nối ngư ời dùng

Tất cả chúng ta đều mong muốn kết nối internet của mình ổn định và đáng tin cậy. Tuy nhiên, không phải lúc nào cũng như vậy; do đó, chúng ta phải giải quyết vấn đề này trong thiết kế của mình. Khi ngư ời dùng ngắt kết nối internet, kết nối liên tục giữa máy khách và máy chủ sẽ bị mất. Một cách ngây thơ để xử lý tình trạng ngắt kết nối của ngư ời dùng là đánh dấu ngư ời dùng là ngoại tuyến và thay đổi trạng thái thành trực tuyến khi kết nối được thiết lập lại. Tuy nhiên, cách tiếp cận này có một sai sót lớn. Ngư ời dùng thường ngắt kết nối và kết nối lại internet thường xuyên trong thời gian ngắn. Ví dụ, kết nối mạng có thể bật và tắt khi ngư ời dùng đi qua đường hầm. Việc cập nhật trạng thái trực tuyến sau mỗi lần ngắt kết nối/kết nối lại sẽ khiến chỉ báo trạng thái thay đổi quá thường xuyên, dẫn đến trải nghiệm ngư ời dùng kém.

Chúng tôi giới thiệu cơ chế nhịp tim để giải quyết vấn đề này. Theo định kỳ, một máy khách trực tuyến sẽ gửi sự kiện nhịp tim đến các máy chủ hiện diện. Nếu máy chủ hiện diện nhận được sự kiện nhịp tim trong một khoảng thời gian nhất định, chẳng hạn như x giây từ máy khách, thì ngư ời dùng đư ợc coi là trực tuyến. Nếu không, thì ngư ời dùng đó sẽ ngoại tuyến.

Trong Hình 12-18, máy khách gửi một sự kiện nhịp tim đến máy chủ sau mỗi 5 giây. Sau khi gửi 3 sự kiện nhịp tim, máy khách bị ngắt kết nối và không kết nối lại trong vòng $x = 30$ giây (Con số này đư ợc chọn tùy ý để chứng minh logic). Trạng thái trực tuyến đư ợc thay đổi thành ngoại tuyến.

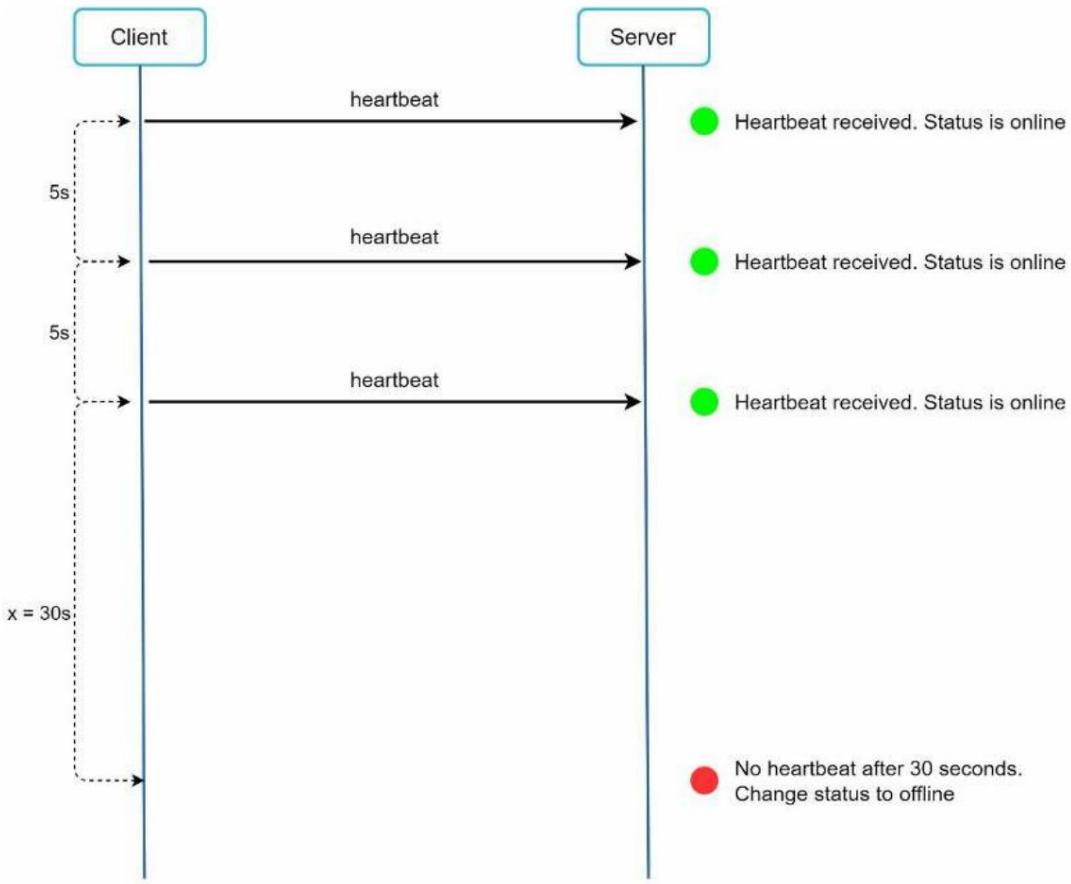


Figure 12-18

Trạng thái trực tuyến lan truyền

Bạn bè của người dùng A biết về những thay đổi trạng thái như thế nào? Hình 12-19 giải thích cách thức hoạt động.

Máy chủ hiện diện sử dụng mô hình đăng ký-xuất bản, trong đó mỗi cặp bạn bè duy trì một kênh. Khi trạng thái trực tuyến của Người dùng A thay đổi, nó sẽ xuất bản sự kiện tới ba kênh, kênh AB, AC và AD. Ba kênh đó được Người dùng B, C và D đăng ký tương ứng. Do đó, bạn bè có thể dễ dàng nhận được cập nhật trạng thái trực tuyến. Giao tiếp giữa máy khách và máy chủ thông qua WebSocket thời gian thực.

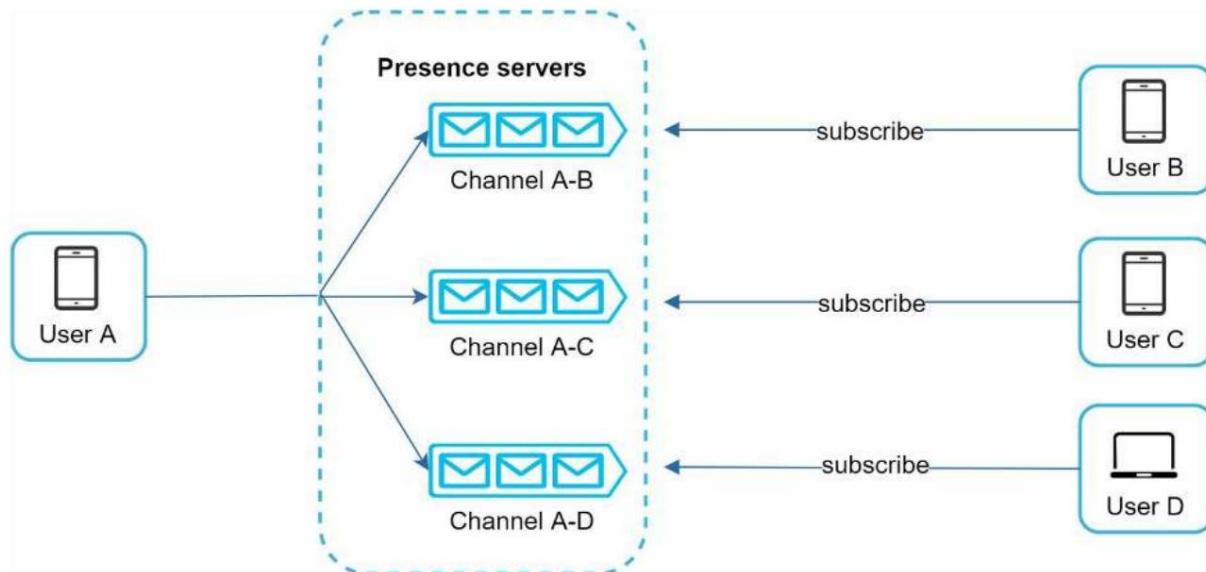


Figure 12-19

Thiết kế trên có hiệu quả đối với một nhóm người dùng nhỏ. Ví dụ, WeChat sử dụng cách tiếp cận tư duy tự vì nhóm người dùng của họ bị giới hạn ở mức 500. Đối với các nhóm lớn hơn, việc thông báo cho tất cả các thành viên về trạng thái trực tuyến là tốn kém và mất thời gian. Giả sử một nhóm có 100.000 thành viên.

Mỗi lần thay đổi trạng thái sẽ tạo ra 100.000 sự kiện. Để giải quyết tình trạng tắc nghẽn hiệu suất, một giải pháp khả thi là chỉ lấy trạng thái trực tuyến khi người dùng vào nhóm hoặc làm mới danh sách bạn bè theo cách thủ công.

Bứ ớc 4 - Tǒng kêt Trong

chư ơng này, chúng tôi đã trình bày một kiến trúc hệ thống trò chuyện hỗ trợ cả trò chuyện 1-1 và trò chuyện nhóm nhỏ. WebSocket được sử dụng để giao tiếp thời gian thực giữa máy khách và máy chủ. Hệ thống trò chuyện bao gồm các thành phần sau: máy chủ trò chuyện để nhắn tin thời gian thực, máy chủ hiện diện để quản lý sự hiện diện trực tuyến, máy chủ thông báo đẩy để gửi thông báo đẩy, kho lưu trữ khóa-giá trị để duy trì lịch sử trò chuyện và máy chủ API cho các chức năng khác.

Nếu bạn còn thời gian vào cuối buổi phòng vấn, sau đây là những điểm cần nói thêm:

- Mở rộng ứng dụng trò chuyện để hỗ trợ các tệp phư ơng tiện như ảnh và video. Tệp phư ơng tiện có kích thư ớc lớn hơn đáng kể so với văn bản. Nên, lưu trữ đám mây và hình thu nhỏ là những chủ đề thú vị để thảo luận.
- Mã hóa đầu cuối. Whatsapp hỗ trợ mã hóa đầu cuối cho tin nhắn. Chỉ người gửi và người nhận mới có thể đọc tin nhắn. Độ giả quan tâm nên tham khảo bài viết trong tài liệu tham khảo [9].
- Lưu trữ đệm tin nhắn ở phía máy khách có hiệu quả trong việc giảm việc truyền dữ liệu giữa máy khách và máy chủ.
- Cải thiện thời gian tải. Slack đã xây dựng một mạng lưới phân tán theo địa lý để lưu trữ dữ liệu, kênh, v.v. của người dùng để có thời gian tải tốt hơn [10].
- Xử lý lỗi. Lỗi máy chủ trò chuyện. Có thể có hàng trăm nghìn hoặc thậm chí nhiều kết nối liên tục hơn đến máy chủ trò chuyện. Nếu máy chủ trò chuyện ngoại tuyến, khám phá dịch vụ (Zookeeper) sẽ cung cấp máy chủ trò chuyện mới để khách hàng thiết lập kết nối mới.
- Cơ chế gửi lại tin nhắn. Thử lại và xếp hàng là các kỹ thuật phổ biến để gửi lại tin nhắn.

Xin chúc mừng vì đã đi đến đây! Bây giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

- [1] Erlang trên Facebook : <https://www.erlang-factory.com/upload/presentations/31/EugeneLetuchy-ErlangatFacebook.pdf>
- [2] Messenger và WhatsApp xử lý 60 tỷ tin nhắn mỗi ngày:
<https://www.theverge.com/2016/4/12/11415198/facebook-messenger-whatsapp-number-messages-vs-sms-f8-2016>
- [3] Độ dài: https://en.wikipedia.org/wiki/Long_tail
- [4] Công nghệ cơ bản của tin nhắn: <https://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919/>
- [5] Discord lưu trữ hàng tỷ tin nhắn như thế nào: <https://blog.discordapp.com/how-discord-stores-billions-of-messages-7fa6ec7ee4c7>
- [6] Công bố Snowflake: https://blog.twitter.com/engineering/en_us/a/2010/announcing-snowflake.html
- [7] Apache ZooKeeper: <https://zookeeper.apache.org/>
- [8] Từ con số không: sự tiến hóa của hệ thống nền tảng WeChat (Bài viết tiếng Trung):
<https://www.infoq.cn/article/the-road-of-the-growth-weixin-background>
- [9] Mã hóa đầu cuối: <https://faq.whatsapp.com/en/android/28030015/>
- [10] Flannel: Bộ nhớ đệm Edge cấp ứng dụng để mở rộng quy mô Slack:
<https://slack.engineering/flannel-an-application-level-edge-cache-to-make-slack-scale-b8a6400e2f6b>

CHƯƠNG 13: THIẾT KẾ TỰ ĐỘNG HOÀN THÀNH TÌM KIẾM

HỆ THỐNG

Khi tìm kiếm trên Google hoặc mua sắm tại Amazon, khi bạn nhập vào hộp tìm kiếm, một hoặc nhiều kết quả khớp với từ khóa tìm kiếm sẽ được hiển thị cho bạn. Tính năng này được gọi là tự động hoàn thành, gợi ý, tìm kiếm khi bạn nhập hoặc tìm kiếm gia tăng. Hình 13-1 trình bày ví dụ về tìm kiếm trên Google hiển thị danh sách các kết quả tự động hoàn thành khi bạn nhập "dinner" vào hộp tìm kiếm. Tự động hoàn thành tìm kiếm là một tính năng quan trọng của nhiều sản phẩm. Điều này dẫn chúng ta đến câu hỏi phỏng vấn: thiết kế hệ thống tự động hoàn thành tìm kiếm, còn được gọi là "thiết kế top k" hoặc "thiết kế top k truy vấn được tìm kiếm nhiều nhất".



Figure 13-1

Bưu ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế Bưu ớc đầu tiên để giải quyết bất kỳ câu hỏi phỏng vấn thiết kế hệ thống nào là đặt đủ câu hỏi để làm rõ các yêu cầu. Sau đây là một ví dụ về tư ơng tác giữa ứng viên và người phỏng vấn: Ứng viên: Việc khớp lệnh chỉ được hỗ trợ ở đầu truy vấn tìm kiếm hay ở giữa?

Người phỏng vấn: Chỉ ở đầu truy vấn tìm kiếm.

Ứng viên: Hệ thống sẽ trả về bao nhiêu gợi ý tự động hoàn thành?

Người phỏng vấn: 5

Ứng viên: Hệ thống biết được 5 gợi ý nào sẽ được trả về bằng cách nào?

Người phỏng vấn: Điều này được xác định bởi mức độ phổ biến, quyết định bởi tần suất truy vấn trong lịch sử.

Ứng viên: Hệ thống có hỗ trợ kiểm tra chính tả không?

Người phỏng vấn: Không, tính năng kiểm tra chính tả hoặc tự động sửa lỗi không được hỗ trợ.

Ứng viên: Các truy vấn tìm kiếm có bằng tiếng Anh không?

Người phỏng vấn: Vâng. Nếu thời gian cho phép, cuối cùng chúng ta có thể thảo luận về hỗ trợ đa ngôn ngữ.

Ứng viên: Chúng tôi có cho phép viết hoa và sử dụng ký tự đặc biệt không?

Người phỏng vấn: Không, chúng tôi cho rằng tất cả các truy vấn tìm kiếm đều có chữ cái viết thường.

Ứng viên: Có bao nhiêu người dùng sử dụng sản phẩm?

Người phỏng vấn: 10 triệu DAU.

Yêu cầu

Sau đây là tóm tắt các yêu cầu:

- Thời gian phản hồi nhanh: Khi người dùng nhập truy vấn tìm kiếm, các gợi ý tự động hoàn thành phải hiển thị đủ nhanh. Một bài viết về hệ thống tự động hoàn thành của Facebook [1] tiết lộ rằng hệ thống cần trả về kết quả trong vòng 100 mili giây. Nếu không, nó sẽ gây ra hiện tượng giật. • Có liên quan: Các gợi ý tự động hoàn thành phải có liên quan đến thuật ngữ tìm kiếm. • Đã sắp xếp: Các kết quả do hệ thống trả về phải được sắp xếp theo mức độ phổ biến hoặc các mô hình xếp hạng khác.
- Có khả năng mở rộng: Hệ thống có thể xử lý lưu lượng truy cập cao. • Tính khả dụng cao: Hệ thống phải luôn khả dụng và có thể truy cập được khi một phần của hệ thống ngoại tuyến, chậm lại hoặc gặp lỗi mạng không mong muốn.

Ước tính sơ bộ • Giả sử có 10 triệu người

dùng hoạt động hàng ngày (DAU). • Trung bình một người thực hiện 10 lượt tìm kiếm mỗi ngày. • 20 byte dữ liệu cho mỗi chuỗi truy vấn:

- Giả sử chúng ta sử dụng mã hóa ký tự ASCII. 1 ký tự = 1 byte • Giả sử một truy vấn chứa 4 từ và mỗi từ chứa trung bình 5 ký tự. • Tức là $4 \times 5 = 20$ byte cho mỗi truy vấn. • Đối với mỗi ký tự được nhập vào hộp tìm kiếm, một máy khách sẽ gửi một yêu cầu đến phần phụ trợ để tự động hoàn thành các gợi ý. Trung bình, có 20 yêu cầu được gửi cho mỗi truy vấn tìm kiếm. Ví dụ: 6 yêu cầu sau được gửi đến phần phụ trợ khi bạn nhập xong "dinner".

tìm kiếm?q=d

tìm kiếm?q=di
tìm kiếm?q=din
tìm kiếm?q=dinn
tìm kiếm?q=dinne
tìm kiếm?q=dinner •

~24.000 truy vấn mỗi giây (QPS) = 10.000.000 người dùng * 10 truy vấn / ngày * 20 ký tự / 24 giờ / 3600 giây.

• QPS đỉnh = QPS * 2 = ~48.000 • Giả sử

20% các truy vấn hàng ngày là mới. 10 triệu * 10 truy vấn / ngày * 20 byte mỗi * 20% = 0,4 GB. Điều này có nghĩa là 0,4 GB dữ liệu mới được thêm vào bộ nhớ hàng ngày. truy vấn

Bài 2 - Đề xuất thiết kế cấp cao và nhận đư ợc sự đồng thuận

Ở cấp độ cao, hệ thống đư ợc chia thành hai phần:

- Dịch vụ thu thập dữ liệu: Thu thập các truy vấn đầu vào của người dùng và tổng hợp chúng theo thời gian thực.

Xử lý thời gian thực không thực tế đối với các tập dữ liệu lớn; tuy nhiên, đây là điểm khởi đầu tốt. Chúng

- ta sẽ khám phá một giải pháp thực tế hơn trong phần phân tích sâu.
- Dịch vụ

truy vấn: Cho một truy vấn tìm kiếm hoặc tiền tố, trả về 5 thuật ngữ đư ợc tìm kiếm thường xuyên nhất.

Dịch vụ thu thập dữ liệu

Chúng ta hãy sử dụng một ví dụ đơn giản để xem dịch vụ thu thập dữ liệu hoạt động như thế nào.

Giả sử chúng ta có một bảng tần suất lưu trữ chuỗi truy vấn và tần suất của nó như thể hiện

trong Hình 13-2. Ban đầu, bảng tần suất là trống. Sau đó, người dùng nhập các truy vấn

"twitch", "twitter", "twitter" và "twillo" theo trình tự. Hình 13-2 cho thấy cách bảng tần suất đư ợc cập nhật.

The figure consists of five separate tables, each labeled with a search query: "query: twitch", "query: twitter", "query: twitter", "query: twillo", and "query: twillo". Each table has two columns: "Query" and "Frequency".

- query: twitch**: An empty table.
- query: twitter**: Contains one row: "twitch" with Frequency 1.
- query: twitter**: Contains two rows: "twitch" with Frequency 1 and "twitter" with Frequency 1.
- query: twillo**: Contains three rows: "twitch" with Frequency 1, "twitter" with Frequency 2, and "twillo" with Frequency 1.
- query: twillo**: An empty table.

Figure 13-2

Dịch vụ truy vấn

Giả sử chúng ta có một bảng tần suất như đư ợc hiển thị trong Bảng 13-1. Bảng

- Truy vấn: lưu

trữ chuỗi truy vấn. • Tần suất: biểu thị số lần truy vấn đư ợc tìm kiếm.

Query	Frequency
Twitter	35
twitch	29
twilight	25
twin peak	21
twitch prime	18
twitter search	14
twillo	10
twin peak sf	8

Table 13-1

Khi người dùng nhập "tw" vào hộp tìm kiếm, 5 truy vấn được tìm kiếm nhiều nhất sau đây sẽ được hiển thị (Hình 13-3), giả sử bảng tần số dựa trên Bảng 13-1.

tw
twitter
twitch
twilight
twin peak
twitch prime

Figure 13-3

Để có được 5 truy vấn được tìm kiếm thường xuyên nhất, hãy thực hiện truy vấn SQL sau:

```
SELECT * FROM frequency_table
WHERE query Like `prefix%`
ORDER BY frequency DESC
LIMIT 5
```

Figure 13-4

Đây là giải pháp chấp nhận được khi tập dữ liệu nhỏ. Khi tập dữ liệu lớn, việc truy cập cơ sở dữ liệu trở thành nút thắt cổ chai. Chúng ta sẽ khám phá tối ưu hóa sâu hơn.

BƯỚC 3 - Thiết kế chuyên sâu

Trong thiết kế cấp cao, chúng ta đã thảo luận về dịch vụ thu thập dữ liệu và dịch vụ truy vấn.

Thiết kế cấp cao không phải là tối ưu, nhưng nó đóng vai trò là điểm khởi đầu tốt. Trong phần này, chúng ta sẽ đi sâu vào một số thành phần và khám phá các tối ưu hóa như sau:

- Cấu trúc dữ liệu Trie
- Dịch vụ thu thập dữ liệu •

Dịch vụ truy vấn •

Mở rộng lưu trữ • Thủ

nghiệm các hoạt động

Cấu trúc dữ liệu Trie

Cơ sở dữ liệu quan hệ được sử dụng để lưu trữ trong thiết kế cấp cao. Tuy nhiên, việc lấy 5 truy vấn tìm kiếm hàng đầu từ cơ sở dữ liệu quan hệ là không hiệu quả. Cấu trúc dữ liệu trie (cây tiền tố) được sử dụng để khắc phục vấn đề này. Vì cấu trúc dữ liệu trie rất quan trọng đối với hệ thống, chúng tôi sẽ dành nhiều thời gian để thiết kế một trie tùy chỉnh. Xin lưu ý rằng một số ý tư ởng đư ợc lấy từ các bài viết [2] và [3].

Hiểu cấu trúc dữ liệu trie cơ bản là điều cần thiết cho câu hỏi phỏng vấn này. Tuy nhiên, đây là câu hỏi về cấu trúc dữ liệu nhiều hơn là câu hỏi về thiết kế hệ thống. Bên cạnh đó, nhiều tài liệu trực tuyến giải thích khái niệm này. Trong chương này, chúng ta sẽ chỉ thảo luận về tổng quan về cấu trúc dữ liệu trie và tập trung vào cách tối ưu hóa trie cơ bản để cải thiện thời gian phản hồi.

Trie (phát âm là "try") là một cấu trúc dữ liệu dạng cây có thể lưu trữ chuỗi một cách gọn gàng. Tên này bắt nguồn từ từ retrieval, cho biết nó đư ợc thiết kế cho các hoạt động truy xuất chuỗi. Ý tư ởng chính của trie bao gồm những điều sau:

- Trie là một cấu trúc dữ liệu dạng cây.

- Gốc biểu diễn một chuỗi rỗng. • Mỗi nút lưu trữ một ký tự và có 26 nút con, một nút cho mỗi ký tự có thể có. Để tiết kiệm không gian, chúng tôi không vẽ các liên kết rỗng. • Mỗi nút cây biểu diễn một từ đơn hoặc một chuỗi tiền tố.

Hình 13-5 hiển thị một bộ ba với các truy vấn tìm kiếm "tree", "try", "true", "toy", "wish", "win".

Các truy vấn tìm kiếm đư ợc đánh dấu bằng đư ờng viền dày hơn.

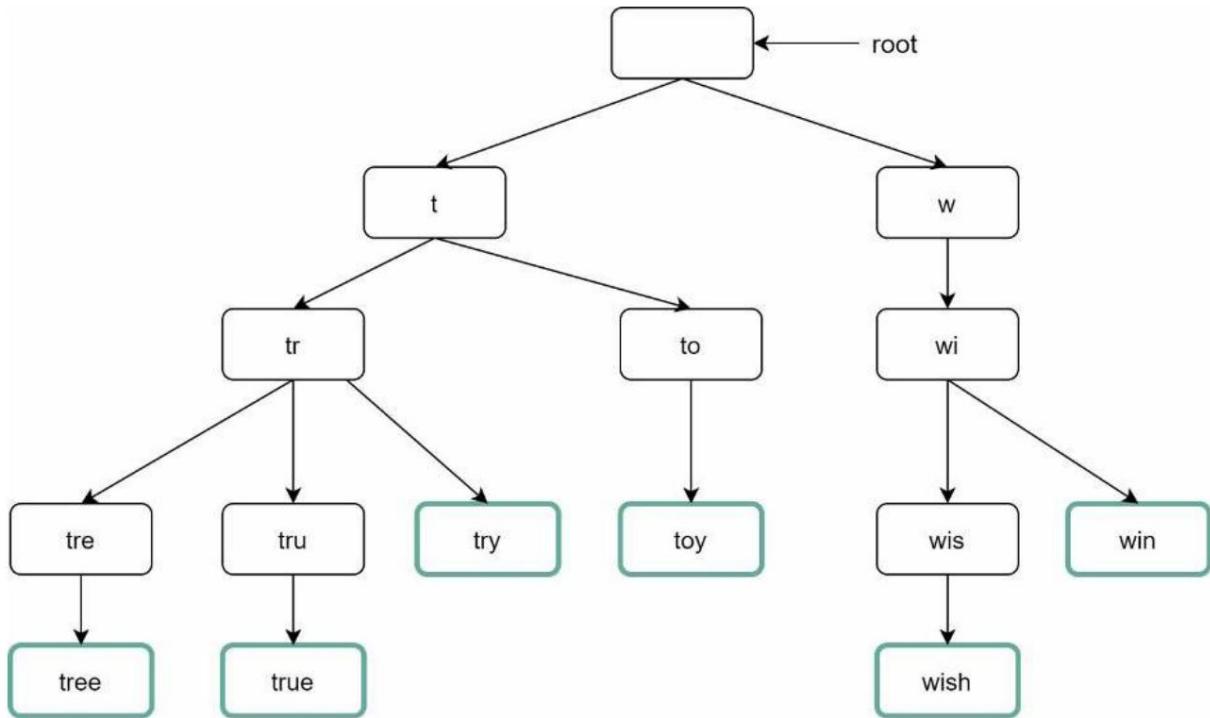


Figure 13-5

Cấu trúc dữ liệu trie cơ bản lưu trữ các ký tự trong các nút. Để hỗ trợ sắp xếp theo tần suất, thông tin tần suất cần được đưa vào các nút. Giả sử chúng ta có bảng tần suất sau.

Query	Frequency
tree	10
try	29
true	35
toy	14
wish	25
win	50

Table 13-2

Sau khi thêm thông tin tần suất vào các nút, cấu trúc dữ liệu trie được cập nhật được hiển thị trong Hình 13-6.

