Hệ thống cơ sở dữ liệu NoSQL:

Hướng dẫn khảo sát và quyết định

Felix Gessert, Wolfram Wingerath, Ste en Friedrich, và Norbert Ritter

Đại học Hamburg, Đức

[{gessert,](mailto:gessert@informatik.uni-hamburg.de) [wingerath,](mailto:wingerath@informatik.uni-hamburg.de) Friedrich, [ritter}](mailto:friedrich@informatik.uni-hamburg.de)[@informatik.uni-](mailto:ritter@informatik.uni-hamburg.de)hamburg.de

Trừu tượng.Ngày nay, dữ liệu được tạo ra và tiêu thụ ở quy mô chưa từng có. Điều này đã dẫn đến các cách tiếp cận mới để quản lý dữ liệu có thể mở rộng được gộp lại dưới thuật ngữ hệ thống cơ sở dữ liệu NoSQL để xử lý khối lượng dữ liệu ngày càng tăng và tải yêu cầu. Tuy nhiên, tính không đồng nhất và đa dạng của nhiều hệ thống hiện có cản trở việc lựa chọn kho lưu trữ dữ liệu phù hợp với bối cảnh ứng dụng nhất định. Do đó, bài viết này cung cấp một cái nhìn tổng quan từ trên xuống về lĩnh vực này: Thay vì đối chiếu các đặc điểm triển khai của các đại diện riêng lẻ, chúng tôi đề xuất một mô hình phân loại so sánh liên quan đến các yêu cầu chức năng và phi chức năng với các kỹ thuật và thuật toán được sử dụng trong cơ sở dữ liệu NoSQL.

# 1. Giới thiệu

Các hệ thống quản lý cơ sở dữ liệu quan hệ truyền thống (RDBMS) cung cấp các cơ chế mạnh mẽ để lưu trữ và truy vấn dữ liệu có cấu trúc dưới sự nhất quán và đảm bảo giao dịch mạnh mẽ, đồng thời đã đạt đến mức độ tin cậy, ổn định và hỗ trợ chưa từng có qua nhiều thập kỷ phát triển. Tuy nhiên, trong những năm gần đây, lượng dữ liệu hữu ích trong một số lĩnh vực ứng dụng đã trở nên quá lớn đến mức không thể lưu trữ hoặc xử lý bằng các giải pháp cơ sở dữ liệu truyền thống. Nội dung do người dùng tạo trong các mạng xã hội hoặc dữ liệu được lấy từ các mạng cảm biến lớn chỉ là hai ví dụ về hiện tượng này thường được gọi là Dữ liệu lớn [35]. Một lớp hệ thống lưu trữ dữ liệu mới có thể đối phó với Dữ liệu lớn được gộp lại theo thuật ngữcơ sở dữ liệu NoSQL, nhiều trong số đó cung cấp khả năng mở rộng theo chiều ngang và tính sẵn sàng cao hơn cơ sở dữ liệu quan hệ bằng cách hy sinh khả năng truy vấn và đảm bảo tính nhất quán. Những trao đổi này là then chốt đối với các mô hình điện toán hướng dịch vụ và dưới dạng dịch vụ, vì bất kỳ dịch vụ trạng thái nào cũng chỉ có thể mở rộng và chịu lỗi như kho lưu trữ dữ liệu cơ bản của nó.

Có hàng tá hệ thống cơ sở dữ liệu NoSQL và thật khó để theo dõi xem chúng vượt trội ở đâu, thất bại ở đâu hoặc thậm chí khác biệt ở đâu, vì các chi tiết triển khai thay đổi nhanh chóng và các bộ tính năng phát triển theo thời gian. Do đó, trong bài viết này, chúng tôi nhằm mục đích cung cấp một cái nhìn tổng quan về bối cảnh NoSQL bằng cách thảo luận về các khái niệm được sử dụng thay vì các thành phố cụ thể của hệ thống và khám phá các yêu cầu thường đặt ra cho các hệ thống cơ sở dữ liệu NoSQL, các kỹ thuật được sử dụng để đáp ứng các yêu cầu này và các trao đổi mà phải được thực hiện trong quá trình. Trọng tâm của chúng tôi nằm ở các cửa hàng khóa-giá trị, tài liệu và cột rộng, vì các danh mục NoSQL này

bao gồm các kỹ thuật phù hợp nhất và các quyết định thiết kế trong không gian quản lý dữ liệu có thể mở rộng.

Trong Phần 2, chúng tôi mô tả các cách tiếp cận cấp cao phổ biến nhất đối với việc phân loại các hệ thống cơ sở dữ liệu NoSQL theo mô hình dữ liệu của chúng thành các kho lưu trữ khóa-giá trị, kho lưu trữ tài liệu và kho lưu trữ nhiều cột hoặc theo các giao dịch đảm bảo tính an toàn trong thiết kế của chúng (CAP và PACELC). Sau đó, chúng tôi khảo sát các kỹ thuật thường được sử dụng chi tiết hơn và thảo luận về mô hình của chúng tôi về mối quan hệ của các yêu cầu và kỹ thuật [trong Phần 3, trước khi chúng](#_bookmark8) tôi đưa ra tổng quan về các hệ thống cơ sở dữ liệu nổi bật bằng cách áp dụng mô hình của chúng tôi cho chúng trong Phần 4. Một mô hình quyết định đơn giản và trừu tượng để hạn chế lựa chọn các hệ thống NoSQL phù hợp dựa trên các yêu cầu của ứng dụng kết thúc bài báo trong Phần 5.

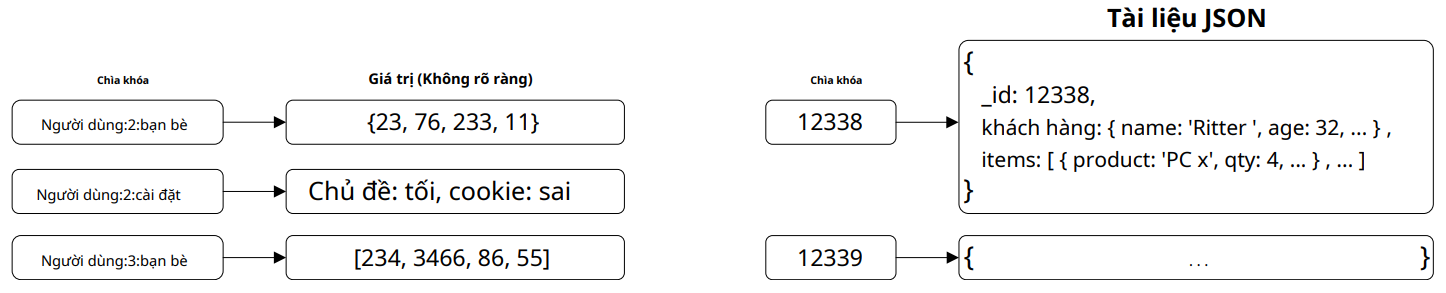
# Phân loại hệ thống cấp cao

Để trừu tượng hóa các chi tiết triển khai của các hệ thống NoSQL riêng lẻ, các tiêu chí phân loại cấp cao có thể được sử dụng để nhóm các kho lưu trữ dữ liệu tương tự thành các danh mục. Trong phần này, chúng tôi giới thiệu hai cách tiếp cận nổi bật nhất: mô hình dữ liệu và các lớp định lý CAP.

* 1. Các mô hình dữ liệu khác nhau

### Sự khác biệt được sử dụng phổ biến nhất giữa các cơ sở dữ liệu NoSQL là

cách họ lưu trữ và cho phép truy cập vào dữ liệu.



Hình 1: Cửa hàng khóa-giá trị cung cấp khả năng lưu trữ và truy xuất hiệu quả các giá trị tùy ý.

Hình 2: Kho lưu trữ tài liệu nhận thức được cấu trúc bên trong của thực thể được lưu trữ và do đó có thể hỗ trợ các truy vấn.

Cửa hàng khóa-giá trị.Kho lưu trữ khóa-giá trị bao gồm một tập hợp các cặp khóa-giá trị với các khóa duy nhất. Do cấu trúc đơn giản này, nó chỉ hỗ trợ các thao tác get và put. Vì bản chất của giá trị được lưu trữ là trong suốt đối với cơ sở dữ liệu, các kho lưu trữ khóa-giá trị thuần túy không hỗ trợ các hoạt động ngoài CRUD đơn giản (Tạo, Đọc, Cập nhật, Xóa). Do đó, các cửa hàng khóa-giá trị thường được gọi làkhông có sơ đồ

[[44]](#_bookmark61): Mọi giả định về cấu trúc của dữ liệu được lưu trữ đều được mã hóa hoàn toàn

trong logic ứng dụng (lược đồ khi đọc [31]) và không được xác định rõ ràng thông qua ngôn ngữ định nghĩa dữ liệu (lược đồ trên ghi).

Ưu điểm rõ ràng của mô hình dữ liệu này nằm ở tính đơn giản của nó. Sự trừu tượng hóa rất đơn giản giúp dễ dàng phân vùng và truy vấn dữ liệu, nhờ đó hệ thống cơ sở dữ liệu có thể đạt được độ trễ thấp cũng như thông lượng cao. Tuy nhiên, nếu một ứng dụng yêu cầu các hoạt động phức tạp hơn, ví dụ như truy vấn phạm vi, thì mô hình dữ liệu này không đủ mạnh. Hình [1 minh họa](#_bookmark1) cách dữ liệu và cài đặt tài khoản người dùng có thể được lưu trữ trong kho khóa-giá trị. Vì các truy vấn phức tạp hơn tra cứu đơn giản không được hỗ trợ, nên dữ liệu phải được phân tích một cách không hiệu quả trong mã ứng dụng để trích xuất thông tin như liệu cookie có được hỗ trợ hay không (cookie: sai).

Cửa hàng tài liệu.Kho lưu trữ tài liệu là kho lưu trữ khóa-giá trị giới hạn các giá trị ở các định dạng bán cấu trúc, chẳng hạn như JSON1các [tài](#_bookmark3) liệu. Hạn chế này so với các kho lưu trữ khóa-giá trị mang lại khả năng truy cập dữ liệu rất linh hoạt. Không chỉ có thể tìm nạp toàn bộ tài liệu theo ID của nó mà còn có thể chỉ truy xuất các phần của tài liệu, ví dụ: tuổi của khách hàng và thực hiện các truy vấn như tổng hợp, truy vấn theo ví dụ hoặc thậm chí tìm kiếm toàn văn.

