LỚP 09CNTT1

NHÓM 10 : NOSQL_SURVEY

Lê Duy Minh Dương (Nhóm trưởng)

Đinh Quốc Dũng

Huỳnh Công Hoàng

NoSQL_SURVEY

1. Giới thiệu

Trong thập kỷ qua, chúng ta đã chứng kiến sự phát triển nhanh chóng của một số loại hình khác nhau của các cơ sở dữ liệu. Nó liên quan đến sự phát triển của Internet, thiết bị di động và đám mây tin học. Tất cả các môi trường này áp đặt các yêu cầu mới cho việc lưu trữ hiệu quả và xử lý dữ liệu.

Để đối phó với những thách thức này, một số công ty duy trì các trung tâm dữ liệu và trang trại trong đó chứa các cụm với hàng nghìn máy phần cứng hàng hóa.

Cơ sở dữ liệu quan hệ kiểu cũ không cung cấp một giải pháp tốt trong tình huống này, do với mô hình dữ liệu chuẩn hóa của họ và hỗ trợ ACID đầy đủ.

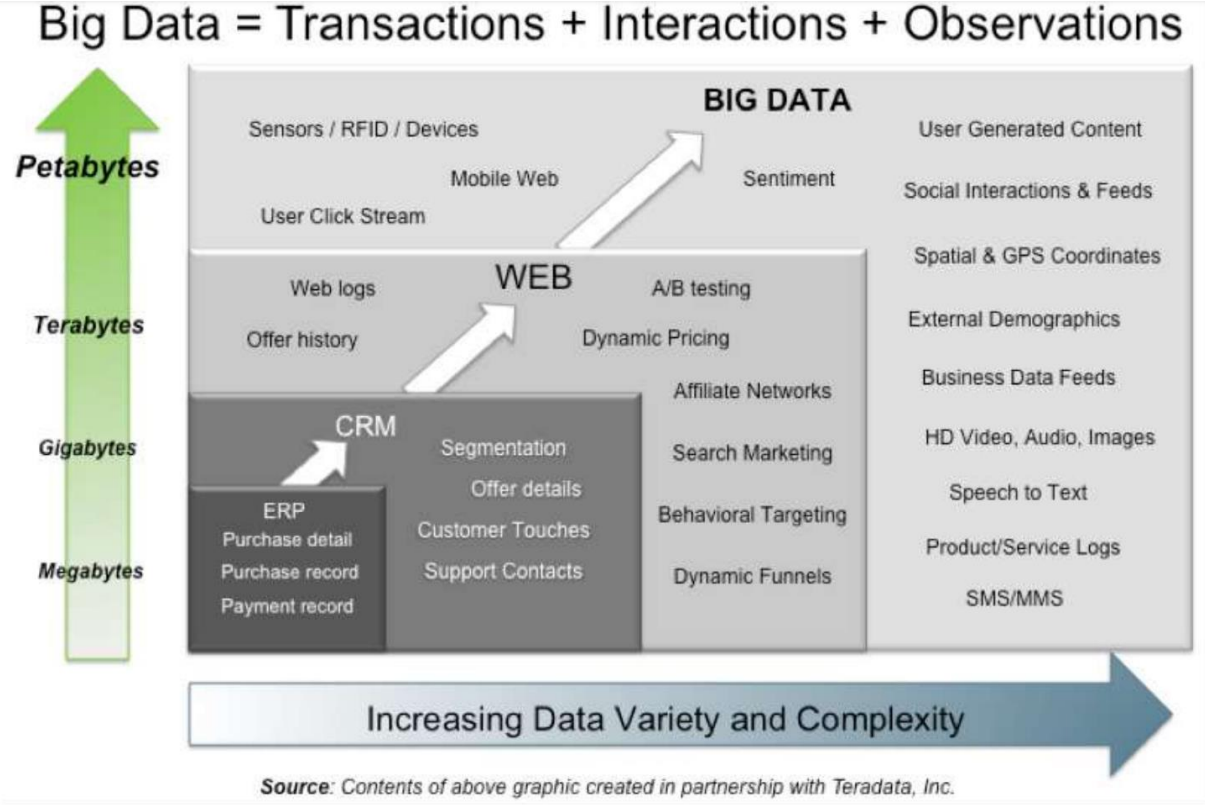
Ngoài ra, hóa ra cách tiếp cận phổ biến "một kích cỡ phù hợp với tất cả" không còn giá trị nữa cho các kịch bản ứng dụng hiện tại. Ngược lại, trong thực tế mới, nó là nhiều hơn phù hợp để thiết kế các hệ thống dựa trên bản chất của các ứng dụng và dữ liệu của nó/ yêu cầu truy vấn.

Do đó, nhiều giải pháp cơ sở dữ liệu thay thế được thiết kế đặc biệt để thỏa mãn nhu cầu đa dạng. Nhiều cơ sở dữ liệu mới này thuộc lớp NoSQL và NewSQL của DBMS. Trong bài báo này, chúng tôi khảo sát, so sánh và đánh giá bốn loại chính của NoSQL và cả cơ sở dữ liệu NewSQL. Cụ thể, chúng tôi phân tích và so sánh DB sau đặc điểm:

- Mô hình dữ liệu
- Khả năng truy vấn
- Đồng thời kiểm soát
- Nhân rộng
- Khả năng mở rộng
- phân vùng
- Tính nhất quán
- Tính năng/nhược điểm bảo mật
- Trường hợp sử dụng/ứng dụng phù hợp
- Phổ biến
- Màn biểu diễn

2. Phân loại

- Key-value stores
- Document stores
- Column family stores
- Graph databases
- NewSQL databases



Mô tả sự gia tăng về khối lượng dữ liệu, tính đa dạng và độ phức tạp trong Big Data

Cơ sở dữ liệu quan hệ truyền thống đang đối mặt với nhiều thách thức mới với quy mô lớn và tính đồng thời cao trong khi xử lý dữ liệu lớn và hơn nữa, một số nhà nghiên cứu trong lĩnh vực này coi trí tuệ DB "cũ" truyền thống là lỗi thời.

Trong khi cơ sở dữ liệu quan hệ truyền thống vẫn chiếm thị phần lớn nhất trên thị trường, một số tập đoàn Internet hiện đại và các công ty khác đã chuyển hướng sang một mô hình lưu trữ dữ liệu khác, cái gọi là NoSQL. Lý do tại sao NoSQL trở nên phổ biến trong vài năm qua chủ yếu là do khi một cơ sở dữ liệu quan hệ phát triển từ một máy chủ, nó không còn dễ sử dụng nữa. Nói cách khác, chúng không mở rộng quy mô tốt trong một hệ thống phân tán.

NoSQL là viết tắt của "Không chỉ ngôn ngữ truy vấn có cấu trúc" và thường xác định một họ các kho lưu trữ dữ liệu loại bỏ các thuộc tính ACID về tính nhất quán và sự cô lập để ủng hộ "tính khả dụng, sự xuống cấp nhanh chóng và hiệu suất" hoặc CƠ SỞ.

Từ viết tắt ACID là viết tắt của

- Atomicity: tất cả các hoạt động trong giao dịch sẽ hoàn thành hoặc không có hoạt động nào.

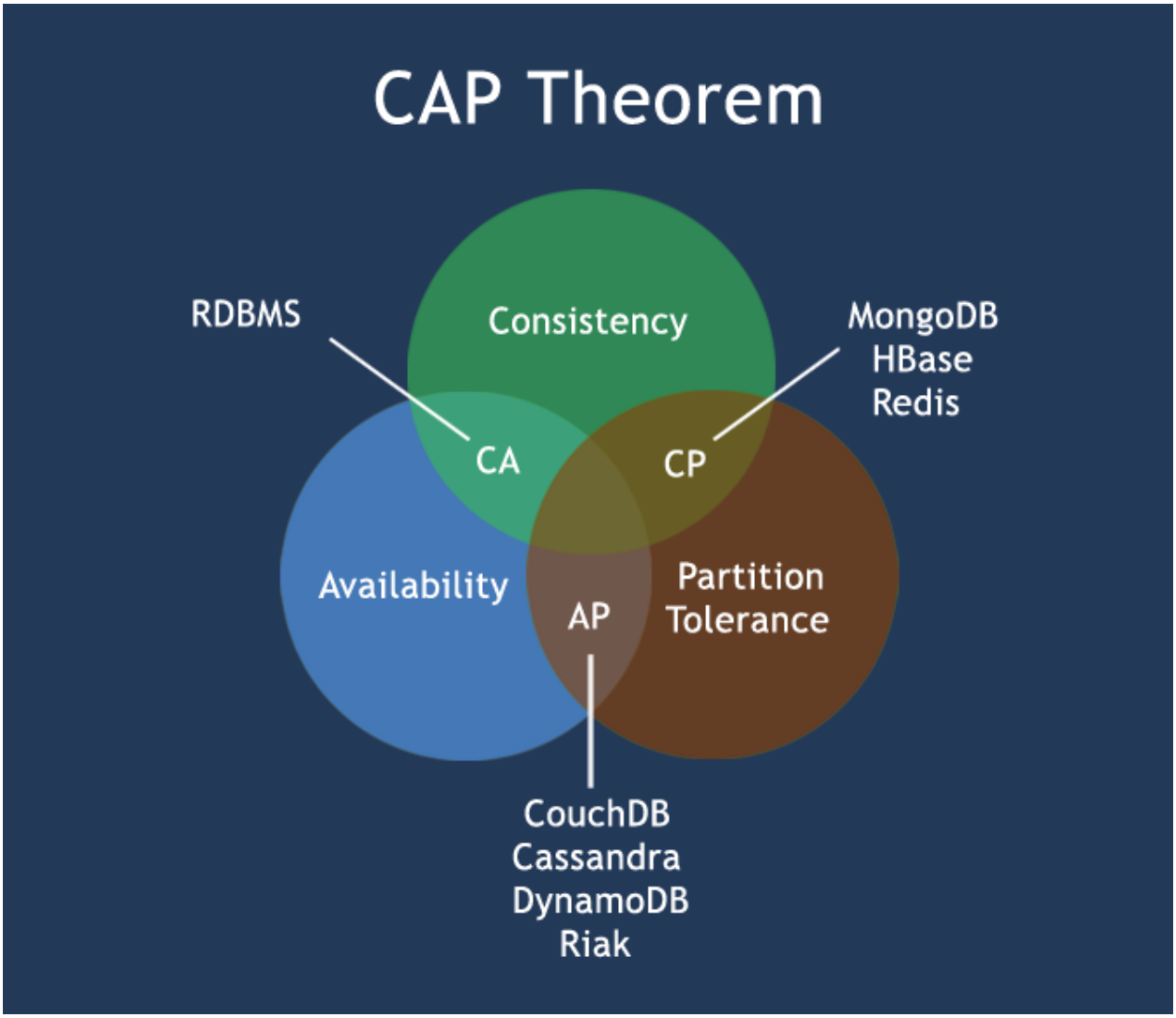
- Tính nhất quán: các giao dịch không bao giờ quan sát hoặc dẫn đến dữ liệu không nhất quán.

- Cách ly: giao dịch sẽ hoạt động như thể nó là hoạt động duy nhất được thực hiện.

- Độ bền: sau khi hoàn thành giao dịch, thao tác sẽ không bị đảo ngược mà sẽ bền bỉ

Nói cách khác, mô hình BASE từ bỏ thuộc tính ACID của

tính nhất quán và sự cô lập có lợi cho “sự sẵn có, sự xuống cấp nhanh chóng và hiệu suất”



Định lý CAP

Định lý CAP do Brewer đưa ra vào năm 2000, tuyên bố rằng không thể có một dịch vụ phân tán nhất quán, khả dụng và có khả năng chịu phân vùng tại cùng một thời điểm.