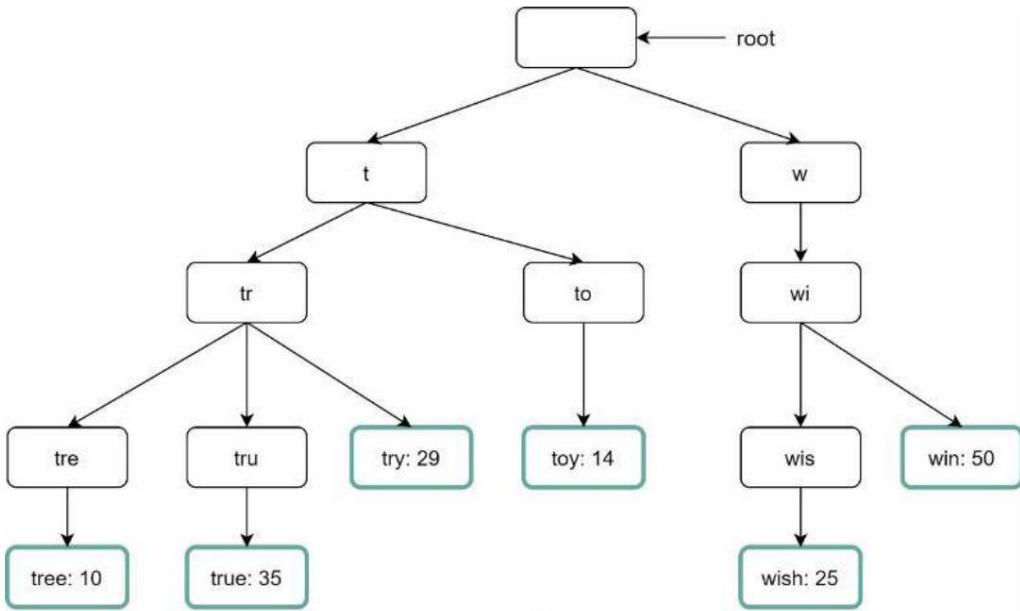


Figure 13-6

Tự động hoàn thành hoạt động như thế nào với trie? Trước khi đi sâu vào thuật toán, chúng ta hãy định nghĩa một số điều khoản.

- p: độ dài của tiền tố •
- n: tổng số nút trong một trie
- c: số lượng con của một nút nhất định

Các bước để có được k truy vấn được tìm kiếm nhiều nhất được liệt kê

- dưới đây:
1. Tìm tiền tố. Độ phức tạp thời gian: $O(p)$.
 2. Duyệt cây con từ nút tiền tố để lấy tất cả các con hợp lệ. Một con hợp lệ nếu nó có thể tạo thành một chuỗi truy vấn hợp lệ. Độ phức tạp thời gian: $O(c)$
 3. Sắp xếp các con và lấy k hàng đầu. Độ phức tạp thời gian: $O(c \log c)$

Chúng ta hãy sử dụng một ví dụ như trong Hình 13-7 để giải thích thuật toán. Giả sử k bằng 2 và người dùng nhập "tr" vào hộp tìm kiếm. Thuật toán hoạt động như sau:

- Bước 1: Tìm nút tiền tố "tr".
- Bước 2:

Duyệt cây con để lấy tất cả các nút con hợp

lệ. Trong trường hợp này, các nút [tree: 10], [true: 35], [try: 29] là hợp lệ. • Bước 3: Sắp xếp các nút con và lấy top 2. [true: 35] và

[try: 29] là 2 truy vấn hàng đầu có tiền tố "tr".

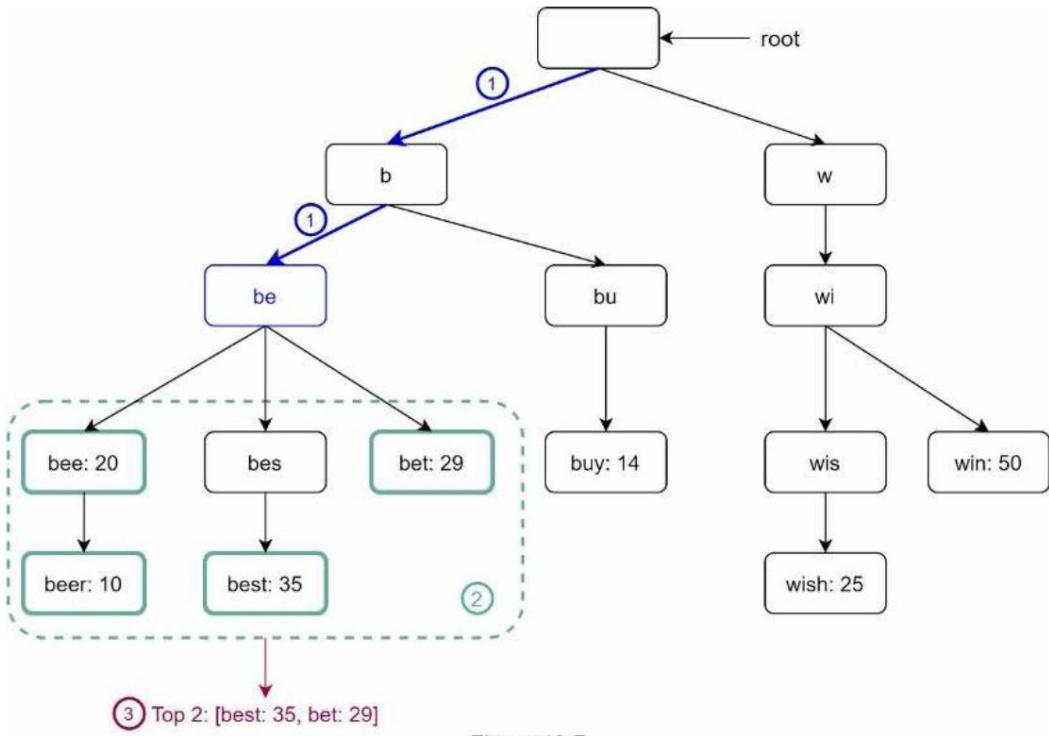


Figure 13-7

Độ phức tạp thời gian của thuật toán này là tổng thời gian dành cho từng bước đư ợc đề cập ở trên: $O(p) + O(c) + O(clocc)$

Thuật toán trên khá đơn giản. Tuy nhiên, nó quá chậm vì chúng ta cần phải duyệt toàn bộ trie để có đư ợc kết quả top k trong trường hợp xấu nhất. Dưới đây là hai tối ưu hóa:

1. Giới hạn độ dài tối đa của tiền tố 2.

Lưu trữ các truy vấn tìm kiếm hàng đầu tại mỗi

nút Chúng ta hãy xem xét từng tối ưu hóa này.

Giới hạn độ dài tối đa của tiền tố Ngươi

dùng hiếm khi nhập truy vấn tìm kiếm dài vào hộp tìm kiếm. Do đó, có thể nói p là một số nguyên nhỏ, chẳng hạn như 50. Nếu chúng ta giới hạn độ dài của tiền tố, độ phức tạp về thời gian cho "Tìm tiền tố" có thể giảm từ $O(p)$ xuống $O(\text{hàng số nhỏ})$, hay còn gọi là $O(1)$.

Lưu trữ các truy vấn tìm kiếm hàng đầu tại mỗi

nút Để tránh phải duyệt toàn bộ bộ ba, chúng tôi lưu trữ k truy vấn đư ợc sử dụng thường xuyên nhất tại mỗi nút.

Vì 5 đến 10 gợi ý tự động hoàn thành là đủ cho người dùng nên k là một con số tương ứng với nhau.

Trong trường hợp cụ thể của chúng tôi, chỉ có 5 truy vấn tìm kiếm hàng đầu đư ợc lưu vào bộ nhớ đệm.

Bằng cách lưu trữ các truy vấn tìm kiếm hàng đầu tại mỗi nút, chúng tôi giảm đáng kể độ phức tạp về thời gian để truy xuất 5 truy vấn hàng đầu. Tuy nhiên, thiết kế này đòi hỏi nhiều không gian đẻ lưu trữ các truy vấn hàng đầu tại mỗi nút. Việc trao đổi không gian lấy thời gian là rất đáng giá vì thời gian phản hồi nhanh là rất quan trọng.

Hình 13-8 cho thấy cấu trúc dữ liệu trie đư ợc cập nhật. 5 truy vấn hàng đầu đư ợc lưu trữ trên mỗi nút. Ví dụ, nút có tiền tố "be" lưu trữ những thông tin sau: [best: 35, bet: 29, bee: 20, be: 15, beer: 10].

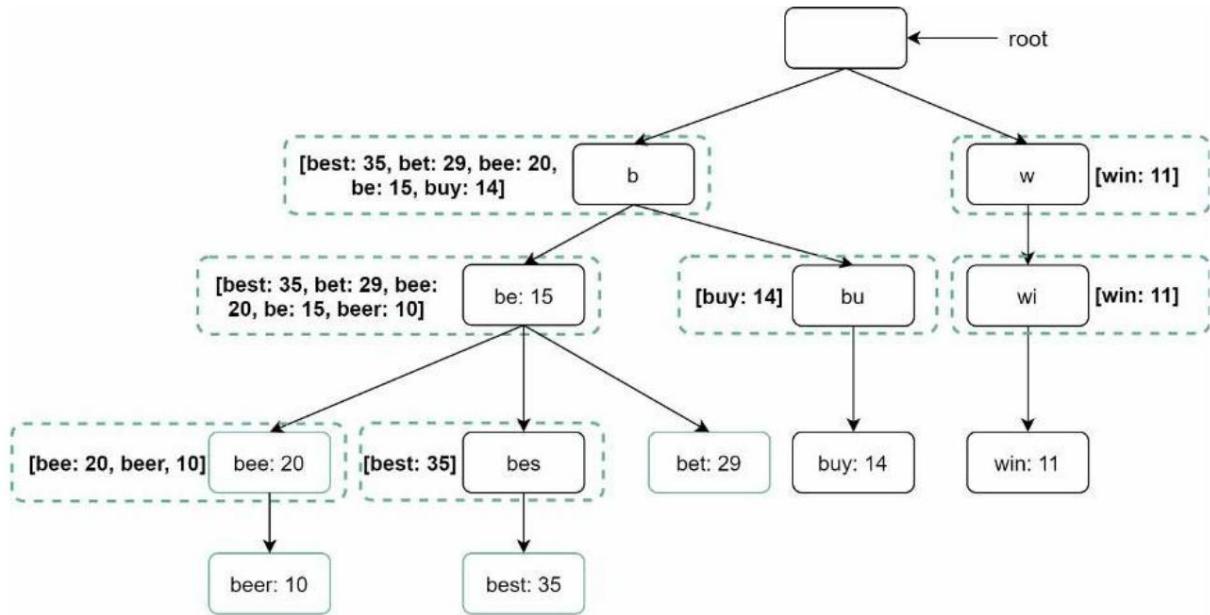


Figure 13-8

Chúng ta hãy xem lại độ phức tạp về thời gian của thuật toán sau khi áp dụng hai tối ưu hóa này:

1. Tìm nút tiền tố. Độ phức tạp thời gian: $O(1)$
2. Trả về k hàng đầu. Vì k truy vấn hàng đầu được lưu trong bộ nhớ đệm nên độ phức tạp thời gian cho bước này là $O(1)$.

Vì độ phức tạp về thời gian cho mỗi bước được giảm xuống còn $O(1)$, nên thuật toán của chúng tôi chỉ mất $O(1)$ để lấy k truy vấn hàng đầu.

Dịch vụ thu thập dữ liệu Trong thiết

kế trước đây, bắt cứ khi nào người dùng nhập truy vấn tìm kiếm, dữ liệu sẽ được cập nhật theo thời gian thực.

Cách tiếp cận này không thực tế vì hai lý do sau:

- Người dùng có thể nhập hàng tỷ truy vấn mỗi ngày. Việc cập nhật trie trên mọi truy vấn làm chậm đáng kể dịch vụ truy vấn.
- Các đề xuất hàng đầu có thể không thay đổi nhiều sau khi trie được xây dựng. Do đó, không cần thiết phải cập nhật trie thường xuyên.

Để thiết kế một dịch vụ thu thập dữ liệu có thể mở rộng, chúng tôi xem xét dữ liệu đến từ đâu và dữ liệu được sử dụng như thế nào. Các ứng dụng thời gian thực như Twitter yêu cầu các đề xuất tự động hoàn thành được cập nhật.

Tuy nhiên, các gợi ý tự động hoàn thành cho nhiều từ khóa của Google có thể không thay đổi nhiều hàng ngày.

Bắt cháp sự khác biệt trong các truy vấn hợp sử dụng, nền tảng cơ bản cho dịch vụ thu thập dữ liệu vẫn như cũ vì dữ liệu được sử dụng để xây dựng trie thường đến từ các dịch vụ phân tích hoặc ghi nhật ký.

Hình 13-9 cho thấy dịch vụ thu thập dữ liệu được thiết kế lại. Mỗi thành phần được kiểm tra từng cái một.



Figure 13-9

Nhật ký phân tích. Lưu trữ dữ liệu thô về truy vấn tìm kiếm. Nhật ký chỉ được thêm vào và không được lập chỉ mục. Bảng 13-3 hiển thị ví dụ về tập nhật ký.

query	time
tree	2019-10-01 22:01:01
try	2019-10-01 22:01:05
tree	2019-10-01 22:01:30
toy	2019-10-01 22:02:22
tree	2019-10-02 22:02:42
try	2019-10-03 22:03:03

Table 13-3

Aggregators. Kích thước của nhật ký phân tích thường rất lớn và dữ liệu không ở đúng định dạng. Chúng tôi cần tổng hợp dữ liệu để hệ thống của chúng tôi có thể dễ dàng xử lý.

Tùy thuộc vào ứng dụng sử dụng, chúng tôi có thể tổng hợp dữ liệu khác nhau. Đối với các ứng dụng thời gian thực như Twitter, chúng tôi tổng hợp dữ liệu trong khoảng thời gian ngắn hơn vì kết quả thời gian thực rất quan trọng. Mặt khác, việc tổng hợp dữ liệu ít thường xuyên hơn, chẳng hạn như một lần mỗi tuần, có thể đủ tốt cho nhiều ứng dụng sử dụng. Trong phiên phòng vấn, hãy xác minh xem kết quả thời gian thực có quan trọng không. Chúng tôi cho rằng trie được xây dựng lại hàng tuần.

Dữ liệu tổng hợp.

Bảng 13-4 hiển thị ví dụ về dữ liệu tổng hợp hàng tuần. Trong “thời gian” biểu thị thời gian bắt đầu của một tuần. Trong “tần suất” là tổng số lần xuất hiện của truy vấn thường ứng trong tuần đó.

query	Time	frequency
tree	2019-10-01	12000
tree	2019-10-08	15000
tree	2019-10-15	9000
toy	2019-10-01	8500
toy	2019-10-08	6256
toy	2019-10-15	8866

Table 13-4

Công nhân. Công nhân là một tập hợp các máy chủ thực hiện các công việc không đồng bộ theo các khoảng thời gian đều đặn. Họ xây dựng cấu trúc dữ liệu trie và lưu trữ nó trong Trie DB.

Trie Cache. Trie Cache là hệ thống bộ nhớ đệm phân tán lưu trữ trie trong bộ nhớ để đọc nhanh. Nó sẽ chụp nhanh DB hàng tuần.

Trie DB. Trie DB là kho lưu trữ liên tục. Có hai tùy chọn để lưu trữ dữ liệu: 1. Kho lưu trữ tài liệu: Vì một trie mới được xây dựng hàng tuần, chúng ta có thể chụp nhanh định kỳ, tuần tự hóa nó và lưu trữ dữ liệu đã tuần tự hóa trong cơ sở dữ liệu. Kho lưu trữ tài liệu như MongoDB [4] phù hợp với dữ liệu tuần tự hóa.

2. Kho lưu trữ khóa-giá trị: Trie có thể được biểu diễn dưới dạng bảng băm [4] bằng cách áp dụng logic sau: •

Mỗi tiền tố trong trie được ánh xạ tới một khóa trong bảng băm. •

Dữ liệu trên mỗi nút trie được ánh xạ tới một giá trị trong bảng băm.

Hình 13-10 hiển thị ánh xạ giữa bảng trie và bảng băm.

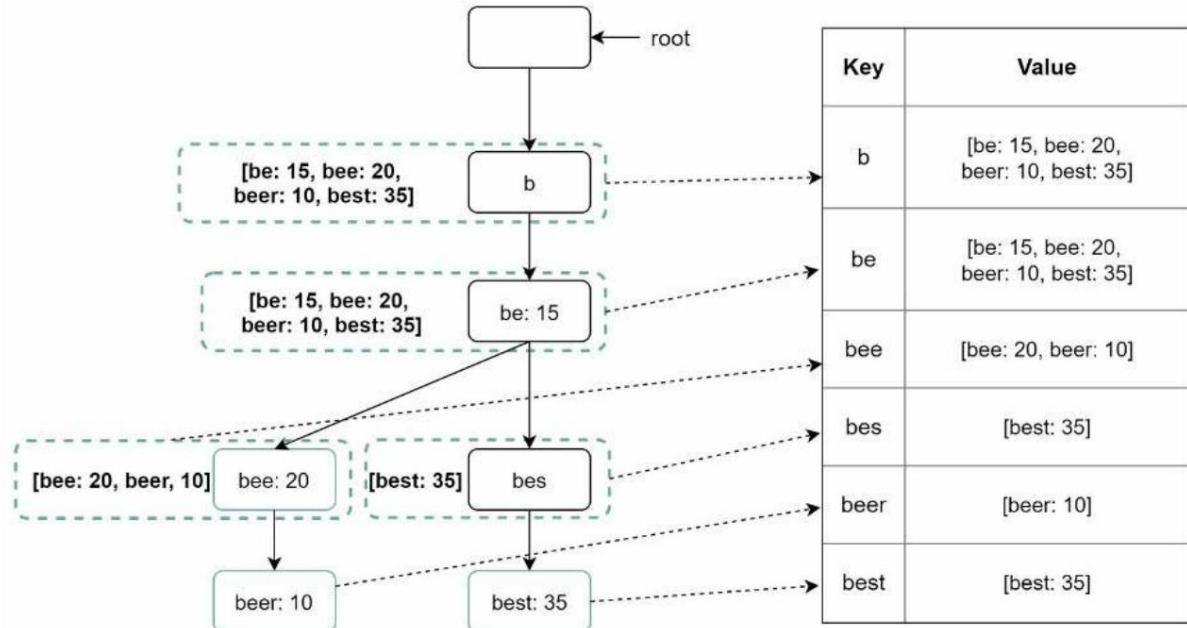


Figure 13-10

Trong Hình 13-10, mỗi nút trie bên trái được ánh xạ tới cặp `<key, value>` bên phải. Nếu bạn không rõ cách hoạt động của kho lưu trữ khóa-giá trị, hãy tham khảo Chương 6: Thiết kế kho lưu trữ khóa-giá trị.

Dịch vụ truy vấn

Trong thiết kế cấp cao, dịch vụ truy vấn sẽ gọi trực tiếp cơ sở dữ liệu để lấy 5 kết quả hàng đầu.

Hình 13-11 cho thấy thiết kế được cải tiến vì thiết kế trước đây không hiệu quả.

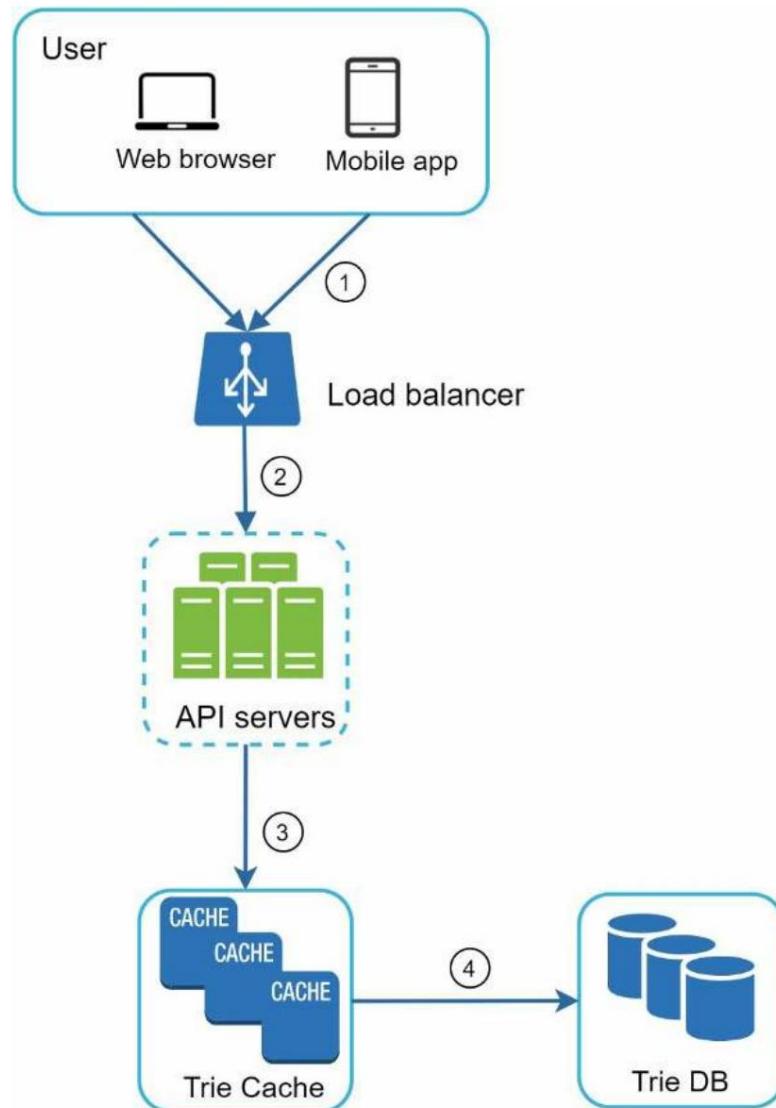


Figure 13-11

1. Truy vấn tìm kiếm được gửi đến bộ cân bằng tải.
2. Bộ cân bằng tải định tuyến yêu cầu đến máy chủ API.
3. Máy chủ API lấy dữ liệu từ Trie Cache và xây dựng các đề xuất tự động hoàn thành cho khách hàng.
4. Trong trường hợp dữ liệu không có trong Trie Cache, chúng tôi sẽ bổ sung dữ liệu trả lại bộ nhớ đệm. Theo cách này, tất cả các yêu cầu tiếp theo cùng một tiền tố sẽ được trả về từ bộ nhớ đệm. Có thể xảy ra lỗi bộ nhớ đệm khi máy chủ bộ nhớ đệm hết bộ nhớ hoặc ngoại tuyến.

Dịch vụ truy vấn yêu cầu tốc độ cực nhanh. Chúng tôi đề xuất các tối ưu hóa sau:

- Yêu cầu AJAX. Đối với các ứng dụng web, trình duyệt thường gửi các yêu cầu AJAX để lấy kết quả tự động hoàn thành. Lợi ích chính của AJAX là việc gửi/nhận yêu cầu/phản hồi không làm mới toàn bộ trang web.
- Bộ nhớ đệm trình duyệt. Đối với nhiều ứng dụng, các gợi ý tìm kiếm tự động hoàn thành có thể không thay đổi nhiều trong thời gian ngắn. Do đó, các gợi ý tự động hoàn thành có thể được lưu trong bộ nhớ đệm trình duyệt để cho phép các yêu cầu tiếp theo nhận kết quả trực tiếp từ bộ nhớ đệm. Công cụ tìm kiếm Google sử dụng một cơ chế bộ nhớ đệm. Hình 13-12 hiển thị tiêu đề phản hồi khi bạn nhập "phỏng vấn thiết kế hệ thống" trên công cụ tìm kiếm Google. Như bạn có thể thấy, Google

lưu trữ kết quả trong trình duyệt trong 1 giờ. Xin lưu ý: "private" trong cache-control có nghĩa là kết quả dành cho một người dùng duy nhất và không được lưu trữ trong bộ nhớ đệm chia sẻ. "max-age=3600" có nghĩa là bộ nhớ đệm có hiệu lực trong 3600 giây, tức là một giờ.

```

Request URL: https://www.google.com/complete/search?q&cp=0&client=psy-ab&xssi=t&gs_ri=gws-wiz&hl=en&authuser=0&pq=system_design_interview
Request method: GET
Remote address: [2607:f8b0:4005:807::2004]:443
Status code: 200 OK
Edit and Resend Raw headers
Version: HTTP/2.0
Filter headers
Response headers (615 B)
alt-svc: quic=":443"; ma=2592000; v="46..00h3-Q043=:443"; ma=2592000
cache-control: private, max-age=3600
content-disposition: attachment; filename="f.txt"
content-encoding: br
content-type: application/json; charset=UTF-8
date: Tue, 17 Dec 2019 22:52:01 GMT
expires: Tue, 17 Dec 2019 22:52:01 GMT
server: gws
strict-transport-security: max-age=31536000
trailer: X-Google-GFE-Current-Request-Cost-From-GWS
X-Firefox-Spdy: h2
x-frame-options: SAMEORIGIN
x-xss-protection: 0

```

Figure 13-12

- Lấy mẫu dữ liệu: Đối với một hệ thống quy mô lớn, việc ghi lại mọi truy vấn tìm kiếm đòi hỏi rất nhiều sức mạnh xử lý và lưu trữ. Lấy mẫu dữ liệu rất quan trọng. Ví dụ, chỉ có 1 trong số mỗi N yêu cầu được hệ thống ghi lại.

Trie operations Trie

là thành phần cốt lõi của hệ thống tự động hoàn thành. Chúng ta hãy xem cách hoạt động của trie operations (create, update, and delete).

Tạo nên

Trie được tạo ra bởi những người lao động sử dụng dữ liệu tổng hợp. Nguồn dữ liệu là từ Analytics Log/DB.

Cập nhật

Có hai cách để cập nhật trie.

Tùy chọn 1: Cập nhật trie hàng tuần. Khi một trie mới được tạo, trie mới sẽ thay thế trie cũ.

Tùy chọn 2: Cập nhật trực tiếp từng nút trie. Chúng tôi cố gắng tránh thao tác này vì nó chậm. Tuy nhiên, nếu kích thước của nút trie nhỏ, thì đây là giải pháp chấp nhận được. Khi chúng tôi cập nhật một nút trie, các nút tổ tiên của nó cho đến nút gốc phải được cập nhật vì các nút tổ tiên lưu trữ các truy vấn hàng đầu của các nút con. Hình 13-13 cho thấy một ví dụ về cách hoạt động của thao tác cập nhật. Ở phía bên trái, truy vấn tìm kiếm "bia" có giá trị ban đầu là 10. Ở phía bên phải, nó được cập nhật thành 30. Như bạn có thể thấy, nút và các nút tổ tiên của nó có giá trị "bia" được cập nhật thành 30.

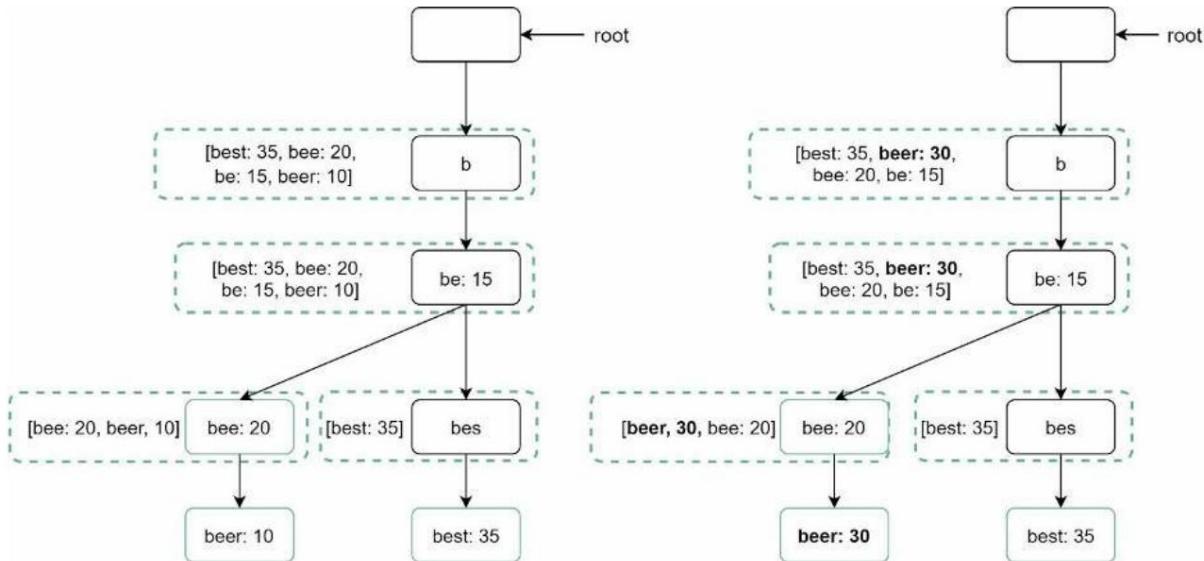


Figure 13-13

Xóa bỏ

Chúng tôi phải xóa các gợi ý tự động hoàn thành mang tính thù hận, bạo lực, khiêu dâm hoặc nguy hiểm.