Cửa hàng cột rộngkế thừa tên của chúng từ hình ảnh thường được sử dụng để giải thích mô hình dữ liệu cơ bản: một bảng quan hệ có nhiều cột thưa thớt. Tuy nhiên, về mặt kỹ thuật, cửa hàng nhiều cột gần với cửa hàng đa cấp phân tán hơn.2

bản đồ được sắp xếp: Các

các cặp giá trị. Các

Các khóa cấp đầu tiên xác định các hàng chứa các khóa cấp đầu tiên được gọiphím hàng,các khóa cấp hai là

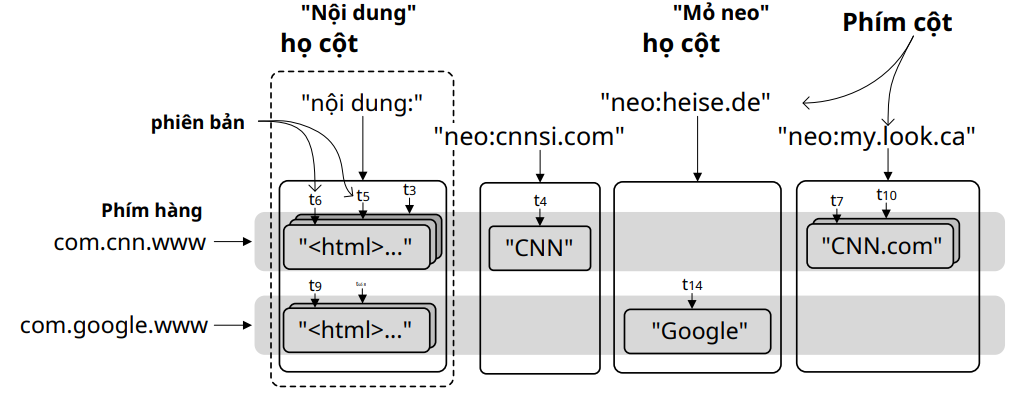
gọi làphím cột.Sơ đồ lưu trữ này làm cho các bảng có nhiều cột tùy ý trở nên khả thi, bởi vì không có khóa cột nào không có giá trị tương ứng. Do đó, các giá trị null có thể được lưu trữ mà không có bất kỳ chi phí không gian nào. Tập hợp tất cả các cột được phân vùng thành cái gọi làhọ cộtđể sắp xếp các cột trên đĩa thường được truy cập cùng nhau. Trên đĩa, các cửa hàng cột rộng không sắp xếp tất cả dữ liệu từ mỗi hàng, mà thay vào đó là các giá trị của cùng một họ cột vàtừ cùng một hàng. Do đó, một thực thể (một hàng) không thể được truy xuất bằng một lần tra cứu đơn lẻ như trong kho lưu trữ tài liệu, mà phải được nối với nhau từ các cột của tất cả các họ cột. Tuy nhiên, cách bố trí lưu trữ này thường cho phép nén dữ liệu rất hiệu quả và chỉ truy xuất một pchổầđnại

của thực thể rất hiệu quả.

cổ đại.

Dữ liệu được lưu trữ theo thứ tự từ điển của các khóa, do đó dữ liệu được truy cập cùng nhau được đặt cùng vị trí vật lý, với thiết kế khóa cẩn thận. Vì tất cả các hàng được phân phối thành các phạm vi liền kề (được gọi làmáy tính bảng)trong số khác nhau máy chủ máy tính

bảng,quét hàng chỉ liên quan đến một vài máy chủ và do đó rất hiệu quả cổ đại.



Hình 3: Dữ liệu trong kho lưu trữ nhiều cột.

## [Bigtable [9],](#_bookmark26) đi tiên phong trong mô hình cột rộng, được phát triển đặc biệt để lưu trữ một tập hợp lớn các trang web như được minh họa [trong](#_bookmark2) Hình

3. Mỗi hàng trong bảng trang web tương ứng với một trang web. Khóa hàng là sự kết hợp của các thành phần URL theo thứ tự đảo ngược và mỗi khóa cột bao gồm họ cột và định tính cột, được phân tách bằng dấu hai chấm. Có hai họ cột: họ cột nội dung chỉ có một cột chứa trang web thực tế và họ cột neo giữ

1Ký hiệu đối tượng JavaScript là một định dạng chuẩn bao gồm thuộc tính lồng nhau-

cặp giá trị và danh sách.

2Trong một số hệ thống(ví dụ: Bigtable và HBase), đa phiên bản là không cần thiết đề cập bởi

thêm dấu thời gian làm khóa cấp ba.

liên kết đến từng trang web, mỗi trang trong một cột riêng biệt. Mọi ô trong bảng (nghĩa là mọi giá trị có thể truy cập bằng tổ hợp phím hàng và cột) đều có thể được tạo phiên bản theo dấu thời gian hoặc số phiên bản. Điều quan trọng cần lưu ý là phần lớn thông tin của một thực thể nằm trong các khóa chứ không chỉ trong các giá trị [9].

* 1. Trao đổi tính nhất quán-Tính sẵn sàng-O s: CAP và PACELC Một thuộc tính xác định khác của cơ sở dữ liệu ngoài cách dữ liệu được lưu trữ và cách chúng có thể được

truy cập là mức độ nhất quán được cung cấp. Một số cơ sở dữ liệu được xây dựng để đảm bảo tính nhất quán mạnh mẽ và khả năng tuần tự hóa (ACID3), trong khi những người khác ủng hộ tính khả dụng (BASE4[). Trao](#_bookmark6) đổi này vốn có đối với mọi hệ thống cơ sở dữ liệu phân tán và số lượng lớn các hệ thống NoSQL khác nhau cho thấy rằng có một phổ rộng giữa hai mô hình. Sau đây, chúng tôi giải thích hai định lý CAP và PACELC theo đó các hệ thống cơ sở dữ liệu có thể được phân loại theo vị trí tương ứng của chúng trong phổ này.

### [MŨ LƯỠI TRAI.Giống như Định lý FLP nổi tiếng](#_bookmark37) [19], Định lý CAP, được trình bày bởi Eric Brewer tại PODC 2000 [7] và sau đó được chứng minh bởi Gilbert [và](#_bookmark39) Lynch [21], là một trong những kết quả bất khả thi thực sự trong lĩnh vực điện toán phân tán, bởi vì nó đặt giới hạn trên cuối cùng cho những gì có thể được thực hiện bởi một hệ thống phân tán. Nó nói rằng một thanh ghi đọc/ghi nhất quán tuần tự5mà [cuối](#_bookmark7) cùng đáp ứng mọi yêu cầu không thể được thực hiện trong một hệ thống không đồng bộ có xu hướng phân vùng mạng. Nói cách khác, nó có thể cùng lúc đảm bảo tối đa hai trong ba đặc tính sau:

Tính nhất quán (C):Việc đọc và ghi luôn được thực thi nguyên tử và nhất quán nghiêm ngặt (có thể tuyến tính hóa [26]). Nói cách khác, tất cả khách hàng luôn có cùng một chế độ xem dữ liệu.

### Sẵn có (A):Mọi nút không bị lỗi trong hệ thống luôn có thể chấp nhận các yêu cầu đọc và ghi của máy khách và cuối cùng sẽ trả về một phản hồi có ý nghĩa, tức là không kèm theo thông báo lỗi.

Dung sai phân vùng (P):Hệ thống duy trì các đảm bảo về tính nhất quán và tính khả dụng đã hiển thị trước đó trong trường hợp mất tin nhắn giữa các nút hoặc lỗi hệ thống một phần.

Brewer lập luận rằng một hệ thống có thể vừa khả dụng vừa nhất quán trong hoạt động bình thường, nhưng với sự có mặt của một phân vùng hệ thống, điều này là không thể: Nếu hệ thống tiếp tục hoạt động bất chấp sự phân vùng, thì có một số nút không bị hỏng có mất liên lạc với các nút khác và do đó phải quyết định tiếp tục xử lý các yêu cầu của máy khách để duy trì tính khả dụng (AP,hệ thống nhất quán cuối cùng)hoặc từ chối yêu cầu của khách hàng để duy trì đảm bảo tính nhất quán (CP). Tùy chọn đầu tiên vi phạm tính nhất quán, bởi vì nó có thể dẫn đến đọc cũ và viết xung đột, trong khi tùy chọn thứ hai rõ ràng là hy sinh tính khả dụng. Cũng có những hệ thống thường có sẵn và nhất quán, nhưng hoàn toàn không hoạt động khi có phân vùng (CA), ví dụ như hệ thống một nút. Người ta đã chứng minh rằng định lý CAP đúng cho mọi tính chất nhất quán

3[AXIT [23]:Một](#_bookmark41)tomicity,Ctính nhất quán,Tôian ủi,Đ.sự phát triển

4[CƠ SỞ [42]:](#_bookmark59)bvề cơ bảnMộtcó sẵn,Strạng thái bình thường,enhất quán

5Một thanh ghi đọc/ghi là một cấu trúc dữ liệu chỉ có hai thao tác: thiết lập một giá trị cụ thể.

giá trị (bộ)và trả về giá trị mới nhất đã được đặt (lấy).

điều đó ít nhất mạnh bằng tính nhất quán nhân quả, cũng bao gồm mọi giới hạn gần đây nhất về độ ổn định cho phép của dữ liệu (*∆*-atomicity) [37]. Khả năng tuần tự hóa như là tiêu chí chính xác của sự cô lập giao dịch không yêu cầu tính nhất quán cao. Tuy nhiên, tương tự như tính nhất quán, khả năng tuần tự hóa cũng không thể đạt được dưới các phân vùng mạng [15].