Những người ủng hộ NOSQL thường trích dẫn định lý CAP của Eric Brewer được tóm tắt là

sau: một hệ thống có thể có không quá hai trong số ba thuộc tính sau

- Tính nhất quán:

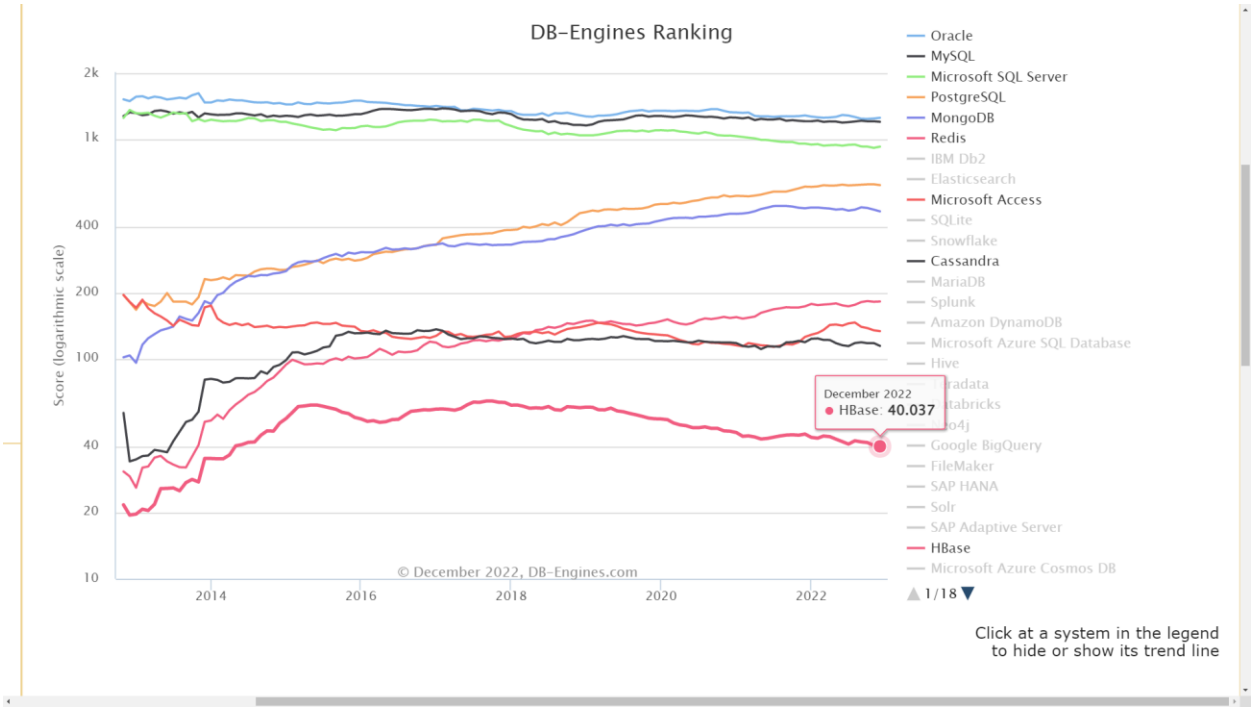
- Khả dụng

- Dung sai phân vùng

ACID	BASE
- Nhất quán mạnh mẽ cho các giao dịch ưu tiên cao nhất - Tính khả dụng ít quan trọng hơn - Bị động - Phân tích nghiêm ngặt - Cơ chế phức tạp	- Tính khả dụng và mở rộng quy mô ưu tiên cao nhất - Tính nhất quán yếu - Chủ động - Tính bền bỉ cao - Đơn giản và nhanh chóng

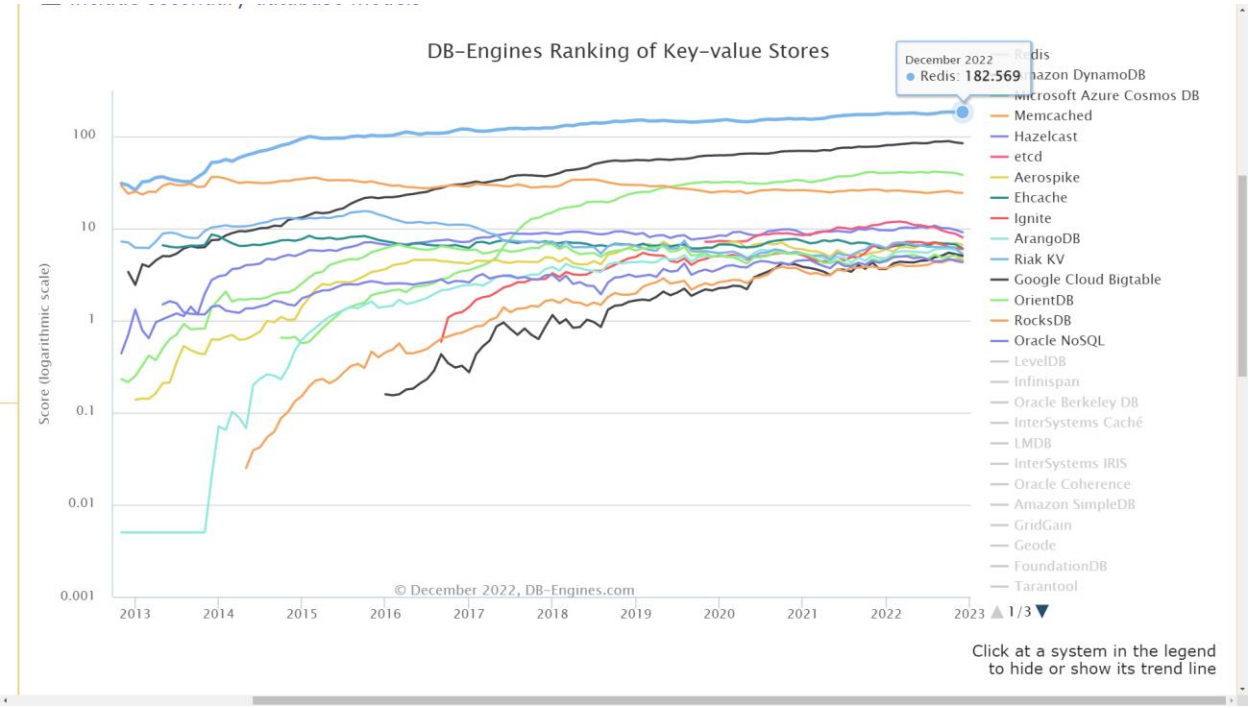
So sánh ACID và BASE theo Brewer

- **Độ phổ biến**

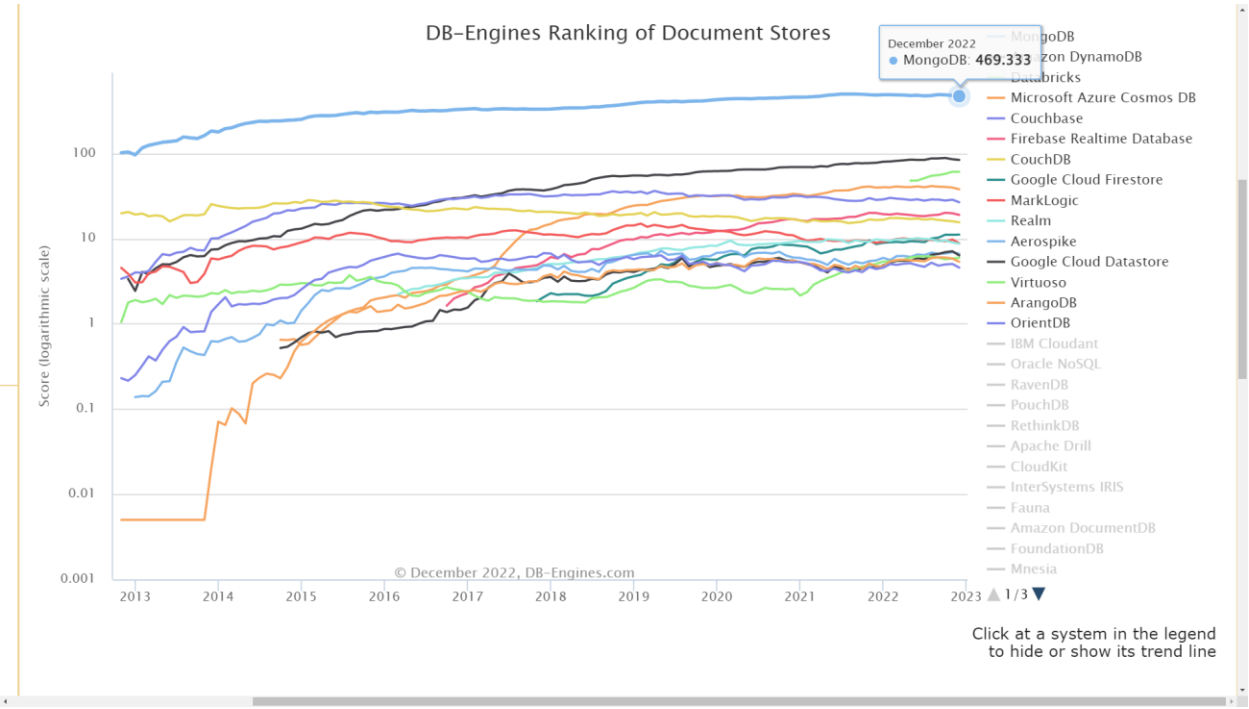


Độ phổ hiến của các loại cơ sở dữ liệu phổ biến đứng đầu là Oracle với 1.250.307 người dùng