Chúng tôi thêm một lớp lọc (Hình 13-14) ở phía trước Trie Cache để lọc ra các gợi ý không mong muốn. Có một lớp lọc giúp chúng tôi có thể linh hoạt xóa kết quả dựa trên các quy tắc lọc khác nhau. Các gợi ý không mong muốn được xóa vật lý khỏi cơ sở dữ liệu theo cách không đồng bộ để bộ dữ liệu chính xác sẽ được sử dụng để xây dựng trie trong chu kỳ cập nhật tiếp theo.

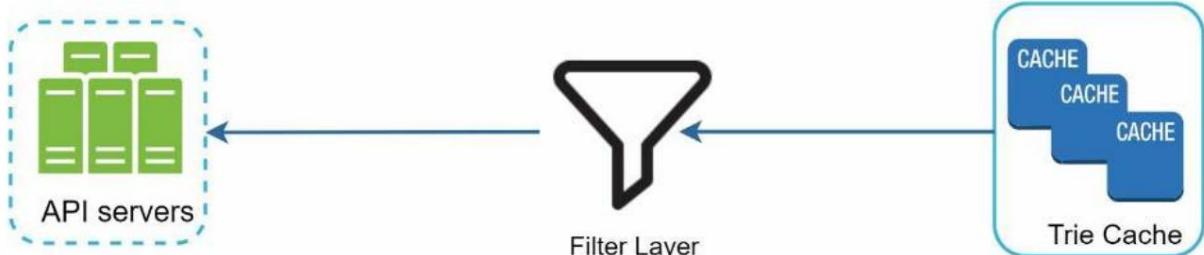


Figure 13-14

Mở rộng dung lư ợng lưu trữ

Trước đây giờ chúng ta đã phát triển một hệ thống để cung cấp các truy vấn tự động hoàn thành cho người dùng, đến lúc giải quyết vấn đề về khả năng mở rộng khi bộ nhớ quá lớn không thể chứa vừa trên một máy chủ.

Vì tiếng Anh là ngôn ngữ duy nhất được hỗ trợ, nên một cách đơn giản để phân mảnh là dựa trên ký tự đầu tiên. Sau đây là một số ví dụ.

- Nếu chúng ta cần hai máy chủ để lưu trữ, chúng ta có thể lưu trữ các truy vấn bắt đầu bằng 'a' đến 'm' trên máy chủ đầu tiên và 'n' đến 'z' trên máy chủ thứ hai.
- Nếu chúng ta cần ba máy chủ, chúng ta có thể chia các truy vấn thành 'a' đến 'i', 'j' đến 'r' và 's' đến 'z'.

Theo logic này, chúng ta có thể chia nhỏ các truy vấn lên đến 26 máy chủ vì tiếng Anh có 26 ký tự chữ cái. Chúng ta hãy định nghĩa phân mảnh dựa trên ký tự đầu tiên là phân mảnh cấp độ đầu tiên. Để lưu trữ dữ liệu vượt quá 26 máy chủ, chúng ta có thể phân mảnh ở cấp độ thứ hai hoặc thậm chí ở cấp độ thứ ba. Ví dụ, các truy vấn dữ liệu bắt đầu bằng 'a' có thể được chia thành 4 máy chủ: 'aa-ag', 'ah-an', 'ao-az' và 'av-az'.

Thoạt nhìn thì cách tiếp cận này có vẻ hợp lý, cho đến khi bạn nhận ra rằng có nhiều từ bắt đầu bằng chữ cái 'c' hơn là 'x'. Điều này tạo ra sự phân bổ không đồng đều.

Để giảm thiểu vấn đề mất cân bằng dữ liệu, chúng tôi phân tích mô hình phân phối dữ liệu lịch sử và áp dụng logic phân mảnh thông minh hơn như thể hiện trong Hình 13-15. Trình quản lý bản đồ phân mảnh duy trì cơ sở dữ liệu tra cứu để xác định nơi lưu trữ các hàng. Ví dụ, nếu có số lượng truy vấn lịch sử tương tự cho 's' và cho 'u', 'v', 'w', 'x', 'y' và 'z' kết hợp, chúng tôi có thể duy trì hai phân mảnh: một cho 's' và một cho 'u' đến 'z'.

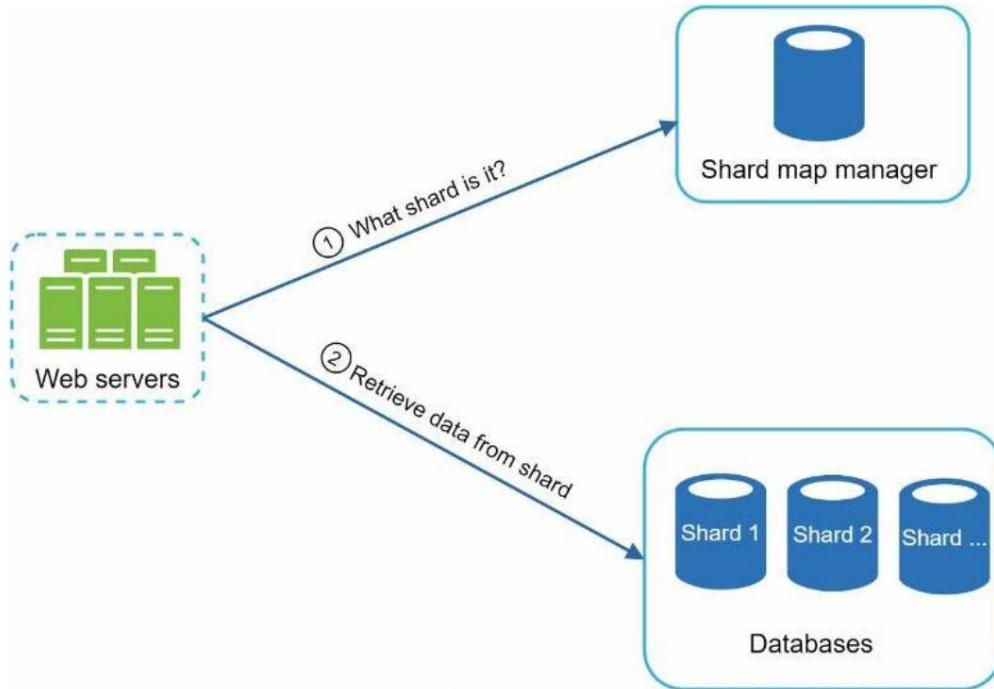


Figure 13-15

Bài 4 - Kết thúc

Sau khi hoàn tất phần tìm hiểu sâu, người phỏng vấn có thể hỏi bạn một số câu hỏi tiếp theo.

Người phỏng vấn: Làm thế nào để mở rộng thiết kế của bạn để hỗ trợ nhiều ngôn ngữ?

Để hỗ trợ các truy vấn không phải tiếng Anh khác, chúng tôi lưu trữ các ký tự Unicode trong các nút trie. Nếu bạn không quen với Unicode, đây là định nghĩa: "một tiêu chuẩn mã hóa bao gồm tất cả các ký tự cho tất cả các hệ thống chữ viết trên thế giới, hiện đại và cổ đại" [5].

Người phỏng vấn: Điều gì sẽ xảy ra nếu các truy vấn tìm kiếm hàng đầu ở một quốc gia khác với các quốc gia khác?

Trong trường hợp này, chúng ta có thể xây dựng các lần thử khác nhau cho các quốc gia khác nhau. Để cải thiện thời gian phản hồi, chúng ta có thể lưu trữ các lần thử trong CDN.

Người phỏng vấn: Làm thế nào chúng ta có thể hỗ trợ các truy vấn tìm kiếm theo xu hướng (thời gian thực)?

Giả sử một sự kiện tin tức nổi ra, một truy vấn tìm kiếm đột nhiên trở nên phổ biến. Thiết kế ban đầu của chúng tôi sẽ không hoạt động vì:

- Người làm việc ngoại tuyến chưa được lên lịch cập nhật trie vì việc này được lên lịch chạy hàng tuần. • Ngay cả khi được lên lịch, việc xây dựng trie cũng mất quá nhiều thời gian.

Xây dựng chức năng tự động hoàn thành tìm kiếm theo thời gian thực rất phức tạp và nằm ngoài phạm vi của cuốn sách này nên chúng tôi sẽ chỉ đưa ra một vài ý tưởng:

- Giảm bộ dữ liệu đang hoạt động bằng cách phân mảnh.
- Thay đổi mô hình xếp hạng và chỉ định nhiều trọng số hơn cho các truy vấn tìm kiếm gần đây.
- Dữ liệu có thể đến dưới dạng luồng, do đó chúng tôi không thể truy cập vào tất cả dữ liệu cùng một lúc. Dữ liệu phát trực tuyến có nghĩa là dữ liệu được tạo liên tục. Xử lý luồng yêu cầu một bộ hệ thống khác: Apache Hadoop MapReduce [6], Apache Spark Streaming [7], Apache Storm [8], Apache Kafka [9], v.v. Vì tất cả các chủ đề đó đều yêu cầu kiến thức chuyên môn cụ thể nên chúng tôi sẽ không đi sâu vào chi tiết ở đây.

Xin chúc mừng vì đã đi được đến đây! Bây giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

[1] Cuộc sống của một truy vấn Typeahead: <https://www.facebook.com/notes/facebook-engineering/the-life-of-a-typeahead-query/389105248919/> [2]

Cách chúng tôi xây dựng Prefixy: Dịch vụ tìm kiếm tiền tố có thể mở rộng để cung cấp năng lực cho tính năng tự động hoàn thành: <https://medium.com/@prefixyteam/how-we-built-prefixy-a-scalable-prefix-search-service-for-powering-autocomplete-c20f98e2eff1> [3]

Cây băm tiền tố Một cấu trúc dữ liệu lập chỉ mục trên các bảng băm phân tán: <https://people.eecs.berkeley.edu/~sylvia/papers/pht.pdf>

[4] Wikipedia MongoDB: <https://en.wikipedia.org/wiki/MongoDB> [5] Câu hỏi thường gặp về Unicode: https://www.unicode.org/faq/basic_q.html

[6] Apache hadoop: <https://hadoop.apache.org/> [7]

Spark phát trực tuyến: <https://spark.apache.org/streaming/>

[8] Apache storm: <https://storm.apache.org/>

[9] Apache kafka: <https://kafka.apache.org/documentation/>

CHƯƠNG 14: THIẾT KẾ YOUTUBE

Trong chương này, bạn được yêu cầu thiết kế YouTube. Giải pháp cho câu hỏi này có thể được áp dụng cho các câu hỏi phòng vấn khác như thiết kế nền tảng chia sẻ video như Netflix và Hulu.

Hình 14-1 hiển thị trang chủ của YouTube.

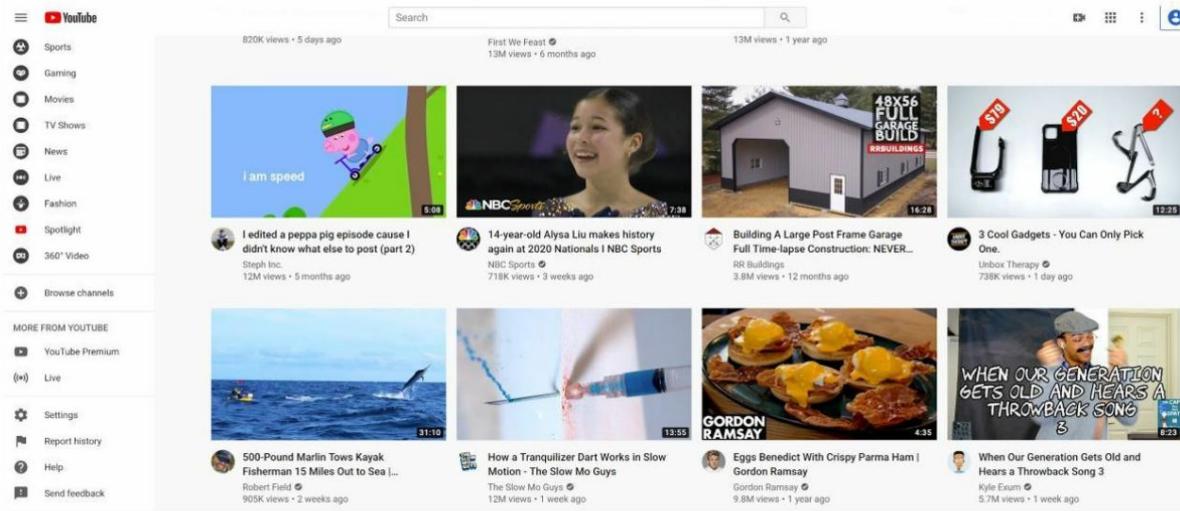


Figure 14-1

YouTube có vẻ đơn giản: người sáng tạo nội dung tải video lên và người xem nhấp vào phát. Có thực sự đơn giản như vậy không? Không hẳn vậy. Có rất nhiều công nghệ phức tạp ẩn chứa bên dưới sự đơn giản đó. Chúng ta hãy cùng xem một số số liệu thống kê, nhân khẩu học và sự thật thú vị ẩn tàng về YouTube vào năm 2020 [1] [2].

- Tổng số người dùng hoạt động hàng tháng: 2 tỷ. • Số video
- được xem mỗi ngày: 5 tỷ. • 73% người lớn ở Hoa Kỳ sử dụng
- YouTube.
- 50 triệu người sáng tạo trên YouTube.
- Doanh thu quảng cáo của YouTube là 15,1 tỷ đô la trong cả năm 2019, tăng 36% so với năm 2018. • YouTube chiếm 37% tổng lưu lượng truy cập internet trên thiết bị di động. • YouTube
- có sẵn bằng 80 ngôn ngữ khác nhau.

Từ những số liệu thống kê này, chúng ta biết YouTube rất lớn, có phạm vi toàn cầu và kiếm được rất nhiều tiền.

Bút ớc 1 - Hiểu vấn đề và thiết lập phạm vi thiết kế Như đư ợc tiết lộ trong Hình 14-1, ngoài việc xem video, bạn có thể làm nhiều việc hơn trên YouTube. Ví dụ, bình luận, chia sẻ hoặc thích video, lưu video vào danh sách phát, đăng ký kênh, v.v. Không thể thiết kế mọi thứ trong một cuộc phỏng vấn kéo dài 45 hoặc 60 phút. Do đó, điều quan trọng là phải đặt câu hỏi để thu hẹp phạm vi.

Ứng viên: Những đặc điểm nào là quan trọng?

Người phỏng vấn: Khả năng tải video lên và xem video.

Ứng viên: Chúng ta cần hỗ trợ những khách hàng nào?

Người phỏng vấn: Ứng dụng di động, trình duyệt web và TV thông minh.

Ứng viên: Chúng ta có bao nhiêu người dùng hoạt động hàng ngày?

Người phỏng vấn: 5 triệu

Ứng viên: Thời gian trung bình mỗi ngày bạn dành cho sản phẩm là bao lâu?

Người phỏng vấn: 30 phút.

Ứng viên: Chúng ta có cần hỗ trợ người dùng quốc tế không?

Người phỏng vấn: Có, phần lớn người dùng là người dùng quốc tế.

Ứng viên: Độ phân giải video đư ợc hỗ trợ là bao nhiêu?

Người phỏng vấn: Hệ thống chấp nhận hầu hết các định dạng và độ phân giải video.

Ứng viên: Có cần mã hóa không?

Người phỏng vấn: Vâng

Ứng viên: Có yêu cầu về kích thước tệp video không?

Người phỏng vấn: Nền tảng của chúng tôi tập trung vào các video có kích thước vừa và nhỏ. Kích thước video tối đa đư ợc phép là 1GB.

Ứng viên: Chúng ta có thể tận dụng một số cơ sở hạ tầng đám mây hiện có do Amazon, Google hoặc Microsoft cung cấp không?

Người phỏng vấn: Đó là một câu hỏi hay. Việc xây dựng mọi thứ từ đầu là không thực tế đối với hầu hết các công ty, nên tận dụng một số dịch vụ đám mây hiện có.

Trong chương này, chúng tôi tập trung vào việc thiết kế một dịch vụ phát video trực tuyến với các tính năng sau:

- Khả năng tải video nhanh • Truyền phát video mượt mà • Khả năng thay đổi chất lượng video • Chi phí cơ sở hạ tầng thấp
- Yêu cầu về tính khả dụng, khả năng mở rộng và độ tin cậy cao • Khách hàng đư ợc hỗ trợ: Ứng dụng di động, trình duyệt web và TV thông minh

Ước tính sơ bộ Các ước tính sau đây

dựa trên nhiều giả định, vì vậy, điều quan trọng là phải trao đổi với người phỏng vấn để đảm bảo rằng họ có cùng quan điểm.

- Giả sử sản phẩm có 5 triệu người dùng hoạt động hàng ngày (DAU). • Người dùng xem 5 video mỗi ngày. • 10% người dùng tải lên 1 video mỗi ngày. • Giả sử kích thước video trung bình là 300 MB. • Tổng dung lượng lưu trữ hàng ngày cần thiết: $5 \text{ triệu} * 10\% * 300 \text{ MB} = 150 \text{ TB}$

- Chi phí CDN.
- Khi mạng CDN đàm mây cung cấp video, bạn sẽ bị tính phí cho dữ liệu được truyền ra khỏi mạng CDN.
- Chúng ta hãy sử dụng CDN CloudFront của Amazon để ước tính chi phí (Hình 14-2) [3]. Giả sử 100% lưu lượng được phục vụ từ Hoa Kỳ. Chi phí trung bình cho mỗi GB là 0,02 đô la.
Để đơn giản, chúng tôi chỉ tính chi phí phát trực tuyến video. • 5 triệu

$$* 5 \text{ video } * 0,3 \text{ GB } * 0,02 \text{ đô la } = 150.000 \text{ đô la mỗi ngày.}$$

Dựa trên ước tính chi phí sơ bộ, chúng ta biết rằng việc phục vụ video từ CDN tốn rất nhiều tiền.

Mặc dù các nhà cung cấp đám mây sẵn sàng giảm đáng kể chi phí CDN cho các khách hàng lớn, như chi phí vẫn còn đáng kể. Chúng tôi sẽ thảo luận về các cách giảm chi phí CDN một cách sâu sắc.

Per Month	United States & Canada	Europe & Israel	South Africa & Middle East	South America	Japan	Australia	Singapore, South Korea, Taiwan, Hong Kong, & Philippines	India
First 10TB	\$0.085	\$0.085	\$0.110	\$0.110	\$0.114	\$0.114	\$0.140	\$0.170
Next 40TB	\$0.080	\$0.080	\$0.105	\$0.105	\$0.089	\$0.098	\$0.135	\$0.130
Next 100TB	\$0.060	\$0.060	\$0.090	\$0.090	\$0.086	\$0.094	\$0.120	\$0.110
Next 350TB	\$0.040	\$0.040	\$0.080	\$0.080	\$0.084	\$0.092	\$0.100	\$0.100
Next 524TB	\$0.030	\$0.030	\$0.060	\$0.060	\$0.080	\$0.090	\$0.080	\$0.100
Next 4PB	\$0.025	\$0.025	\$0.050	\$0.050	\$0.070	\$0.085	\$0.070	\$0.100
Over 5PB	\$0.020	\$0.020	\$0.040	\$0.040	\$0.060	\$0.080	\$0.060	\$0.100

Figure 14-2 On-demand pricing for Data Transfer to the Internet (per GB).

Bưu ớc 2 - Đề xuất thiết kế cấp cao và nhận được sự đồng thuận Như đã thảo luận trước đó, người phỏng vấn đề xuất nên tận dụng các dịch vụ đám mây hiện có thay vì xây dựng mọi thứ từ đầu. CDN và lưu trữ blob là các dịch vụ đám mây mà chúng tôi sẽ tận dụng. Một số độc giả có thể hỏi tại sao không tự mình xây dựng mọi thứ? Các lý do dưới đây liệt kê dưới đây:

- Phỏng vấn thiết kế hệ thống không phải là về việc xây dựng mọi thứ từ đầu. Trong khung thời gian hạn chế, việc lựa chọn đúng công nghệ để hoàn thành đúng công việc quan trọng hơn là giải thích chi tiết cách thức công nghệ hoạt động. Ví dụ, đề cập đến lưu trữ blob để lưu trữ video nguồn là đủ cho cuộc phỏng vấn. Nói về thiết kế chi tiết cho lưu trữ blob có thể là quá mức cần thiết.
- Xây dựng lưu trữ blob có thể mở rộng hoặc CDN cực kỳ phức tạp và tốn kém. Ngay cả các công ty lớn như Netflix hoặc Facebook cũng không tự xây dựng mọi thứ. Netflix tận dụng các dịch vụ đám mây của Amazon [4] và Facebook sử dụng CDN của Akamai [5].

Ở cấp độ cao, hệ thống bao gồm ba thành phần (Hình 14-3).

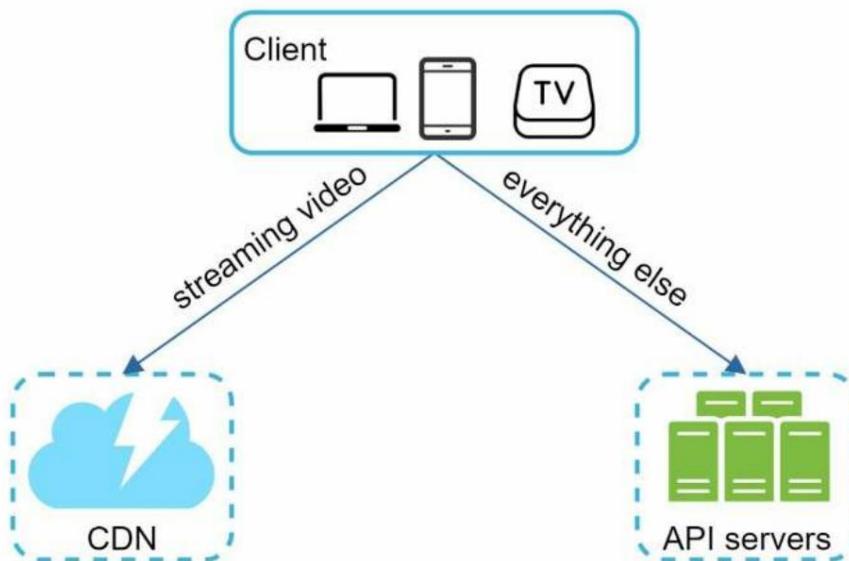


Figure 14-3

Khách hàng: Bạn có thể xem YouTube trên máy tính, điện thoại di động và TV thông minh.

CDN: Video được lưu trữ trong CDN. Khi bạn nhấn phát, video sẽ được phát trực tuyến từ CDN.

Máy chủ API: Mọi thứ khác ngoại trừ phát trực tuyến video đều đi qua máy chủ API. Điều này bao gồm đề xuất nguồn cấp dữ liệu, tạo URL tải lên video, cập nhật cơ sở dữ liệu siêu dữ liệu và bộ nhớ đệm, đăng ký người dùng, v.v.

Trong phần hỏi/đáp, người phỏng vấn đã thể hiện sự quan tâm đến hai luồng: • Luồng tải video lên • Luồng phát trực tuyến video

Chúng ta sẽ khám phá thiết kế cấp cao cho từng luồng.

Luồng tải video lên Hình 14-4

hiển thị thiết kế cấp cao cho việc tải video lên.

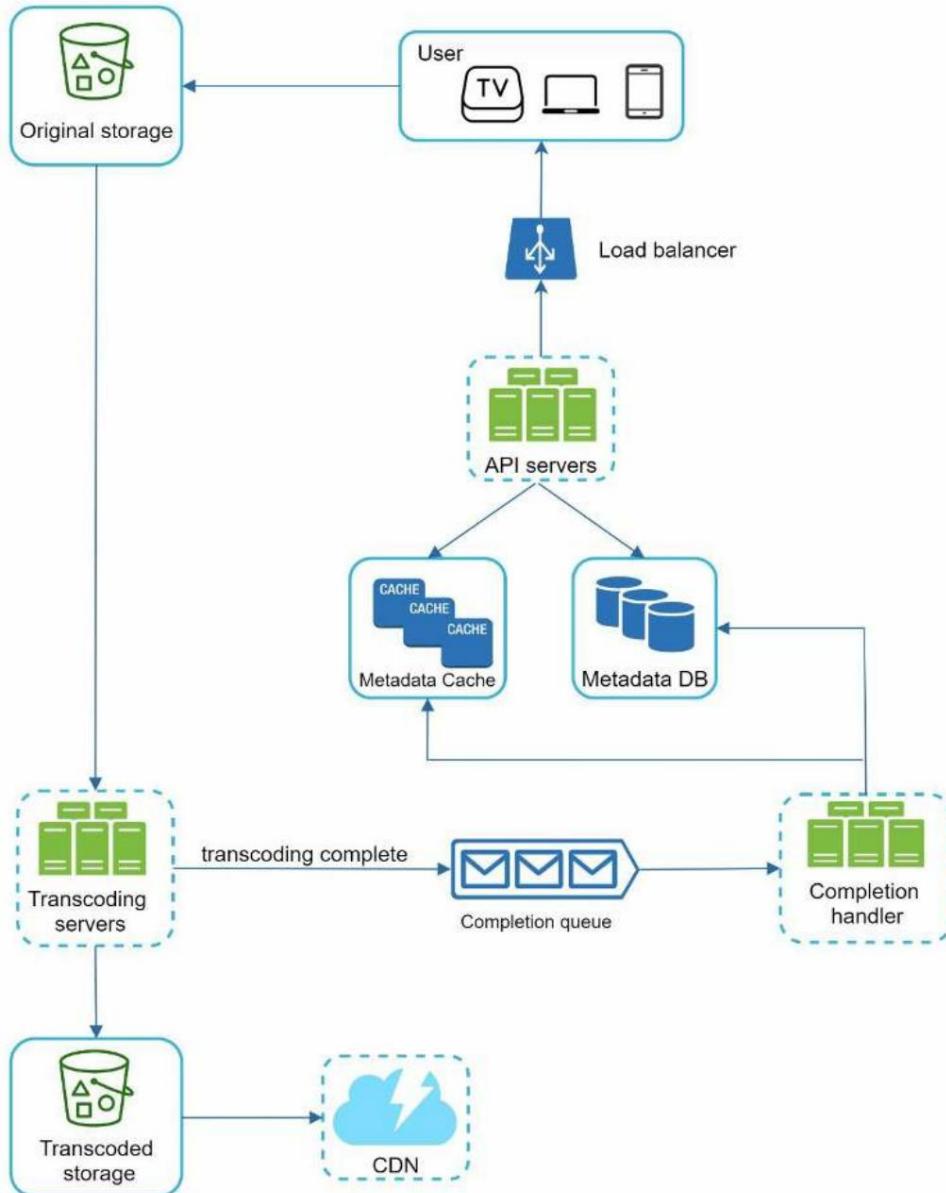


Figure 14-4

Nó bao gồm các thành phần sau:

- Người dùng: Người dùng xem YouTube trên các thiết bị như máy tính, điện thoại di động hoặc TV thông minh.
- Bộ cân bằng tải: Bộ cân bằng tải phân phối đều các yêu cầu giữa các máy chủ API.
- Máy chủ API: Tất cả các yêu cầu của người dùng đều đi qua máy chủ API ngoại trừ phát trực tuyến video.
- Cơ sở dữ liệu siêu dữ liệu: Siêu dữ liệu video được lưu trữ trong Cơ sở dữ liệu siêu dữ liệu. Nó được phân mảnh và sao chép để đáp ứng các yêu cầu về hiệu suất và tính khả dụng cao.
- Bộ đệm siêu dữ liệu: Để có hiệu suất tốt hơn, siêu dữ liệu video và đối tượng người dùng được lưu vào bộ đệm.
- Lưu trữ gốc: Hệ thống lưu trữ blob được sử dụng để lưu trữ video gốc. Một trích dẫn trên Wikipedia về lưu trữ blob cho thấy rằng: "Một Đối tượng nhị phân lớn (BLOB) là một tập hợp dữ liệu nhị phân được lưu trữ dưới dạng một thực thể duy nhất trong hệ thống quản lý cơ sở dữ liệu" [6].
- Máy chủ chuyển mã: Chuyển mã video cũng được gọi là mã hóa video. Đây là quá trình chuyển đổi định dạng video sang các định dạng khác (MPEG, HLS, v.v.), cung cấp các luồng video tốt nhất có thể cho các thiết bị và khả năng băng thông khác nhau.