Việc phân loại các hệ thống NoSQL là AP, CP hoặc CA phản ánh một cách mơ hồ khả năng của các hệ thống riêng lẻ và do đó được chấp nhận rộng rãi như một phương tiện để so sánh cấp cao. Tuy nhiên, điều quan trọng cần lưu ý là Định lý CAP thực sự không phát biểu bất cứ điều gì trong hoạt động bình thường; nó chỉ cho chúng ta biết liệu một hệ thống ủng hộ tính khả dụng hay tính nhất quánkhi đối mặt với một phân vùng mạng. Ngược lại với Định lý FLP, định lý CAP giả định một mô hình lỗi cho phép các thông điệp tùy ý bị loại bỏ, sắp xếp lại hoặc trì hoãn không xác định. Theo giả định yếu hơn về các kênh liên lạc đáng tin cậy (tức là các tin nhắn luôn đến nhưng không đồng bộ và có thể được sắp xếp lại), trên thực tế, hệ thống CAP có thể sử dụng thuật toán Attiya, Bar-Noy, Dolev [2], miễn là phần lớn các nút đang hoạt động6.

PACELC.Việc thiếu Định lý CAP này được đề cập trong một bài báo của Daniel Abadi [1], trong đó ông chỉ ra rằng Định lý CAP không nắm bắt được sự trao đổi giữa độ trễ và tính nhất quán trong quá trìnhthông thườnghoạt động, mặc dù nó đã được chứng minh là có ảnh hưởng nhiều hơn đến thiết kế của các hệ thống phân tán so với sự đánh đổi tính khả dụng-tính nhất quán trong các tình huống lỗi. Ông xây dựng PACELC hợp nhất cả hai giao dịch và do đó mô tả không gian thiết kế của các hệ thống phân tán chính xác hơn. Từ PACELC, chúng tôi biết rằng trong trường hợpPnghệ thuật, có mộtMộtkhả dụng-Cgiao dịch nhất quán-o ;else, tức là trong hoạt động bình thường, có mộtLđộ chính xác-Cgiao dịch nhất quán-o .

Sự phân loại này về cơ bản cung cấp hai lựa chọn khả thi cho kịch bản phân vùng (A/C) và hai lựa chọn cho hoạt động bình thường (L/C) và do đó xuất hiện nhiều hơn ne-grained hơn so với phân loại CAP. Tuy nhiên, nhiều hệ thống không thể được gán riêng cho một lớp PACELC duy nhất và một trong bốn lớp PACELC, cụ thể là PC/EL, khó có thể được gán cho bất kỳ hệ thống nào.

# kỹ thuật

Mọi cơ sở dữ liệu thành công đáng kể đều được thiết kế cho một lớp ứng dụng cụ thể hoặc để đạt được sự kết hợp cụ thể của các thuộc tính hệ thống mong muốn. Lý do đơn giản giải thích tại sao có quá nhiều hệ thống cơ sở dữ liệu khác nhau là vì không một hệ thống nào có thể đạt được tất cả các thuộc tính mong muốn cùng một lúc. Các cơ sở dữ liệu SQL truyền thống như PostgreSQL đã được xây dựng để cung cấp gói chức năng đầy đủ: một mô hình dữ liệu rất linh hoạt, khả năng truy vấn tinh vi bao gồm các liên kết, ràng buộc toàn vẹn toàn cầu và đảm bảo giao dịch. Ở đầu bên kia của phạm vi thiết kế, có các kho lưu trữ khóa- giá trị như Dynamo mở rộng quy mô theo dữ liệu và khối lượng yêu cầu, đồng thời cung cấp thông lượng đọc và ghi cao cũng như độ trễ thấp, nhưng hầu như không có bất kỳ chức năng nào ngoài tra cứu đơn giản.

6Do đó, sự đồng thuận được sử dụng để phối hợp trong nhiều hệ thống NoSQL hoặc na- tích cực [4] hoặc thông qua các dịch vụ phối hợp như Chubby và Zookeeper [28] thậm chí còn khó đạt được với tính sẵn sàng cao hơn là tính nhất quán cao [19].

Trong phần này, chúng tôi nêu bật không gian thiết kế của các hệ thống cơ sở dữ liệu phân tán, tập trung vào phân mảnh, sao chép, quản lý lưu trữ và xử lý truy vấn. Chúng tôi khảo sát các kỹ thuật có sẵn và thảo luận về cách chúng liên quan đến các thuộc tính (mục tiêu) chức năng và phi chức năng khác nhau của các hệ thống quản lý dữ liệu. Để minh họa những kỹ thuật nào phù hợp để đạt được các thuộc tính hệ thống nào, chúng tôi cung cấpHộp công cụ NoSQL (Hình [4) trong đó mỗi kỹ thuật](#_bookmark10) được kết nối với các thuộc tính chức năng và phi chức năng mà nó cho phép (chỉ các cạnh dương).

chức năng kỹ thuật phi chức năng

Khả năng mở rộng dữ liệu

Phạm vi-Sharding Hash-Sharding

Chia cắt thực thể-nhóm

Băm nhất quán đĩa chia sẻ

Viết khả năng mở rộng

Đọc khả năng mở rộng

độ đàn hồi

Tính nhất quán

ghi độ trễ Đọc độ trễ

Viết thông lượng

Đọc sẵn có

Lập chỉ mục phụ toàn cầu Lập chỉ mục phụ cục bộ Lập kế hoạch truy vấn

Khung phân tích

Chế độ xem cụ thể hóa

Viết khả dụng

Độ bền

Tìm kiếm toàn văn

Xử lý truy vấn

ghi nhật ký

Cập nhật tại chỗ

Bộ nhớ đệm

Lưu trữ trong bộ nhớ

Bộ nhớ chỉ nối thêm

Quản lý lưu trữ

Đồng bộ Giao thức Cam kết/ Đồng thuận

không đồng bộ

Bản chính

Cập nhật mọi nơi

Nhân rộng

sharding

Tổng hợp và phân tích

Lọc truy vấn

Sắp xếp

tham gia

Viết có điều kiện hoặc nguyên tử

Giao dịch ACID

Truy vấn quét

Hình 4: Hộp công cụ NoSQL: Nó kết nối các kỹ thuật của cơ sở dữ liệu NoSQL với các thuộc tính hệ thống chức năng và phi chức năng mong muốn mà chúng hỗ trợ.

## Chia nhỏ

Một số hệ thống cơ sở dữ liệu quan hệ phân tán như Oracle RAC hoặc IBM DB2 pureScale dựa trên mộtkiến trúc đĩa chia sẻtrong đó tất cả các nút cơ sở dữ liệu truy cập vào cùng một kho lưu trữ dữ liệu trung tâm (ví dụ: NAS hoặc SAN). Do đó, các hệ thống này luôn cung cấp dữ liệu nhất quán, nhưng cũng rất khó mở rộng quy mô. Ngược lại, các hệ thống cơ sở dữ liệu (NoSQL) được tập trung trong bài viết này được xây dựng dựa trên một kiến trúc không chia sẻ,nghĩa là mỗi hệ thống bao gồm nhiều máy chủ có bộ nhớ riêng và đĩa riêng được kết nối qua mạng. Do đó, khả năng mở rộng cao về thông lượng và khối lượng dữ liệu đạt được bằng cáchsharding (phân vùng) dữ liệu trên các nút khác nhau (mảnh vỡ)trong hệ thống.

Có ba kỹ thuật phân phối cơ bản: phân mảnh phạm vi, phân mảnh băm và phân mảnh nhóm thực thể. Để có thể quét hiệu quả, dữ liệu có thể được phân vùng thành các phạm vi giá trị được sắp xếp theo thứ tự và liền kề theochia sẻ phạm vi.Tuy nhiên, cách tiếp cận này đòi hỏi một số sự phối hợp thông qua một bậc thầy quản lý các nhiệm vụ. Để đảm bảo tính linh hoạt, hệ thống phải có khả năng tự động phát hiện và giải quyết các điểm nóng bằng cách chia nhỏ thêm một phân đoạn quá tải.

Bảo vệ phạm vi được hỗ trợ bởi các kho lưu trữ nhiều cột như BigTable, HBase [hoặc Hypertable](#_bookmark67) [49] và các kho lưu trữ tài liệu, ví dụ: MongoDB, RethinkDB, Espresso

[43] và [DocumentDB [46].](#_bookmark64) Một cách khác để phân vùng dữ liệu trên nhiều máy là băm nhỏtrong đó mọi mục dữ liệu được gán cho một máy chủ phân đoạn theo một số giá trị băm được tạo từ khóa chính. Cách tiếp cận này không yêu cầu điều phối viên và cũng đảm bảo dữ liệu được phân phối đồng đều trên các phân đoạn, miễn là hàm băm được sử dụng tạo ra phân phối đồng đều. Tuy nhiên, nhược điểm rõ ràng là nó chỉ cho phép tra cứu và khiến việc quét không khả thi. Phân đoạn hàm băm được sử dụng trong các kho lưu trữ khóa-giá trị và cũng có sẵn trong một số kho lưu trữ nhiều cột như Cassandra [34] hoặc Azure Tables [8].

Ví dụ, máy chủ phân đoạn chịu trách nhiệm về một bản ghi có thể được xác định là *người phục vụTôi*=*băm*(*Tôi*)*máy chủ mod*. Tuy nhiên, lược đồ băm này yêu cầu tất cả các bản ghi phải được chỉ định lại mỗi khi một máy chủ mới tham gia hoặc rời đi, bởi vì nó thay đổi theo số lượng máy chủ phân đoạn (*may chủ*), vì vậy nó thực sự không được sử dụng trong các hệ thống đàn hồi như Dynamo, Riak hoặc Cassandra, cho phép thêm các tài nguyên bổ sung theo yêu cầu và lại bị xóa khi cần thiết. Thay vào đó, hệ thống đàn hồi sử dụngbăm nhất quán [30] trong đó chỉ một phần dữ [liệu phải](#_bookmark48) được gán lại khi hệ thống thay đổi như vậy.

Phân mảnh nhóm thực thểlà một sơ đồ phân vùng dữ liệu với mục tiêu cho phép các giao dịch một phân vùng trên dữ liệu cùng vị trí. Các phân vùng được gọi là nhóm thực thể và được ứng dụng khai báo rõ ràng (ví dụ: trong G-Store [14] và MegaStore [[4]) hoặc được](#_bookmark32) lấy từ các mẫu truy cập của giao dịch (ví dụ: trong Relational Cloud [13] và Cloud SQL Server [5 ]). [Nếu](#_bookmark31) một giao dịch truy cập dữ [liệu](#_bookmark23) mở rộng trên nhiều nhóm, thì quyền sở hữu dữ liệu có thể được chuyển giao giữa các nhóm thực thể hoặc trình quản lý giao dịch phải chuyển sang các giao thức giao dịch đa nút đắt tiền hơn.