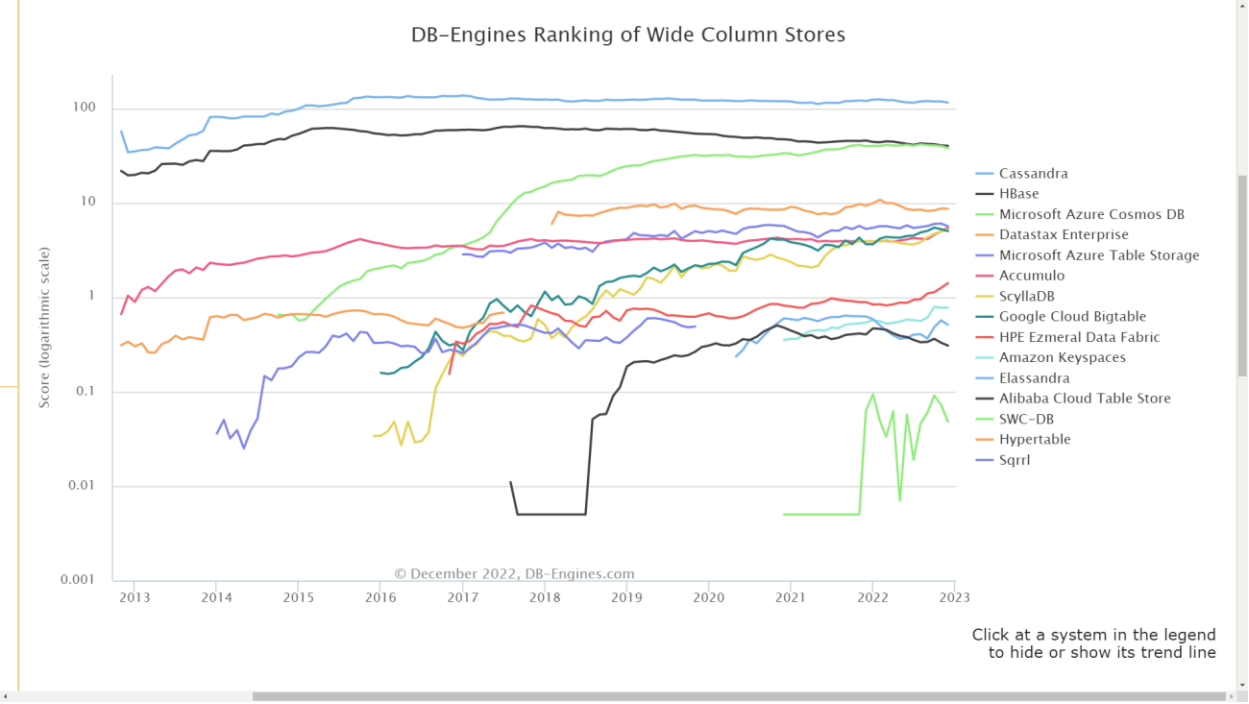
- **Key-value stores**



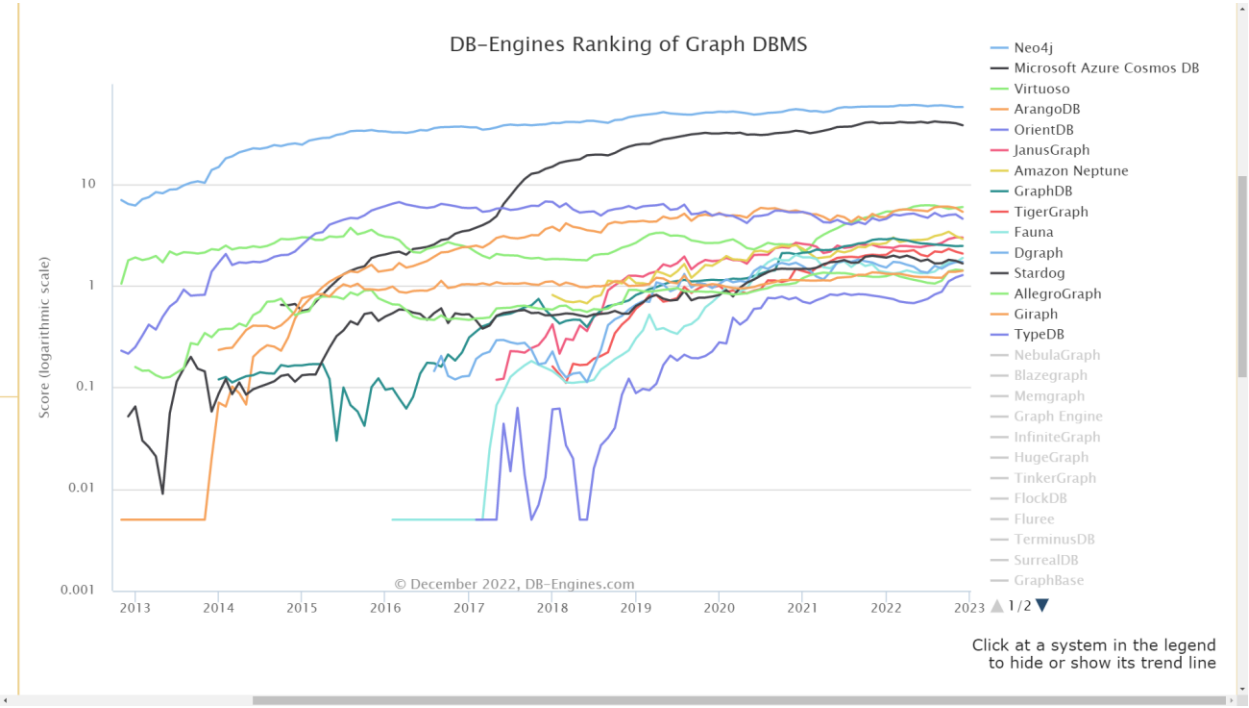
- **Document stores**



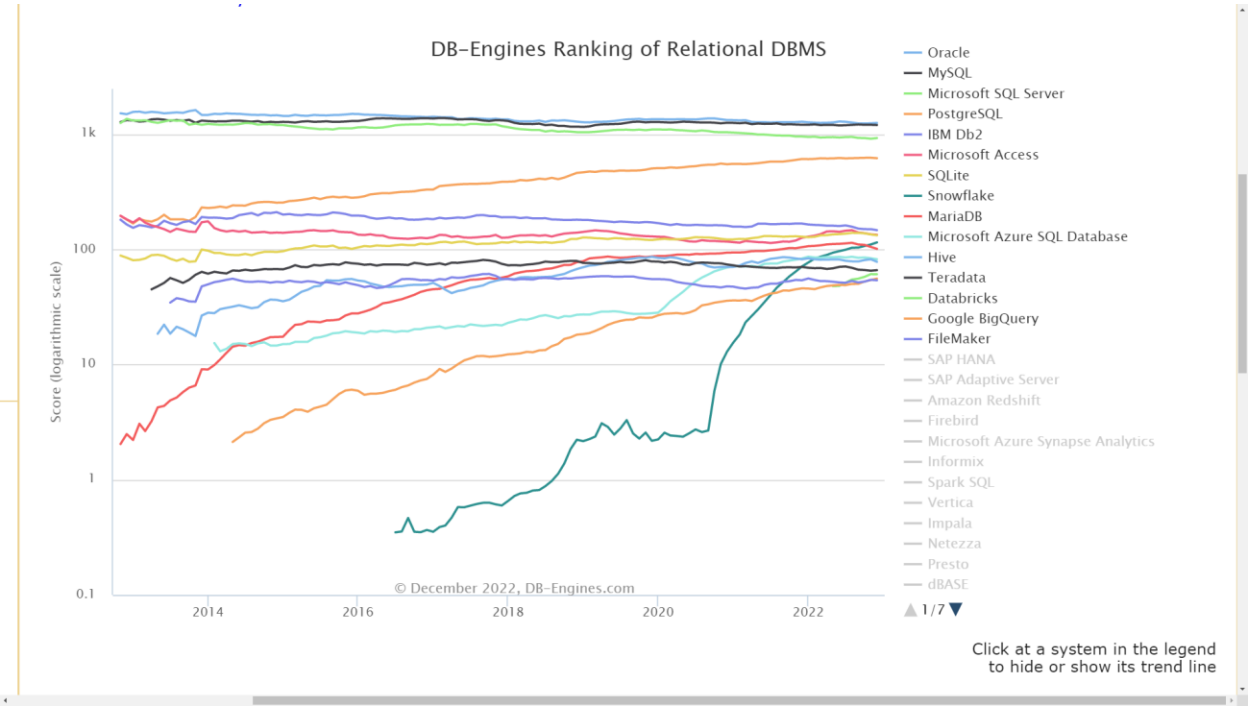
- **Column family stores**



- **Graph databases**



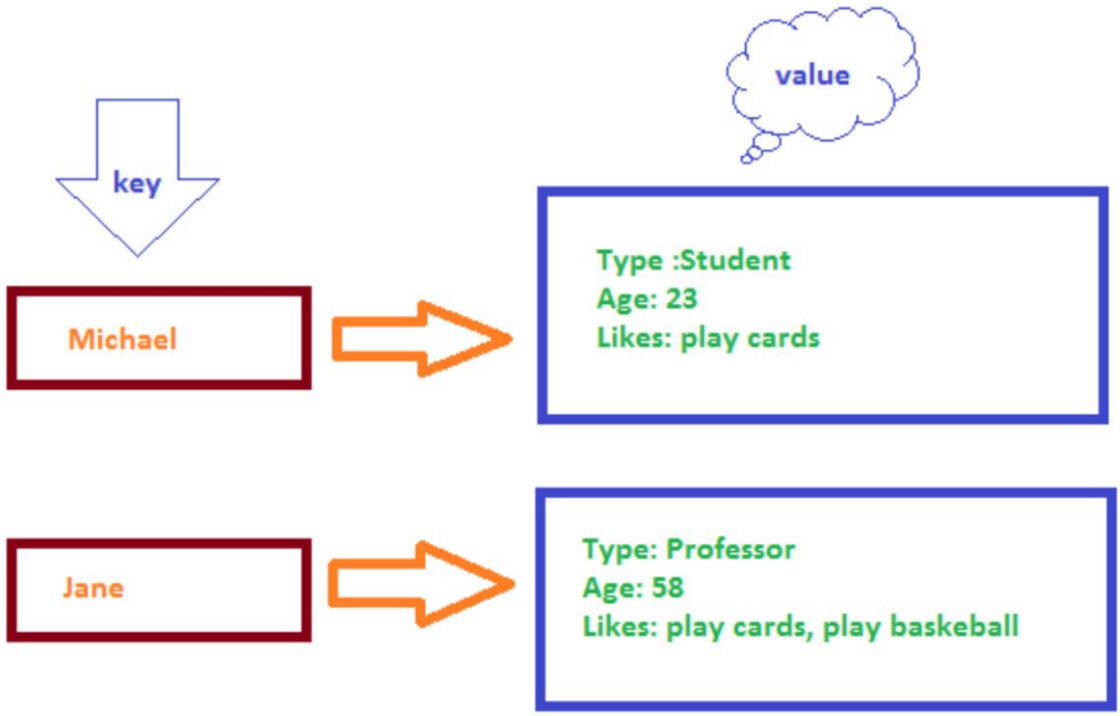
- **NewSQL databases**



- **Mô hình dữ liệu**
  - **Key-value stores**

Các kho lưu trữ dữ liệu khóa-giá trị đôi khi được coi là dạng cơ sở dữ liệu "đơn giản nhất".

Những cơ sở dữ liệu đó thường không có lược đồ. Dữ liệu được lưu trữ dưới dạng một cặp khóa-giá trị, do đó, mô hình dữ liệu khóa-giá trị giống với cấu trúc tương tự như bản đồ hoặc từ điển. Các khóa được sử dụng làm chỉ mục để tìm nạp dữ liệu (giá trị) và điều này làm cho các kho lưu trữ dữ liệu đó hiệu quả hơn nhiều đối với việc truy xuất dữ liệu so với RDBMS cổ điển.



Mô hình dữ liệu KV rất đơn giản và đôi khi nó được sử dụng làm cơ sở trong khi các mô hình dữ liệu phức tạp hơn được triển khai trên nó. Mô hình khóa-giá trị có thể được mở rộng thành mô hình được sắp xếp để duy trì các khóa theo thứ tự từ điển. Tiện ích mở rộng này rất mạnh ở chỗ nó có thể xử lý các phạm vi chính một cách hiệu quả .

Theo định lý CAP, các cơ sở dữ liệu đó thường ưu tiên tính khả dụng hơn tính nhất quán.

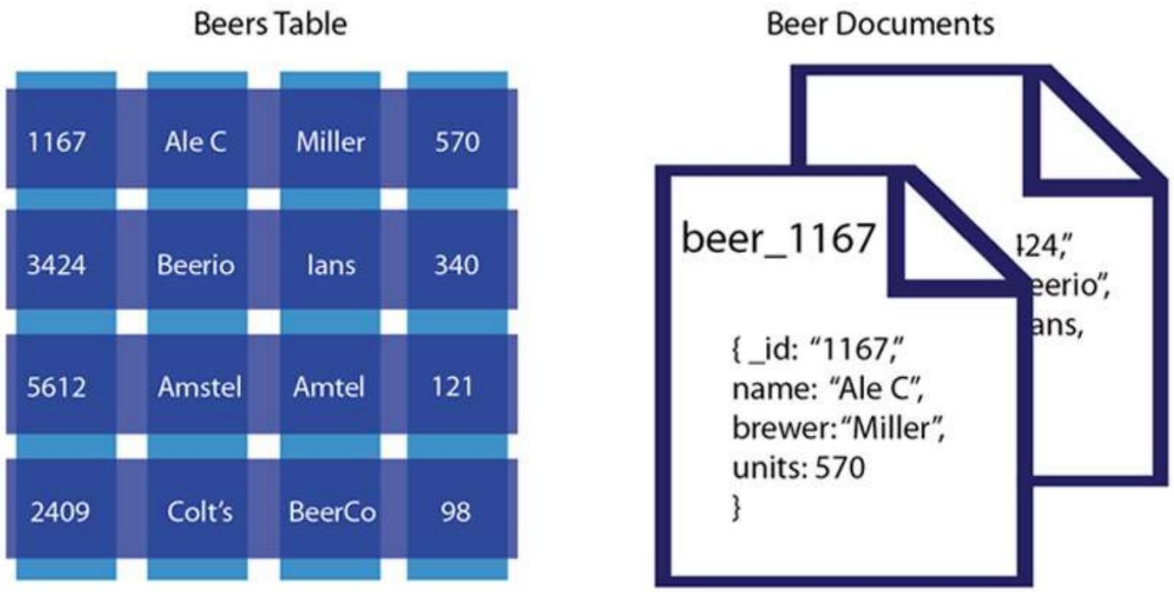
Ví dụ về cơ sở dữ liệu giá trị khóa bao gồm Voldemort và Amazon Dynamo DB. Mô hình DB của Amazon Dynamo cung cấp dịch vụ cơ sở dữ liệu NOSQL nhanh, có độ tin cậy cao và tiết kiệm chi phí được thiết kế cho các ứng dụng quy mô internet. Nó cung cấp độ trễ thấp, có thể dự đoán được ở mọi quy mô. Hơn nữa trong bài báo cuối cùng này, chúng tôi sẽ tập trung vào Dự án Voldemort như một ví dụ về truy vấn KV. Project Voldemort là một kho lưu trữ khóa-giá trị nâng cao, được viết bằng Java . Nó là nguồn mở, với sự đóng góp đáng kể từ LinkedIn.

- **Document stores**

Cơ sở dữ liệu dựa trên tài liệu có lẽ phổ biến nhất trong số các loại NoSQL khác và mức độ phổ biến của chúng không ngừng tăng lên. Các cơ sở dữ liệu này lưu trữ dữ liệu của họ dưới dạng tài liệu. Nói chung, loại cơ sở dữ liệu này gói gọn khái niệm “khóa-giá trị”, trong khi khóa là ID của tài liệu và giá trị là chính tài liệu đó, có thể được truy xuất bằng ID. Các định dạng khác nhau có thể được sử dụng làm siêu dữ liệu cho các DB hướng tài liệu: XML, JSON và một số định dạng khác.

Trái ngược với RDBMS truyền thống, nơi mọi hàng tuân theo lược đồ, trong các DB hướng tài liệu, mỗi tài liệu có thể có cấu trúc khác nhau. Và thông thường các truy vấn định hướng tài liệu cung cấp chỉ mục bổ sung dựa trên nội dung tài liệu. Đó là một trong những cải tiến chính của kho lưu trữ tài liệu so với mô hình lưu trữ khóa-giá trị cơ bản hơn.

Cả hai đều cung cấp cơ chế truy vấn dựa trên “khóa chính”, nhưng trong mô hình lưu trữ tài liệu, người ta thường có thể truy vấn dữ liệu theo nội dung giá trị (tài liệu). Tương tự như KV-stores, loại hệ thống cơ sở dữ liệu này kém hiệu quả hơn khi ứng dụng yêu cầu giao dịch nhiều khóa.



So sánh cấu trúc của cơ sở dữ liệu quan hệ và cơ sở dữ liệu dạng tài liệu

- **Column family stores**

Họ cột tiêu chuẩn là một đối tượng NoSQL chứa các cột dữ liệu liên quan. Cách tiếp cận truy cập và xử lý dữ liệu theo cột thay vì theo hàng đã được sử dụng trong các công cụ phân tích từ khá lâu. Tuy nhiên, lớp con của Cơ sở dữ liệu NoSQL được gọi là Truy vấn cột-gia đình đề cập cụ thể đến các hệ thống có nguồn gốc từ Google Bigtable. Các nhà nghiên cứu của Google mô tả mô hình này là “các bản đồ được sắp xếp thưa thớt, phân tán, liên tục và đa chiều” . Trong Bigtable, tập dữ liệu bao gồm một số hàng, mỗi hàng có thể được xử lý bằng một khóa - khóa chính. Một số phổ biến nhất

Các kho lưu trữ dữ liệu trong danh mục này triển khai mô hình Google Bigtable cơ bản, ví dụ: HBase. Trong khi những người khác, như Cassandra mở rộng mô hình bằng cách giới thiệu các siêu cột, trong đó giá trị có thể là họ cột của chính nó

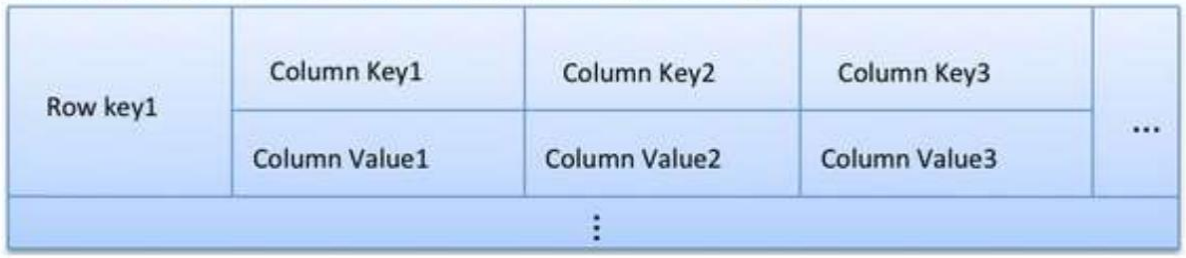
Thuật ngữ Họ cột có nghĩa là một cặp bao gồm khóa và giá trị, trong đó khóa được ánh xạ tới một giá trị là một tập hợp các cột. Tương tự với cơ sở dữ liệu quan hệ, một họ cột có thể được coi là một "bảng", trong khi mỗi cặp khóa-giá trị đại diện cho một "hàng". Tuy nhiên, bằng cách sử dụng phép loại suy này, chúng ta có thể minh họa một trong những khác biệt chính giữa mô hình RDBMS truyền thống và mô hình họ Colum, đó là thực tế là cùng một “bảng” (họ cột) có thể chứa các cột khác nhau và số lượng cột khác nhau, trong khi mô hình quan hệ truyền thống mô hình là hoàn toàn nghiêm ngặt về điều này.