- Lưu trữ chuyển mã: Đây là lưu trữ blob lưu trữ các tệp video đã chuyển mã. • CDN:
Video được lưu trong bộ nhớ đệm CDN. Khi bạn nhấp vào nút phát, video sẽ được phát trực tuyến từ CDN.

- Hàng đợi hoàn tất: Đây là hàng đợi tin nhắn lưu trữ thông tin về các sự kiện hoàn tất chuyển mã video. •
Trình xử lý hoàn tất:

Bao gồm danh sách các công nhân lấy dữ liệu sự kiện từ hàng đợi hoàn tất và cập nhật bộ đệm siêu dữ
liệu và cơ sở dữ liệu.

Bây giờ chúng ta đã hiểu từng thành phần riêng lẻ, hãy cùng xem xét cách hoạt động của luồng tải
video lên. Luồng được chia thành hai quy trình chạy song song. a. Tải video thực tế lên. b. Cập nhật
siêu dữ liệu video. Siêu dữ

liệu chứa thông tin về URL video, kích thước, độ phân giải, định dạng, thông tin người dùng, v.v.

Luồng a: tải video thực tế lên

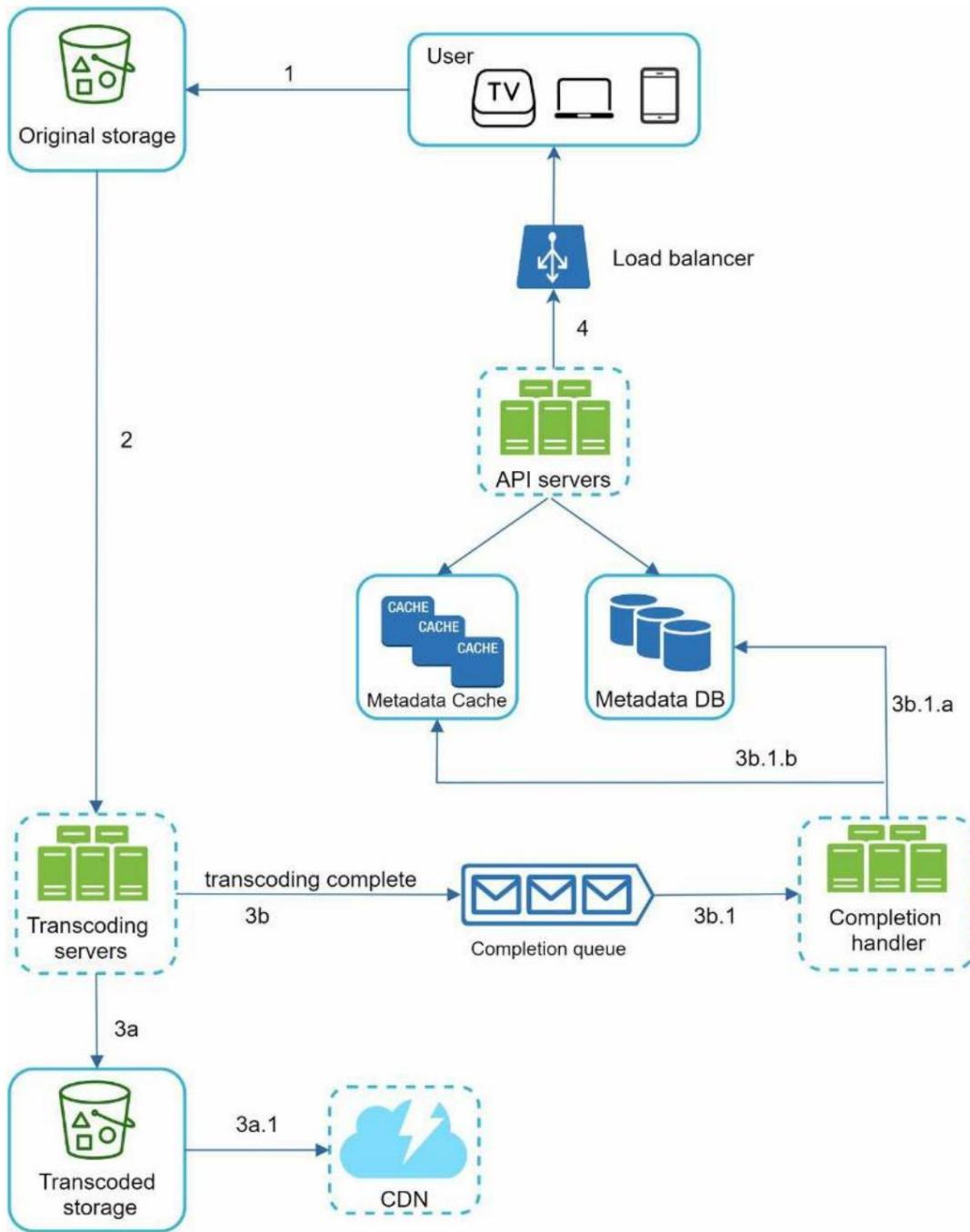


Figure 14-5

Hình 14-5 cho thấy cách tải video thực tế lên. Giải thích đư ợc hiển thị bên dưới:

1. Video đư ợc tải lên bộ nhớ gốc.
2. Máy chủ chuyển mã sẽ lấy video từ bộ lưu trữ gốc và bắt đầu chuyển mã.
3. Sau khi quá trình chuyển mã hoàn tất, hai bước sau đư ợc thực hiện song song:
 - 3a. Video đã chuyển mã đư ợc gửi đến bộ lưu trữ đã chuyển mã.
 - 3b. Các sự kiện hoàn tất chuyển mã đư ợc xếp hàng trong hàng đợi hoàn tất.
- 3a.1. Video đã chuyển mã đư ợc phân phối tới CDN.
- 3b.1. Trình xử lý hoàn tất chứa một nhóm công nhân liên tục kéo dữ liệu sự kiện từ hàng đợi.

3b.1.a. và 3b.1.b. Trình xử lý hoàn tất cập nhật cơ sở dữ liệu siêu dữ liệu và bộ nhớ đệm khi quá trình chuyển mã video hoàn tất.

4. Máy chủ API thông báo cho khách hàng rằng video đã được tải lên thành công và sẵn sàng để phát trực tuyến.

Luồng b: cập nhật siêu dữ liệu Trong

khi tệp đang được tải lên bộ lưu trữ gốc, máy khách song song gửi yêu cầu cập nhật siêu dữ liệu video như thể hiện trong Hình 14-6. Yêu cầu chứa siêu dữ liệu video, bao gồm tên tệp, kích thước, định dạng, v.v. Máy chủ API cập nhật bộ đệm siêu dữ liệu và cơ sở dữ liệu.

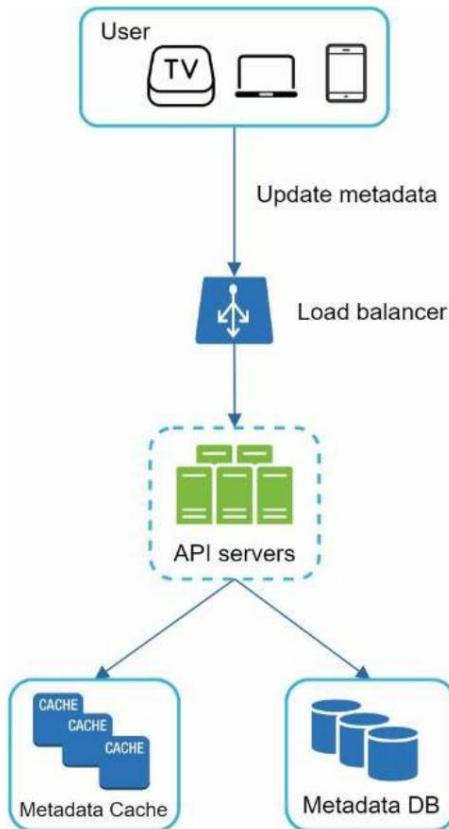


Figure 14-6

Luồng phát trực tuyến video Bắt

cứ khi nào bạn xem một video trên YouTube, video đó thường bắt đầu phát trực tuyến ngay lập tức và bạn không phải đợi cho đến khi toàn bộ video được tải xuống. Tải xuống có nghĩa là toàn bộ video được sao chép vào thiết bị của bạn, trong khi phát trực tuyến có nghĩa là thiết bị của bạn liên tục nhận được luồng video từ các video nguồn từ xa. Khi bạn xem video phát trực tuyến, máy khách của bạn sẽ tải một ít dữ liệu tại một thời điểm để bạn có thể xem video ngay lập tức và liên tục.

Trước khi thảo luận về luồng phát trực tuyến video, chúng ta hãy xem xét một khái niệm quan trọng: giao thức phát trực tuyến. Đây là một cách chuẩn hóa để kiểm soát việc truyền dữ liệu cho phát trực tuyến video. Các giao thức phát trực tuyến phổ biến là:

- MPEG-DASH. MPEG là viết tắt của "Moving Picture Experts Group" và DASH là viết tắt của "Dynamic Adaptive Streaming over HTTP".
- Apple HLS. HLS là viết tắt của "HTTP Live Streaming".
- Microsoft Smooth Streaming.
- Adobe HTTP Dynamic Streaming (HDS).

Bạn không cần phải hiểu đầy đủ hoặc thậm chí nhớ tên các giao thức phát trực tuyến đó vì chúng là các chi tiết cấp thấp đòi hỏi kiến thức chuyên môn. Điều quan trọng ở đây là phải hiểu rằng các giao thức phát trực tuyến khác nhau hỗ trợ các mã hóa video và trình phát lại khác nhau. Khi chúng tôi thiết kế một dịch vụ phát trực tuyến video, chúng tôi phải chọn đúng giao thức phát trực tuyến để hỗ trợ các trường hợp sử dụng của mình. Để tìm hiểu thêm về các giao thức phát trực tuyến, đây là một bài viết tuyệt vời [7].

Video được truyền trực tiếp từ CDN. Máy chủ biên gần bạn nhất sẽ phân phối video. Do đó, độ trễ rất thấp. Hình 14-7 cho thấy mức độ thiết kế cao cho việc truyền phát video.

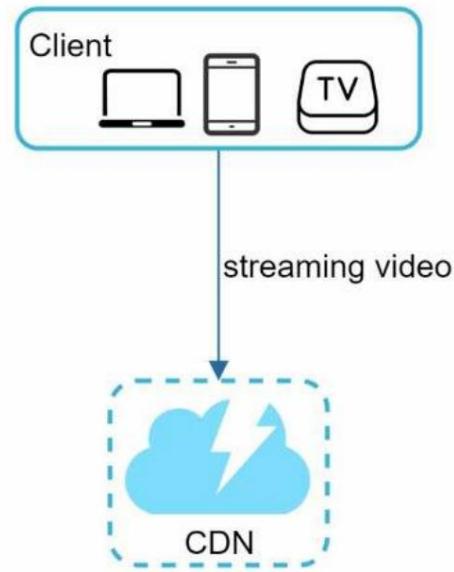


Figure 14-7

Bứ ớc 3 - Thiết kế chuyên sâu

Trong thiết kế cấp cao, toàn bộ hệ thống được chia thành hai phần: luồng tải video lên và luồng phát trực tuyến video. Trong phần này, chúng tôi sẽ tinh chỉnh cả hai luồng với các tối ưu hóa quan trọng và giới thiệu các cơ chế xử lý lỗi.

Chuyển mã video Khi bạn

quay video, thiết bị (thường là điện thoại hoặc máy ảnh) sẽ cung cấp cho tệp video một định dạng nhất định. Nếu bạn muốn video được phát mượt mà trên các thiết bị khác, video phải được mã hóa thành các bitrate và định dạng tương thích. Bitrate là tốc độ mà các bit được xử lý theo thời gian. Bitrate cao hơn thường có nghĩa là chất lượng video cao hơn. Các luồng bitrate cao cần nhiều sức mạnh xử lý hơn và tốc độ internet nhanh hơn.

Việc chuyển mã video rất quan trọng vì những lý do sau:

- Video thường chiếm nhiều dung lượng lưu trữ. Một video độ nét cao dài một giờ được ghi ở tốc độ 60 khung hình/giây có thể chiếm tới vài trăm GB dung lượng. • Nhiều thiết bị và trình duyệt chỉ hỗ trợ một số loại định dạng video nhất định. Do đó, điều quan trọng là phải mã hóa video sang các định dạng khác nhau vì lý do tương thích. • Để đảm bảo người dùng xem video chất lượng cao trong khi vẫn duy trì phát lại mượt mà, bạn nên cung cấp video có độ phân giải cao hơn cho những người dùng có băng thông mạng cao và video có độ phân giải thấp hơn cho những người dùng có băng thông thấp.
- Điều kiện mạng có thể thay đổi, đặc biệt là trên thiết bị di động. Để đảm bảo video được phát liên tục, việc chuyển đổi chất lượng video tự động hoặc thủ công dựa trên điều kiện mạng là điều cần thiết để có trải nghiệm người dùng mượt mà.

Có nhiều loại định dạng mã hóa; tuy nhiên, hầu hết chúng đều bao gồm hai phần:

- Container: Giống như một cái giỏ chứa tệp video, âm thanh và siêu dữ liệu. Bạn có thể biết định dạng container bằng phần mở rộng tệp, chẳng hạn như .avi, .mov hoặc .mp4. • Codec: Đây là các thuật toán nén và giải nén nhằm mục đích giảm kích thước video trong khi vẫn giữ nguyên chất lượng video. Các codec video được sử dụng nhiều nhất là H.264, VP9 và HEVC.

Mô hình đồ thị không chu trình có hướng (DAG)

Chuyển mã video vốn kém về mặt tính toán và thời gian. Bên cạnh đó, những người sáng tạo nội dung khác nhau có thể có các yêu cầu xử lý video khác nhau. Ví dụ, một số người sáng tạo nội dung yêu cầu hình mờ trên đầu video của họ, một số tự cung cấp hình ảnh thu nhỏ và một số tải lên video độ nét cao, trong khi những người khác thì không.

Để hỗ trợ các đường ống xử lý video khác nhau và duy trì tính song song cao, điều quan trọng là phải thêm một số mức độ trừu tượng và để các lập trình viên khách hàng xác định những tác vụ nào cần thực hiện. Ví dụ, công cụ phát video trực tuyến của Facebook sử dụng mô hình lập trình đồ thị không có chu trình có hướng (DAG), mô hình này xác định các tác vụ theo từng giai đoạn để chúng có thể được thực hiện tuần tự hoặc song song [8]. Trong thiết kế của mình, chúng tôi áp dụng một mô hình DAG tương tự để đạt được tính linh hoạt và tính song song. Hình 14-8 biểu diễn một DAG để chuyển mã video.

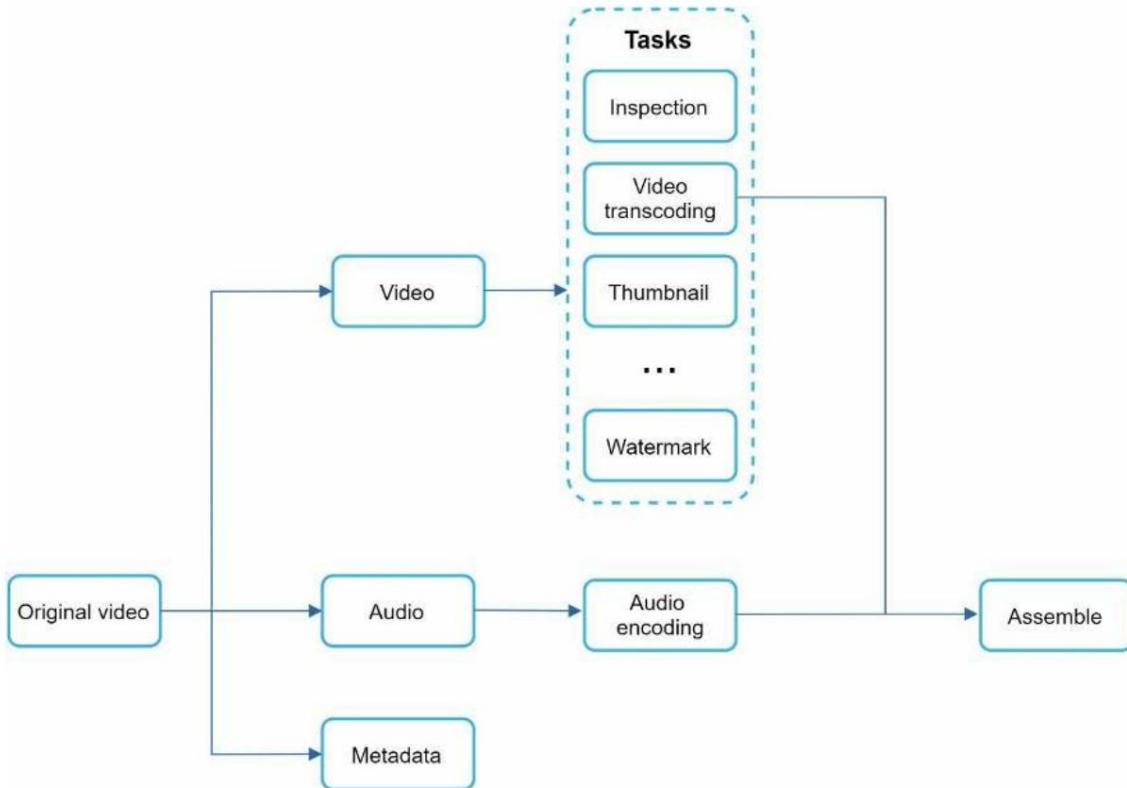


Figure 14-8

Trong Hình 14-8, video gốc được chia thành video, âm thanh và siêu dữ liệu. Sau đây là một số tác vụ có thể áp dụng trên tệp

- video:
 - Kiểm tra: Đảm bảo video có chất lượng tốt và không bị lỗi định dạng.
 - Mã hóa video: Video được chuyển đổi để hỗ trợ các độ phân giải, codec, tốc độ bit, v.v. khác nhau. Hình 14-9 hiển thị một ví dụ về tệp video được mã hóa.
 - Hình thu nhỏ: Hình thu nhỏ có thể được ứng dụng ngay khi tải lên hoặc
 - được hệ thống tự động tạo.
 - Hình mờ: Lớp phủ hình ảnh trên đầu video của bạn chứa thông tin nhận dạng về video của bạn.

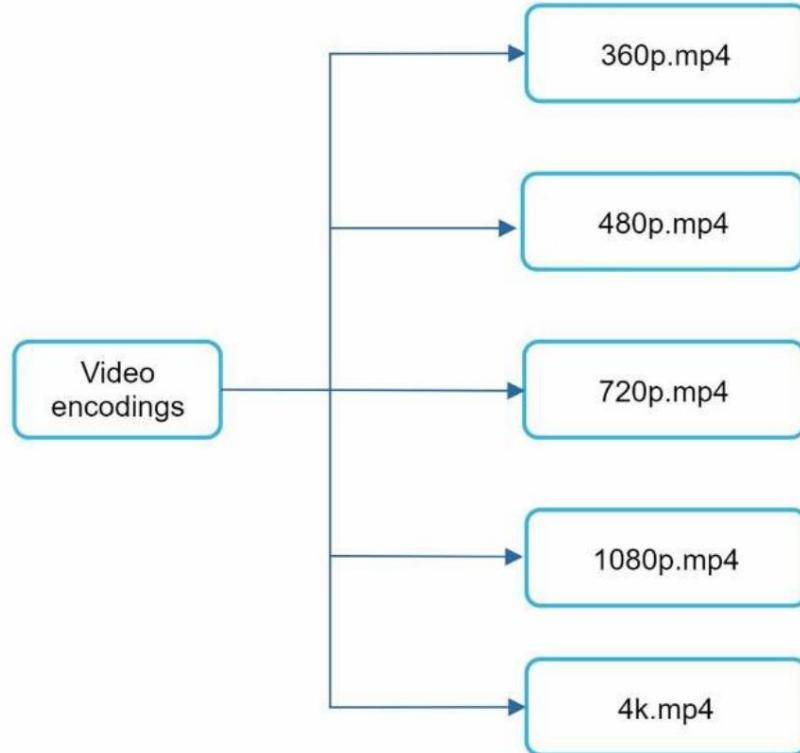


Figure 14-9

Kiến trúc chuyển mã video

chuyển mã video được đề xuất tận dụng các dịch vụ đám mây được thể hiện trong Hình 14-10.

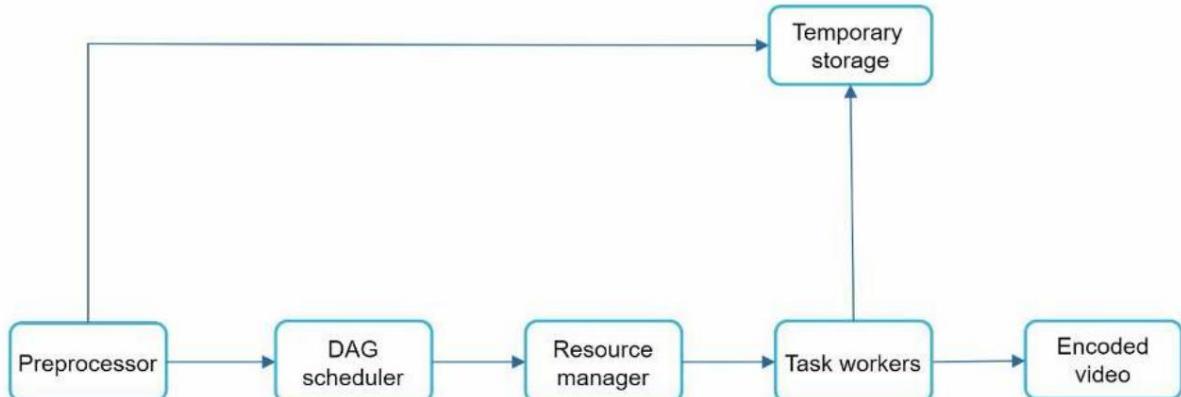


Figure 14-10

Kiến trúc có sáu thành phần chính: bộ tiền xử lý, bộ lập lịch DAG, trình quản lý tài nguyên, công nhân tác vụ, bộ lưu trữ tạm thời và video được mã hóa làm đầu ra. Chúng ta hãy xem xét kỹ từng thành phần.

Bộ tiền xử lý

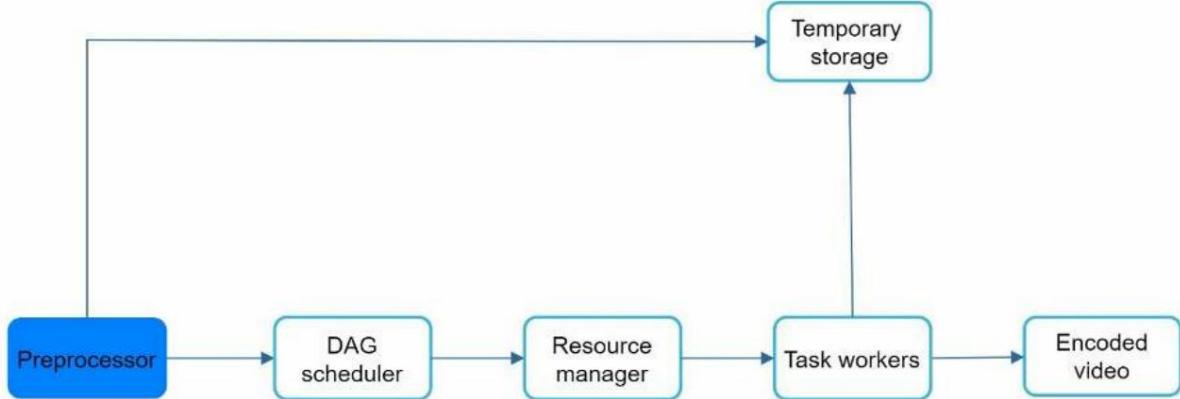


Figure 14-11

Bộ tiền xử lý có 4 trách nhiệm:

1. Phân chia video. Luồng video được chia hoặc chia nhỏ hơn nữa thành các nhóm sắp xếp hình ảnh (GOP) nhỏ hơn. GOP là một nhóm/khối các khung hình được sắp xếp theo thứ tự cụ thể. Mỗi khối là một đơn vị có thể phát độc lập, thường dài vài giây.

2. Một số thiết bị di động hoặc trình duyệt cũ có thể không hỗ trợ chia tách video. Bộ xử lý trước chia tách video theo cẩn chỉnh GOP cho các máy khách cũ.

3. Tạo DAG. Bộ xử lý tạo DAG dựa trên các tệp cấu hình mà lập trình viên máy khách viết. Hình 14-12 là biểu diễn DAG đơn giản hóa có 2 nút và 1 cạnh:



Figure 14-12

Biểu diễn DAG này được tạo từ hai tệp cấu hình bên dưới (Hình 14-13):

```

task {
    name 'download-input'
    type 'Download'
    input {
        url config.url
    }
    output { it->
        context.inputVideo = it.file
    }
    next 'transcode'
}

task {
    name 'transcode'
    type 'Transcode'
    input {
        input context.inputVideo
        config config.transConfig
    }
    output { it->
        context.file = it.outputVideo
    }
}
  
```

Figure 14-13 (source: [9])

4. Bộ nhớ đệm dữ liệu. Bộ tiền xử lý là bộ nhớ đệm cho các video được phân đoạn. Để có độ tin cậy cao hơn, bộ tiền xử lý lưu trữ GOP và siêu dữ liệu trong bộ nhớ tạm thời. Nếu mã hóa video không thành công, hệ thống có thể sử dụng dữ liệu đã lưu để thử lại các hoạt động.

Lịch trình NGÀY

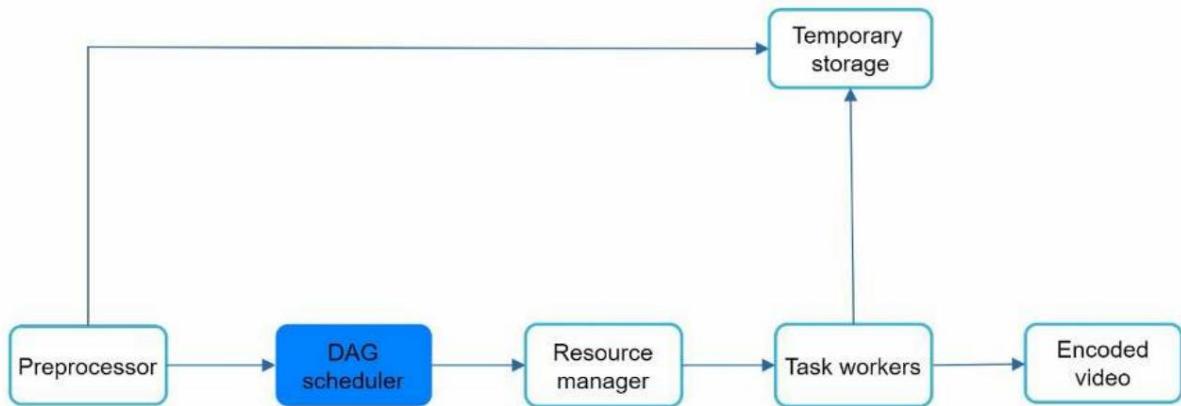


Figure 14-14

Bộ lập lịch DAG chia đồ thị DAG thành các giai đoạn tác vụ và đưa chúng vào hàng đợi tác vụ trong trình quản lý tài nguyên. Hình 14-15 cho thấy ví dụ về cách bộ lập lịch DAG hoạt động.