## Sao chép

Về mặt CAP, các RDBMS thông thường thường là các hệ thống CA chạy ở chế độ một máy chủ: Toàn bộ hệ thống sẽ không khả dụng khi máy bị lỗi. Và do đó, các nhà khai thác hệ thống đảm bảo tính toàn vẹn và tính khả dụng của dữ liệu thông qua phần cứng cao cấp đắt tiền nhưng đáng tin cậy. Ngược lại, các hệ thống NoSQL như Dynamo, BigTable hoặc Cassandra được thiết kế cho khối lượng dữ liệu và yêu cầu mà một máy đơn lẻ không thể xử lý được và do đó chúng chạy trên các cụm bao gồm hàng nghìn máy chủ7. Vì lỗi là [không](#_bookmark11) thể tránh khỏi và sẽ xảy ra thường xuyên trong bất kỳ hệ thống phân tán quy mô lớn nào, nên phần mềm phải đối phó với chúng hàng ngày [24]. Năm 2009, [Je](#_bookmark42) Dean, một đồng nghiệp của Google, đã tuyên bố [16] rằng một cụm mới điển hình tại Google gặp phải hàng nghìn lỗi ổ cứng, 1.000 lỗi máy đơn, 20 lỗi giá đỡ và một số phân vùng mạng do các trường hợp dự kiến và không mong muốn chỉ trong năm đầu tiên. Nhiều trường hợp gần đây về phân vùng mạng và ngừng hoạt động trong các trung tâm dữ liệu đám mây lớn đã được báo cáo [3]. Sao chép cho phép [hệ](#_bookmark21) thống duy trì tính khả dụng và độ bền khi đối mặt với các lỗi như vậy. Nhưng lưu trữ các bản ghi giống nhau trên các máy khác nhau (máy chủ bản sao)trong cụm giới thiệu vấn đề đồng bộ hóa giữa chúng

7Phần cứng cấp thấp được sử dụng vì nó tiết kiệm chi phí hơn đáng kể so với phần cứng cấp cao.

phần cứng cuối [27, Mục 3.1].

## và do đó, một mặt là sự đánh đổi giữa tính nhất quán và mặt khác là độ trễ và tính khả dụng.

### Xám và cộng sự. [22] đề xuất một phân loại hai cấp của các chiến lược sao chép khác nhau theokhi nàocác bản cập nhật được truyền đến các bản sao vàở đâucập nhật được chấp nhận. Có hai lựa chọn có thể có ở cấp một ( khi ):Hăng hái (đồng bộ)bản sao lan truyền các thay đổi đến một cách đồng bộ tới tất cả các bản sao trước khi một cam kết có thể được trả lại cho máy khách, trong khilười biếng ( không đồng bộ) sao chép chỉ áp dụng các thay đổi tại bản sao nhận và chuyển chúng không đồng bộ. Ưu điểm lớn củahăng háisao chép là tính nhất quán giữa các bản sao, nhưng phải trả giá bằng độ trễ ghi cao hơn do phải chờ các bản sao [khác và tính khả dụng bị suy giảm [22].Lười biếng](#_bookmark40)sao chép nhanh hơn, bởi vì nó cho phép các bản sao phân kỳ; do đó, dữ liệu cũ có thể được cung cấp. Ở tầng thứ hai ( where ), một lần nữa, có thể có hai cách tiếp cận khác nhau: Hoặc làchủ-tớ ( bản chính)lược đồ được theo đuổi khi các thay đổi chỉ có thể được chấp nhận bởi một bản sao (chính) hoặc, trong mộtcập nhật ở bất cứ đâu (đa chủ)cách tiếp cận, mọi bản sao đều có thể chấp nhận ghi. Trongchủ nôgiao thức, điều khiển đồng thời không phức tạp hơn trong một hệ thống phân tán không có bản sao, nhưng toàn bộ bộ bản sao sẽ không khả dụng ngay khi bản chính bị lỗi. Các giao thức đa chủ yêu cầu các cơ chế phức tạp để ngăn ngừa hoặc phát hiện và điều chỉnh các thay đổi xung đột. Các kỹ thuật thường được sử dụng cho các mục đích này là lập phiên bản, đồng hồ véc-tơ, sửa lỗi sai và đọc (ví dụ: trong Dynamo [18]) và các kiểu dữ liệu hội tụ hoặc giao hoán [45] (ví dụ: trong Riak).

Về cơ bản, tất cả bốn kết hợp của phân loại hai tầng đều có thể thực hiện được. Hệ thống quan hệ phân tán thường thực hiệnchủ nô háo hứcsao chép để duy trì tính nhất quán mạnh mẽ.Háo hức cập nhật mọi lúc mọi nơisao chép như ví dụ đặc trưng trong các sản phẩm Megastore của Google khỏi chi phí liên lạc nặng do đồng bộ hóa tạo ra và có thể gây ra các bế tắc phân tán rất tốn kém để phát hiện. Các hệ thống cơ sở dữ liệu NoSQL thường dựa vàolười biếngsao chép, kết hợp với master-slave (hệ thống CP, ví dụ HBase và MongoDB) hoặc phương pháp cập nhật ở mọi nơi (hệ thống AP, ví dụ Dynamo và Cassandra). Nhiều hệ thống NoSQL để máy khách lựa chọn giữa độ trễ và tính nhất quán, nghĩa là đối với mọi yêu cầu, máy khách quyết định có chờ phản hồi từ bất kỳ bản sao nào để đạt được độ trễ tối thiểu hay để có phản hồi chắc chắn nhất quán (bởi phần lớn các bản sao hoặc master) để ngăn dữ liệu cũ.

Một khía cạnh của bản sao không được đề cập trong sơ đồ hai tầng là khoảng cách giữa các bản sao. Lợi thế rõ ràng của việc đặt các bản sao gần nhau là độ trễ thấp, nhưng việc đặt các bản sao ở gần nhau cũng có thể làm giảm tác động tích cực đến tính khả dụng; ví dụ: nếu hai bản sao của cùng một mục dữ liệu được đặt trong cùng một giá, thì mục dữ liệu đó không khả dụng khi giá bị lỗi mặc dù đã sao chép. Một kỹ thuật thay thế để giảm độ trễ được sử dụng trong Orestes [20], trong đó dữ liệu được [lưu vào bộ đệm](#_bookmark38) ẩn gần với các ứng dụng sử dụng cơ sở hạ tầng bộ nhớ đệm web và các giao thức kết hợp bộ đệm.

Sao chép địa lý có thể bảo vệ hệ thống khỏi mất dữ liệu hoàn toàn và cải thiện độ trễ đọc đối với quyền truy cập phân tán từ máy khách.Hăng háisao chép địa lý, như được triển khai [trong Google's Megastore](#_bookmark22) [4], Spanner [[12],](#_bookmark30) MDCC [32] và Mạnh Tử

[[38]](#_bookmark56) đạt được tính nhất quán cao với chi phí là độ trễ ghi cao hơn (thường là 100 mili giây

[[12] đến 600](#_bookmark30) mili giây [4]). Vớilười biếngsao chép địa lý như trong Dynamo [18], PNUTS [11], [Walter](#_bookmark29) [47], [COPS](#_bookmark65) [36], [Cassandra](#_bookmark54) [34] và [BigTable](#_bookmark51) [9] các thay đổi [gần](#_bookmark26) đây có thể bị mất, nhưng hệ thống hoạt động tốt hơn và vẫn khả dụng trong quá trình phân vùng . Charron- Bost và cộng sự. [10, [Chương](#_bookmark28) 12] và Öszu và Valduriez [41, Chương 13] [cung](#_bookmark60) cấp một cuộc thảo luận toàn diện về sao chép cơ sở dữ liệu.

## Quản lý lưu trữ

## 

Hình 5: Kim tự tháp lưu trữ và vai trò của nó trong các hệ thống NoSQL.

Để có hiệu suất tốt nhất, các hệ thống cơ sở dữ liệu cần được tối ưu hóa cho phương tiện lưu trữ mà chúng sử dụng để phục vụ và duy trì dữ liệu. Đây thường là bộ nhớ chính (RAM), ổ đĩa thể rắn (SSD) và ổ đĩa quay (HDD) có thể được sử dụng trong mọi kết hợp. Không giống như RDBMS trong thiết lập doanh nghiệp, cơ sở dữ liệu NoSQL phân tán tránh các kiến trúc đĩa chia sẻ chuyên dụng để ưu tiên các cụm không chia sẻ dựa trên máy chủ hàng hóa (sử dụng phương tiện lưu trữ hàng hóa). Các thiết bị lưu trữ thường được hình dung dưới dạng kim tự tháp lưu trữ (xem Hình 5) [25]. Ngoài ra còn có một tập hợp các bộ đệm trong suốt (ví dụ: bộ đệm CPU L1-L3 và bộ đệm đĩa, không được hiển thị trong Hình), chỉ được tận dụng hoàn toàn thông qua các thuật toán cơ sở dữ liệu được thiết kế tốt nhằm thúc đẩy vị trí dữ liệu. Các đặc điểm hiệu [suất và chi phí](#_bookmark43) rất khác nhau của RAM, Lưu trữ SSD và HDD và các chiến lược khác nhau để tận dụng điểm mạnh của chúng (quản lý lưu trữ) là một trong những lý do tạo nên sự đa dạng của cơ sở dữ liệu NoSQL. Quản lý lưu trữ có kích thước không gian (nơi lưu trữ dữ liệu) và kích thước thời gian (thời điểm lưu trữ dữ liệu). Update-in-place và append-only-IO là hai kỹ thuật tổ chức dữ liệu không gian bổ sung; trong bộ nhớ quy định RAM là vị trí của dữ liệu, trong khi ghi nhật ký là một kỹ thuật tạm thời giúp tách rời bộ nhớ chính và bộ lưu trữ liên tục và do đó cung cấp quyền kiểm soát khi dữ liệu thực sự được lưu giữ. Update-in-place và append-only-IO là hai kỹ thuật tổ chức dữ liệu không gian bổ sung; trong bộ nhớ quy định RAM là vị trí của dữ liệu, trong khi ghi nhật ký là một kỹ thuật tạm thời giúp tách rời bộ nhớ chính và bộ lưu trữ liên tục và do đó cung cấp quyền kiểm soát khi dữ liệu thực sự được lưu giữ. Update-in-place và append-only-IO là hai kỹ thuật tổ chức dữ liệu không gian bổ sung; trong bộ nhớ quy định RAM là vị trí của dữ liệu, trong khi ghi nhật ký là một kỹ thuật tạm thời giúp tách rời bộ nhớ chính và bộ lưu trữ liên tục và do đó cung cấp quyền kiểm soát khi dữ liệu thực sự được lưu giữ.