Cũng có một điểm tương đồng với mô hình khóa-giá trị cơ bản nhất, vì các chức năng của khóa hàng

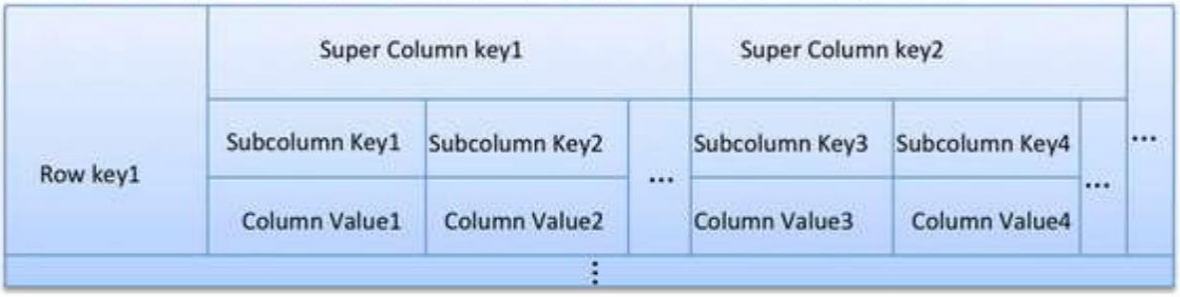
dưới dạng "khóa", trong khi tập hợp các cột giống với "giá trị". Do các chi tiết cụ thể của mô hình dữ liệu, họ Cột thường không xử lý logic quan hệ phức tạp. Do đó, giống như trường hợp lưu trữ Khóa-giá

trị cơ bản, nếu chức năng truy vấn quan hệ phức tạp được yêu cầu, thì nó phải được triển khai ở phía máy khách.

The following layout represents a row in a Column Family (CF):



The following layout represents a row in a Super Column Family (SCF):

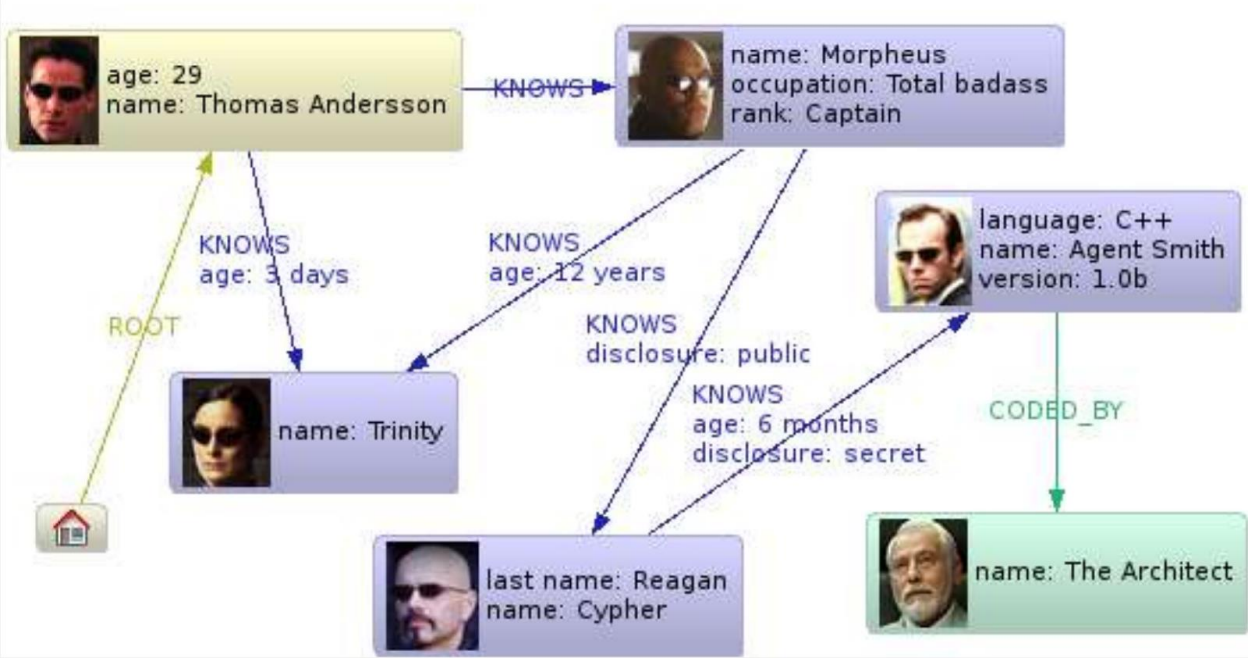


- **Graph databases**

Cơ sở dữ liệu đồ thị là mô hình có nguồn gốc từ lý thuyết đồ thị. Ngoài ra, mô hình dữ liệu của các cơ sở dữ liệu NoQuery này dựa trên cấu trúc biểu đồ. Về cơ bản, điều đó có nghĩa là dữ liệu được lưu trữ dưới dạng các nút (đỉnh), trong khi các mối quan hệ giữa các dữ liệu được trình bày dưới dạng các cạnh kết nối các đỉnh với nhau.

Các kho lưu trữ dữ liệu này được coi là cơ sở dữ liệu NoSQL, bởi vì các hệ thống này không cung cấp hỗ trợ SQL và mô hình dữ liệu không giống với mô hình quan hệ. Tuy nhiên, nhiều cơ sở dữ liệu Đồ thị, bao gồm Neo4j, hoàn toàn tuân thủ ACID. Một điểm khác biệt giữa nhóm phụ này và các danh mục NoSQL khác là mô hình này không phải là phần mở rộng của khái niệm “khóa-giá trị” và mô hình này hiệu quả hơn để lưu trữ dữ liệu được kết nối với nhau và xử lý truy vấn quan hệ. Do đó, một cách tự nhiên, chúng phù hợp hơn để giải quyết vấn đề phân tích phụ thuộc và một số tình huống mạng xã hội.

Mặc dù có hiệu quả trong các lĩnh vực đó, nhưng cơ sở dữ liệu đồ thị có thể kém phù hợp hơn trong việc xử lý các sự cố Dữ liệu lớn khác. Đặc biệt, các cơ sở dữ liệu này không hiệu quả khi chia tỷ lệ theo chiều ngang dưới dạng khóa-giá trị hoặc họ cột. “Điểm yếu” này xuất phát từ thực tế là nếu dữ liệu liên quan được lưu trữ trên một máy chủ khác, thì việc duyệt qua một biểu đồ như vậy có thể hoạt động rất “đắt” về mặt hiệu suất.



- **NewSQL databases**

NewSQL là một lớp khác của hệ thống quản lý cơ sở dữ liệu hiện đại. Cho đến gần đây, việc triển khai kiến trúc mở rộng yêu cầu một số giải pháp NoSQL vì cơ sở dữ liệu quan hệ kiểu cũ không cung cấp hỗ trợ tốt cho quy mô "ngang". Như đã đề cập ở trên, các giải pháp NoSQL như vậy thường không cung cấp ACID và thay vào đó cung cấp một số loại nhất quán cuối cùng. Sự căng thẳng này là điều đã truyền cảm hứng cho phong trào NewSQL.

Các giải pháp NewSQL theo định nghĩa dựa trên mô hình quan hệ. Bên cạnh đó, các cơ sở dữ liệu này tìm cách cung cấp cùng hiệu suất có thể mở rộng của các hệ thống NoSQL cho khối lượng công việc đọc-ghi xử lý giao dịch trực tuyến (OLTP), trong khi vẫn duy trì các đảm bảo ACID của hệ thống cơ sở dữ liệu truyền thống .

Tuy nhiên, mặc dù họ sử dụng SQL làm ngôn ngữ giao diện chính và các máy khách tương tác với NewSQL theo các thuật ngữ DB quan hệ truyền thống, chẳng hạn như "bảng" và "quan hệ", biểu diễn bên trong thực tế có thể hoàn toàn khác với biểu diễn của DB truyền thống. Ví dụ: NuoDB có thể lưu trữ dữ liệu của nó vào kho khóa-giá trị .

Thuật ngữ "NewSQL" đã được tạo ra bởi Matt Aslett. Anh ấy viết: "'NewSQL' là cách viết tắt của chúng tôi dành cho các nhà cung cấp cơ sở dữ liệu SQL hiệu năng cao/có khả năng mở rộng mới khác nhau. Trước đây, chúng tôi đã gọi các sản phẩm này là 'ScalableSQL' để phân biệt chúng với các sản phẩm cơ sở dữ liệu quan hệ hiện tại. Vì điều này ngụ ý khả năng mở rộng theo chiều ngang, vốn không nhất thiết phải là một tính năng của tất cả các sản phẩm, nên chúng tôi đã sử dụng thuật ngữ 'NewSQL' trong báo cáo mới.

Và để làm rõ, giống như NoSQL, NewSQL không được hiểu theo nghĩa đen: điều mới về các nhà cung cấp NewSQL là nhà cung cấp, không phải SQL." Theo Giáo sư Stonebraker, các mục tiêu của NewSQL là mang

lại lợi ích của mô hình quan hệ cho các kiến trúc phân tán hoặc cung cấp hiệu suất tốt đến mức khả năng mở rộng theo chiều ngang không còn cần thiết nữa.

	Old SQL	NoSQL	NewSQL
Relational	Yes	No	Yes
SQL	Yes	No	Yes
ACID transactions	Yes	No	Yes
Horizontal scalability	No	Yes	Yes
Performance/ big volume	No	Yes	Yes
Schema-less	No	Yes	No

Bảng này cung cấp sự so sánh các đặc điểm chính của OldSQL, NoSQL và NewSQL. Chẳng hạn, chỉ các cơ sở dữ liệu NoSQL mới được coi là không có lược đồ, nghĩa là chúng cho phép lưu trữ dữ liệu phi cấu trúc mà không cần biết trước về lược đồ; hệ thống SQL truyền thống ("OldSQL") hoặc NewSQL không cho phép điều đó.

3. **So sánh các tính năng**
    - **Khả năng truy vấn**

Nói chung, các mô hình dữ liệu được kết hợp chặt chẽ với các khả năng truy vấn. Do đó phân tích các truy vấn sẽ cần thiết cho một ứng dụng là một quá trình quan trọng trong để tìm giải pháp cơ sở dữ liệu phù hợp. Nhiều kho lưu trữ dữ liệu NoSQL/NewSQL khác nhau không chỉ khác nhau về mô hình dữ liệu được cung cấp, nhưng cũng có các API và giao diện khác nhau để tương tác với họ. Điều này một lần nữa phụ thuộc trực tiếp vào mô hình dữ liệu được sử dụng bởi dữ liệu truy vấn.

Chẳng hạn, kho lưu trữ khóa-giá trị thường không thể cung cấp truy vấn dựa trên giá trị. Giá trị trong kho lưu trữ dữ liệu KV không rõ ràng và chúng chỉ cung cấp "đặt", "lấy" và "xóa" dựa trên khóa hoạt động. Vì vậy, bất kỳ ngôn ngữ truy vấn nào sẽ là chi phí không cần thiết cho các truy vấn này. Và trong các hệ thống khi cần thêm các chức năng truy vấn phức tạp hơn, chúng phải được triển khai trên lớp ứng dụng, điều này có thể dẫn đến nhiều hệ thống hơn hình phạt phức tạp và hiệu suất. Do đó, các truy vấn khóa-giá trị không nên được sử dụng trong các ứng dụng yêu cầu truy vấn phức tạp hoặc truy vấn dựa trên giá trị.

Mặt khác, các truy vấn tài liệu cung cấp API phong phú hơn bao gồm các truy vấn dựa trên giá trị. Loại kho lưu trữ dữ liệu này cung cấp các truy vấn phạm vi về giá trị, chỉ mục phụ, truy vấn các tài liệu và hoạt động lồng nhau như "and", "or" và "between". Truy vấn MongoDB có thể được mở rộng với các biểu thức chính quy và bên cạnh đó MongoDB cung cấp hỗ trợ đến các hoạt động như đếm và phân biệt.

Hầu hết các kho lưu trữ tài liệu cũng hỗ trợ giao diện REST . Tuy nhiên những kho dữ liệu đó không cung cấp bất kỳ ngôn ngữ truy vấn nào có cú pháp giống như SQL.

Truy vấn họ cột thường hỗ trợ các truy vấn phạm vi và hoạt động "trong", "và", "hoặc" và biểu thức chính quy. Tuy nhiên, những thao tác đó chỉ có thể được áp dụng trên các phím hàng. Ngay cả khi kho lưu trữ dữ liệu CF cung cấp ngôn ngữ truy vấn (ví dụ: CQL cho Cassandra) – chỉ các phím hàng có thể được xem xét trong mệnh đề where chứ không phải các giá trị.