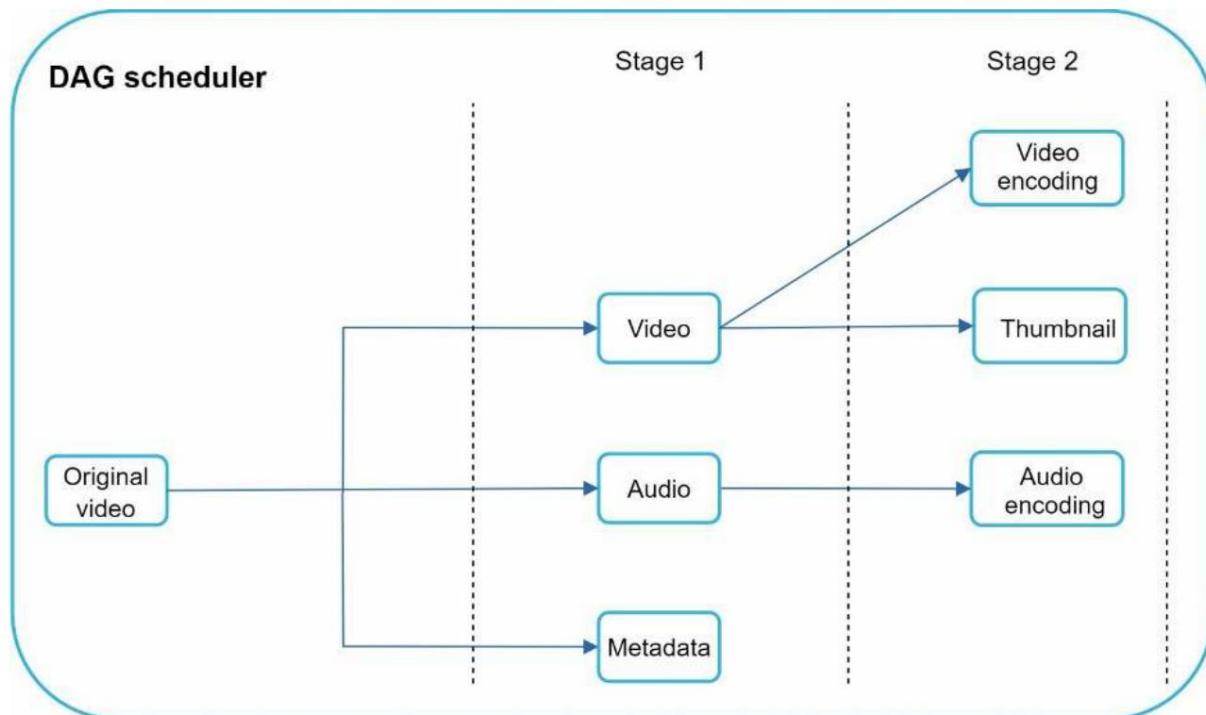


Figure 14-15

Như thể hiện trong Hình 14-15, video gốc được chia thành ba giai đoạn: Giai đoạn 1: video, âm thanh và siêu dữ liệu. Tệp video được chia thành hai tác vụ trong giai đoạn 2: mã hóa video và hình thu nhỏ. Tệp âm thanh yêu cầu mã hóa âm thanh như một phần của các tác vụ giai đoạn 2.

Quản lý tài nguyên

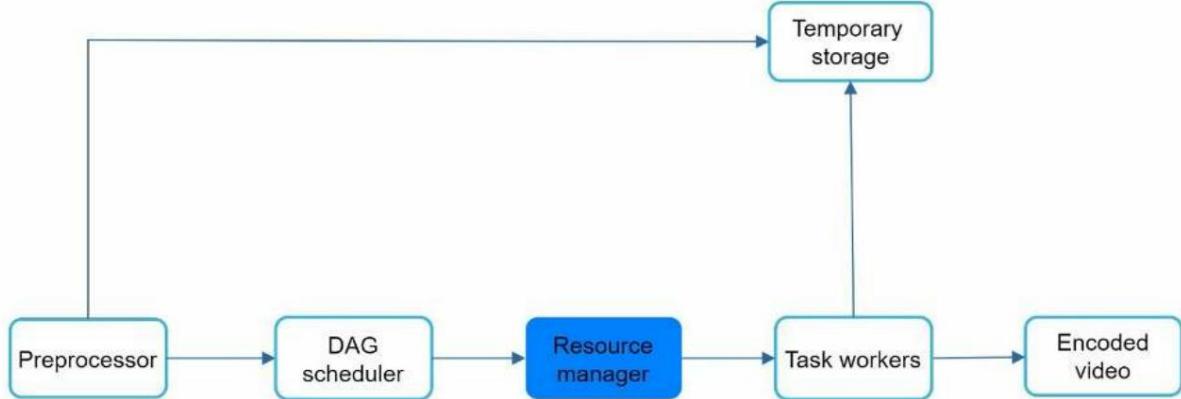


Figure 14-16

Trình quản lý tài nguyên chịu trách nhiệm quản lý hiệu quả phân bổ tài nguyên. Nó bao gồm 3 hàng đợi và một trình lập lịch tác vụ như thể hiện trong Hình 14-17.

- Hàng đợi tác vụ: Đây là hàng đợi ưu tiên chứa các tác vụ cần thực thi.
- Hàng đợi công nhân: Đây là hàng đợi ưu tiên chứa thông tin sử dụng công nhân.
- Hàng đợi đang chạy: Đây chứa thông tin về các tác vụ đang chạy và các công nhân đang chạy các tác vụ đó.
- Trình lập lịch tác vụ: Chọn tác vụ/người làm việc tối ưu và hướng dẫn người làm việc được chọn thực hiện công việc.

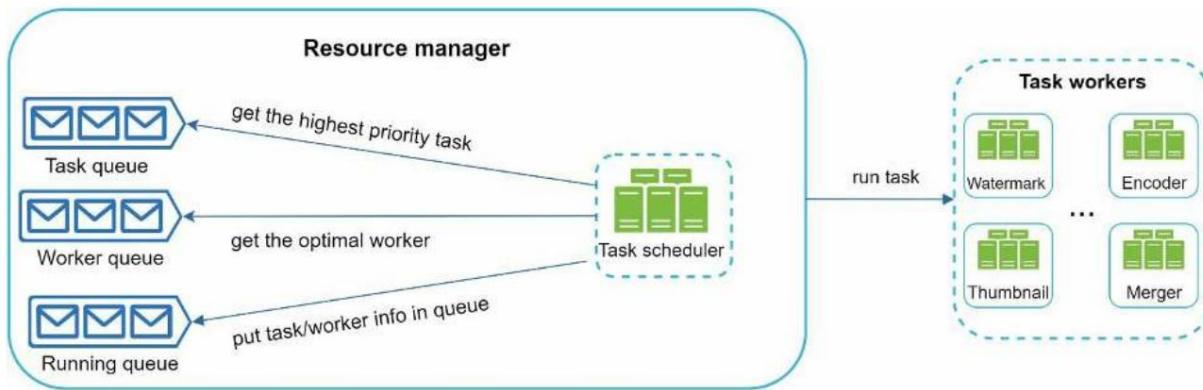


Figure 14-17

Trình quản lý tài nguyên hoạt động như sau:

- Trình lập lịch tác vụ lấy tác vụ có mức ưu tiên cao nhất từ hàng đợi tác vụ.
- Trình lập lịch tác vụ lấy công nhân tác vụ tối ưu để chạy tác vụ từ hàng đợi công nhân.
- Trình lập lịch tác vụ hướng dẫn công nhân tác vụ được chọn chạy tác vụ.
- Bộ lập lịch tác vụ liên kết thông tin tác vụ/người làm việc và đưa vào hàng đợi đang chạy.
- Bộ lập lịch tác vụ xóa công việc khỏi hàng đợi đang chạy khi công việc hoàn tất.

Người làm nhiệm vụ

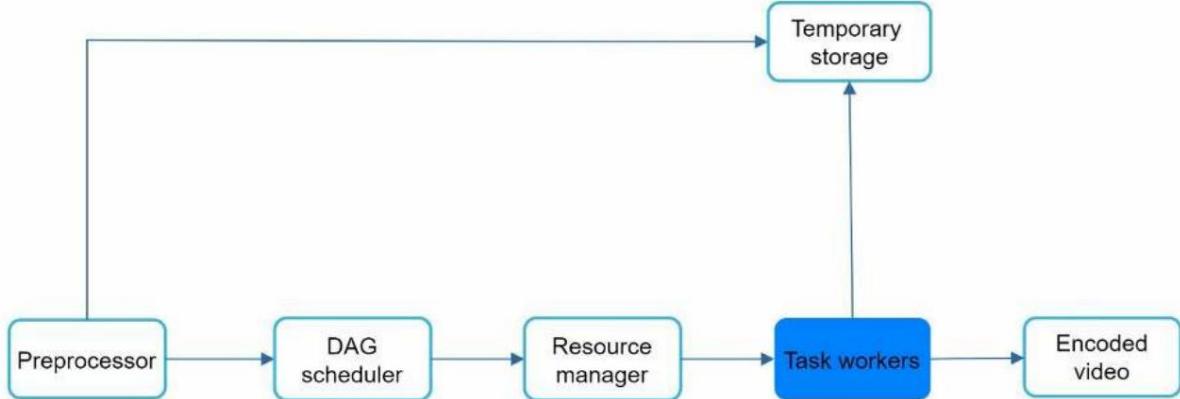


Figure 14-18

Các công nhân tác vụ chạy các tác vụ được xác định trong DAG. Các công nhân tác vụ khác nhau có thể chạy các tác vụ khác nhau như thể hiện trong Hình 14-19.

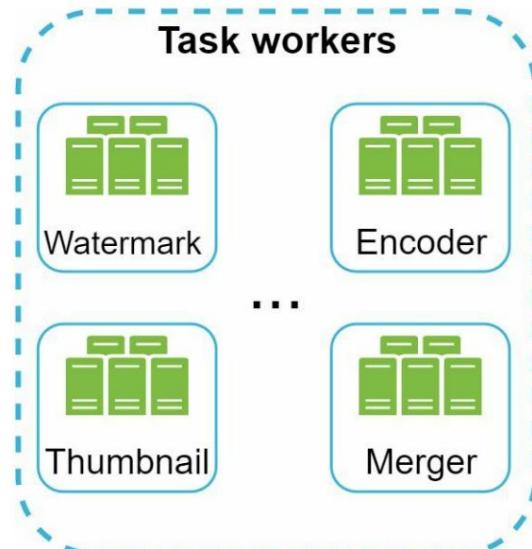


Figure 14-19

Lưu trữ tạm thời

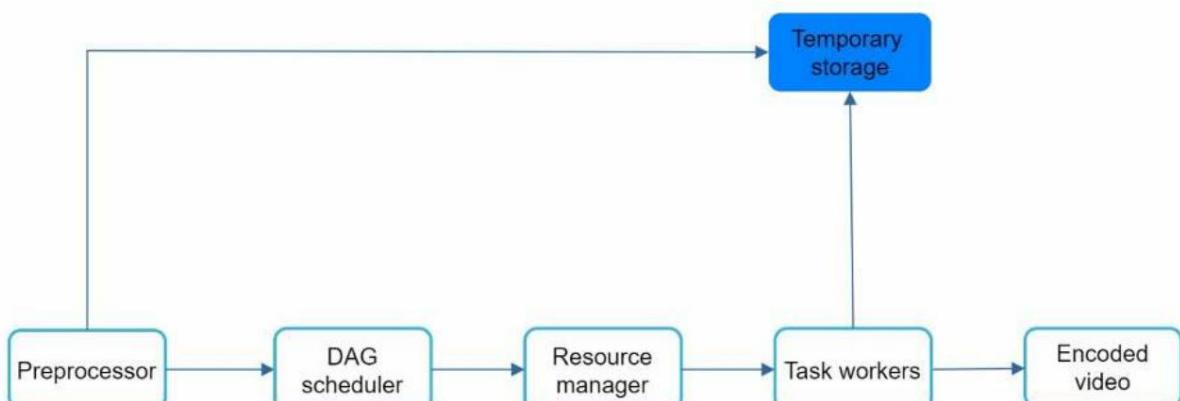
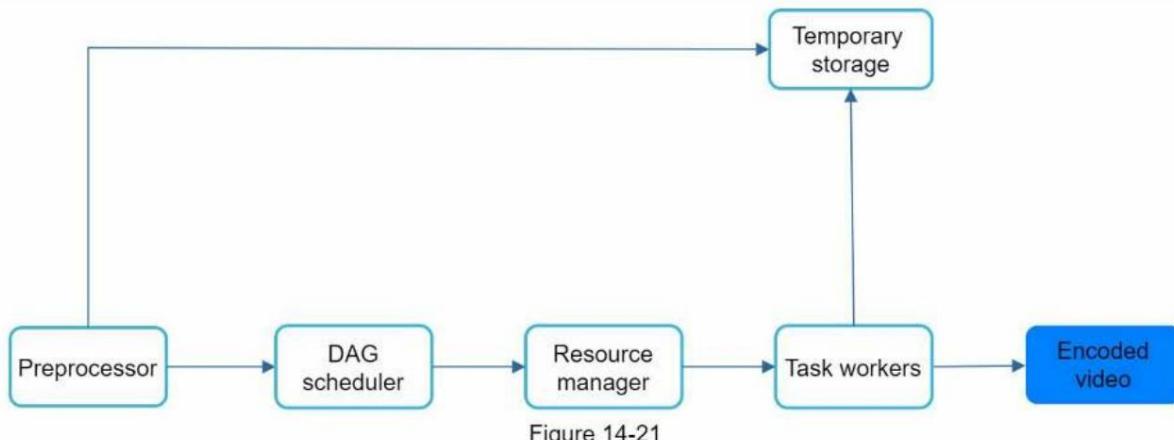


Figure 14-20

Nhiều hệ thống lưu trữ được sử dụng ở đây. Việc lựa chọn hệ thống lưu trữ phụ thuộc vào các yếu tố như loại dữ liệu, kích thước dữ liệu, tần suất truy cập, tuổi thọ dữ liệu, v.v. Ví dụ, siêu dữ liệu thư ờng xuyên

đư ợc truy cập bởi các công nhân, và kích thư ớc dữ liệu thư ờng nhỏ. Do đó, lưu trữ siêu dữ liệu trong bộ nhớ đệm là một ý tư ờng hay. Đối với dữ liệu video hoặc âm thanh, chúng tôi đặt chúng vào bộ lưu trữ blob. Dữ liệu trong bộ lưu trữ tạm thời đư ợc giải phóng sau khi quá trình xử lý video tương ứng hoàn tất.

Video đư ợc mã hóa



Video đư ợc mã hóa là đầu ra cuối cùng của đư ờng ống mã hóa. Sau đây là ví dụ về đầu ra: funny_720p.mp4.

Tối ưu hóa hệ thống Tại thời

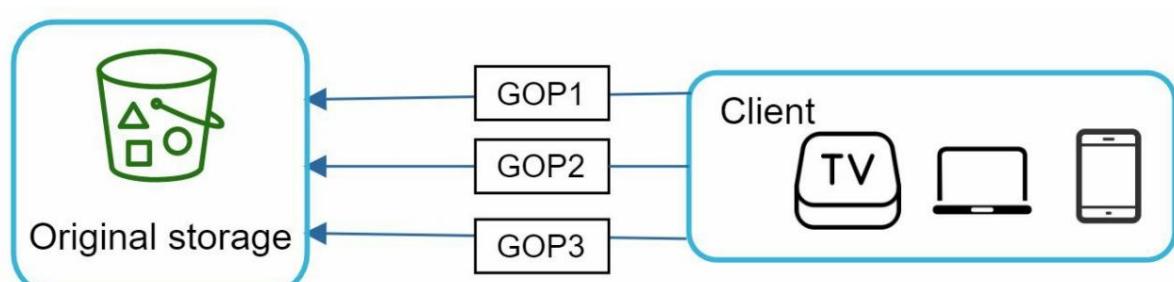
diểm này, bạn cần hiểu rõ về luồng tải video, luồng phát trực tuyến video và chuyển mã video. Tiếp theo, chúng tôi sẽ tinh chỉnh hệ thống bằng cách tối ưu hóa, bao gồm tốc độ, tính an toàn và tiết kiệm chi phí.

Tối ưu hóa tốc độ: song song hóa việc tải video lên Tải video

lên như một đơn vị toàn bộ là không hiệu quả. Chúng ta có thể chia video thành các phần nhỏ hơn bằng cách căn chỉnh GOP như thể hiện trong Hình 14-22.



Điều này cho phép tải lên có thể tiếp tục nhanh khi tải lên trước đó không thành công. Công việc chia tách tệp video bằng GOP có thể đư ợc khách hàng triển khai để cải thiện tốc độ tải lên như thể hiện trong Hình 14-23.



Tối ưu tốc độ: đặt các trung tâm tải lên gần ngư ời dùng

Một cách khác để cải thiện tốc độ tải lên là thiết lập nhiều trung tâm tải lên trên

quả địa cầu (Hình 14-24). Người dân ở Hoa Kỳ có thể tải video lên trung tâm tải lên Bắc Mỹ và người dân ở Trung Quốc có thể tải video lên trung tâm tải lên Châu Á. Để đạt được điều này, chúng tôi sử dụng CDN làm trung tâm tải lên.

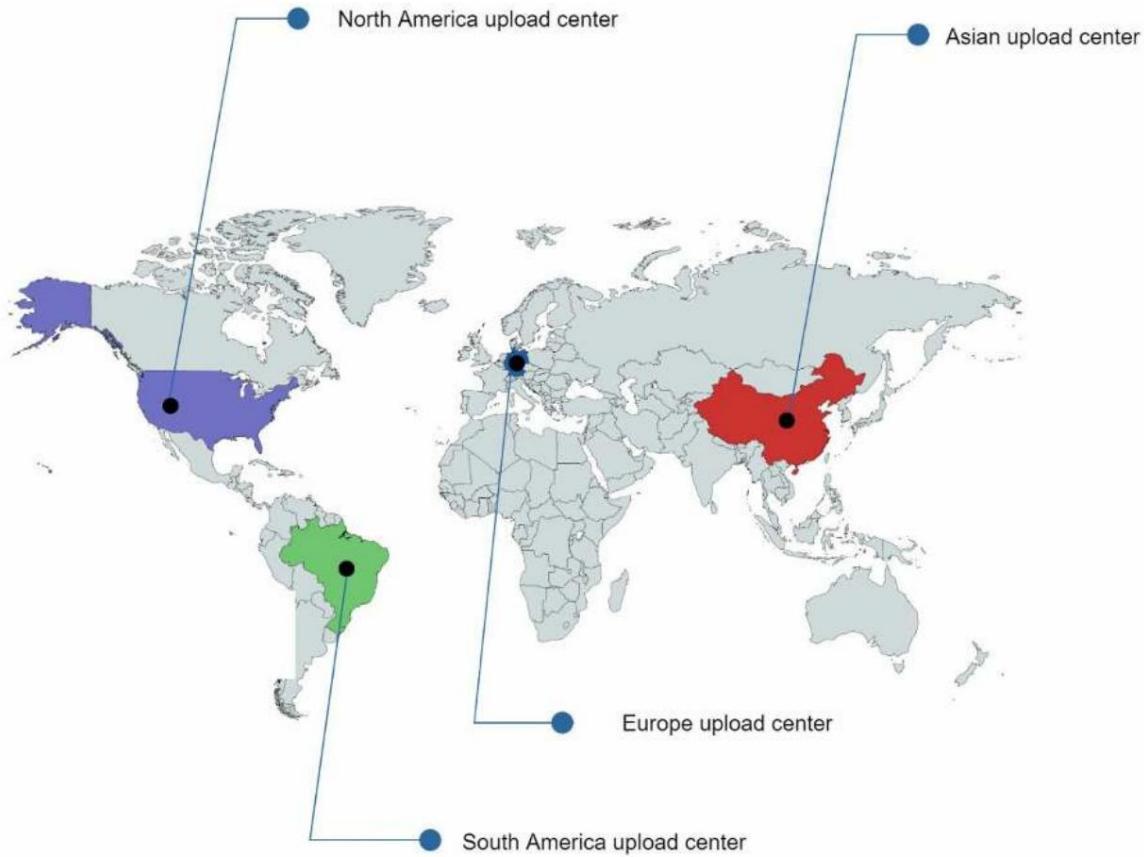


Figure 14-24

Tối ưu hóa tốc độ: song song ở mọi nơi. Để đạt được độ

trễ thấp đòi hỏi những nỗ lực nghiêm túc. Một cách tối ưu hóa khác là xây dựng một hệ thống kết nối lồng lèo và cho phép song song cao.

Thiết kế của chúng tôi cần một số sửa đổi để đạt được tính song song cao. Chúng ta hãy phóng to luồng video được chuyển từ bộ lưu trữ gốc sang CDN. Luồng được hiển thị trong Hình 14-25, cho thấy đầu ra phụ thuộc vào đầu vào của bức tranh đó. Sự phụ thuộc này làm cho tính song song trở nên khó khăn.

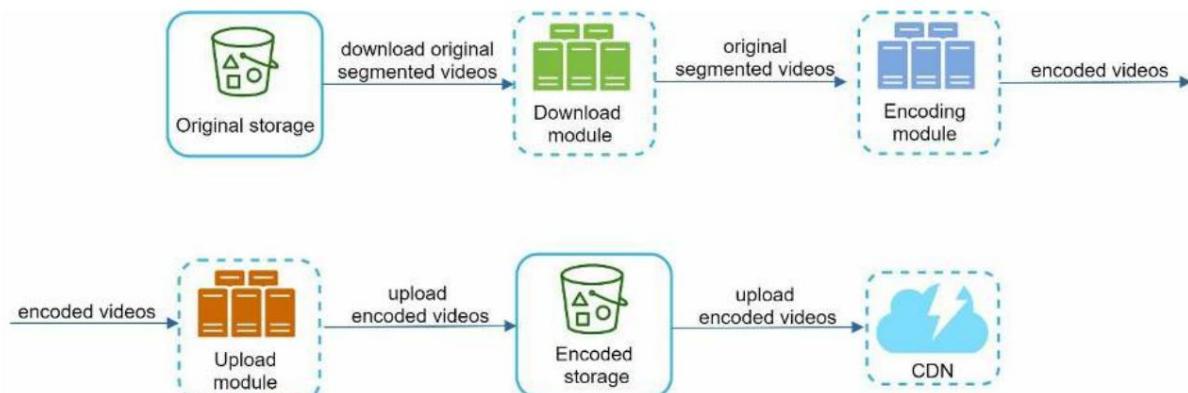


Figure 14-25

Để làm cho hệ thống được kết nối lồng lèo hơn, chúng tôi đã giới thiệu hàng đợi tin nhắn như thể hiện trong Hình

14-26. Chúng ta hãy sử dụng một ví dụ để giải thích cách hàng đợi tin nhắn làm cho hệ thống được kết nối lỏng lẻo hơn.

- Trừ ớc khi hàng đợi tin nhắn được đưa vào, mô-đun mã hóa phải chờ đầu ra của mô-đun tải xuống.
- Sau khi hàng đợi tin nhắn được giới thiệu, mô-đun mã hóa không cần phải chờ đầu ra của mô-đun tải xuống nữa. Nếu có sự kiện trong hàng đợi tin nhắn, mô-đun mã hóa có thể thực hiện các công việc đó song song.

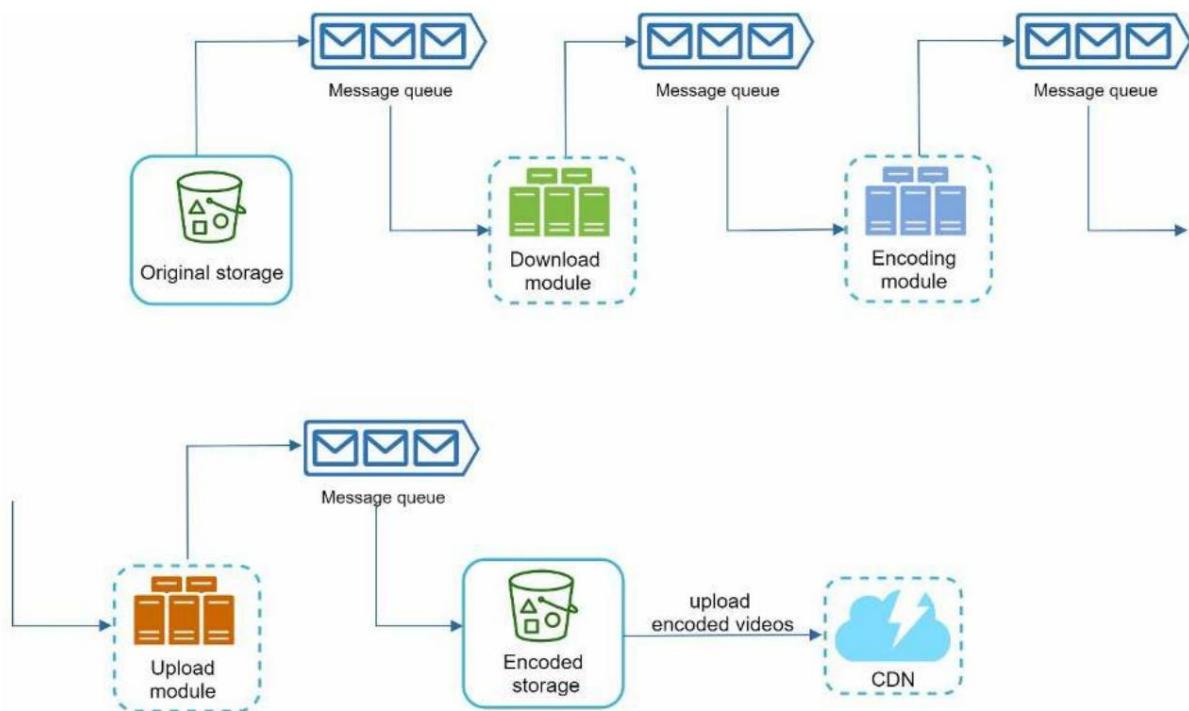


Figure 14-26

Tối ưu hóa an toàn: URL tải lên được ký trự ớc An toàn

là một trong những khía cạnh quan trọng nhất của bất kỳ sản phẩm nào. Để đảm bảo chỉ những người dùng được ủy quyền mới tải video lên đúng vị trí, chúng tôi giới thiệu URL được ký trự ớc như thể hiện trong Hình 14-27.

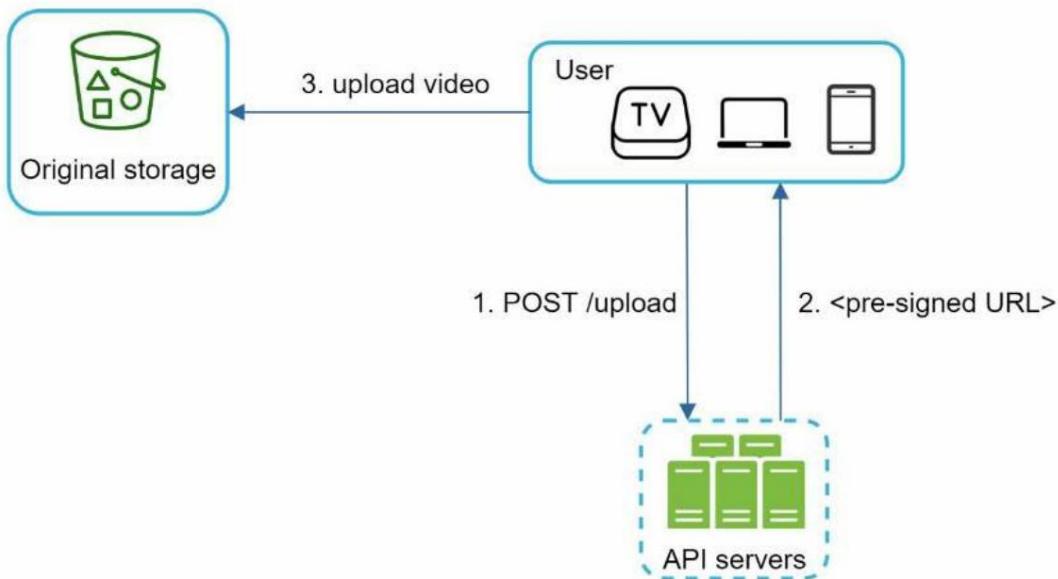


Figure 14-27

Luồng tải lên đư ợc cập nhật như sau:

1. Khách hàng thực hiện yêu cầu HTTP tới máy chủ API để lấy URL đã ký truy cập, cấp quyền truy cập cho đối tượng đư ợc xác định trong URL. Thuật ngữ URL đã ký truy cập đư ợc sử dụng bằng cách tải tệp lên Amazon S3. Các nhà cung cấp dịch vụ đám mây khác có thể sử dụng tên khác. Ví dụ, bộ lưu trữ blob của Microsoft Azure hỗ trợ cùng một tính năng, nhưng gọi là "Chữ ký truy cập đư ợc chia sẻ" [10].
2. Máy chủ API phản hồi bằng URL đã ký truy cập.
3. Sau khi nhận đư ợc phản hồi, máy khách sẽ tải video lên bằng URL đã ký truy cập.