Trong bài báo chuyên đề về sự kết thúc của một kỷ nguyên [kiến trúc [48],](#_bookmark66) Stonebraker et al. đã phát hiện ra rằng trong các RDBMS điển hình, chỉ có 6,8% thời gian thực thi được dành cho

công việc hữu ích , trong khi phần còn lại được dành cho:

quản lý bộ đệm (34,6%), tức là bộ nhớ đệm để giảm thiểu truy cập đĩa chậm hơn

chốt (14,2%), để bảo vệ cấu trúc dữ liệu được chia sẻ khỏi các điều kiện tương tranh do đa luồng gây ra

### khóa (16,3%), để đảm bảo cách ly hợp lý nhật ký giao dịch (11,9%), để đảm bảo độ bền khi đối mặt với các lỗi tối ưu hóa được mã hóa thủ công (16,2%)

Điều này thúc đẩy rằng có thể mong đợi những cải tiến lớn về hiệu suất nếu RAM được sử dụng làm bộ lưu trữ chính (trong trí nhớcơ sở dữ liệu [50]). Nhược điểm là chi phí lưu trữ cao và thiếu độ bền, một sự cố mất điện nhỏ có thể phá hủy trạng thái cơ sở dữ liệu. Điều này có thể được giải quyết theo hai cách: Trạng thái có thể được nhân rộng trên*N*các nút máy chủ trong bộ nhớ bảo vệ chống lại*n −*1lỗi nút đơn (ví dụ: HStore, VoltDB [29]) hoặc bởi [khai thác gỗsang lưu trữ lâu](#_bookmark47) bền (ví dụ: Redis hoặc SAP Hana). Thông qua ghi nhật ký, một mẫu truy cập ghi ngẫu nhiên có thể được chuyển đổi thành một mẫu tuần tự bao gồm các thao tác đã nhận và các thuộc tính liên quan của chúng (ví dụ: thông tin làm lại). Trong hầu hết các hệ thống NoSQL, quy tắc cam kết để ghi nhật ký được tôn trọng, quy tắc này yêu cầu mọi thao tác ghi được xác nhận là thành công phải được ghi lại và nhật ký sẽ được đưa vào bộ lưu trữ liên tục. Để tránh độ trễ quay của ổ cứng HDD phát sinh bằng cách ghi nhật ký từng hoạt động riêng lẻ, các lần ghi nhật ký có thể được nhóm lại với nhau (cam kết nhóm), điều này làm tăng nhẹ độ trễ của từng thao tác ghi, nhưng cải thiện đáng kể thông lượng.

## SSD và nói chung là tất cả các thiết bị lưu trữ dựa trên bộ nhớ tro NAND về cơ bản khác với HDD ở nhiều khía cạnh: (1) tốc độ đọc và ghi không đối xứng,

(2) không ghi đè tại chỗ phải xóa toàn bộ khối trước khi ghi đè bất kỳ trang nào trong khối đó và (3) chu trình xóa/chương trình hạn chế [40]. Do đó, quản [lý](#_bookmark58) lưu trữ của hệ thống cơ sở dữ liệu không được coi SSD và HDD là RAM liên tục, chậm hơn một chút, vì ghi ngẫu nhiên vào SSD chậm hơn gần một bậc so với ghi tuần tự. Mặt khác, các lần đọc ngẫu nhiên có thể được thực hiện mà không có bất kỳ hình phạt nào về hiệu suất. Có một số hệ thống cơ sở dữ liệu (ví dụ: Oracle Exadata, Aerospike) được thiết kế rõ ràng cho các đặc tính hiệu suất này của SSD. Trong ổ cứng, cả đọc và ghi ngẫu nhiên đều chậm hơn

10-100 lần so với truy cập tuần tự.

Đối với cơ sở dữ liệu trong bộ nhớ, mộtcập nhật tại chỗmẫu truy cập là lý tưởng: Nó đơn giản hóa việc triển khai và ghi ngẫu nhiên vào RAM về cơ bản nhanh như nhau như các mẫu tuần tự, với những khác biệt nhỏ bị ẩn bởi đường ống và hệ thống phân cấp bộ đệm CPU. Tuy nhiên, RDBMS và nhiều hệ thống NoSQL (ví dụ: MongoDB) cũng sử dụng mẫu cập nhật tại chỗ để lưu trữ liên tục. Để giảm thiểu việc truy cập ngẫu nhiên chậm vào bộ lưu trữ liên tục, bộ nhớ chính thường được sử dụng làm bộ đệm và được bổ sung bằng cách ghi nhật ký để đảm bảo độ bền. Trong các RDBMS, điều này đạt được thông qua một nhóm bộ đệm phức tạp không chỉ sử dụng các thuật toán thay thế bộ đệm thích hợp cho các mẫu truy cập dựa trên SQL điển hình mà còn đảm bảo ngữ nghĩa ACID. Cơ sở dữ liệu NoSQL có vùng đệm đơn giản hơn, thu được lợi nhuận từ các truy vấn đơn giản hơn và thiếu các giao dịch ACID. Giải pháp thay thế cho mô hình vùng đệm là để lại bộ nhớ đệm cho HĐH thông qua bộ nhớ ảo (ví dụ: được sử dụng trong công cụ lưu trữ MMAP của MongoDB). Điều này đơn giản là cơ sở dữ liệu

kiến trúc, nhưng có nhược điểm là cung cấp ít quyền kiểm soát hơn đối với mục hoặc trang dữ liệu nào nằm trong bộ nhớ và khi chúng bị loại bỏ. Ngoài ra, tính năng đọc trước (đọc suy đoán) và ghi sau (bộ đệm ghi) được thực hiện minh bạch với bộ đệm hệ điều hành thiếu sự phức tạp vì chúng dựa trên logic hệ thống tập tin thay vì truy vấn cơ sở dữ liệu.

chỉ nối thêmlưu trữ (còn được gọi là cấu trúc nhật ký) cố gắng tối đa hóa thông lượng bằng cách ghi tuần tự. Mặc dù các hệ thống tập tin có cấu trúc nhật ký đã có một lịch sử nghiên cứu lâu dài, nhưng I/O chỉ gắn thêm mới chỉ được phổ biến gần đây cho cơ sở dữ liệu nhờ BigTable sử dụng các cây Hợp nhất có [cấu trúc nhật ký (LSM) [9] bao](#_bookmark26) gồm một bộ nhớ đệm trong bộ nhớ, một log và các tập tin lưu trữ được ghi định kỳ, không thay đổi. Các cây LSM và các biến thể như Sorted Array Merge Trees (SAMT) và Cache-Oblivious Look-ahead Arrays (COLA) đã được áp dụng trong nhiều hệ thống NoSQL (Cassandra, CouchDB, LevelDB, RethinkDB, RocksDB, In uxDB, TokuDB) [31].

Thiết kế cơ sở dữ liệu để đạt được hiệu suất ghi tối đa bằng cách luôn ghi vào nhật ký khá đơn giản, khó khăn nằm ở việc cung cấp các lần đọc tuần tự và ngẫu nhiên nhanh chóng. Điều này yêu cầu một cấu trúc chỉ mục thích hợp được cập nhật vĩnh viễn dưới dạng cấu trúc dữ liệu sao chép khi ghi (COW) (ví dụ: COW B-tree của CouchDB) hoặc chỉ được duy trì định kỳ dưới dạng cấu trúc dữ liệu bất biến (ví dụ: trong các hệ thống kiểu BigTable). Một vấn đề của tất cả các phương pháp lưu trữ có cấu trúc nhật ký là bộ sưu tập rác (nén) tốn kém để lấy lại không gian của các mục đã cập nhật hoặc đã xóa.

## Xử lý truy vấn

### Khả năng truy vấn của cơ sở dữ liệu NoSQL chủ yếu tuân theo mô hình phân phối, đảm bảo tính nhất quán và mô hình dữ liệu của nó.Tra cứu khóa chính,

tức là truy xuất các mục dữ liệu bằng một ID duy nhất, được mọi hệ thống NoSQL hỗ trợ, vì nó tương thích với phân vùng phạm vi cũng như phân vùng băm.Lọc truy vấn trả về tất cả các mục (hoặc phép chiếu) đáp ứng một vị từ được chỉ định trên các thuộc tính của các mục dữ liệu từ một bảng. Ở dạng đơn giản nhất, chúng có thể được thực hiện nhưquét toàn bộ bảng đã lọc.Đối với cơ sở dữ liệu được phân vùng băm, điều này có nghĩa làphân tán-tụ tập mẫu trong đó mỗi phân vùng thực hiện quét dự đoán và kết quả được hợp nhất. Đối với các hệ thống được phân vùng theo phạm vi, bất kỳ điều kiện nào trên thuộc tính phạm vi đều có thể được khai thác để chọn các phân vùng.