Cơ sở dữ liệu đồ thị cung cấp khả năng truy vấn theo hai cách: so khớp mẫu đồ thị và duyệt đồ thị. Khớp mẫu thường đề cập đến một cơ chế thực hiện một tìm kiếm các phần của biểu đồ ban đầu sẽ khớp với mẫu. Đồ thị đi qua là một kỹ thuật tìm kiếm đồ thị khác đi qua đồ thị theo một số mô tả và bắt đầu từ nút đã chọn. Khả năng đi ngang có thể hỗ trợ nhiều các chiến lược như BFS (tìm kiếm theo chiều rộng trước) và DFS (tìm kiếm theo chiều sâu).

Hầu hết các cơ sở dữ liệu đồ thị đều cung cấp API REST và bổ sung trong cơ sở dữ liệu đồ thị có một ngôn ngữ chung được hỗ trợ bởi nhiều cơ sở dữ liệu đồ thị SPARQL bao gồm cả Neo4j.

API REST là API (Giao diện chương trình ứng dụng) tuân thủ các nguyên tắc của REST (Chuyển trạng thai biểu hiện). REST là một kiểu kiến trúc sử dụng HTTP đơn giản yêu cầu liên lạc giữa các máy. Sử dụng REST có nghĩa là các lệnh gọi API sẽ là thông báodựa trên tiêu chuẩn HTTP.

Tóm lại, các lớp lưu trữ dữ liệu NoSQL khác nhau khác nhau đáng kể trong truy vấn của chúng khả năng và hoàn toàn cần thiết để xem xét các yêu cầu ứng dụng cụ thể để chọn loại cơ sở dữ liệu NoSQL phù hợp nhất với mô hình dữ liệu phù hợp và các giao diện được cung cấp. Ngược lại, hầu hết các kho lưu trữ dữ liệu NewSQL đang cung cấp hỗ trợ SQL như một trong những các tính năng chính của họ. Tuy nhiên, NuoDB được coi là tuân thủ SQL hơn, trong khi.

Ví dụ, VoltDB có nhiều hạn chế khác nhau: không thể sử dụng “có” mệnh đề, các bảng không thể tự nối và tất cả các bảng đã nối phải được phân vùng trên các giá trị giống nhau.

- **Map Reduce**

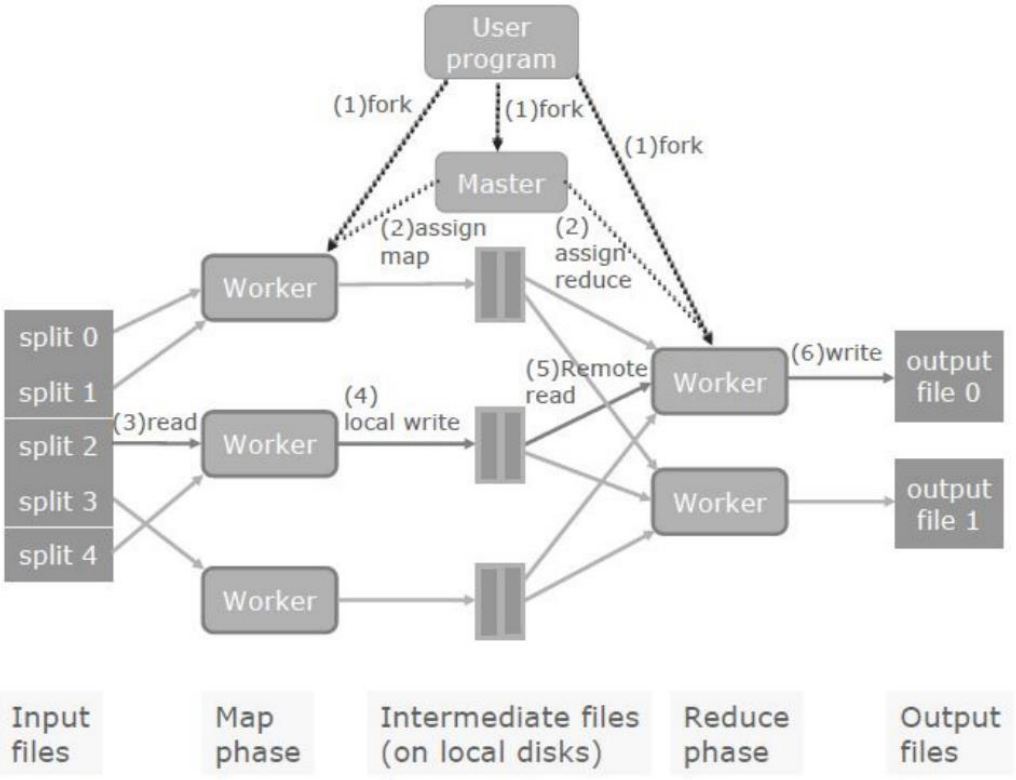
MapReduce là một mô hình lập trình song song để xử lý tập dữ liệu lớn được giới thiệu bởi Google. MapReduce thường được sử dụng để thực hiện tính toán phân tán trên các cụm máy tính.

Trong mô hình này, người dùng chỉ định tính toán bằng hai chức năng, Bản đồ và Giảm:

- Bản đồGiảm Trong giai đoạn lập bản đồ, MapReduce lấy dữ liệu đầu vào và cung cấp từng dữ liệu phần tử cho người ánh xạ.

- Trong giai đoạn giảm, bộ giảm tốc xử lý tất cả các đầu ra từ trình ánh xạ và đi đến một kết quả cuối cùng. Nói một cách đơn giản, trình ánh xạ có nghĩa là lọc và chuyển đổi đầu vào thành thứ gì đó mà bộ giảm tốc có thể tổng hợp lại.

Thư viện MapReduce bên dưới tự động song song hóa việc tính toán và xử lý các vấn đề phức tạp như phân phối dữ liệu, cân bằng tải và khả năng chịu lỗi. Việc triển khai MapReduce ban đầu của Google, cũng như mã nguồn mở của nó đối tác, Hadoop [46], nhằm mục đích tính toán song song trong các cụm lớn máy móc hàng hóa. Map Giảm đã trở nên phổ biến vì nó duyên dáng và tự động đạt được khả năng chịu lỗi. Nó tự động xử lý việc thu thập kết quả trên nhiều nút và trả về một kết quả hoặc một tập hợp. Mô hình MapReduce lợi thế là dễ dàng mở rộng quy mô xử lý dữ liệu trên nhiều nút điện toán.



Mô tả tổng quan về thực thi của MapReduce

Khi được áp dụng cho cơ sở dữ liệu, MapReduce có nghĩa là xử lý một bộ khóa bằng cách gửi logic quy trình (ánh xạ và rút gọn mã chức năng – xem ở trên) tới các nút lưu trữ, đến lượt họ áp dụng cục bộ chức năng bản đồ cho các khóa mà họ sở hữu.

Các cụm cửa hàng họ tài liệu và cột có thể lưu trữ một lượng lớn dữ liệu có cấu trúc, do đó các truy vấn có thể rất kém hiệu quả nếu chỉ một máy có để xử lý dữ liệu cần thiết. Đó là lý do tại sao tất cả họ tài liệu và cột các cửa hàng cung cấp khung MapReduce cho phép tính toán song song trên diện rộng bộ dữ liệu trên nhiều máy.

Cửa hàng NoSQL Key-Value, cơ sở dữ liệu đồ thị và NewSQL thường không hỗ trợ MapReduce.

NoSQL/NewSQL databases		Querying capabilities			
		Rest API	Query Language	Other API	MapReduce Support
Key Value Store	Voldemort	Yes	No	Clients for several languages	Yes
Document Store	MongoDB	Yes	No	CLI and API in different languages. Support Thrift interface	Yes
Column Family Store	Cassandra	Yes	Cassandra Query Language (CQL)	CLI and API in different languages	Yes
Graph Database	Neo4j	Yes	Cypher, Gremlin and SparQL	CLI and API in different languages	No
NewSQL	VoltDB	Yes	SQL	CLI and API in different languages. JDBC support.	No
	NuoDB	No	SQL	CLI and drivers for most common data access APIs (JDBC, ODBC, ADO.NET). Also provides a C++ support.	No

Khả năng Truy vấn NoSQL/NewSQL

- **Khả năng mở rộng**

Khả năng mở rộng theo chiều dọc là khả năng tăng quy mô hoặc thêm tài nguyên vào một nút trong một hệ thống, thường liên quan đến việc bổ sung CPU hoặc bộ nhớ cho một máy tính.

Khả năng mở rộng của ứng dụng đề cập đến hiệu suất được cải thiện của các ứng dụng đang chạy trên một phiên bản mở rộng của hệ thống Cơ sở dữ liệu truyền thống được thiết kế chủ yếu để mở rộng quy mô theo chiều dọc bằng cách bổ sung thêm sức mạnh đến một nút duy nhất.

Khả năng mở rộng theo chiều ngang là khả năng mở rộng quy mô hoặc thêm nhiều nút (máy chủ) vào hệ thống. Khi giá máy tính đã giảm và hiệu suất tiếp tục tăng, các ứng dụng tính toán hiệu suất đã áp dụng các hệ thống "hàng hóa" chi phí thấp cho những nhiệm vụ mà trước đây cần phải có siêu máy tính. Khả năng mở rộng kích thước là tối đa số lượng bộ xử lý mà một hệ thống có thể chứa. Khả năng mở rộng theo chiều ngang là một trong những các đặc điểm chính của hệ thống NoSQL và NewSQL.

Có những khía cạnh của cây được xem xét liên quan đến khả năng mở rộng:

- Mở rộng yêu cầu đọc

- Mở rộng yêu cầu ghi
- Mở rộng quy mô lưu trữ

Trong các chương tiếp theo, chúng tôi sẽ tập trung vào các chiến lược chính có ảnh hưởng lớn về các khả năng mở rộng này trong NoSQL/NewSQL: Kiểm soát tương tranh, Sao chép, Phân vùng và nhất quán cuối cùng.

- **Khả năng kiểm soát**

Kiểm soát tương tranh được quan tâm đặc biệt trong kho lưu trữ dữ liệu NoSQL và NewSQL, bởi vì họ thường cần đáp ứng một lượng lớn người dùng có quyền truy cập vào dữ liệu nguồn song song.

Cơ sở dữ liệu truyền thống (RDBMS) sử dụng chiến lược đồng thời bi quan với độc quyền truy cập trên một tập dữ liệu. Kiểm soát đồng thời bi quan hoặc giả định khóa bi quan mở rộng yêu cầu đọc mở rộng yêu cầu ghi mở rộng quy mô lưu trữ rằng hai hoặc nhiều người dùng đồng thời sẽ cố gắng cập nhật cùng một bản ghi cùng một lúc. Để ngăn chặn tình trạng này, một khóa được đặt vào thực thể được truy cập, để độc quyền quyền truy cập chỉ được đảm bảo cho một hoạt động của người dùng, các máy khách khác truy cập cùng một đối tượng sẽ phải đợi cho đến khi đối tượng ban đầu hoàn thành công việc của nó. Những chiến lược này có thể được phù hợp, nếu chi phí khóa thấp. Tuy nhiên trong các cụm cơ sở dữ liệu được phân phối trên khoảng cách lớn, các chiến lược nhất quán bi quan có thể dẫn đến hiệu suất xuống cấp, đặc biệt là khi các ứng dụng phải hỗ trợ tỷ lệ yêu cầu đọc cao.

Kiểm soát đồng thời lạc quan hoặc khóa lạc quan giả định rằng xung đột có thể xảy ra, nhưng chúng không phổ biến. Vì vậy, trước khi dữ liệu thay đổi được cam kết, mọi giao dịch kiểm tra xem các giao dịch khác có thực hiện bất kỳ sửa đổi xung đột nào đối với giao dịch tương tự hay không bộ dữ liệu. Nếu có xung đột được xác định, giao dịch sẽ được khôi phục. Đây chiến lược có thể hoạt động tốt nếu các bản cập nhật rất hiếm và do đó có cơ hội xung đột tương đối thấp. Trong tình huống này, khôi phục lại sẽ rẻ hơn khi khóa tập dữ liệu độc quyền như trong khóa bi quan.

Một số kho lưu trữ dữ liệu bao gồm Voldemort và NuoDB triển khai nhiều phiên bản điều khiển tương tranh (MVCC). Trong truy cập đồng thời không được quản lý bằng khóa mà bằng tổ chức của nhiều phiên bản được sắp xếp theo trình tự thời gian không thể sửa đổi. Nhiều phiên bản là được lưu trữ, nhưng chỉ một cái được đánh dấu là hiện tại, tất cả những cái còn lại được đánh dấu là lỗi thời. Sử dụng chiến lược này, thao tác đọc có thể xem dữ liệu như khi nó bắt đầu đọc, trong khi một quy trình đồng thời có thể thực hiện thao tác ghi trên cùng một tập dữ liệu trong trong khi đó .