Tối ưu hóa an toàn: bảo vệ video của bạn Nhiều

nhà sản xuất nội dung không muốn đăng video trực tuyến vì họ sợ video gốc của mình sẽ bị đánh cắp. Để bảo vệ video có bản quyền, chúng ta có thể áp dụng một trong ba tùy chọn an toàn sau:

- Hệ thống quản lý quyền kỹ thuật

số (DRM): Ba hệ thống DRM chính là Apple FairPlay, Google Widevine và Microsoft PlayReady.

- Mã hóa AES: Bạn có thể mã hóa video và định cấu hình chính sách ủy quyền. Video đư ợc mã hóa sẽ đư ợc giải mã khi phát lại. Điều này đảm bảo rằng chỉ những người dùng đư ợc ủy quyền mới có thể xem video đư ợc mã hóa.
- Hình mờ trực quan: Đây là lớp phủ hình ảnh lén trên video của bạn, chứa thông tin nhận dạng cho video của bạn. Đó có thể là logo hoặc tên công ty của bạn.

Tối ưu hóa tiết kiệm chi phí CDN

là một thành phần quan trọng trong hệ thống của chúng tôi. Nó đảm bảo phân phối video nhanh chóng trên phạm vi toàn cầu. Tuy nhiên, xét về mặt tính toán, chúng ta biết rằng CDN rất tốn kém, đặc biệt là khi kích thước dữ liệu lớn. Làm thế nào chúng ta có thể giảm chi phí?

Các nghiên cứu trước đây cho thấy luồng video trên YouTube tuân theo phân phối đuôi dài [11] [12]. Điều này có nghĩa là một số video phổ biến đư ợc truy cập thường xuyên như nhiều video khác có ít hoặc không có người xem. Dựa trên quan sát này, chúng tôi triển khai một số

tối ưu hóa:

1. Chỉ phục vụ các video phổ biến nhất từ CDN và các video khác từ dung lư ợng cao của chúng tôi

máy chủ lưu trữ video (Hình 14-28).

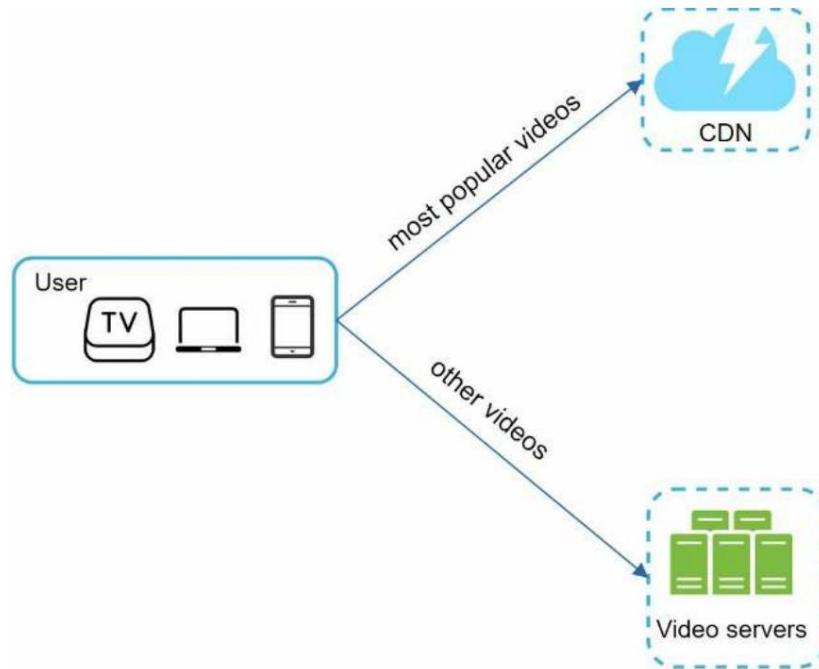


Figure 14-28

2. Đối với nội dung ít phổ biến hơn, chúng tôi có thể không cần lưu trữ nhiều phiên bản video đư ợc mã hóa. Các video ngắn có thể đư ợc mã hóa theo yêu cầu.

3. Một số video chỉ phổ biến ở một số khu vực nhất định. Không cần phải phân phối những video này đến các khu vực khác.

4. Xây dựng CDN của riêng bạn như Netflix và hợp tác với Nhà cung cấp dịch vụ Internet (ISP).

Xây dựng CDN của bạn là một dự án không lồ; tuy nhiên, điều này có thể có ý nghĩa đối với các công ty phát trực tuyến lớn. Một ISP có thể là Comcast, AT&T, Verizon hoặc các nhà cung cấp internet khác. Các ISP nằm trên khắp thế giới và gần với người dùng. Bằng cách hợp tác với các ISP, bạn có thể cải thiện trải nghiệm xem và giảm phí băng thông.

Tất cả các tối ưu hóa đó đều dựa trên mức độ phổ biến của nội dung, mẫu truy cập của người dùng, kích thước video, v.v. Điều quan trọng là phải phân tích các mẫu xem lịch sử trước khi thực hiện bất kỳ tối ưu hóa nào. Sau đây là một số bài viết thú vị về chủ đề này: [12] [13].

Xử lý lỗi Đối với một

hệ thống quy mô lớn, lỗi hệ thống là không thể tránh khỏi. Để xây dựng một hệ thống có khả năng chịu lỗi cao, chúng ta phải xử lý lỗi một cách khéo léo và phục hồi nhanh chóng. Có hai loại lỗi:

- Lỗi có thể phục hồi. Đối với các lỗi có thể phục hồi như phân đoạn video không chuyển mã đư ợc, ý tưởng chung là thử lại thao tác một vài lần. Nếu tác vụ tiếp tục không thành công và hệ thống cho rằng không thể phục hồi đư ợc, hệ thống sẽ trả về mã lỗi thích hợp cho máy khách. • Lỗi không thể phục hồi.
- Đối với các lỗi không thể phục hồi như định dạng video không đúng, hệ thống sẽ dùng các tác vụ đang chạy liên quan đến video và trả về mã lỗi thích hợp cho máy khách.

Các lỗi điển hình cho từng thành phần hệ thống đư ợc đề cập trong số tay hướng dẫn sau: • Lỗi tải lên: thử lại một vài lần.

- Lỗi chia tách video: nếu các phiên bản cũ hơn của máy khách không thể chia tách video theo cản chỉnh GOP, toàn bộ video sẽ được chuyển đến máy chủ. Công việc chia tách video được thực hiện ở phía máy chủ. • Lỗi chuyển mã: thử lại. • Lỗi tiền xử lý: tạo lại sơ đồ DAG. • Lỗi trình lập lịch DAG: lên lịch lại tác vụ.

- Hàng đợi quản lý tài nguyên bị sập: sử dụng bản sao. • Công nhân tác vụ bị sập: thử lại tác vụ trên một công nhân mới. • Máy chủ API bị sập: Máy chủ API không có trạng thái nên các yêu cầu sẽ được chuyển hướng đến một API khác máy chủ.

- Máy chủ bộ đệm siêu dữ liệu ngừng hoạt động: dữ liệu được sao chép nhiều lần. Nếu một nút ngừng hoạt động, bạn vẫn có thể truy cập các nút khác để lấy dữ liệu. Chúng tôi có thể đưa máy chủ bộ đệm mới lên để thay thế máy chủ đã ngừng hoạt động. •

Máy chủ DB siêu dữ liệu ngừng hoạt động:

- Chủ nhân đã xuống. Nếu chủ nhân đã xuống, hãy thăng chức cho một trong những nô lệ để làm chủ nhân mới bậc thầy.
- Slave bị hỏng. Nếu một slave bị hỏng, bạn có thể sử dụng một slave khác để đọc và đưa một máy chủ cơ sở dữ liệu khác lên để thay thế cho slave đã hỏng.

Bài 4 - Tóm tắt

Trong chương này, chúng tôi đã trình bày thiết kế kiến trúc cho các dịch vụ phát trực tuyến video như YouTube. Nếu có thêm thời gian vào cuối buổi phỏng vấn, sau đây là một số điểm bổ sung:

- Mở rộng tầng API: Vì máy chủ API không có trạng thái nên có thể dễ dàng mở rộng tầng API

theo chiều ngang. • Mở rộng cơ sở dữ liệu: Bạn có thể nói về sao chép và phân mảnh cơ sở dữ liệu. • Phát trực tiếp: Nó đề cập đến quy trình ghi lại và phát video theo thời gian thực. Mặc dù hệ thống của chúng tôi không được thiết kế riêng cho phát trực tiếp, phát trực tiếp và không phát trực tiếp có một số điểm tương đồng: cả hai đều yêu cầu tải lên, mã hóa và phát trực tuyến. Những điểm khác biệt đáng chú ý là:

- Phát trực tiếp có yêu cầu về độ trễ cao hơn, do đó có thể cần một giao thức phát trực tiếp khác.
- Phát trực tiếp có yêu cầu thấp hơn về tính song song vì các khôi dữ liệu nhỏ đã được xử lý theo thời gian thực. • Phát trực tiếp yêu cầu các bộ xử lý lõi khác nhau. Bất kỳ xử lý lõi nào mất quá nhiều thời gian đều không được chấp nhận. •

Gỡ video: Các video vi phạm bản quyền, khiêu dâm hoặc các hành vi bất hợp pháp khác sẽ bị xóa. Một số có thể được hệ thống phát hiện trong quá trình tải lên, trong khi những video khác có thể được phát hiện thông qua việc người dùng đánh dấu.

Xin chúc mừng vì đã đi được đến đây! Böyle giờ hãy tự khen mình nhé. Làm tốt lắm!

Tài liệu tham khảo

[1] YouTube theo số liệu: <https://www.omnicoreagency.com/youtube-statistics/> [2] Nhân khẩu học _____

YouTube năm 2019: <https://blog.hubspot.com/marketing/youtube-demographics> [3] Giá Cloudfront: <https://aws.amazon.com/cloudfront/pricing/> [4] Netflix trên AWS: <https://aws.amazon.com/solutions/case-studies/netflix/> [5] Trang chủ Akamai: <https://www.akamai.com/> [6] Đổi tư ợng nhị phân lớn: https://en.wikipedia.org/wiki/Binary_large_object [7] Đây là những gì bạn cần biết về giao thức phát trực tuyến: <https://www.dacast.com/blog/streaming-protocols/> [8] SVE: Xử lý video phân tán ở quy mô Facebook: <https://www.cs.princeton.edu/~wlloyd/papers/sve-sosp17.pdf> [9] Kiến trúc xử lý video Weibo (trong Tiếng Trung): <https://www.upyun.com/opentalk/399.html> [10] Ủy quyền truy cập với chữ ký truy cập được chia sẻ: <https://docs.microsoft.com/en-us/rest/api/storageservices/delegate-access-with-shared-access-signature> [11] _____

Bài nói chuyện về khả năng mở rộng của YouTube do nhân viên

YouTube thời kỳ đầu thực hiện: <https://www.youtube.com/watch?v=w5WVu624fY8> _____

[12] Hiểu các đặc điểm của việc chia sẻ video ngắn trên internet: Một nghiên cứu đo lường dựa trên youtube. <https://arxiv.org/pdf/0707.3670.pdf> [13] Mức độ phổ biến _____

của nội dung cho Open Connect: <https://netflixtechblog.com/content-popularity-for-open-connect-b86d56f613b> _____

CHƯƠNG 15: THIẾT KẾ GOOGLE DRIVE

Trong những năm gần đây, các dịch vụ lưu trữ đám mây như Google Drive, Dropbox, Microsoft OneDrive và Apple iCloud đã trở nên rất phổ biến. Trong chương này, bạn được yêu cầu thiết kế Google Drive.

Hãy dành chút thời gian để hiểu Google Drive trước khi bắt đầu thiết kế. Google Drive là dịch vụ lưu trữ và đồng bộ hóa tệp giúp bạn lưu trữ tài liệu, ảnh, video và các tệp khác trên đám mây. Bạn có thể truy cập tệp của mình từ bất kỳ máy tính, điện thoại thông minh và máy tính bảng nào. Bạn có thể dễ dàng chia sẻ các tệp đó với bạn bè, gia đình và đồng nghiệp [1]. Hình 15-1 và 15-2 cho thấy Google Drive trông như thế nào trên trình duyệt và ứng dụng di động.

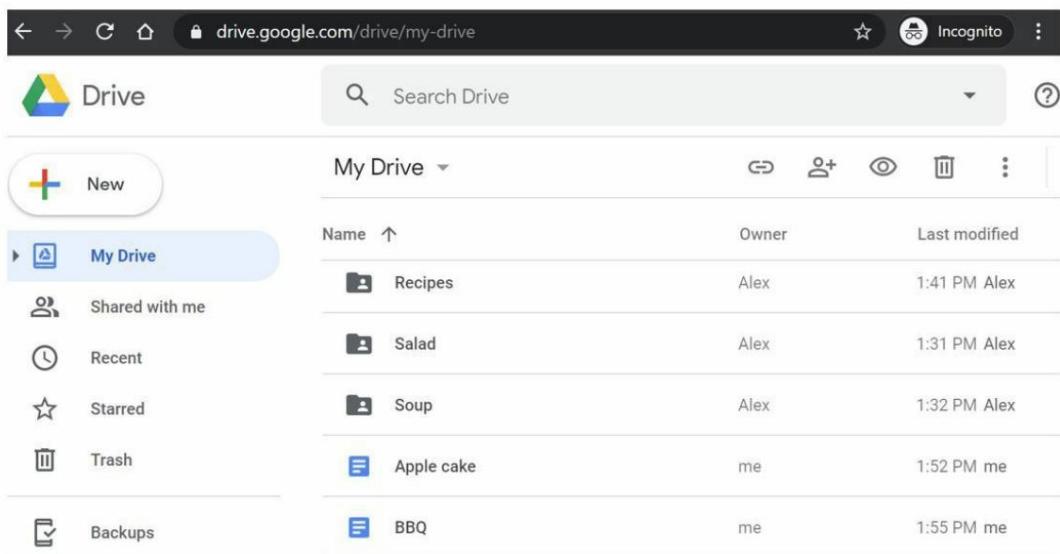


Figure 15-1

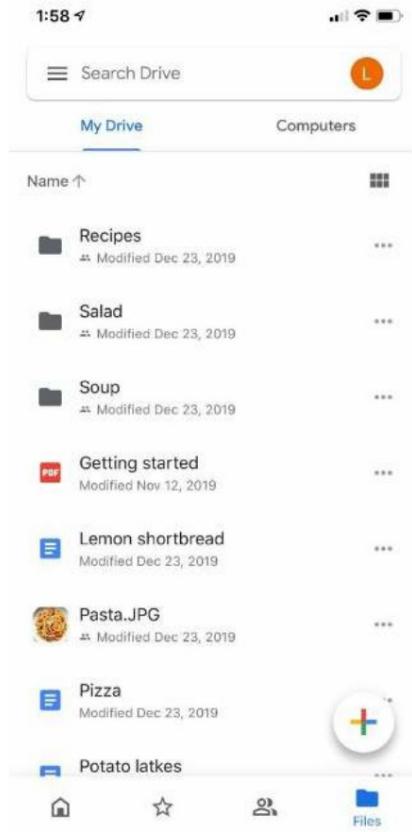


Figure 15-2

Bút ớc 1 - Hiểu rõ vấn đề và thiết lập phạm vi thiết kế Thiết kế Google Drive là một dự án lớn, vì vậy, điều quan trọng là phải đặt câu hỏi để thu hẹp phạm vi.

Ứng viên: Những đặc điểm quan trọng nhất là gì?

Người phỏng vấn: Tải lên và tải xuống tệp, đồng bộ hóa tệp và thông báo.

Ứng viên: Đây là ứng dụng di động, ứng dụng web hay cả hai?

Người phỏng vấn: Cả hai.

Ứng viên: Định dạng tập tin nào được hỗ trợ?

Người phỏng vấn: Tất cả các định dạng.

Ứng viên: Các tập tin có cần phải được mã hóa không?

Phỏng vấn: Có, tất cả các tập tin trong bộ lưu trữ phải được mã hóa.

Ứng viên: Có giới hạn kích thước tập tin không?

Phỏng vấn: Có, tập tin phải có dung lượng 10 GB hoặc nhỏ hơn.

Ứng viên: Sản phẩm có bao nhiêu người dùng?

Người phỏng vấn: 10M DAU.

Trong chương này, chúng tôi tập trung vào các tính năng sau:

Thêm tệp. Cách dễ nhất để thêm tệp là kéo và thả tệp vào Google Drive.

• Tải xuống tệp.

• Đồng bộ hóa tệp trên nhiều thiết bị. Khi một tệp được thêm vào một thiết bị, nó sẽ tự động được đồng bộ hóa với các thiết bị khác.

• Xem các bản sửa đổi tệp.

• Chia sẻ tệp với bạn bè, gia đình và đồng nghiệp.

• Gửi thông báo khi tệp được chỉnh sửa, xóa hoặc chia sẻ với bạn.

Các tính năng không được thảo luận trong chương này bao

gồm:

• Chỉnh sửa và cộng tác Google Doc. Google Doc cho phép nhiều người chỉnh sửa cùng một tài liệu cùng lúc.

Điều này nằm ngoài phạm vi thiết kế của chúng tôi.

Ngoài việc làm rõ các yêu cầu, điều quan trọng là phải hiểu các yêu cầu không chức năng:

• Độ tin cậy. Độ tin cậy cực kỳ

quan trọng đối với hệ thống lưu trữ. Mất dữ liệu là không thể chấp nhận được.

• Tốc độ đồng bộ nhanh.

Nếu quá trình

đồng bộ tệp mất quá nhiều thời gian, người dùng sẽ mất kiên nhẫn và từ bỏ sản phẩm.

• Sử dụng băng thông.

Nếu một sản phẩm chiếm

nhiều băng thông mạng không cần thiết, người dùng sẽ không hài lòng, đặc biệt là khi họ sử dụng gói dữ liệu di động.

• Khả năng mở rộng.

Hệ thống phải có khả năng xử lý lưu lượng truy cập

lớn.

• Tính khả dụng cao.

Người dùng vẫn có thể sử dụng hệ thống khi một số máy chủ ngoại tuyến,

chạm lại hoặc gặp lỗi mạng không mong muốn.

Ước tính sơ bộ

• Giả sử ứng dụng có 50 triệu người dùng đã đăng ký và 10 triệu DAU.

• Người dùng có 10 GB dung

lượng trống.

• Giả sử người dùng tải

tải lên 2 tệp mỗi ngày. Kích thước tệp trung bình là 500 KB.

• Tỷ lệ đọc ghi là 1:1.

- Tổng dung lư ợng đư ợc phân bō: 50 triệu * 10 GB = 500 Petabyte • QPS cho API tải lên: 10 triệu * 2 lần tải lên / 24 giờ / 3600 giây = ~ 240 • QPS định = QPS * 2 = 480

BƯỚC 2 - ĐỀ XUẤT THIẾT KẾ CẤP CAO VÀ NHẬN ĐƯỢC SỰ ĐỒNG THUẬN

Thay vì hiển thị sơ đồ thiết kế cấp cao ngay từ đầu, chúng tôi sẽ sử dụng một cách tiếp cận hơi khác. Chúng tôi sẽ bắt đầu bằng một điều đơn giản: xây dựng mọi thứ trong một máy chủ duy nhất. Sau đó, dần dần mở rộng quy mô để hỗ trợ hàng triệu người dùng. Bằng cách thực hiện bài tập này, bạn sẽ làm mới lại trí nhớ của mình về một số chủ đề quan trọng được đề cập trong sách.

Chúng ta hãy bắt đầu với một thiết lập máy chủ duy nhất như được liệt kê dưới đây:

- Một máy chủ web để tải lên và tải xuống các tệp.
- Một cơ sở dữ liệu để theo dõi siêu dữ liệu như dữ liệu người dùng, thông tin đăng nhập, thông tin tệp, v.v.
- Một hệ thống lưu trữ để lưu trữ các tệp. Chúng tôi phân bổ 1TB dung lượng lưu trữ để lưu trữ các tệp.

Chúng tôi dành vài giờ để thiết lập máy chủ web Apache, cơ sở dữ liệu MySQL và thư mục có tên là drive/ làm thư mục gốc để lưu trữ các tệp đã tải lên. Trong thư mục drive/, có một danh sách các thư mục, được gọi là không gian tên. Mỗi không gian tên chứa tất cả các tệp đã tải lên cho người dùng đó. Tên tệp trên máy chủ được giữ nguyên như tên tệp gốc. Mỗi tệp hoặc thư mục có thẻ được xác định duy nhất bằng cách nối không gian tên và đường dẫn tương đối.

Hình 15-3 cho thấy ví dụ về cách thư mục /drive trông như thế nào ở phía bên trái và chế độ xem mở rộng của nó ở phía bên phải.

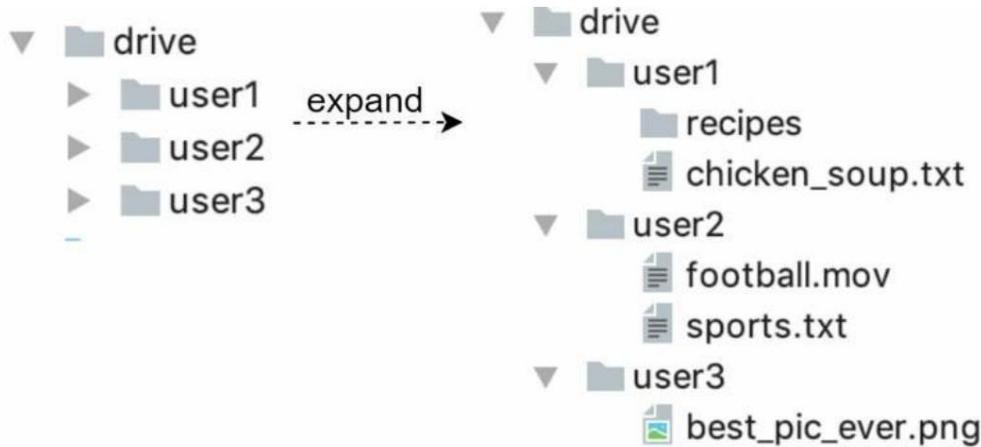


Figure 15-3

API

API trông như thế nào? Về cơ bản, chúng ta cần 3 API: tải tệp lên, tải tệp xuống và nhận bản sửa đổi tệp.

1. Tải tệp lên Google Drive

Có hai loại tải lên được hỗ trợ:

- Tải lên đơn giản.

Sử dụng loại tải lên này khi kích thước tệp nhỏ.

• Tải lên có thể tiếp tục.

Sử dụng loại tải lên này khi kích thước tệp lớn và có khả năng cao bị gián đoạn mạng.

Sau đây là một ví dụ về API tải lên có thể tiếp tục:

<https://api.example.com/files/upload?uploadType=resumable>

Tham số:

- uploadType=resumable
- data: Tệp cục bộ sẽ được tải lên.

Tải lên có thể tiếp tục đư ợc thực hiện theo 3 bư ớc sau [2]: • Gửi yêu cầu ban đầu để lấy URL có thể tiếp tục. • Tải dữ liệu lên và theo dõi trạng thái tải lên. • Nếu quá trình tải lên bị gián đoạn, hãy tiếp tục tải lên.

2. Tải xuống tệp từ API ví dụ của Google

Drive: <https://api.example.com/files/download>

Tham số:

- đường dẫn: đường dẫn tải tệp tin.

Ví dụ tham số: {

```
"đường dẫn": "/recipes/soup/best_soup.txt"  
}
```

3. Nhận bản sửa đổi tệp

API ví dụ: https://api.example.com/files/list_revisions

Tham số:

- path: Đường dẫn đến tệp bạn muốn lấy lịch sử sửa đổi.
- limit: Số lượng sửa đổi tối đa trả về.

Ví dụ tham số: {

```
"đường dẫn": "/recipes/soup/best_soup.txt",  
"giới hạn": 20  
}
```

Tất cả các API đều yêu cầu xác thực ngư ời dùng và sử dụng HTTPS. Secure Sockets Layer (SSL) bảo vệ việc truyền dữ liệu giữa máy khách và máy chủ phụ trợ.

Di chuyển khỏi máy chủ đơn Khi tải lên

nhiều tệp hơn, cuối cùng bạn sẽ nhận đư ợc cảnh báo không gian đầy như thể hiện trong Hình 15-4.



Figure 15-4

Chỉ còn 10 MB dung lư ợng lưu trữ! Đây là tru ờng hợp khẩn cấp vì ngư ời dùng không thể tải tệp lên nữa. Giải pháp đầu tiên xuất hiện trong đầu là phân mảnh dữ liệu, do đó dữ liệu đư ợc lưu trữ trên nhiều máy chủ lưu trữ. Hình 15-5 cho thấy một ví dụ về phân mảnh dựa trên user_id .

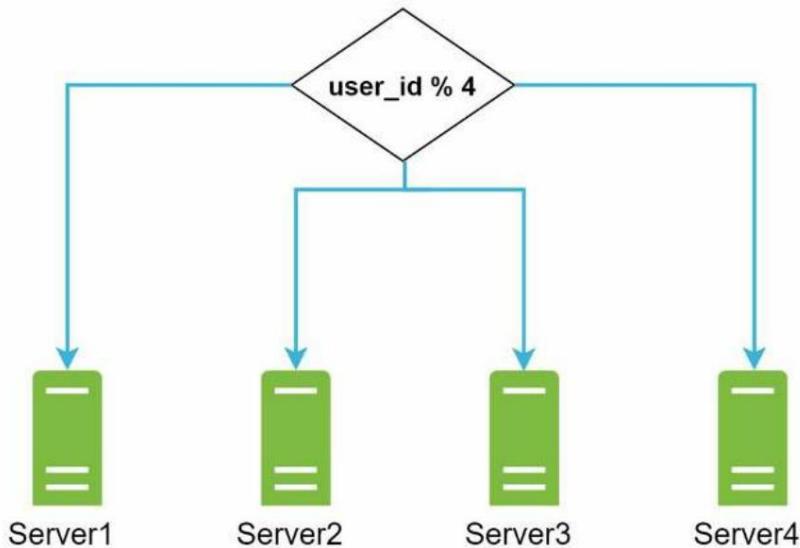


Figure 15-5

Bạn thức trăng đêm để thiết lập phân mảnh cơ sở dữ liệu và theo dõi chặt chẽ. Mọi thứ lại hoạt động trơn tru. Bạn đã dập tắt đư ợc đám cháy, nhưng bạn vẫn lo lắng về khả năng mất dữ liệu trong trường hợp máy chủ lưu trữ ngừng hoạt động. Bạn hỏi xung quanh và người bạn chuyên gia về backend Frank nói với bạn rằng nhiều công ty hàng đầu như Netflix và Airbnb sử dụng Amazon S3 để lưu trữ.

"Amazon Simple Storage Service (Amazon S3) là dịch vụ lưu trữ đối tượng cung cấp khả năng mở rộng, tính khả dụng của dữ liệu, bảo mật và hiệu suất hàng đầu trong ngành" [3]. Bạn quyết định thực hiện một số nghiên cứu để xem liệu nó có phù hợp hay không.

Sau khi đọc nhiều, bạn sẽ hiểu rõ về hệ thống lưu trữ S3 và quyết định lưu trữ tệp trong S3. Amazon S3 hỗ trợ sao chép cùng vùng và liên vùng. Vùng là khu vực địa lý nơi Amazon web services (AWS) có trung tâm dữ liệu. Như thể hiện trong Hình 15-6, dữ liệu có thể được sao chép trên cùng vùng (bên trái) và liên vùng (bên phải).

Các tệp dự phòng được lưu trữ ở nhiều vùng để bảo vệ dữ liệu khỏi bị mất và đảm bảo tính khả dụng. Một thùng giống như một thư mục trong hệ thống tệp.