Để phá vỡ sự bất hợp lý của*Ô*(*N*)quét, các chỉ mục phụ có thể được sử dụng. Đây có thể làchỉ số phụ cục bộđược quản lý trong mỗi phân vùng hoặcchỉ số phụ toàn cầudữ liệu chỉ mục đó trên tất cả các phân vùng [4]. Vì bản thân chỉ mục toàn cầu phải được [phân](#_bookmark22) phối trên các phân vùng, nên việc duy trì chỉ mục thứ cấp nhất quán sẽ yêu cầu các giao thức cam kết chậm và có khả năng không khả dụng. Do đó, trên thực tế, hầu hết các hệ thống chỉ cung cấp tính nhất quán cuối cùng cho các chỉ mục này (ví dụ: Megastore, Google AppEngine Datastore, DynamoDB) hoặc hoàn toàn không hỗ trợ chúng (ví dụ: HBase, Azure Tables). Khi thực hiện các truy vấn toàn cầu trên các chỉ mục phụ cục bộ, truy vấn chỉ có thể được nhắm mục tiêu đến một tập hợp con các phân vùng nếu vị từ truy vấn và các quy tắc phân vùng giao nhau. Mặt khác, kết quả phải được tập hợp thông qua tập hợp phân tán. Ví dụ: một bảng người dùng có phân vùng phạm vi trên một trường tuổi có thể phục vụ các truy vấn có điều kiện bình đẳng về tuổi từ một phân vùng trong khi các truy vấn trên tên cần

được đánh giá tại mỗi phân vùng. Một trường hợp đặc biệt của lập chỉ mục phụ toàn cầu là tìm kiếm toàn văn bản, trong đó các trường đã chọn hoặc các mục dữ liệu hoàn chỉnh được đưa vào chỉ mục đảo ngược bên trong cơ sở dữ liệu (ví dụ: MongoDB) hoặc vào một nền tảng tìm kiếm bên ngoài như ElasticSearch hoặc Solr (Riak Search, DataStax) Cassandra).

## Lập kế hoạch truy vấnlà nhiệm vụ tối ưu hóa kế hoạch truy vấn để giảm

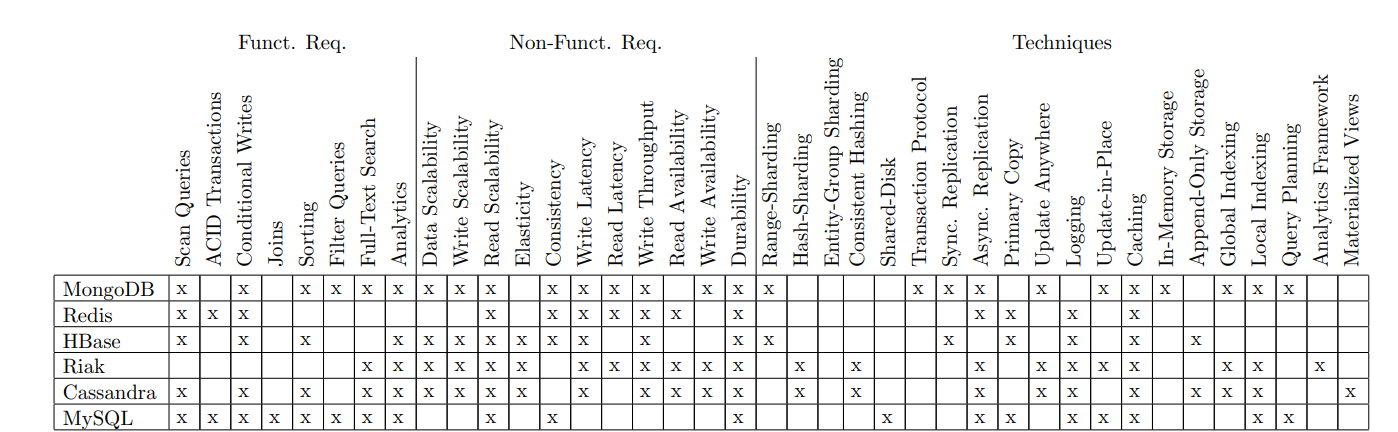
thiểu [chi phí thực](#_bookmark43) hiện [25]. Đối với các tập hợp và liên kết, lập kế hoạch truy vấn là điều cần thiết vì các truy vấn này rất kém hiệu quả và khó thực hiện trong mã ứng dụng. Sự phong phú của tài liệu và kết quả về xử lý truy vấn quan hệ phần lớn bị bỏ qua trong các hệ thống NoSQL hiện tại vì hai lý do. Đầu tiên, mô hình khóa-giá trị và cột rộng được tập trung xung quanh CRUD và quét các hoạt động trên các khóa chính để lại ít chỗ cho việc tối ưu hóa truy vấn. Thứ hai, hầu hết công việc xử lý truy vấn phân tán tập trung vào khối lượng công việc OLAP ưu tiên thông lượng hơn độ trễ trong khi tối ưu hóa truy vấn một nút không dễ dàng áp dụng cho cơ sở dữ liệu được phân vùng và sao chép. Tuy nhiên,số 8.

Phân tích trong cơ sở dữ liệucó thể được thực hiện tự nhiên (ví dụ: trong MongoDB, Riak, CouchDB) hoặc thông qua các nền tảng phân tích bên ngoài như Hadoop, Spark và Flink (ví dụ: trong Cassandra và HBase). Sự trừu tượng hóa phân tích hàng loạt bản địa phổ biến được đưa ra bởi các hệ thống NoSQL là MapReduce [17]. Do I/O, chi phí giao tiếp và tối ưu hóa kế hoạch thực hiện hạn chế, các phương pháp tiếp cận theo định hướng theo lô và vi mô này có thời gian phản hồi cao.Quan điểm cụ thể hóalà một giải pháp thay thế với thời gian phản hồi truy vấn thấp hơn. Chúng được khai báo tại thời điểm thiết kế và được cập nhật liên tục khi thay đổi hoạt động (ví dụ: trong CouchDB và Cassandra). Tuy nhiên, tương tự như lập chỉ mục thứ cấp toàn cầu, tính nhất quán của chế độ xem thường được nới lỏng để hỗ trợ ghi nhanh, có tính sẵn sàng cao khi hệ thống được phân phối. Vì chỉ có một số hệ thống cơ sở dữ liệu có hỗ trợ tích hợp để nhập và truy vấn các luồng dữ liệu không giới hạn,phân tích gần thời gian thực đường ống thường thực hiện một trong haiKiến trúc Lambda [39] [hoặc](#_bookmark57)Kiến trúc Kappa [33]: Cái trước bổ [sung](#_bookmark52) cho khung xử lý hàng loạt như Hadoop MapReduce với bộ xử lý luồng như Storm [6] và cái sau hoàn toàn dựa vào xử lý luồng và bỏ [qua](#_bookmark24) hoàn toàn xử lý hàng loạt.

1. Nghiên cứu điển hình về hệ thống

Trong phần này, chúng tôi cung cấp một so sánh định tính của một số kho lưu trữ khóa-giá trị, tài liệu và cột rộng nổi bật nhất. Chúng tôi trình bày kết quả dưới dạng so sánh cô đọng và tham khảo tài liệu của các hệ thống riêng lẻ để biết thông tin chi tiết. Hộp công cụ NoSQL được đề xuất (xem Hình 4, trang 6) là một phương tiện trừu tượng có [thể được](#_bookmark10) sử dụng để phân loại các hệ thống cơ sở dữ liệu theo ba khía cạnh: yêu cầu chức năng, yêu cầu phi chức năng và các kỹ thuật được sử dụng để triển khai chúng. Chúng tôi lập luận rằng sự phân loại này đặc trưng cho nhiều hệ thống cơ sở dữ liệu và do đó có thể được sử dụng để đối chiếu một cách có ý nghĩa các hệ thống cơ sở dữ liệu khác nhau: Bảng 1 cho thấy sự so sánh trực [tiếp](#_bookmark15)

số 8Hiện tại chỉ có RethinkDB có thể thực hiện chung*θ*-tham gia. tổng hợp của MongoDB framework có hỗ trợ cho các phép nối equi bên ngoài bên trái trong khung tổng hợp của nó và CouchDB cho phép các phép nối cho các chế độ xem thu nhỏ bản đồ được khai báo trước.

của MongoDB, Redis, HBase, Riak, Cassandra và MySQL trong cấu hình mặc định tương ứng của chúng. Bảng so sánh chi tiết hơn về các thuộc tính của hệ thống trung tâm được trình [bày trong Bảng 2 (xem trang](#_bookmark18) 15).

## Bảng 1: So sánh trực tiếp các yêu cầu chức năng, yêu cầu phi chức năng và kỹ thuật giữa MongoDB, Redis, HBase, Riak, Cassandra và MySQL theo Hộp công cụ NoSQL của chúng tôi.

### Sự so sánh làm sáng tỏ cách cơ sở dữ liệu SQL và NoSQL được thiết kế để đáp ứng các nhu cầu rất khác nhau: RDBMS cung cấp mức chức năng chưa từng có trong khi cơ sở dữ liệu NoSQL vượt trội ở khía cạnh phi chức năng thông qua khả năng mở rộng, tính khả dụng, độ trễ thấp và/hoặc thông lượng cao. Tuy nhiên, cũng có sự khác biệt lớn giữa các cơ sở dữ liệu NoSQL. Ví dụ, Riak và Cassandra có thể được cấu hình để đáp ứng nhiều yêu cầu phi chức năng, nhưng cuối cùng chỉ nhất quán và không có nhiều khả năng chức năng ngoài phân tích dữ liệu và, trong trường hợp của Cassandra, cập nhật có điều kiện. Mặt khác, MongoDB và HBase cung cấp tính nhất quán mạnh mẽ hơn và các khả năng chức năng phức tạp hơn như truy vấn quét và chỉ truy vấn MongoDB: lter, nhưng không duy trì khả năng đọc và ghi trong các phân vùng và có xu hướng hiển thị độ trễ đọc cao hơn. Redis, với tư cách là hệ thống không phân vùng duy nhất trong so sánh này ngoài MySQL, cho thấy một tập hợp các giao dịch đặc biệt tập trung vào khả năng duy trì thông lượng cực cao ở độ trễ thấp bằng cách sử dụng cấu trúc dữ liệu trong bộ nhớ và sao chép chính-phụ không đồng bộ .