Bên cạnh việc cắt giảm tính nhất quán, MVCC cũng gây ra các yêu cầu về không gian cao hơn khi nhiều các phiên bản được giữ song song. Vì vậy, thông thường để làm việc với MVCC, cần phải có một trình thu gom rác xóa các phiên bản không còn cần thiết và cả một số xung đột thuật toán giải quyết để đối phó với sự không nhất quán. Điều này gây ra hệ thống bổ sung sự phức tạp. Một số cơ sở dữ liệu NoSQL cho phép các ứng dụng triển khai đồng thời tối ưu kiểm soát bằng cách cung cấp các nguyên tắc như "kiểm tra và thiết lập" (CAS). Phương pháp CAS được sử dụng để đảm bảo rằng thao tác ghi sẽ chỉ được thực hiện nếu không có thao tác máy khách nào khác được thực hiện ghi khác kể từ khi dữ liệu được đọc lần cuối.

Một số giải pháp NewSQL cũng triển khai các cách tiếp cận sáng tạo để xử lý đồng thời điều khiển. NuoDB chủ yếu dựa vào MVCC như đã thảo luận ở trên. Ngược lại, VoltDB có một cách tiếp cận khác về đồng thời. Kho dữ liệu này giả định rằng toàn bộ cơ sở dữ liệu có thể vừa với bộ nhớ, nghĩa là có đủ bộ

nhớ và các giao dịch chỉ tồn tại trong thời gian ngắn. Dựa trên những giả định này, các giao dịch được thực hiện tuần tự trong một môi trường luồng đơn, không khóa.

Cơ sở dữ liệu NoSQL/NewSQL		**Khả năng kiểm soát**
Key-value stores	Voldermort	MVCC với đồng hồ vector
Document stores	MongoDB	Khóa đọc - viết
Column family stores	Cassandra	Khách hàng cung cấp là đã sử dụng đến xác định bản cập nhật gần nhất cho một cột. Dấu thời gian mới nhất luôn thắng và cuối cùng vẫn tồn tại
Graph databases	neo4j	Các khóa ghi được yêu cầu trên các nút và các mối quan hệ cho đến khi được cam kết
NewSQL databases	VoltDB	Mô hình đơn luồng, không kiểm soát đồng thời
	NuoDB	MVCC

Bảng 4: Tóm tắt các cơ chế kiểm soát tương tranh được sử dụng trong NoSQL/NewSQL cơ sở dữ liệu.

- **Nhân rộng**

Bên cạnh khả năng mở rộng ngày càng tăng và hiệu suất được cải thiện thông qua cân bằng tải, sao chép cũng mang lại độ tin cậy, khả năng chịu lỗi và độ bền tốt hơn. Như những gì chúng ta có đã được đề cập trong chương "Định lý CAP" Eric Brewer nhận thấy rằng chỉ có đầy đủ tính khả dụng hoặc tính nhất quán đầy đủ có thể được đảm bảo cùng một lúc trong các hệ thống phân tán.

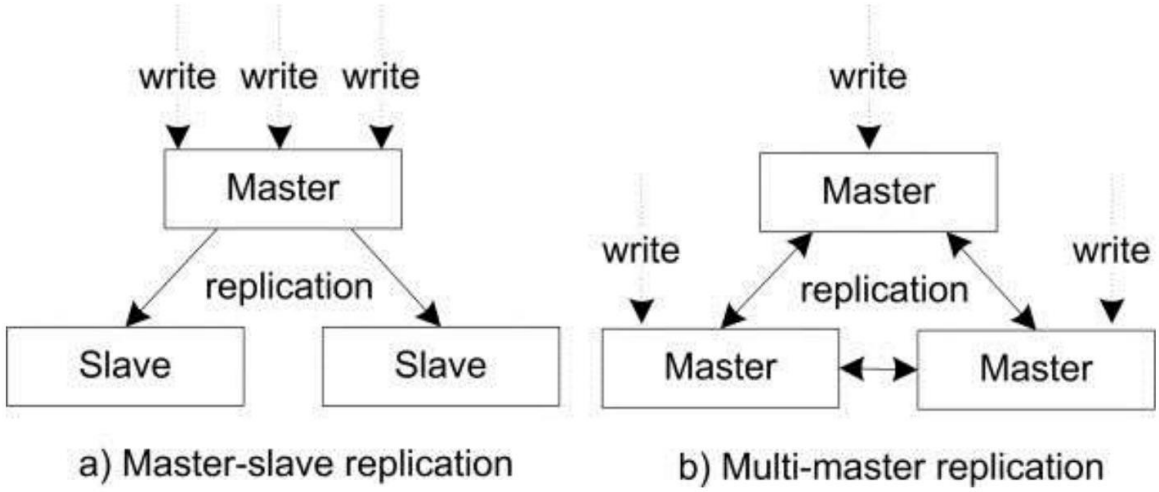
Do đó, nếu tất cả các bản sao trong bản gốc được cập nhật đồng bộ, hệ thống sẽ không có sẵn cho đến khi tất cả các nô lệ đã thực hiện hoạt động. Nhưng nếu tin nhắn bị mất do sự cố mạng, thì hệ thống sẽ không khả dụng cho đến khi tất cả nô lệ có thực hiện thao tác ghi.

Giải pháp này có thể không phù hợp với các hệ thống dựa trên tính sẵn sàng cao, như Web các nhà bán lẻ như Amazon hoặc hệ thống thương mại internet như eBay. Do đó những các hệ thống thường thực hiện sao chép không đồng bộ hoặc lười biếng, điều này có thể dẫn đến tình huống mà số lần đọc trên các bản sao có thể không nhất quán trong một khoảng thời gian ngắn. Đây cách tiếp cận được sử dụng bởi phần lớn các hệ thống NoSQL trong khi các hệ thống đó có thể được phân loại theo Brewer cuối cùng nhất quán hoặc cung cấp BASE (ngược lại với AXIT). Ngược lại, trong bản sao đồng bộ hoặc háo hức, tất cả các thay đổi đều được được truyền đến các nút trước khi gửi xác nhận về việc ghi thành công hoạt động. Điều này thực sự có nghĩa là độ trễ bổ sung được đưa ra khi thực thi thao tác ghi, bởi vì nó sẽ kết hợp tất cả thời gian lan truyền. Điều này có thể dẫn đến trong tình trạng suy giảm hiệu suất nghiêm trọng và do đó phương pháp này hiếm khi được sử dụng trong NoSQL các hệ thống.

Một đặc điểm xác định khác là chế độ sao chép, có hai loại: chính sao chép nô lệ và chủ-chủ.

Sao chép chủ-tớ là một sơ đồ trong đó một nút duy nhất được định nghĩa là chủ và đây là nút duy nhất chấp nhận và xử lý các yêu cầu ghi. Sau đó các thay đổi đang được truyền từ nút chủ sang các nút nô lệ. Hai trong số bốn NoSQL cơ sở dữ liệu được xem xét trong bài báo cuối cùng này hỗ trợ sao chép chủ-tớ: MongoDB và Neo4j.

Trong bản sao đa chủ, nhiều nút có thể xử lý ghi các yêu cầu, sau đó được truyền đến các nút còn lại. Vì vậy, về cơ bản trong master-slave việc truyền bá chỉ hoạt động theo một hướng: từ chủ đến nô lệ, theo nhiều hướng sự lan truyền bản sao chính có thể xảy ra theo nhiều hướng khác nhau.



Hình thể hiện sự khác biệt giữa hai kiểu sao chép: chủ-tớ và đa chủ

Các cơ sở dữ liệu NoSQL còn lại được khảo sát trong bài báo cuối cùng, Dự án Voldemort và Cassandra, hỗ trợ sao chép vô chủ, tương tự như đa master được mô tả ở trên, bởi vì nhiều nút chấp nhận yêu cầu "ghi". Tuy nhiên đó là được gọi là sao chép không chủ, bởi vì tất cả các nút đóng vai trò giống nhau trong hệ thống sao chép, nên không có chủ. Lưu ý rằng dữ liệu lưu trữ với cơ chế sao chép không chủ cũng sử dụng hàm băm nhất quán như một chiến lược phân vùng. Chiến lược sắp xếp bản sao liên quan rất nhiều đến vị trí nút trên vòng phân vùng.

Trong cả Cassandra và Voldemort, tổng số bản sao trong cụm là được gọi là hệ số sao chép và đó là một tham số hệ thống có thể định cấu hình. Các điều chỉnh cấu hình của RF có tác động lớn đến hiệu suất và chúng tôi có thực tế đã quan sát thấy rằng với Cassandra trong một trong những thí nghiệm chúng tôi tiến hành như một phần của Dự án cuối cùng. Chúng tôi đã thực hiện tải dữ liệu bằng 3 nút.

Cụm Cassandra: một lần với “hệ số sao chép” (RF) được đặt thành 1 và lần khác với RF được đặt thành 3, giúp thao tác ghi hoàn toàn bền bỉ. Kết quả thật thú vị mặc dù có thể dự đoán về mặt lý thuyết: thời gian tải với cài đặt RF cao hơn nhiều hơn lâu hơn hai lần đối với cùng kích thước của cơ sở dữ liệu.

Các mô hình sao chép NewSQL có thể được coi là không chủ hoặc đa chủ, bởi vì bất kỳ nút nào cũng có thể nhận được câu lệnh cập nhật. Trong NuoDB, các hàng nằm trong nội bộ được biểu diễn dưới dạng các đối tượng “trong bộ nhớ” giao tiếp không đồng bộ để sao chép trạng thái của chúng thay đổi. VoltDB có một trình quản lý giao dịch/phiên chịu trách nhiệm về nhận các bản cập nhật và chuyển tiếp nó tới tất cả các bản sao để được thực thi song song. Bảng 5 mô tả các kỹ thuật sao chép khác nhau được sử dụng trong cơ sở dữ liệu NoSQL/NewSQL.

Cơ sở dữ liệu NoSQL/NewSQL		Nhân rộng
Key-value stores	Voldermort	Masterless, sao chép không đồng bộ. Các bản sao nằm trên các nút R đầu tiên di chuyển qua vòng phân vùng theo chiều kim đồng hồ.
Document stores	MongoDB	Master-Slave, sao chép không đồng bộ. Được thiết kế cho hoạt động ngoại tuyến. Nhiều bản sao có thể duy trì các bản sao của cùng một dữ liệu và đồng bộ hóa chúng sau đó.
Column family stores	Cassandra	Masterless, sao chép không đồng bộ. Hai chiến lược để đặt các bản sao: các bản sao được đặt trên các nút R tiếp theo dọc theo vòng; hoặc, bản sao 2 được đặt trên nút đầu tiên dọc theo vòng thuộc về một trung tâm dữ liệu khác, với các bản sao còn lại trên các nút dọc theo vòng trên cùng một giá lúc đầu
Graph databases	neo4j	Master-slave, nhưng có thể xử lý các yêu cầu ghi trên tất cả các nút máy chủ. Yêu cầu viết cho nô lệ phải truyền đồng bộ đến chủ.
NewSQL databases	VoltDB	Cập nhật được thực hiện trên tất cả các bản sao cùng một lúc
	NuoDB	Đa chủ (sao chép đối tượng phân tán). không đồng bộ

Bảng 5: Các chế độ sao chép NoSQL/NewSQL

- **Phân vùng**

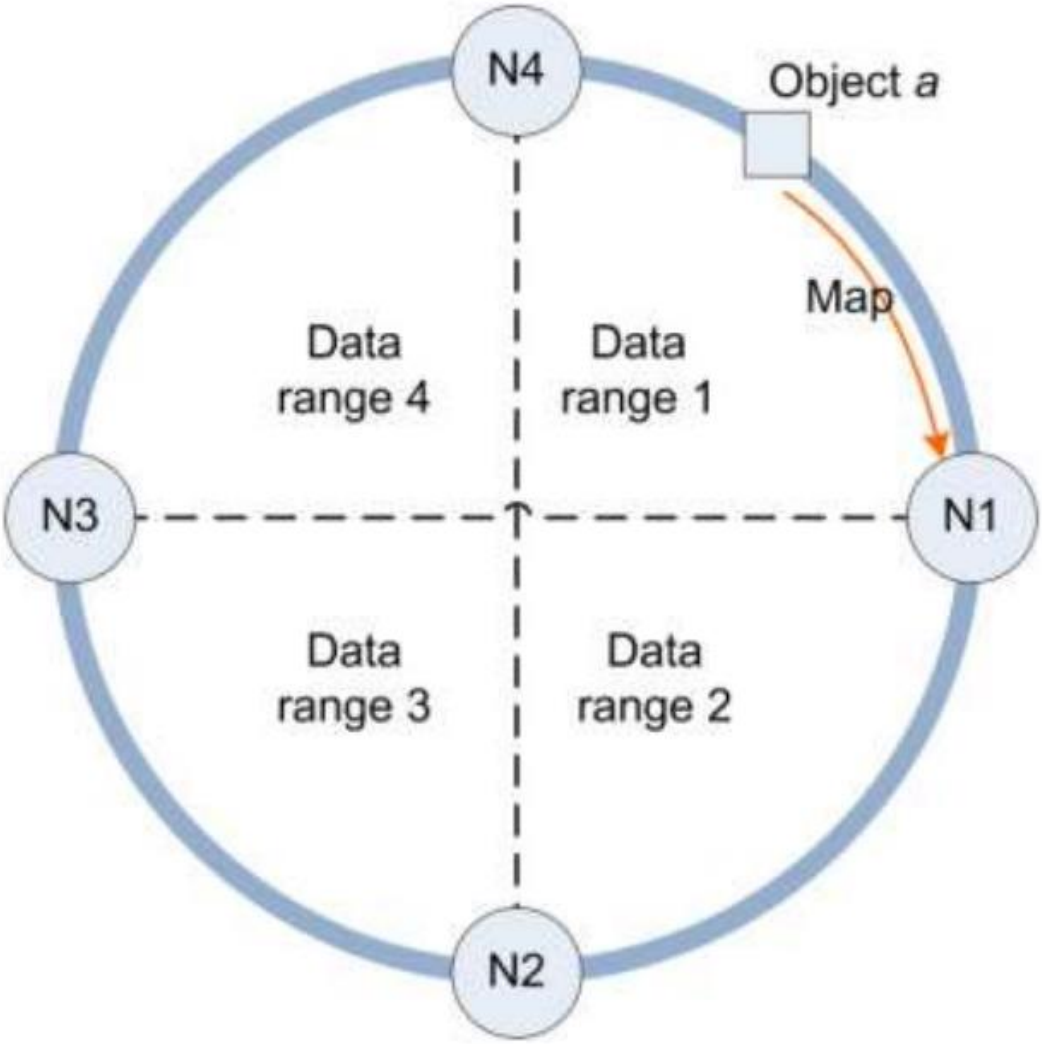
Khi chúng tôi xử lý Dữ liệu lớn, trong khi lượng dữ liệu khổng lồ và tỷ lệ đọc và đọc rất cao tỷ lệ yêu cầu ghi vượt quá khả năng của một máy, cơ sở dữ liệu phải được được phân vùng trên các cụm. Cơ sở dữ liệu quan hệ truyền thống (RDBMS) thường không hỗ trợ khả năng mở rộng theo chiều ngang, do mô hình dữ liệu được chuẩn hóa và ACID của chúng 45 bảo lãnh. Do đó, việc tăng gấp đôi số lượng máy chủ trong các cụm RDBMS không tăng gấp đôi hiệu suất. Vấn đề này là một trong những lý do tại sao những người chơi Web lớn thích Google, Amazon và Facebook bắt đầu phát triển cơ sở dữ liệu của riêng họ, đó là được thiết kế để mở rộng theo chiều ngang và do đó đáp ứng các yêu cầu rất cao về hiệu suất và năng lực của các hệ thống này.