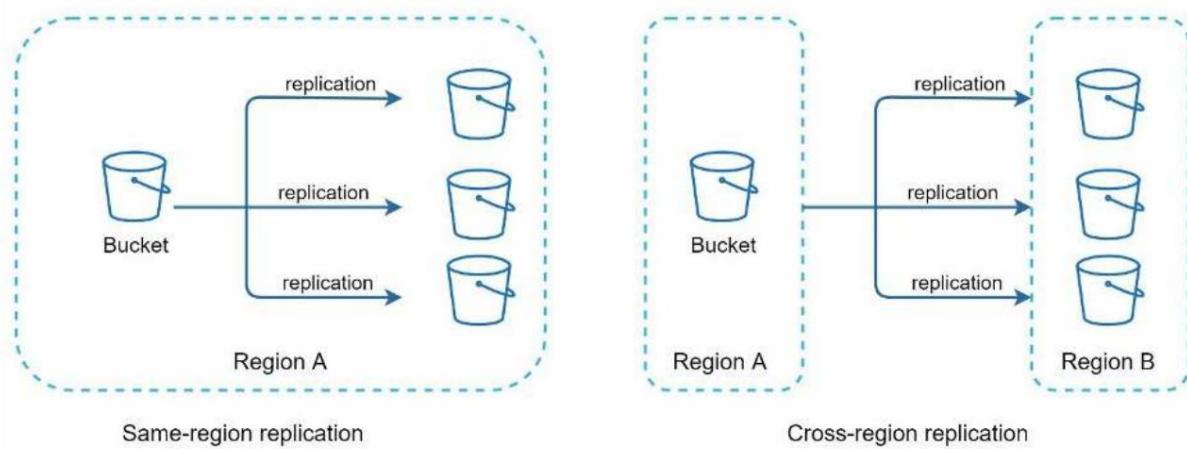


Figure 15-6

Sau khi đưa các tập tin vào S3, cuối cùng bạn có thể có một đêm ngon giấc mà không phải lo lắng về việc mất dữ liệu. Để ngăn chặn các vấn đề tương tự xảy ra trong tương lai, bạn quyết định nghiên cứu thêm về các lĩnh vực bạn có thể cải thiện. Sau đây là một số lĩnh vực bạn tìm thấy:

- Bộ cân bằng tài: Thêm bộ cân bằng tài để phân phối lưu lượng mạng. Bộ cân bằng tài đảm bảo

phân phối lưu lượng đều đặn và nếu máy chủ web ngừng hoạt động, nó sẽ phân phối lại lưu lượng. • Máy chủ web: Sau khi bộ cân bằng tải được thêm vào, có thể dễ dàng thêm/xóa nhiều máy chủ web hơn, tùy thuộc vào tải lưu lượng. • Cơ sở dữ liệu

siêu dữ liệu: Di chuyển cơ sở dữ liệu ra khỏi máy chủ để tránh điểm lỗi đơn.

Trong khi đó, hãy thiết lập sao chép và phân mảnh dữ liệu để đáp ứng các yêu cầu về tính khả dụng và khả năng mở rộng. •

Lưu trữ tệp: Amazon S3 được sử dụng để lưu trữ tệp. Để đảm bảo tính khả dụng và độ bền, các tệp được sao chép ở hai vùng địa lý riêng biệt.

Sau khi áp dụng các cải tiến trên, bạn đã tách thành công máy chủ web, cơ sở dữ liệu siêu dữ liệu và lưu trữ tệp khỏi một máy chủ duy nhất. Thiết kế được cập nhật được hiển thị trong Hình 15-7.

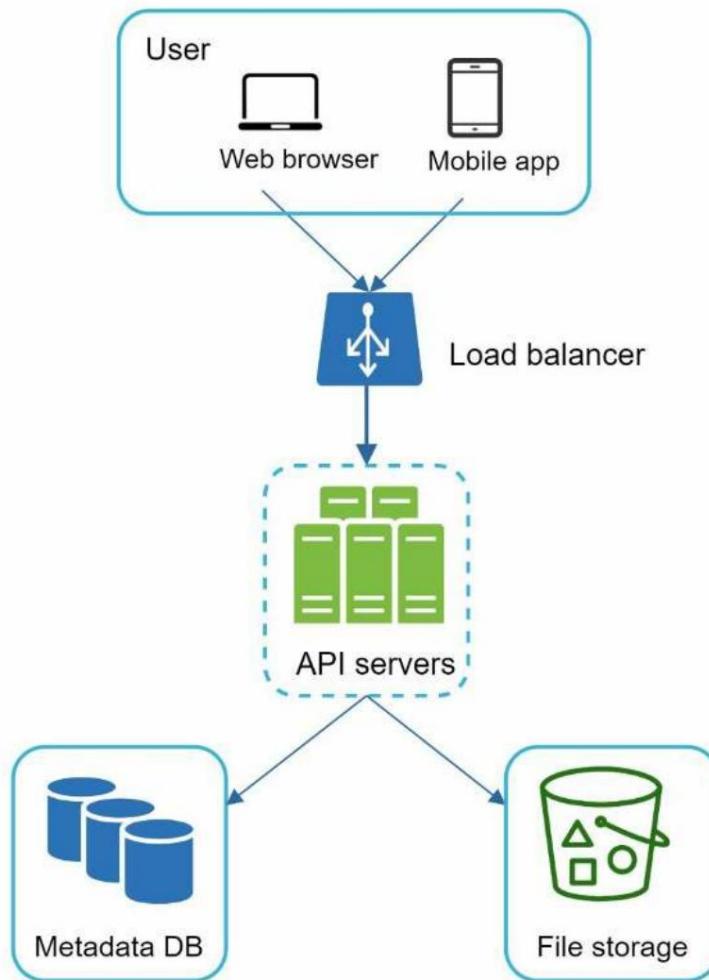


Figure 15-7

Xung đột đồng bộ

Đối với một hệ thống lưu trữ lớn như Google Drive, xung đột đồng bộ xảy ra theo thời gian. Khi hai người dùng sửa đổi cùng một tệp hoặc thư mục cùng một lúc, xung đột sẽ xảy ra. Làm thế nào chúng ta có thể giải quyết xung đột? Đây là chiến lược của chúng tôi: phiên bản đầu tiên được xử lý sẽ thắng và phiên bản được xử lý sau sẽ gặp xung đột. Hình 15-8 cho thấy một ví dụ về xung đột đồng bộ.

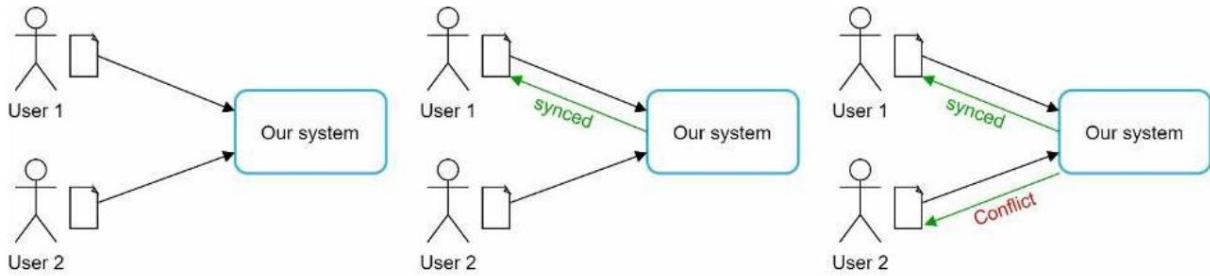


Figure 15-8

Trong Hình 15-8, người dùng 1 và người dùng 2 cố gắng cập nhật cùng một tệp cùng lúc, như tệp của người dùng 1 được hệ thống của chúng tôi xử lý trước. Hoạt động cập nhật của Người dùng 1 diễn ra, nhưng người dùng 2 gặp xung đột đồng bộ. Làm thế nào chúng ta có thể giải quyết xung đột cho người dùng 2? Hệ thống của chúng tôi trình bày cả hai bản sao của cùng một tệp: bản sao cục bộ của người dùng 2 và phiên bản mới nhất từ máy chủ (Hình 15-9). Người dùng 2 có tùy chọn hợp nhất cả hai tệp hoặc ghi đè một phiên bản bằng phiên bản kia.

≡ SystemDesignInterview ☺

≡❗ SystemDesignInterview_user 2_conflicted_copy_2019-05-01 ☺

Figure 15-9

Trong khi nhiều người dùng cùng chỉnh sửa một tài liệu cùng lúc, việc giữ cho tài liệu được đồng bộ hóa là một thách thức. Độc giả quan tâm nên tham khảo tài liệu tham khảo [4] [5].

Thiết kế cấp cao Hình

15-10 minh họa thiết kế cấp cao được đề xuất. Chúng ta hãy xem xét từng thành phần của hệ thống.

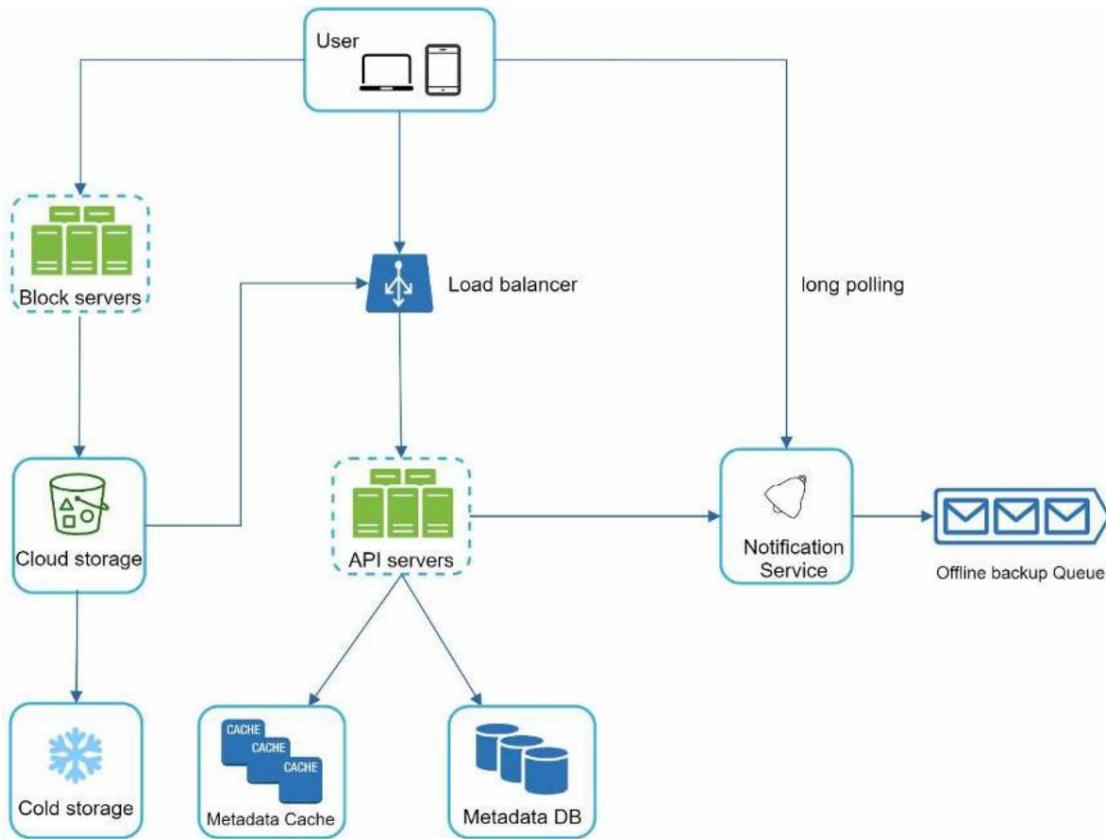


Figure 15-10

Người dùng: Người dùng sử dụng ứng dụng thông qua trình duyệt hoặc ứng dụng di động.

Máy chủ khôi: Máy chủ khôi tải các khôi lên bộ nhớ đệm máy. Bộ nhớ khôi, được gọi là bộ nhớ cấp khôi, là công nghệ lưu trữ các tệp dữ liệu trên môi trường đám mây. Một tệp có thể được chia thành nhiều khôi, mỗi khôi có một giá trị băm duy nhất, được lưu trữ trong cơ sở dữ liệu siêu dữ liệu của chúng tôi. Mỗi khôi được coi là một đối tượng độc lập và được lưu trữ trong hệ thống lưu trữ của chúng tôi (S3). Để tái tạo một tệp, các khôi được nối theo một thứ tự cụ thể. Đối với kích thước khôi, chúng tôi sử dụng Dropbox làm tài liệu tham khảo: Dropbox đặt kích thước tối đa của một khôi là 4MB [6].

Lưu trữ đám mây: Một tệp được chia thành các khôi nhỏ hơn và lưu trữ trên đám mây.

Lưu trữ lạnh: Lưu trữ lạnh là hệ thống máy tính được thiết kế để lưu trữ dữ liệu không hoạt động, nghĩa là các tệp tin sẽ không được truy cập trong một thời gian dài.

Bộ cân bằng tải: Bộ cân bằng tải phân bổ đều các yêu cầu giữa các máy chủ API.

Máy chủ API: Chúng chịu trách nhiệm cho hầu hết mọi thứ ngoại trừ luồng tải lên. Máy chủ API được sử dụng để xác thực người dùng, quản lý hồ sơ người dùng, cập nhật siêu dữ liệu tệp, v.v.

Cơ sở dữ liệu siêu dữ liệu: Lưu trữ siêu dữ liệu của người dùng, tệp, khôi, phiên bản, v.v. Xin lưu ý rằng các tệp được lưu trữ trên đám mây và cơ sở dữ liệu siêu dữ liệu chỉ chứa siêu dữ liệu.

Bộ nhớ đệm siêu dữ liệu: Một số siêu dữ liệu được lưu vào bộ nhớ đệm để truy xuất nhanh.

Dịch vụ thông báo: Đây là hệ thống nhà xuất bản/người đăng ký cho phép dữ liệu được chuyển từ dịch vụ thông báo đến máy khách khi một số sự kiện nhất định xảy ra. Trong trường hợp cụ thể của chúng tôi, dịch vụ thông báo sẽ thông báo cho máy khách có liên quan khi tệp được thêm/sửa/xóa ở nơi khác để họ có thể kéo những thay đổi mới nhất.

Hàng đợi sao lưu ngoại tuyến: Nếu máy khách ngoại tuyến và không thể kéo các thay đổi tệp mới nhất, thì máy khách ngoại tuyến

Hàng đợi sao lưu lưu trữ thông tin để những thay đổi sẽ được đồng bộ hóa khi máy khách trực tuyến.

Chúng tôi đã thảo luận về thiết kế của Google Drive ở cấp độ cao. Một số thành phần phức tạp và đáng để xem xét cẩn thận; chúng tôi sẽ thảo luận chi tiết về những thành phần này trong phần phân tích sâu.

Bài 3 - Thiết kế chuyên sâu

Trong phần này, chúng ta sẽ xem xét kỹ lưỡng những nội dung sau: máy chủ khối, cơ sở dữ liệu siêu dữ liệu, luồng tải lên, luồng tải xuống, dịch vụ thông báo, tiết kiệm không gian lưu trữ và xử lý lỗi.

Chặn máy chủ

Đối với các tệp lớn được cập nhật thường xuyên, việc gửi toàn bộ tệp trên mỗi bản cập nhật sẽ tiêu tốn rất nhiều băng thông. Hai tối ưu hóa được đề xuất để giảm thiểu lưu lượng lưu trữ lưu lượng mạng được truyền:

- Đồng bộ delta. Khi một tệp được sửa đổi, chỉ các khối đã sửa đổi được đồng bộ hóa thay vì toàn bộ tệp bằng thuật toán đồng bộ hóa [7] [8].
- Nén.

Áp dụng nén trên các khối có thể làm giảm đáng kể kích thước dữ liệu.

Do đó, các khối được nén bằng thuật toán nén tùy thuộc vào loại tệp. Ví dụ, gzip và bzip2 được sử dụng để nén tệp văn bản. Cần có các thuật toán nén khác nhau để nén hình ảnh và video.

Trong hệ thống của chúng tôi, máy chủ khối thực hiện công việc nặng nhọc để tải tệp lên. Máy chủ khối xử lý các tệp được truyền từ máy khách bằng cách chia tệp thành các khối, nén từng khối và mã hóa chúng. Thay vì tải toàn bộ tệp lên hệ thống lưu trữ, chỉ các khối đã sửa đổi mới được chuyển.

Hình 15-11 cho thấy máy chủ khối hoạt động như thế nào khi một tệp mới được thêm vào.

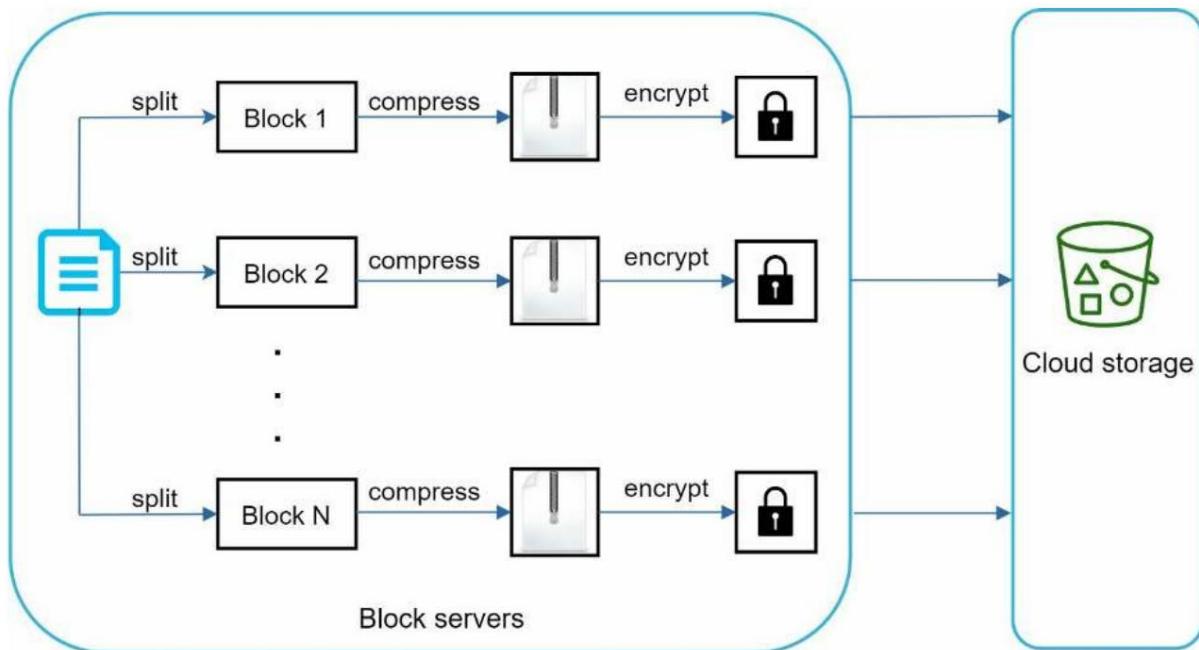


Figure 15-11

- Một tệp được chia thành các khối nhỏ
- Mỗi khối được nén bằng thuật toán nén.
- Để đảm bảo an ninh, mỗi khối được mã hóa trước khi được gửi đến bộ nhớ đám mây.
- Các khối được tải lên bộ nhớ đám mây.

Hình 15-12 minh họa delta sync, nghĩa là chỉ các khối đã sửa đổi mới được chuyển đến lưu trữ đám mây. Các khối được tô sáng "block 2" và "block 5" biểu diễn các khối đã thay đổi. Sử dụng delta sync, chỉ có hai khối đó được tải lên lưu trữ đám mây.

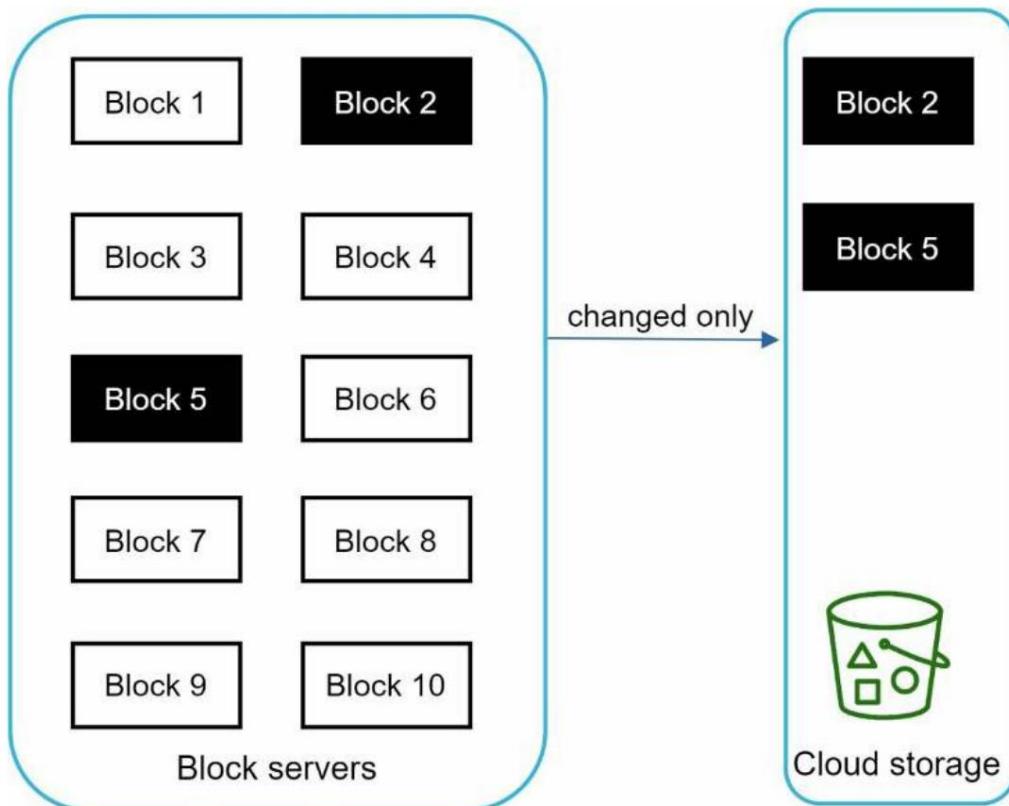


Figure 15-12

Máy chủ khối cho phép chúng ta tiết kiệm lưu lượng mạng bằng cách cung cấp đồng bộ delta và nén.

Yêu cầu về tính nhất quán cao Hệ thống

của chúng tôi yêu cầu tính nhất quán cao theo mặc định. Không thể chấp nhận việc một tệp được hiển thị khác nhau bởi nhiều máy khách cùng một lúc. Hệ thống cần cung cấp tính nhất quán cao cho bộ đệm siêu dữ liệu và các lớp cơ sở dữ liệu.

Bộ nhớ đệm áp dụng mô hình nhất quán cuối cùng theo mặc định, nghĩa là các bản sao khác nhau có thể có dữ liệu khác nhau. Để đạt được tính nhất quán mạnh, chúng ta phải đảm bảo những điều sau:

- Dữ liệu trong bản sao bộ nhớ đệm và bản chính là nhất quán.

- Vô hiệu hóa bộ nhớ đệm khi ghi cơ sở dữ liệu để đảm bảo bộ nhớ đệm và cơ sở dữ liệu giữ cùng một giá trị.

Đạt được tính nhất quán mạnh mẽ trong cơ sở dữ liệu quan hệ là điều dễ dàng vì nó duy trì các thuộc tính ACID (Atomicity, Consistency, Isolation, Durability) [9]. Tuy nhiên, cơ sở dữ liệu NoSQL không hỗ trợ các thuộc tính ACID theo mặc định. Các thuộc tính ACID phải được tích hợp theo chương trình trong logic đồng bộ hóa. Trong thiết kế của chúng tôi, chúng tôi chọn cơ sở dữ liệu quan hệ vì ACID được hỗ trợ gốc.

Cơ sở dữ liệu siêu dữ liệu

Hình 15-13 cho thấy thiết kế lưu trữ đòn cơ sở dữ liệu. Xin lưu ý rằng đây là phiên bản được đơn giản hóa rất nhiều vì nó chỉ bao gồm các bảng quan trọng nhất và các trường thú vị.

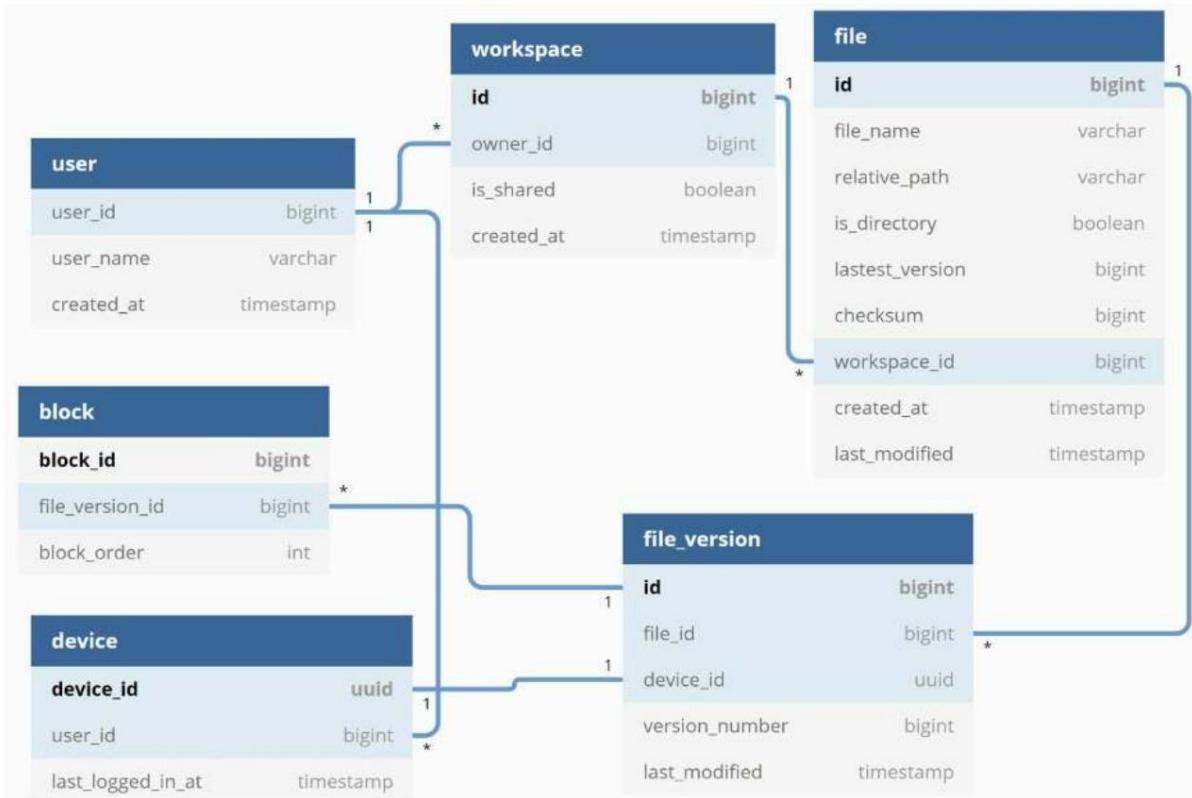


Figure 15-13

Người dùng: Bảng người dùng chứa thông tin cơ bản về người dùng như tên người dùng, email, ảnh đại diện, v.v.

Thiết bị: Bảng thiết bị lưu trữ thông tin thiết bị. Push_id được sử dụng để gửi và nhận thông báo đẩy trên thiết bị di động. Xin lưu ý rằng một người dùng có thể có nhiều thiết bị.

Không gian tên: Không gian tên là thư mục gốc của người dùng.

Tệp: Bảng tệp lưu trữ mọi thứ liên quan đến tệp mới nhất.

File_version: Lưu trữ lịch sử phiên bản của một tệp. Các hàng hiện có chỉ đọc để giữ tính toàn vẹn của lịch sử sửa đổi tệp.

Block: Lưu trữ mọi thứ liên quan đến một khối tệp. Một tệp ở bất kỳ phiên bản nào cũng có thể được xây dựng lại bằng cách nối tất cả các khối theo đúng thứ tự.

Luồng tải lên

Chúng ta hãy thảo luận về những gì xảy ra khi một máy khách tải lên một tệp. Để hiểu rõ hơn về luồng, chúng ta vẽ sơ đồ trình tự như trong Hình 15-14.