5. Kết Luận

Chọn một hệ thống cơ sở dữ liệu luôn có nghĩa là chọn một tập hợp các thuộc tính mong muốn hơn một tập hợp các thuộc tính khác. Để phá vỡ sự phức tạp của [lựa chọn này, chúng tôi trình bày một](#_bookmark17) cây quyết định nhị phân trong Hình 6 ánh xạ các quyết định trao đổi cho các ứng dụng mẫu và các hệ thống cơ sở dữ liệu phù hợp tiềm năng. Các nút lá bao gồm các ứng dụng từ bộ nhớ đệm đơn giản (trái) đến phân tích Dữ liệu lớn (phải). Đương nhiên, quan điểm này về không gian vấn đề là không đầy đủ, nhưng nó mơ hồ chỉ ra một giải pháp cho một vấn đề quản lý dữ liệu cụ thể. Sự phân chia đầu tiên trong cây là dọc theo mẫu truy cập của ứng dụng: Chúng chỉ dựa vào tra cứu nhanh (nửa bên trái) hoặc yêu cầu khả năng truy vấn phức tạp hơn (nửa bên phải). Các ứng dụng tra cứu nhanh có thể được phân biệt thêm bằng khối lượng dữ liệu mà chúng xử lý: Nếu bộ nhớ chính của một máy duy nhất có thể chứa tất cả dữ liệu,

tùy thuộc vào việc chức năng (Redis) hay tính đơn giản (Memcache) được ưu tiên. Nếu khối lượng dữ liệu đang hoặc có thể tăng vượt quá dung lượng RAM hoặc thậm chí không bị giới hạn, thì một hệ thống nhiều nút chia tỷ lệ theo chiều ngang có thể phù hợp hơn. Quyết định quan trọng nhất trong trường hợp này là ưu tiên tính khả dụng (AP) hay tính nhất quán (CP) như được mô tả bởi định lý CAP. Các hệ thống như Cassandra và Riak có thể mang lại trải nghiệm luôn bật, trong khi các hệ thống như HBase, MongoDB và DynamoDB mang lại sự nhất quán mạnh mẽ.

Truy cập

Tra cứu nhanh

Truy vấn phức tạp

Âm lượng

Âm lượng

ĐẬP

không giới hạn

Kích thước ổ cứng

không giới hạn

MŨ LƯỠI TRAI

Tính nhất quán

Mẫu truy vấn

AP

CP AXIT

khả dụng đặc biệt

phân tích

**làm lại Cassandra HBase RDBMS Đi văngDB MongoDB Hadoop, tia lửa**

bộ nhớ đệm **đồng riak** MongoDB neo4j **MongoDB** Suy nghĩ lại về DB **DWH song song** Voldermort Đi văngCơ sở RavenDB Đơn giảnDB HBase, Tích lũy Cassandra, HBase khí cầu DynamoDB đánh dấuLogic Đàn hồi, Solr Riak, MongoDB

Bộ nhớ cache

Mua sắm-

rổ

Gọi món

Lịch sử

OLTP

Trang mạng

Xã hội

Mạng

Dữ liệu lớn

Ứng dụng ví dụ

Hình 6: Cây quyết định để ánh xạ các yêu cầu tới hệ thống cơ sở dữ liệu (NoSQL).

Nửa bên phải của cây bao gồm các ứng dụng yêu cầu truy vấn phức tạp hơn so với tra cứu đơn giản. Ở đây cũng vậy, đầu tiên chúng tôi phân biệt các hệ thống theo khối lượng dữ liệu mà chúng phải xử lý tùy theo việc hệ thống một nút có khả thi hay không (kích thước ổ cứng) hay yêu cầu phân phối (khối lượng không giới hạn). Đối với khối lượng công việc OLTP phổ biến trên khối lượng dữ liệu lớn vừa phải, RDBMS truyền thống hoặc cơ sở dữ liệu đồ thị như Neo4J là tối ưu vì chúng cung cấp ngữ nghĩa ACID. Tuy nhiên, nếu tính khả dụng là điều cốt yếu, thì các hệ thống phân tán như MongoDB, CouchDB hoặc DocumentDB sẽ được ưu tiên hơn.

Nếu khối lượng dữ liệu vượt quá giới hạn của một máy, thì việc lựa chọn hệ thống phù hợp sẽ phụ thuộc vào mẫu truy vấn phổ biến: Khi các truy vấn phức tạp phải được tối ưu hóa về độ trễ, chẳng hạn như trong các ứng dụng mạng xã hội, thì MongoDB rất hấp dẫn, bởi vì nó tạo điều kiện cho các truy vấn ad-hoc biểu cảm. HBase và Cassandra cũng hữu ích trong trường hợp như vậy, nhưng vượt trội trong phân tích Dữ liệu lớn được tối ưu hóa thông qua khả năng, khi được kết hợp với Hadoop.

### Tóm lại, chúng tôi tin rằng mô hình từ trên xuống được đề xuất là một hỗ trợ

ra quyết định hiệu quả để lọc số lượng lớn hệ thống cơ sở dữ liệu NoSQL dựa trên các yêu cầu trung tâm. Ngoài ra, Hộp công cụ NoSQL còn cung cấp ánh xạ từ các yêu cầu chức năng và phi chức năng tới các kỹ thuật triển khai chung để phân loại không gian NoSQL không ngừng phát triển.

Bảng 2: Một số tính chất của MongoDB, HBase, Cassandra, Riak và Redik

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Kích thước | MongoDB | HBase | Cassandra | | đồng riak | | làm lại | |
| Người mẫu | Tài liệu | Cột rộng | Cột rộng | | Giá trị cốt lõi | | Giá trị cốt lõi | |
| MŨ LƯỠI TRAI | CP | CP | AP | | AP | | CP | |
| Hiệu suất quét | Cao (với mức độ phù hợp  khóa mảnh) | Cao (chỉ trên phím hàng) | Cao (dùng hợp chất  mục lục) | | không áp dụng | | Cao (phụ thuộc vào cấu trúc  dữ liệu) | |
| Độ trễ đĩa trên mỗi  Nhận bằng phím Row | *∼*Một số đĩa tìm kiếm | *∼*Một số đĩa tìm kiếm | *∼*Một số đĩa tìm kiếm | | *∼*Tìm kiếm một đĩa | | Trong trí nhớ | |
| Viết hiệu suất | Cao (I/O chỉ nối thêm) | Cao (I/O chỉ nối thêm) | Cao (I/O chỉ nối thêm) | | Cao (I/O chỉ nối thêm) | | Rất cao, trong bộ nhớ | |
| Độ trễ mạng | Có thể định cấu hình: nô lệ gần  nhất, chủ (“ưu tiên đọc”) Có thể | Máy chủ khu vực được chỉ định | Có thể định cấu hình: Bản sao R  đã liên hệ | | Có thể định cấu hình: Bản sao R  đã liên hệ | | chủ được chỉ định | |
| Độ bền | định cấu hình: không có, WAL,  sao chép (“mối quan tâm ghi”) | WAL, cấp hàng  phiên bản | Bản sao WAL, W được viết | | Có thể định cấu hình: ghi, ghi  lâu bền, ghi bản sao W | | Có thể định cấu hình: không có, ghi  nhật ký định kỳ, WAL | |
| Nhân rộng | Chủ-tớ, đồng bộ  có thể định cấu hình | Tập tin - nhất quán cấp hệ thống | (HDFS) h | hóa tro | Băm nhất quán | | không đồng bộ  nô lệ | bậc thầy- |
| sharding | Hash- hoặc phạm vi dựa trên  thuộc tính) | ra Ndựa trên địa lý (phím hàng) | nhất quán h hóa tro | | Băm nhất quán | | Chỉ trong Redis Cluster:  băm | |
| Tính nhất quán | Thạc sĩ viết, với đại biểu  đọc tuyến tính hóa và cuối cùng khác | Lâmnghe được | Cuối cùng, op chuyên môn  có thể tuyến tính hóa bpạdnates  (“nhẹ t  giao dịch ” ) | | Cuối cùng, phía khách hàng cphạm tội  nghị quyết | | Thầy đọc: tuyến tính hóa-  có thể, nô lệ đọc: cuối cùng | |
| nguyên tử | tài liệu duy nhất | Tộigle hàng, hoặc rõ ràng  lộcking | cột đơn N  (nhiều màu tôni bản cập nhật  có thể gây d bẩn viết) | | Cặp khóa/giá trị đơn | | Lạc quan  giao dịch,  kịch bản | đa phím  Lua nguyên tử |
| có điều kiện  cập nhật | Có (làm chủ) | Đún(glàm chủ) | Có (Paxos-c phối hợp) | | Không | | Có (làm chủ) | |
| giao diện | TCP nhị phân | Thrnếu | Tiết kiệm hoặc T CP/CQL | | REST hoặc TCP/Proto buf | | TCP/Văn bản thuần túy | |
| Dữ liệu đặc biệt các loại | Đối tượng, mảng, tập hợp,  bộ đếm, tập tin | đồnbngạnter | quầy | | CRDT cho bộ đếm,  sổ đăng ký, bản đồ | cờ, | Bộ, hàm băm, bộ đếm, Bộ được sắp xếp, danh sách, Hyper-  LogLog, bitvector | |
| Truy vấn | Truy vấn theo ví dụ (bộ lọc, sắp xếp, dự án), truy vấn phạm vi,  MapReduce, tổng hợp | Lấy theo phím hàng, quét  lò nướnrg các phạm vi phím hàng,  chuyên ntgừhiệpchối CF/cột | Nhận bởi Parti phím chức năng và  lọc/sắp xếp ov cụm er  phím, FT-sear ch | | Nhận theo ID hoặc chỉ mụcdây dẫn  se cục bộ, cụ thể hóa vi lỗi,  MapReduce, FT-sear ch | | Dữ liệu Cấu trúc Opera-  lời đề nghị | |
| Sơ trung  lập chỉ mục | Băm, B-Tree, không gian địa lý  chỉ mục | Không Ne | Sắp xếp cục bộ chỉ mục, toàn cầu  thứ cấp trong dex  (dựa trên hàm băm), Tìm kiếm  chỉ mục (Solr) | | Chỉ số thứ cấp địa phương xlà,  chỉ mục tìm kiếm (Solr) | | Không rõ ràng | |
| Giấy phép | GPL 3.0 | Apache 2 | Apache 2 | | Apache 2 | | BSD | |