Có nhiều kỹ thuật phân vùng được sử dụng bởi các kho lưu trữ dữ liệu khác nhau. Hầu hết NoSQL và các kho lưu trữ dữ liệu NewSQL thực hiện một số loại phân vùng ngang hoặc phân mảnh, thực sự liên quan đến việc lưu trữ các tập hợp hàng vào các phân đoạn/phân đoạn khác nhau. trên ngược lại, phân vùng theo chiều dọc liên quan đến việc lưu trữ các tập hợp cột thành các cột khác nhau phân đoạn và phân phối chúng cho phù hợp. Chiến lược phân vùng mạnh mẽ tùy thuộc vào mô hình dữ liệu. Ví dụ: trong các kho lưu trữ dữ liệu của Cột Family, họ thường hỗ trợ phân vùng theo chiều dọc, điều này là tự nhiên vì các phân đoạn phân vùng có chứa tập hợp các cột được xác định trước.

Vì các mô hình dữ liệu của khóa-giá trị, lưu trữ dữ liệu tài liệu và cơ sở dữ liệu họ cột các cửa hàng được định hướng rất quan trọng, hai chiến lược chính liên quan đến phân vùng theo chiều ngang cũng được định hướng chính.

- Đầu tiên được gọi là phân vùng phạm vi - chiến lược này phân phối tập dữ liệu cho các phân vùng cư trú trên các máy chủ khác nhau dựa trên phạm vi của khóa phân vùng. ban đầu định tuyến máy chủ chia toàn bộ tập dữ liệu theo phạm vi khóa của chúng. Sau đó mỗi nút chịu trách nhiệm lưu trữ và xử lý

đọc/ghi một dãy khóa cụ thể. Các lợi thế của phương pháp này là truy vấn theo phạm vi có thể trở nên rất hiệu quả, bởi vì các khóa lân cận thường nằm trên cùng một nút (trong cùng một phạm vi). Tuy nhiên, có một số nhược điểm của phương pháp này, chẳng hạn như các điểm nóng 46 và vấn đề cân bằng tải. Ví dụ, vì máy chủ định tuyến chịu trách nhiệm cân bằng tải, phân bổ tải chính và lời khuyên về khối phân vùng - có nghĩa là quá trình xử lý sẽ luôn tập trung vào một máy chủ đó (hoặc một vài máy chủ khi máy định tuyến được sao chép). Vì vậy, sự sẵn có của toàn bộ cụm phụ thuộc trên máy định tuyến cụ thể đó, nếu nó bị lỗi, nó có thể ảnh hưởng nghiêm trọng đến hoạt động của cụm.



Hình mô tả ví dụ băm nhất quán

Một trong những ưu điểm của hàm băm nhất quán là không cần ánh xạ dịch vụ như trong phân vùng phạm vi và vị trí mục tiêu của một đối tượng có thể được tính toán rất nhanh. Bên cạnh đó, phương pháp này cũng hiệu quả trong việc thay đổi kích thước động: nếu các nút được thêm vào hoặc xóa khỏi vòng, chỉ các vùng lân cận mới được gán lại cho các nút khác nhau và phần lớn các bản ghi vẫn không bị

ảnh hưởng. Tuy nhiên, băm nhất quán có thể có tác động xấu đến các truy vấn phạm vi vì các khóa liền kề được phân phối trên một số nút khác nhau.

Trong số các hệ thống được khảo sát trong bài báo này, hệ thống sau sử dụng hàm băm nhất quán: Dự án Voldermort, Cassandra và VoltDB.

Phân vùng trong cơ sở dữ liệu đồ thị là vấn đề khó khăn hơn nhiều so với các NoSQL khác cơ sở dữ liệu. Cửa hàng khóa-giá trị, kho lưu trữ tài liệu cơ sở dữ liệu họ cột thường cung cấp phân vùng dựa trên giá trị khóa, tương đối ổn định và có thể tra cứu bằng sử dụng các kỹ thuật tìm kiếm. Ngược lại, cơ sở dữ liệu đồ thị không có khóa ổn định và những cơ sở dữ liệu đó thực hiện tra cứu, nhưng bằng cách khai thác các mối quan hệ giữa các thực thể.

Do đó, có 2 yêu cầu trái ngược nhau liên quan đến cơ sở dữ liệu đồ thị: trước tiên, tất cả các nút liên quan của biểu đồ phải được giữ trên một nút để kích hoạt truyền tải hiệu quả mặc dù vậy, mặt khác, chúng tôi không thể giữ quá nhiều nút trên cùng một máy chủ, vì nó có thể gây ra tải nặng và các vấn đề về hiệu suất khác. Có nhiều cơ chế phân vùng cơ sở dữ liệu đồ thị được đề xuất, tuy nhiên việc triển khai rất hạn chế trong các hệ thống Graph DB thực tế, bởi vì chuyên sâu các hoạt động tái cân bằng có thể là kết quả của việc thay đổi dữ liệu nhanh chóng trong kho lưu trữ dữ liệu.

Hệ thống cơ sở dữ liệu đồ thị được quan sát trong bài viết này: Neo4j không triển khai phân vùng; tuy nhiên nó cung cấp một sharding bộ nhớ cache.

Sharding là một loại phân vùng cơ sở dữ liệu được sử dụng để phân tách các cơ sở dữ liệu rất lớn thành các phần nhỏ hơn, nhanh hơn, dễ quản lý hơn được gọi là phân đoạn dữ liệu.

Các hệ thống NewSQL cũng cung cấp nhiều loại giải pháp phân vùng khác nhau. Ví dụ VoltDB sử dụng phương pháp băm nhất quán trong đó mỗi bảng được phân vùng bằng một khóa duy nhất và các hàng được phân phối giữa các máy chủ. Thủ tục lưu trữ có thể được thực hiện trên một phân vùng hoặc trên tất cả chúng; tuy nhiên, nhược điểm là trong VoltDB, người dùng chịu trách nhiệm lựa chọn giữa các tùy chọn này. Một giải pháp NewSQL khác được phân tích trong bài báo cuối cùng này NuoDB sử dụng một cách tiếp cận hoàn toàn khác để phân vùng dữ liệu.

Một triển khai NuoDB được tạo thành từ một số Trình quản lý lưu trữ (SM) và Giao dịch Các nhà quản lý (TM). SM là các nút chịu trách nhiệm duy trì dữ liệu, trong khi TM là các nút xử lý các truy vấn. Mỗi SM có một bản sao hoàn chỉnh của toàn bộ dữ liệu, về cơ bản có nghĩa là không có phân vùng nào diễn ra trong SM.

Tuy nhiên, kho lưu trữ khóa-giá trị cơ bản được SM sử dụng có thể phân vùng dữ liệu theo chính nó, mặc dù người dùng không thể kiểm soát hoặc xem được.

- **Tính nhất quán**

Tính nhất quán có liên quan rất chặt chẽ đến việc triển khai phân vùng được thảo luận trong chương trước. Nói chung có 2 loại nhất quán chính có thể được phân biệt: tính nhất quán mạnh mẽ và tính nhất quán cuối cùng.

Tính nhất quán cao đảm bảo rằng khi tất cả các yêu cầu ghi được xác nhận, dữ liệu được hiển thị cho tất cả các hoạt động đọc tiếp theo. Thường mạnh (ngay lập tức) tính nhất quán có thể đạt được bằng cách sao chép đồng bộ hoặc thiếu hoàn toàn nhân rộng. Tuy nhiên, 2 tùy chọn này không được chấp nhận

cho các mục đích mà NoSQL các cửa hàng dữ liệu được thiết kế để, bởi vì nó đưa ra độ trễ lớn và tác động tính khả dụng hoặc có ảnh hưởng xấu đến khả năng mở rộng.

Do đó, hầu hết các cơ sở dữ liệu NoSQL, bao gồm các hệ thống được phân tích trong bài viết này, đang triển khai loại nhất quán thứ 2 - nhất quán cuối cùng. cuối cùng mô hình nhất quán các thay đổi lan truyền đến các bản sao thông qua hệ thống đã cho đủ thời gian. Điều đó có nghĩa là một số nút có thể lỗi thời (không nhất quán) đối với một số khoảng thời gian. Sao chép không đồng bộ sẽ dẫn đến tính nhất quán cuối cùng, vì ở đó là độ trễ giữa xác nhận ghi và lan truyền thay đổi.

Mặc dù phần lớn các hệ thống NoSQL được liên kết với tính nhất quán cuối cùng, nhiều người trong số họ cung cấp một số mức cấu hình để thiết lập sự cân bằng giữa hiệu suất và mức độ nhất quán

- Một trong những hệ thống NoSQL được xem xét trong bài báo này, MongoDB có thể đạt được hiệu suất mạnh mẽ tính nhất quán bằng cách sử dụng 2 tham số: đầu tiên được đặt để chỉ đọc từ bản gốc, nghĩa là chỉ một nút sẽ được truy cập để đọc và "khả năng mở rộng đọc" sẽ được loại bỏ. Một cách khác là đặt tham số "ghi mối quan tâm" thành "bản sao được công nhận", điều này đảm bảo rằng quá trình ghi được hoàn tất thành công trên tất cả các bản sao. Những kỹ thuật này thực sự buộc lưu trữ dữ liệu đồng bộ sao chép và do đó làm giảm hiệu suất.
- Hai cơ sở dữ liệu NoSQL khác: Project Voldemort và Cassandra sử dụng nhất quán băm và sao chép không chủ, các hệ thống đó cũng sử dụng một đại biểu trong Tính nhất quán. Số đại biểu có nghĩa là số lượng bản sao tối thiểu phải đáp ứng một yêu cầu cho nó được coi là thành công. Đối với thao tác đọc, họ sử dụng "đọc số đại biểu" và cho các hoạt động ghi – đó là một "số đại biểu ghi" riêng biệt. Nếu phát biểu sau là đúng: R + W > RF, trong đó R là "số đại biểu đã đọc", W là "viết số đại biểu" và RF là Hệ số sao chép (RF đã được giải thích trong chương "sao chép" ở trên), sau đó chúng tôi kết thúc với một dữ liệu nhất quán mạnh mẽ cửa hàng.
- Một trong hai hệ thống NewSQL – VoltDB nhất quán mạnh mẽ, giao dịch đầy đủ DB trong khi cái còn lại NuoDB sử dụng bản sao không đồng bộ và do đó có thể được định nghĩa là kho lưu trữ dữ liệu nhất quán cuối cùng.
- Bảng 6 tóm tắt các kiểu phân vùng và nhất quán của tất cả NoSQL và NewSQL các hệ thống được khảo sát trong bài báo cuối cùng này.

Cơ sở dữ liệu NoSQL/NewSQL		**Phân vùng**	**Tính nhất quán**
Key-value stores	Voldermort	Băm nhất quán	Có thể cấu hình, dựa trên đại biểu đọc và viết yêu cầu.
Document stores	MongoDB	Phân vùng phạm vi dựa trên khóa phân đoạn (một hoặc nhiều trường tồn tại trong mọi tài liệu trong bộ sưu tập). Ngoài ra, các khóa phân đoạn được băm có thể được sử dụng để phân vùng dữ liệu.	Có thể định cấu hình. Hai phương pháp để đạt được sự nhất quán mạnh mẽ: - Đặt kết nối để chỉ đọc từ chính - Đặt viết tham số quan tâm đến "Bản sao công nhận"
Column family stores	Cassandra	Băm nhất quán và phân vùng phạm vi (được gọi là phân vùng bảo toàn thứ tự trong thuật ngữ Cassandra) không được khuyến nghị do khả năng xảy ra các vấn đề về cân bằng tải và điểm nóng.	Có thể cấu hình, dựa trên đại biểu đọc và viết yêu cầu.