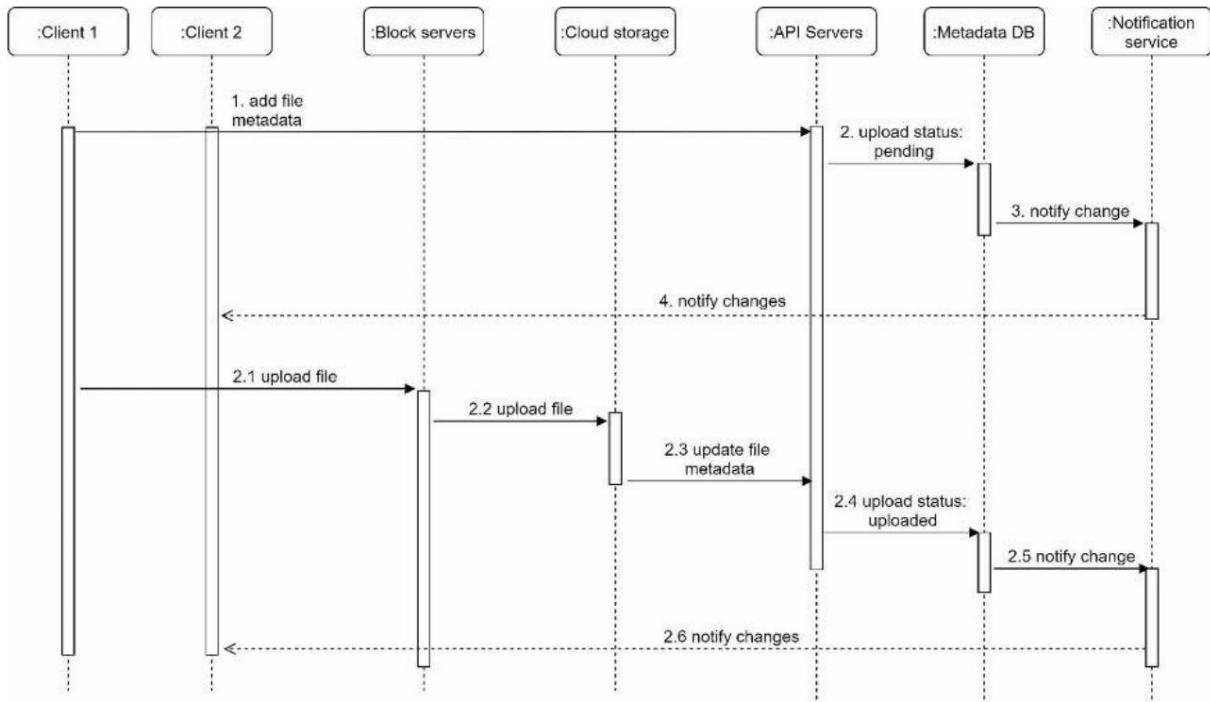


Figure 15-14

Trong Hình 15-14, hai yêu cầu được gửi song song: thêm siêu dữ liệu tệp và tải tệp lên bộ nhớ đám mây. Cả hai yêu cầu đều bắt nguồn từ máy khách 1. • Thêm siêu dữ liệu tệp.

1. Máy khách 1 gửi yêu cầu thêm siêu dữ liệu của tệp mới.
2. Lưu trữ siêu dữ liệu tệp mới trong cơ sở dữ liệu siêu dữ liệu và thay đổi trạng thái tệp lên thành “đang chờ xử lý”.
3. Thông báo cho dịch vụ thông báo rằng có tệp mới đang được thêm vào.
4. Dịch vụ thông báo sẽ thông báo cho các máy khách liên quan (máy khách 2) rằng một tệp đang được tải lên.
- Tải tệp lên bộ nhớ đám mây.
- 2.1 Máy khách 1 tải nội dung của tệp lên máy chủ chặn.
- 2.2 Máy chủ khôi phục chia nhỏ các tệp thành các khôi, nén, mã hóa các khôi và tải chúng lên bộ nhớ đám mây.
- 2.3 Sau khi tệp được tải lên, lưu trữ đám mây sẽ kích hoạt lệnh gọi lại hoàn tất tải lên. Yêu cầu được gửi đến máy chủ API.
- 2.4 Trạng thái tệp đã được thay đổi thành “đã tải lên” trong Cơ sở dữ liệu siêu dữ liệu.
- 2.5 Thông báo cho dịch vụ thông báo rằng trạng thái của tệp đã được thay đổi thành “đã tải lên”.
- 2.6 Dịch vụ thông báo sẽ thông báo cho các máy khách liên quan (máy khách 2) rằng tệp đã được tải lên đầy đủ.

Khi chỉnh sửa một tập tin, quy trình sẽ tương tự, vì vậy chúng tôi sẽ không lặp lại quy trình đó.

Luồng tải xuống Luồng

tải xuống được kích hoạt khi một tệp được thêm hoặc chỉnh sửa ở nơi khác. Làm thế nào để máy khách biết tệp được máy khách khác thêm hoặc chỉnh sửa? Có hai cách để máy khách biết:

- Nếu máy khách A đang trực tuyến trong khi tệp được máy khách khác thay đổi, dịch vụ thông báo sẽ thông báo cho máy khách A rằng có thay đổi ở đâu đó nên cần phải lấy dữ liệu mới nhất.

- Nếu máy khách A ngoại tuyến trong khi tệp được máy khách khác thay đổi, dữ liệu sẽ được lưu vào bộ nhớ đệm. Khi máy khách ngoại tuyến trực tuyến trở lại, nó sẽ kéo những thay đổi mới nhất.

Khi máy khách biết một tệp đã thay đổi, trước tiên nó sẽ yêu cầu siêu dữ liệu qua máy chủ API, sau đó tải xuống các khôi để xây dựng tệp. Hình 15-15 hiển thị luồng chi tiết. Lưu ý, chỉ các thành phần quan trọng nhất được hiển thị trong sơ đồ do hạn chế về không gian.

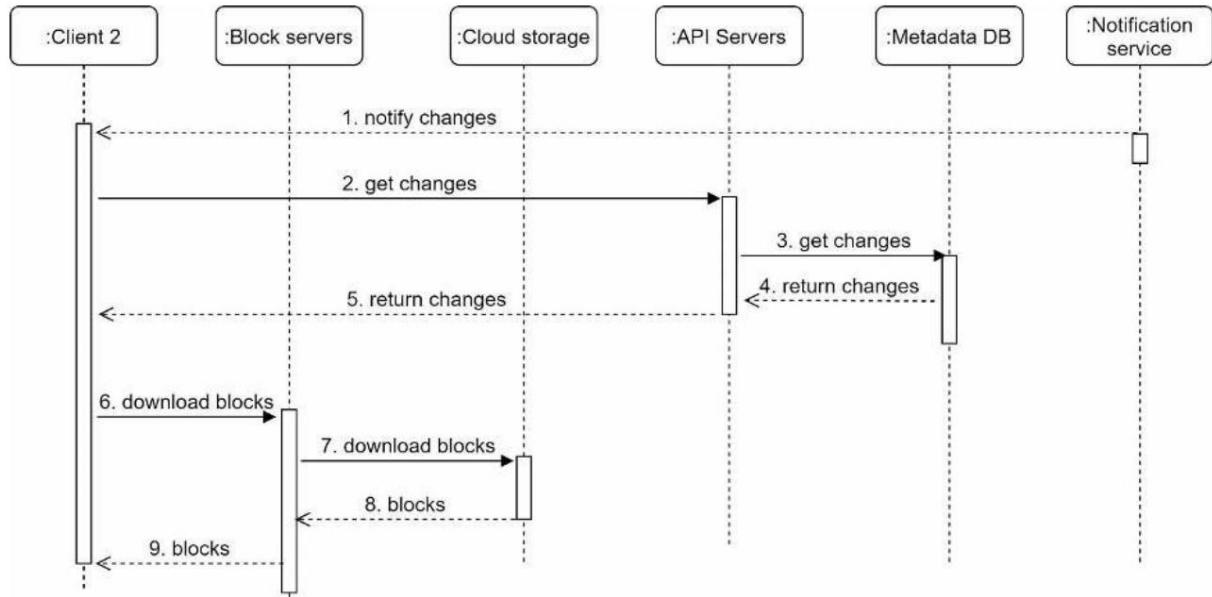


Figure 15-15

1. Dịch vụ thông báo thông báo cho máy khách 2 rằng có một tập tin bị thay đổi ở nơi khác.
2. Khi máy khách 2 biết có bản cập nhật mới, nó sẽ gửi yêu cầu để lấy siêu dữ liệu.
3. Máy chủ API gọi DB siêu dữ liệu để lấy siêu dữ liệu về các thay đổi.
4. Siêu dữ liệu được trả về máy chủ API.
5. Máy khách 2 nhận được siêu dữ liệu.
6. Khi máy khách nhận được siêu dữ liệu, nó sẽ gửi yêu cầu đến máy chủ chặn để tải xuống các khôi.
7. Trước tiên, máy chủ chặn tải xuống các khôi từ bộ nhớ đệm máy.
8. Lưu trữ đệm máy trả các khôi về máy chủ khôi.
9. Máy khách 2 tải xuống tất cả các khôi mới để tái tạo tệp.

Dịch vụ thông báo

Để duy trì tính nhất quán của tệp, bất kỳ đột biến nào của tệp được thực hiện cục bộ đều cần được thông báo cho các máy khách khác để giảm xung đột. Dịch vụ thông báo được xây dựng để phục vụ mục đích này. Ở cấp độ cao, dịch vụ thông báo cho phép dữ liệu được chuyển đến máy khách khi các sự kiện xảy ra. Sau đây là một số tùy chọn:

- Thăm dò dài. Dropbox sử dụng thăm dò dài [10]. •
- WebSocket. WebSocket cung cấp kết nối liên tục giữa máy khách và máy chủ. Giao tiếp là hai chiều.

Mặc dù cả hai tùy chọn đều hoạt động tốt, chúng tôi chọn thăm dò dài vì hai lý do sau:

- Giao tiếp cho dịch vụ thông báo không phải là hai chiều. Máy chủ gửi thông tin về các thay đổi tệp cho máy khách, nhưng không phải ngược lại.
- WebSocket phù hợp cho giao tiếp hai chiều theo thời gian thực như ứng dụng trò chuyện. Đối với

Google Drive, thông báo được gửi không thường xuyên và không có dữ liệu lớn.

Với long polling, mỗi máy khách thiết lập một kết nối long poll đến dịch vụ thông báo. Nếu phát hiện thấy thay đổi đối với tệp, máy khách sẽ đóng kết nối long poll. Đóng kết nối có nghĩa là máy khách phải kết nối đến máy chủ siêu dữ liệu để tải xuống các thay đổi mới nhất. Sau khi nhận được phản hồi hoặc đạt đến thời gian chờ kết nối, máy khách sẽ ngay lập tức gửi yêu cầu mới để giữ kết nối mở.

Tiết kiệm không gian lưu trữ

Để hỗ trợ lịch sử phiên bản tệp và đảm bảo độ tin cậy, nhiều phiên bản của cùng một tệp được lưu trữ trên nhiều trung tâm dữ liệu. Không gian lưu trữ có thể được lấp đầy nhanh chóng bằng cách sao lưu thường xuyên tất cả các bản sửa đổi tệp. Có ba kỹ thuật được đề xuất để giảm chi phí lưu trữ:

- Loại bỏ các khôi dữ liệu trùng lặp. Việc loại bỏ các khôi dư thừa ở cấp độ tài khoản là một cách dễ dàng để tiết kiệm không gian. Hai khôi giống nhau nếu chúng có cùng giá trị băm. • Áp dụng chiến lược sao lưu dữ liệu thông minh. Có thể áp dụng hai chiến lược tối ưu hóa:

- Đặt giới hạn: Chúng ta có thể đặt giới hạn cho số phiên bản lưu trữ. Nếu đạt đến giới hạn, phiên bản cũ nhất sẽ được thay thế bằng phiên bản mới. • Chỉ giữ các phiên bản có giá trị: Một số tệp có thể được chỉnh sửa thường xuyên. Ví dụ, lưu mọi phiên bản đã chỉnh sửa cho một tài liệu được chỉnh sửa nhiều có thể có nghĩa là tệp được lưu hơn 1000 lần trong một thời gian ngắn. Để tránh các bản sao không cần thiết, chúng ta có thể giới hạn số lượng phiên bản đã lưu. Chúng ta chú trọng hơn vào các phiên bản gần đây.

Thử nghiệm hữu ích để tìm ra số lượng phiên bản tối ưu để lưu. • Di chuyển dữ liệu ít sử dụng đến kho lưu trữ lạnh. Dữ liệu lạnh là dữ liệu không hoạt động trong nhiều tháng hoặc nhiều năm. Kho lưu trữ lạnh như Amazon S3 glacier [11] rẻ hơn nhiều so với S3.

Xử lý lỗi Lỗi có thể xảy

ra trong một hệ thống quy mô lớn và chúng ta phải áp dụng các chiến lược thiết kế để giải quyết những lỗi này. Người phòng vấn của bạn có thể muốn biết cách bạn xử lý các lỗi hệ thống sau:

- Lỗi bộ cân bằng tải: Nếu bộ cân bằng tải bị lỗi, bộ cân bằng tải thứ cấp sẽ hoạt động và tiếp nhận lưu lượng. Các bộ cân bằng tải thứ cấp giám sát lẫn nhau bằng cách sử dụng nhịp tim, một tín hiệu định kỳ được gửi giữa các bộ cân bằng tải. Một bộ cân bằng tải được coi là bị lỗi nếu nó không gửi nhịp tim trong một thời gian.

- Lỗi máy chủ khồi: Nếu một máy chủ khồi bị lỗi, các máy chủ khác sẽ tiếp nhận các công việc chưa hoàn thành hoặc

đang chờ xử lý. • Lỗi lưu trữ đám mây: Các thùng S3 được sao chép nhiều lần ở các vùng khác nhau. Nếu các tệp không khả dụng ở một vùng, chúng có thể được truy xuất từ các vùng khác nhau. • Lỗi máy chủ API: Đây là dịch vụ không có trạng thái. Nếu một máy chủ API bị lỗi, bộ cân bằng tải sẽ chuyển hướng lưu lượng đến các máy chủ API khác. • Lỗi bộ đệm siêu dữ liệu:

Máy chủ bộ đệm siêu dữ liệu được sao chép nhiều lần. Nếu một nút ngừng hoạt động, bạn vẫn có thể truy cập các nút khác để truy xuất dữ liệu. Chúng tôi sẽ đưa lên một máy chủ bộ đệm mới để thay thế máy chủ bị lỗi. • Lỗi DB siêu dữ liệu.

- Master down: Nếu master down, hãy tăng cấp một trong các slave để hoạt động như một master mới và đưa lên một nút slave mới. • Slave down: Nếu một slave down, bạn có thể sử dụng một slave khác cho các hoạt động đọc và

mang một máy chủ cơ sở dữ liệu khác đến để thay thế máy chủ bị hỏng.

- Lỗi dịch vụ thông báo: Mỗi người dùng trực tuyến đều duy trì kết nối thăm dò dài với máy chủ thông báo. Do đó, mỗi máy chủ thông báo được kết nối với nhiều người dùng. Theo bài nói chuyện của Dropbox năm 2012 [6], hơn 1 triệu kết nối được mở trên mỗi máy. Nếu một máy chủ ngừng hoạt động, tất cả các kết nối thăm dò dài đều bị mất nên các máy khách phải kết nối lại với một máy chủ khác. Mặc dù một máy chủ có thể duy trì nhiều kết nối mở, nhưng nó không thể kết nối lại tất cả các kết nối đã mất cùng một lúc. Việc kết nối lại với tất cả các máy khách đã mất là một quá trình tương đối chậm. • Lỗi hàng đợi sao lưu ngoại tuyến: Hàng đợi được sao chép nhiều lần. Nếu một hàng đợi bị lỗi, người dùng của hàng đợi có thể cần đăng ký lại hàng đợi sau lưu.

Bứ ớc 4 - Tǒng kêt Trong

chư ơ ng nà y, chung tōi đã đè xuâ t môt thiêt kê hệ thô ng để hổ tró Google Drive. Sư kết hợp giûa tính nhâ t quâ n mạn h mẽ, bă ng thô ng mạn g thâ p và đồng bộ hóa nhanh lâm cho thiêt kê trở nê n thû vi. Thiêt kê của chung tōi bao gồm hai luô ng: quản lý siê u dữ liê u tê p và đồng bộ hóa tê p. Dịch vụ thông báo là môt thàn h phâ n quan trọng khâ c của hệ thô ng. Nô sủ dụng thă m dò dài đê giûp khâ ch hàng cập nhât các thay đổi của tê p.

Giống như bất kỳ câu hỏi phỏng vấn thiêt kê hệ thô ng nà o, khô ng có giải pháp hoàn hảo. Mỗi công ty đều có nhữ ng hạn ché riê ng và bạn phâi thiêt kê môt hệ thô ng phù hợp với nhữ ng hạn ché đó. Biết đư ợc sự đán h đỗi giûa thiêt kê và lựa chọn công nghệ của bạn là điều quan trọng. Nếu còn vài phút nǔa, bạn có thể nói về các lựa chọn thiêt kê khâ c nhau.

Ví dụ, chung ta có thể tải tê p trực tiếp lêm luh trũ đám mây từ máy khâ ch thay vì phâi thông qua máy chủ khô i. Ưu đì em của cách tiếp cận này là nó giûp tải tê p lêm nhanh hơ n vì tê p chí càn đư ợc chuyê n môt lân đên luh trũ đám mây. Trong thiêt kê của chung tōi, tê p đư ợc chuyê n đên máy chủ khô i trư ớc, sau đô đên luh trũ đám mây. Tuy nhiên, cách tiếp cận mới có môt số nhữ ợc đì em:

- Đầu tiên, cùng môt logic chunking, compression và encryption phâi đư ợc triển khai trên các nền tảng khâ c nhau (iOS, Android, Web). Nó dẽ bị lâ i và đài hỏi nhie u nô lực kỹ thuât. Trong thiêt kê của chung tōi, tất cả các logic đó đư ợc triển khai ở môt nơ i tâp trung: block servers. • Thứ hai, vì máy khâ ch có thể dẽ dàng bị hack hoặc bị thao túng, nê n việc triển khai logic mã hóa ở phâ n mây khâ ch là khô ng lý tư ờng.

Một sự phát triễn thú vi khâ c của hệ thô ng là di chuyê n logic trực tuyê n/ngoại tuyê n sang môt dịch vụ riê ng biê t. Chung ta hâ y gọi đó là dịch vụ hiện dien. Bằng cách di chuyê n dịch vụ hiện dien ra khô i máy chủ thông báo, chức nă ng trực tuyê n/ngoại tuyê n có thể dẽ dàng đư ợc tích hợp bời các dịch vụ khâ c.

Xin chúc mừng vì đã đì đư ợc đê n đâ y! Bây giờ hâ y tự khen mìn h nhé. Lâ m tốt lâ m!

Tài liệu tham khảo

- [1] Google Drive: <https://www.google.com/drive/>
- [2] Tải dữ liệu tệp lên: <https://developers.google.com/drive/api/v2/manage-uploads>
- [3] Amazon S3: <https://aws.amazon.com/s3>
- [4] Đồng bộ hóa khác biệt <https://neil.fraser.name/writing/sync/>
- [5] Bài nói chuyện trên youtube về Đồng bộ hóa khác biệt https://www.youtube.com/watch?v=S2Hp_1jqpY8
- [6] Cách chúng tôi mở rộng Dropbox: <https://youtu.be/PE4gwstWhmc>
- [7] Tridgell, A., & Mackerras, P. (1996). Thuật toán rsync.
- [8] Librsync. (nd). Truy cập ngày 18 tháng 4 năm 2015, từ <https://github.com/librsync/librsync>
- [9] ACID: <https://en.wikipedia.org/wiki/ACID>
- [10] Sách trắng bảo mật Dropbox: https://www.dropbox.com/static/business/resources/Security_Whitepaper.pdf
- [11] Amazon S3 Glacier: <https://aws.amazon.com/glacier/faqs/>

CHƯƠNG 16: VIỆC HỌC TẬP VĂN TIẾP TỤC

Thiết kế hệ thống tốt đòi hỏi nhiều năm tích lũy kiến thức. Một cách nhanh chóng là tìm hiểu sâu về kiến trúc hệ thống trong thế giới thực. Dưới đây là bộ sưu tập tài liệu đọc hữu ích.

Chúng tôi thực sự khuyên bạn nên chú ý đến cả các nguyên tắc chung và các công nghệ cơ bản. Nghiên cứu từng công nghệ và hiểu được những vấn đề mà công nghệ đó giải quyết là một cách tuyệt vời để củng cố cơ sở kiến thức của bạn và tinh chỉnh quy trình thiết kế.

Hệ thống thực tế Các

tài liệu sau đây có thể giúp bạn hiểu được ý tư ởng thiết kế chung về kiến trúc hệ thống thực tế của các công ty khác nhau.

Dòng thời gian Facebook: Mang đến cho bạn sức mạnh của sự phi chuẩn hóa: <https://goo.gl/FCNrhm>

Mở rộng quy mô trên Facebook: <https://goo.gl/NGTdCs>

Xây dựng dòng thời gian: Mở rộng quy mô để lưu giữ câu chuyện cuộc đời ban: <https://goo.gl/8p5wDV>

Erlang trên Facebook (trò chuyện trên Facebook): <https://goo.gl/zSLHrj>

Trò chuyện trên Facebook: <https://goo.gl/qzSiWC>

Tìm kim trong đồng cỏ khô: Lưu trữ ảnh của Facebook: <https://goo.gl/edj4FL>

Phục vụ Facebook Multifeed: Hiệu quả, hiệu suất tăng lên thông qua thiết kế lại: <https://goo.gl/adFVMQ>

Mở rộng Memcache tại Facebook: <https://goo.gl/rZiAhX>

TAO: Kho dữ liệu phân tán của Facebook dành cho Social Graph: <https://goo.gl/Tk1DyH> Kiến trúc Amazon: <https://goo.gl/k4feoW> Dynamo: [Kho lưu trữ khóa-giá](https://goo.gl/Khóá-giá)

tri có tính khả dụng cao của Amazon: <https://goo.gl/C7zxDL>

Góc nhìn 360 độ về toàn bộ Netflix Stack: <https://goo.gl/rYSDTz>

Tất cả đều là về thử nghiệm: Nền tảng thử nghiệm của Netflix: <https://goo.gl/agbA4K> Đề xuất của

Netflix: Ngoài 5 sao (Phần 1): <https://goo.gl/A4FkYi>

Đề xuất của Netflix: Ngoài 5 sao (Phần 2): <https://goo.gl/XNPMXm>

Kiến trúc Google: <https://goo.gl/dvkDiY>

Hệ thống tệp Google (Google Docs): <https://goo.gl/xj5n9R>

Đồng bộ hóa khác biệt (Google Docs): <https://goo.gl/9zgG7x>

Kiến trúc YouTube: <https://goo.gl/mCPRUF>

Hội nghị Seattle về khả năng mở rộng: Khả năng mở rộng của YouTube: <https://goo.gl/dH3zYq>

Bigtable: Hệ thống lưu trữ phân tán cho dữ liệu có cấu trúc: <https://goo.gl/6NaZca>

Kiến trúc Instagram: 14 triệu người dùng, hàng Terabyte ảnh, hàng trăm trang hợp, hàng chục công nghệ: <https://goo.gl/s1VcW5>

Kiến trúc mà Twitter sử dụng để xử lý 150 triệu người dùng đang hoạt động: <https://goo.gl/EwvfRd>

Mở rộng quy mô Twitter: Làm cho Twitter nhanh hơn 10000 phần trăm: <https://goo.gl/nYGC1k>

Giới thiệu Snowflake (Snowflake là dịch vụ mạng để tạo số ID duy nhất ở quy mô lớn với một số đảm bảo đơn giản): <https://goo.gl/GzVWYm>

Dòng thời gian theo quy mô: <https://goo.gl/8KbaTy>

Uber mở rộng quy mô nền tảng thị trường thời gian thực của họ như thế nào: <https://goo.gl/kGZuVy>

Mở rộng Pinterest: <https://goo.gl/KtmjW3>

Cập nhật kiến trúc Pinterest: <https://goo.gl/w6rRsf>

Lịch sử tóm tắt về việc mở rộng quy mô LinkedIn: <https://goo.gl/8A1Pi8>

Kiến trúc Flickr: <https://goo.gl/dWtgYa>

Cách chúng tôi mở rộng Dropbox: <https://goo.gl/NjBDtC>

Kiến trúc WhatsApp mà Facebook đã mua với giá 19 tỷ đô la: <https://bit.ly/2AHJnFn>

Blog kỹ thuật của công ty Nếu bạn

sắp phỏng vấn với một công ty, bạn nên đọc blog kỹ thuật của họ và làm quen với các công nghệ và hệ thống được áp dụng và triển khai tại đó. Bên cạnh đó, blog kỹ thuật cung cấp những hiểu biết vô giá về một số lĩnh vực nhất định. Đọc chúng thường xuyên có thể giúp chúng ta trở thành những kỹ sư giỏi hơn.

Sau đây là danh sách các blog kỹ thuật của các công ty lớn và công ty khởi nghiệp nổi tiếng.

Airbnb: <https://medium.com/airbnb-engineering> Amazon:
<https://developer.amazon.com/blogs> Asana: <https://blog.asana.com/category/eng> Atlassian: <https://developer.atlassian.com/blog> BitTorrent: <http://engineering.bittorrent.com> Cloudera: <https://blog.cloudera.com> Docker: <https://blog.docker.com> Dropbox: <https://blogs.dropbox.com/tech> eBay : <http://www.ebaytechblog.com> Facebook: <https://code.facebook.com/posts> GitHub: <https://githubengineering.com> Google: <https://developers.googleblog.com> Groupon: <https://engineering.groupon.com> Khả năng mở rộng quy mô cao: <http://highscalability.com> Instacart:
<https://tech.instacart.com> Instagram:
<https://engineering.instagram.com> LinkedIn: <https://engineering.linkedin.com/blog> Mixpanel: <https://mixpanel.com/blog> Netflix: <https://medium.com/netflix-techblog> Nextdoor: <https://engblog.nextdoor.com> PayPal: <https://www.paypal-engineering.com> Pinterest: <https://engineering.com/pinterest> Quora: <https://engineering.pinterest.com/quora> Reddit: <https://engineering.quora.com> Salesforce: <https://redditblog.com> Shopify: <https://developer.shopify.com/blogs/engineering> Slack: <https://engineering.shopify.com> SoundCloud:
<https://developers.soundcloud.com/blog> Spotify: <https://labs.spotify.com> Stripe: <https://stripe.com/blog/engineering> Tài liệu cơ bản về thiết kế hệ thống: <https://github.com/donnemartin/system-design-primer> Twitter: https://blog.twitter.com/engineering/en_us.html Thumbtack: <https://www.thumbtack.com/engineering> Uber: <http://eng.uber.com>

Yahoo: <https://yahooblog.tumblr.com>

Yelp: <https://engineeringblog.yelp.com>

Zoom: <https://medium.com/zoom-developer-blog>

LỜI BÊN KẾT

Xin chúc mừng! Bạn đã đến cuối hư ứng dẫn phòng vấn này. Bạn đã tích lũy được các kỹ năng và kiến thức để thiết kế hệ thống.

Không phải ai cũng có đủ kỹ luật để học những gì bạn đã học.

Hãy dành chút thời gian và tự khen ngợi bản thân. Công sức của bạn sẽ được đền đáp xứng đáng.

Việc có được công việc mơ ước là một hành trình dài và đòi hỏi nhiều thời gian và công sức. Luyện tập sẽ tạo nên sự hoàn hảo. Chúc bạn may mắn!

Cảm ơn bạn đã mua và đọc cuốn sách này. Nếu không có độc giả như bạn, tác phẩm của chúng tôi sẽ không tồn tại. Chúng tôi hy vọng bạn thích cuốn sách này!

Nếu bạn không phiền, vui lòng đánh giá cuốn sách này trên Amazon: <https://tinyurl.com/y7d3ltbc> Nó sẽ giúp tôi thu hút thêm nhiều độc giả tuyệt vời như bạn.

Vui lòng đăng ký danh sách email của chúng tôi nếu bạn muốn được thông báo khi có chương mới: <https://bit.ly/3dtIcsE>

Nếu bạn có ý kiến hoặc thắc mắc về cuốn sách này, vui lòng gửi email cho chúng tôi theo địa chỉ systemdesigninsider@gmail.com. Ngoài ra, nếu bạn nhận thấy bất kỳ lỗi nào, vui lòng cho chúng tôi biết để chúng tôi có thể sửa lỗi trong lần tái bản tiếp theo. Cảm ơn bạn!