Người giới thiệu

1. Abadi, D.: Giao dịch nhất quán trong thiết kế hệ thống cơ sở dữ liệu phân tán hiện đại: Giới hạn chỉ là một phần của câu chuyện. Máy tính 45(2), 37 42 (2012)
2. Attiya, H., Bar-Noy, A., Dolev, D., khác: Chia sẻ bộ nhớ một cách mạnh mẽ trong các hệ thống chuyển thông điệp. JACM 42(1), 124 142 (1995)
3. Bailis, P., Kingsbury, K.: Mạng đáng tin cậy. cộng đồng. ACM 57(9), 48 55 (2014)
4. Baker, J., Bond, C., Corbett, JC, khác: Megastore: Cung cấp dung lượng lưu trữ có thể mở rộng, khả dụng cao cho các dịch vụ tương tác. Trong: CIDR. trang 223 234 (2011)
5. Bernstein, PA, Cseri, I., Dani, N., khác: Điều chỉnh máy chủ sql microsoft cho điện toán đám mây. Trong: ICDE lần thứ 27. trang 1255 1263. IEEE (2011)
6. Boykin, O., Ritchie, S., O'Connell, I., Lin, J.: Summingbird: Một khuôn khổ để tích hợp

tính toán mapreduce hàng loạt và trực tuyến. VLDB 7(13), 1441 1451 (2014)

1. Brewer, EA: Hướng tới các hệ thống phân tán mạnh mẽ. (2000)
2. Calder, B., Wang, J., Ogus, A., khác: Windows azure storage: dịch vụ lưu trữ đám mây có tính sẵn sàng cao với tính nhất quán cao. Trong: SOSP thứ 23. ACM (2011)
3. Chang, F., Dean, J., Ghemawat, S., khác: Bigtable: Một hệ thống lưu trữ phân tán cho dữ liệu có cấu trúc. Trong: OSDI thứ 7. trang 15 15. Hiệp hội USENIX (2006)
4. Charron-Bost, B., Pedone, F., Schiper, A. (eds.): Nhân rộng: Lý thuyết và Thực hành, Bài giảng Khoa học Máy tính, tập. 5959. Mùa xuân (2010)
5. Cooper, BF, Ramakrishnan, R., Srivastava, U., khác: Pnuts: Nền tảng phục vụ dữ liệu được lưu trữ của Yahoo!. Kỷ yếu của VLDB Endowment 1(2), 1277 1288 (2008)
6. Corbett, JC, Dean, J., Epstein, M., khác: Spanner: Cơ sở dữ liệu được phân phối toàn cầu của Google. Trong: Kỷ yếu của OSDI. trang 251 264. Hiệp hội USENIX (2012)
7. Curino, C., Jones, E., Popa, RA, other: Đám mây quan hệ: Một dịch vụ cơ sở dữ liệu cho đám mây. Trong: CIDR lần thứ 5 (2011)
8. Das, S., Agrawal, D., El Abbadi, A., khác: G-store: kho lưu trữ dữ liệu có thể mở rộng để truy cập đa khóa giao dịch trên đám mây. Trong: SoCC đầu tiên. trang 163 174. ACM (2010)
9. Davidson, SB, Garcia-Molina, H., Skeen, D., khác: Tính nhất quán trong mạng được phân vùng: một cuộc khảo sát. SUR 17(3), 341 370 (1985)
10. Dean, J.: Thiết kế, bài học và lời khuyên từ việc xây dựng các hệ thống phân tán lớn (2009), bài phát biểu quan trọng tại LADIS 2009
11. Dean, J., Ghemawat, S.: Mapreduce: xử lý dữ liệu biên tập đơn giản trên các cụm

lớn. Truyền thông của ACM 51(1), 107 113 (2008)

1. DeC andia, G., Hastorun, D., khác: Dynamo: kho khóa-giá trị có sẵn cao của amazon. Trong: SOSP thứ 21. trang 205 220. ACM (2007)
2. Fischer, MJ, Lynch, NA, Paterson, MS: Không thể có sự đồng thuận phân tán với một quy trình bị lỗi. J. ACM 32(2), 374 382 (tháng 4 năm 1985)
3. Gessert, F., Schaarschmidt, M., Wingerath, W., Friedrich, S., Ritter, N.: The cache sketch: Xem lại bộ nhớ đệm dựa trên thời hạn hết hạn trong thời đại quản lý dữ liệu đám mây.

Trong: BTW. trang 53 72 (2015)

1. Gilbert, S., Lynch, N.: Phỏng đoán của Brewer và tính khả thi của các dịch vụ web nhất quán, có sẵn, chịu phân vùng. SIGACT News 33(2), 51 59 (Tháng 6 năm 2002)
2. Gray, J., Hell and, P., other: Những mối nguy hiểm của việc sao chép và giải pháp. SIGMOD Rec. 25(2), 173 182 (Tháng 6 năm 1996)
3. Haerder, T., Reuter, A.: Nguyên tắc khôi phục cơ sở dữ liệu hướng giao dịch. Máy tính

ACM. sống sót 15(4), 287 317 (12/1983)

1. Hamilton, J.: Về thiết kế và triển khai các dịch vụ quy mô internet. Trong: LISA thứ 21. trang 18:1 18:12. Hiệp hội USENIX (2007)
2. Hellerstein, JM, Stonebraker, M., Hamilton, J.: Kiến trúc của một hệ thống cơ sở dữ liệu. Bây giờ Nhà xuất bản Inc (2007)
3. Herlihy, MP, Wing, JM: Tính tuyến tính: điều kiện đúng cho các đối tượng đồng thời. ACM Trans. Chương trình. Láng. hệ thống. 12(3), 463 492 (tháng 7 năm 1990)
4. Hoelzle, U., Barroso, LA: Trung tâm dữ liệu như một máy tính: Giới thiệu về thiết kế

máy quy mô kho hàng. Nhà xuất bản Morgan và Claypool (2009)

1. Hunt, P., Konar, M., Junqueira, FP, Reed, B.: Zookeeper: điều phối miễn phí cho các hệ thống quy mô internet. Trong: USENIXATC. Hiệp hội USENIX (2010)
2. Kallman, R., Kimura, H., Natkins, J., khác: H-store: hệ thống xử lý giao dịch bộ nhớ chính phân tán, hiệu suất cao. Tài trợ VLDB (2008)
3. Karger, D., Lehman, E., Leighton, T., khác: Cây băm nhất quán và cây ngẫu nhiên: giao thức bộ nhớ đệm phân tán để loại bỏ các điểm nóng trên mạng toàn cầu. Trong: CỔ PHIẾU THỨ 29. trang 654 663. ACM (1997)
4. Kleppmann, M.: Thiết kế các ứng dụng sử dụng nhiều dữ liệu. O Reilly, xuất hiện (2016)
5. Kraska, T., Pang, G., Franklin, MJ, khác: Mdcc: Tính nhất quán của trung tâm đa dữ liệu. Trong: EuroSys lần thứ 8. trang 113 126. ACM (2013)
6. Kreps, J.: Đặt câu hỏi về kiến trúc lambda (2014), truy cập: 2015-12-17
7. Lakshman, A., Malik, P.: Cassandra: Một hệ thống lưu trữ có cấu trúc phi tập trung. Nhà điều hành SIGOPS hệ thống. Rev. 44(2), 35 40 (Tháng 4 năm 2010)
8. Laney, D.: Quản lý dữ liệu 3d: Kiểm soát khối lượng, tốc độ và sự đa dạng của dữ liệu. Công

nghệ. đại diện, META Group (tháng 2 năm 2001)

1. Lloyd, W., Freedman, MJ, Kaminsky, M., khác: Đừng giải quyết vấn đề cuối cùng: Tính nhất quán nhân quả có thể mở rộng để lưu trữ diện rộng với cảnh sát. Trong: SOSP thứ 23. ACM (2011)
2. Mahajan, P., Alvisi, L., Dahlin, M., khác: Nhất quán, sẵn có và hội tụ. Đại học Texas

tại Austin Tech Report 11 (2011)

1. Mao, Y., Junqueira, FP, Marzullo, K.: Mạnh Tử: xây dựng các cỗ máy trạng thái sao chép hiệu quả cho wans. Trong: OSDI. tập 8, trang 369 384 (2008)
2. Marz, N., Warren, J.: Dữ liệu lớn: Nguyên tắc và thực tiễn tốt nhất của hệ thống dữ liệu thời gian thực có thể mở rộng. Manning Publications Co. (2015)
3. Min, C., Kim, K., Cho, H., khác: Sfs: ghi ngẫu nhiên được coi là có hại trong ổ đĩa trạng thái rắn.

Trong: NHANH CHÓNG. P. 12 (2012)

1. "Ozsu, MT, Valduriez, P.: Các nguyên tắc của hệ thống cơ sở dữ liệu phân tán. Springer Science & Business Media (2011)
2. Pritchett, D.: Bazơ: Một chất thay thế axit. Hàng đợi 6(3), 48 55 (Tháng 5 năm 2008)
3. Qiao, L., Surlaker, K., Das, S., other: Về pha cà phê espresso tươi: Nền tảng cung cấp dữ liệu phân tán của Linkedin. Trong: SIGMOD. trang 1135 1146. ACM (2013)
4. Sadalage, PJ, Fowler, M.: NoSQL chưng cất: hướng dẫn ngắn gọn về thế giới mới nổi của sự kiên trì đa ngôn ngữ. Addison-Wesley, Thượng nguồn sông Saddle, NJ (2013)
5. Shapiro, M., Preguicc a, N., Baquero, C., other: Một nghiên cứu toàn diện về các loại dữ

liệu sao chép hội tụ và giao hoán. Bằng tiến sĩ. luận án, INRIA (2011)

1. Shukla, D., Thota, S., Raman, K., khác: Lập chỉ mục lược đồ bất khả tri với azure documentdb. Kỷ yếu của VLDB Endowment 8(12), 1668 1679 (2015)
2. Sovran, Y., Power, R., Aguilera, MK, Li, J.: Lưu trữ giao dịch cho các hệ thống tái tạo địa lý. Trong: SOSP thứ 23. trang 385 400. ACM (2011)
3. Stonebraker, M., Madden, S., Abadi, DJ, khác: Sự kết thúc của một kỷ nguyên kiến trúc: (đã đến lúc viết lại toàn bộ). Trong: VLDB lần thứ 33. trang 1150 1160 (2007)
4. Wiese, L., other: Quản lý dữ liệu nâng cao: Dành cho SQL, NoSQL, Đám mây và Cơ

sở dữ liệu phân tán. Walter de Gruyter GmbH & Co KG (2015)

1. Zhang, H., Chen, G., Ooi, BC, other: Xử lý và quản lý dữ liệu lớn trong bộ nhớ: Một cuộc khảo sát. TKDE (2015)