Graph databases	neo4j	Không phân vùng (chỉ sharding bộ nhớ cache).	Nhất quán cuối cùng.
NewSQL databases	VoltDB	Băm nhất quán. Người dùng xác định xem các thủ tục được lưu trữ sẽ chạy trên một máy chủ hay trên tất cả các máy chủ	Tính nhất quán mạnh mẽ.
	NuoDB	Không phân vùng. Kho lưu trữ giá trị khóa cơ bản có thể phân vùng dữ liệu nhưng người dùng không nhìn thấy được	Tính nhất quán mạnh mẽ.

Bảng 6: Kiểu phân vùng và tính nhất quán của NoSQL/NewSQL

- **Khả năng bảo vệ**

Bảo mật là một trong những khía cạnh quan trọng nhất của kho lưu trữ dữ liệu hiện đại. Vì khác nhau các loại dữ liệu nhạy cảm hoặc cá nhân có thể được lưu trữ trong NoSQL và NewSQL, bảo mật có trở thành mối quan tâm lớn đối với một công ty sẽ chọn công nghệ nào để nhận nuôi. Trong tiểu mục này, chúng ta sẽ khảo sát các kho dữ liệu và so sánh chúng với liên quan đến những điều sau đây:

- Xác thực
- Uỷ quyền
- Kiểm toán
- Mã hoá

Trong bài báo "Security Issues in NoSQL Databases" từ năm 2011, các nhà nghiên cứu từ Đại học Ben-Gurion đã khảo sát hai trong số các hệ thống mà chúng tôi liên quan đến trong bài báo này: MongoDB và Cassandra.

Họ đã tìm thấy nhiều vấn đề khác nhau, chẳng hạn như thiếu mã hóa trong giao tiếp và thiếu kiểm toán trong Cassandra. Tuy nhiên, một số trong những vấn đề đã được giải quyết trong hầu hết các các phiên bản gần đây của các kho lưu trữ dữ liệu đó và các khía cạnh bảo mật đó không còn cung cấp nguyên nhân cho mối quan tâm.

Cơ sở dữ liệu NoSQL/NewSQL		**Xác thực**	**Uỷ quyền**	**Kiểm toán**
Key-value stores	Voldermort	Không	Không	Không
Document stores	MongoDB	Có, cửa hàng thông tin xác thực trong một hệ thống thu thập. giao diện REST không hỗ trợ xác thực. doanh nghiệp Phiên bản hỗ trợ cũng Kerberos	Có, quyền bao gồm đọc, đọc viết, dbAdmin và userAdmin. độ chi tiết của bộ sưu tập	Không
Column family stores	Cassandra	Có, cửa hàng thông tin xác thực trong một bảng hệ thống. có thể cung cấp cắm được triển khai	Vâng, tương tự như CẤP SQL/ REVOKE phương pháp tiếp cận. Có thể cung cấp cắm được triển khai	Phiên bản doanh nghiệp chỉ có. Dựa trên log4j khuôn khổ. Danh mục ghi nhật ký

				bao gồm QUẢN TRỊ, TẤT CẢ, XÁC THỰC, DML, DDL, DCL, và QUERY. Khả thi để vô hiệu hóa đăng nhập cho không gian chính cụ thể
Graph databases	neo4j	Không, tuy nhiên nhà phát triển có thể tạo Bảo mật Quy tắc và đăng ký với máy chủ	Không	Không
NewSQL databases	VoltDB	Có, người dùng đang định nghĩa trong một tập tin triển khai đó cần phải được sao chép vào mỗi nút	Vâng, vai trò là xác định tại mức giản đồ, và từng được lưu trữ thủ tục xác định vai trò nào được phép thực hiện nó	Có, ghi nhật ký danh mục, bao gồm kết nối, SQL các câu lệnh, ảnh chụp nhanh, xuất khẩu, xác thực/ ủy quyền và khác
	NuoDB	Đúng	Có, giống như SQL	Có, ghi nhật ký danh mục, bao gồm câu lệnh SQL, sự kiện an ninh, thống kê chung và khác

Bảng 7: Bảo mật NoSQL/NewSQL: AAA

- **Mã hoá**

Điều này đề cập đến các cơ chế mã hóa dữ liệu để nó không thể đọc được bởi các bên trái phép. Một giải pháp mã hóa hoàn chỉnh phải có mặt trong ít nhất ba cấp độ khác nhau:

- Dữ liệu ở trạng thái nghỉ
- Máy khách đến máy chủ
- Máy chủ đến máy chủ

Cơ sở dữ liệu NoSQL/NewSQL		**Hỗ trợ mã hoá**		
		Dữ liệu ở trạng thái nghỉ	Máy khách / Máy chủ	Máy chủ/Máy chủ
Key-value stores	Voldermort	Có thể nếu BerkleyDB được sử dụng như một động cơ lưu trữ	Không	Không
Document stores	MongoDB	Không, tuy nhiên một phần ba đối tác bên (Gazzang) cung cấp một plug-in mã hóa Trong	Có – dựa trên SSL	Đúng

Column family stores	Cassandra	Phiên bản doanh nghiệp chỉ có. Nhật ký cam kết là không được mã hóa	Có – dựa trên SSL	Đúng, có thể định cấu hình: tất cả máy chủ đến người phục vụ liên lạc , chỉ giữa trung tâm dữ liệu hoặc giữa máy chủ trong cùng giá đỡ
Graph databases	neo4j	Không	Có – dựa trên SSL	Không
NewSQL databases	VoltDB	Không	Không	Không
	NuoDB	Cửa hàng bản địa không không hỗ trợ nó. Về mặt lý thuyết, nó có thể sử dụng một pluggable cửa hàng mà hỗ trợ nó	Đúng	Đúng

Bảo mật NoSQL/NewSQL: mã hóa

Quan sát chính về bảo mật là các giải pháp NoSQL vẫn chưa hoàn thiện như những hệ thống có trong hệ thống RDBMS truyền thống.

Một phát hiện thú vị khác là MongoDB và Cassandra cung cấp bảo mật bổ sung các chức năng trong phiên bản doanh nghiệp của họ, bởi vì bảo mật là một yếu tố đặc biệt quan trọng mối quan tâm đối với các công ty lớn. Ví dụ: kiểm tra và mã hóa dữ liệu tại phần còn lại là chỉ có sẵn trong Cassandra Enterprise Edition.

4. Ứng dụng và sự phù hợp với kịch bản

- **Key-value stores**

Do các cửa hàng giá trị khóa mô hình dữ liệu của chúng rất hữu ích cho các hoạt động đơn giản, đó là chỉ dựa trên các thuộc tính chính. Để tăng tốc độ trang web được hiển thị cụ thể của người dùng, các phần của trang này có thể được tính toán trước và phục vụ nhanh chóng và dễ dàng ra khỏi lưu trữ theo ID người dùng khi cần. Ngoài ra còn có một số trường hợp sử dụng khác cho khóa-giá trị dựa trên kho dữ liệu, ví dụ kho KV được sử dụng rộng rãi cho các loại ứng dụng sau:

- Bộ nhớ đệm: Vì hầu hết các kho lưu trữ giá trị khóa giữ tập dữ liệu của chúng trong bộ nhớ nên chúng đôi khi được sử dụng để lưu vào bộ đệm các truy vấn SQL tốn nhiều thời gian hơn.
- Sắp xếp: Một số cửa hàng K/V hỗ trợ danh sách, bộ, hàng đợi, v.v
- Phân phối thông tin / nhiệm vụ
- Lưu trữ thông tin trực tiếp: Các ứng dụng cần giữ trạng thái có thể sử dụng các cửa hàng K/V một cách dễ dàng.
- Ứng dụng giống như MySQL, truy vấn động, nhiều cập nhật dữ liệu

- **Document stores**

Cửa hàng tài liệu cung cấp tra cứu đa thuộc tính trên các bản ghi có thể đã hoàn thành các loại cặp giá trị khóa khác nhau, đó là một trong những lý do các hệ thống đó rất thuận tiện trong các tác vụ tích hợp dữ liệu và di chuyển lược đồ.

Những kho lưu trữ dữ liệu đó thường cũng hỗ trợ cấu trúc dữ liệu lồng nhau, liên kết và JSON các tài liệu bên cạnh các cửa hàng tài liệu như MongoDB được coi là rất dễ bảo trì và do đó phù hợp với các ứng dụng web linh hoạt. Phổ biến nhất trường hợp sử dụng là:

- Các trường hợp sử dụng thông tin lồng nhau: Kho lưu trữ dữ liệu dựa trên tài liệu cho phép bạn làm việc với dữ liệu được lồng sâu, cấu trúc dữ liệu phức tạp
- Phân tích thời gian thực
- Ghi nhật ký
- Lớp lưu trữ của các trang web nhỏ và linh hoạt như blog
- Các ứng dụng thân thiện với JavaScript khác nhau: Do đó, một trong những chức năng quan trọng nhất của dữ liệu dựa trên tài liệu các cửa hàng là cách chúng giao tiếp với các ứng dụng: sử dụng JSON thân thiện với JS.

Lưu trữ giá trị khóa, lưu trữ tài liệu và lưu trữ họ cột có điểm chung là chúng lưu trữ dữ liệu được định dạng hóa để đạt được lợi thế trong phân phối. Kết quả là các mối quan hệ phải được xử lý hoàn toàn trong logic ứng dụng. xã hội nổi tiếng Sự cố miền mạng - truy vấn bạn bè và đề xuất tương tự sẽ dẫn đến nhiều truy vấn, hình phạt hiệu suất và mã phức tạp trong ứng dụng lớp nếu một trong những cơ sở dữ liệu này đã được sử dụng cho các trường hợp sử dụng đó. NoSQL còn lại danh mục lưu trữ dữ liệu – Cơ sở dữ liệu đồ thị là giải pháp phù hợp hơn nhiều trong việc này miền.

- **Column family stores**

Cửa hàng họ cột chỉ lưu trữ một cặp giá trị khóa trong một hàng, nếu tập dữ liệu cần. Của nó cách tiếp cận hoàn toàn khác với cơ sở dữ liệu quan hệ truyền thống sẽ lưu trữ giá trị null trong mỗi cột mà tập dữ liệu không có giá trị. Do đó họ cột phù hợp hơn trong các miền có lượng dữ liệu khổng lồ với số lượng khác nhau thuộc tính. Do đó, các trường hợp sử dụng chính cho lưu trữ dữ liệu CF sẽ là:

- Giữ thông tin không có cấu trúc, không dễ bay hơi: Nếu một bộ sưu tập lớn các thuộc tính và giá trị cần được lưu giữ lâu dài trong một khoảng thời gian, kho lưu trữ dữ liệu dựa trên cột trở nên cực kỳ tiện dụng.
- Mở rộng quy mô: Các kho lưu trữ dữ liệu dựa trên cột có khả năng mở rộng cao về bản chất. Họ có thể xử lý một lượng thông tin khủng khiếp.
- Công cụ tìm kiếm, phân tích dữ liệu nhật ký.
    - **Graph databases**

Cơ sở dữ liệu đồ thị, chẳng hạn như Neo4j phù hợp trong các tình huống như:

- Khớp mẫu
- Phân tích phụ thuộc
- Hệ thống khuyến nghị
- Ứng dụng mạng xã hội
- Giải bài toán tìm đường trong hệ thống định vị

Một số cơ sở dữ liệu đồ thị và đặc biệt là Neo4j hoàn toàn tuân thủ ACID. Tuy nhiên, chúng không đủ như các loại NoSQL khác được xem xét ở trên trong các tình huống khác ngoài việc xử lý đồ thị và các mối quan hệ

- **NewSQL databases**

Cơ sở dữ liệu NewSQL đang cung cấp ACID và hỗ trợ các giao dịch, do đó đối sánh tự nhiên cho NewSQL sẽ là các ứng dụng liên quan đến OLTP (giao dịch trực tuyến Chế biến). Ví dụ, theo Giáo sư Stonebreaker NewSQL sẽ phù hợp với các lĩnh vực sau:

- Các ứng dụng trên nền web như
  - Trò chơi nhiều người,
  - Các trang mạng xã hội, và
  - Mạng cờ bạc trực tuyến.
- Các ứng dụng điện thoại thông minh sử dụng điện thoại làm cảm biến địa lý và cung cấp dịch vụ dựa trên địa điểm.
- Các ứng dụng giữa máy với máy (M2M).

	Aerospike	Cassandra	Couchbase	CouchDB	HBase	MongoDB	Voldemort
Availability	⊞	⊞	⊞	⊞	−	−	⊞
Consistency	⊞	⊞	+	+	□	⊞	+
Durability	−	+	+	−	+	+	+
Maintainability	+	□	+	+	−	□	−
Read-Performance	+	−	⊞	□	−	⊞	+
Recovery Time	⊞	⊖	+	?	?	⊞	?
Reliability	−	+	−	+	+	⊞	?
Robustness	+	+	□	□	⊖	□	?
Scalability	⊞	⊞	⊞	−	⊞	−	+
Stabilization Time	⊖	+	+	?	?	⊖	?
Write-Performance	+	⊞	+	−	+	−	⊞

Legend:
⊞ Great
\+ Good
□ Average
− Mediocre
⊖ Bad
? Unknown/N.A.

Bảng tóm tắt các thuộc tính chất lượng khác nhau được nghiên cứu cho các cơ sở dữ liệu phổ biến

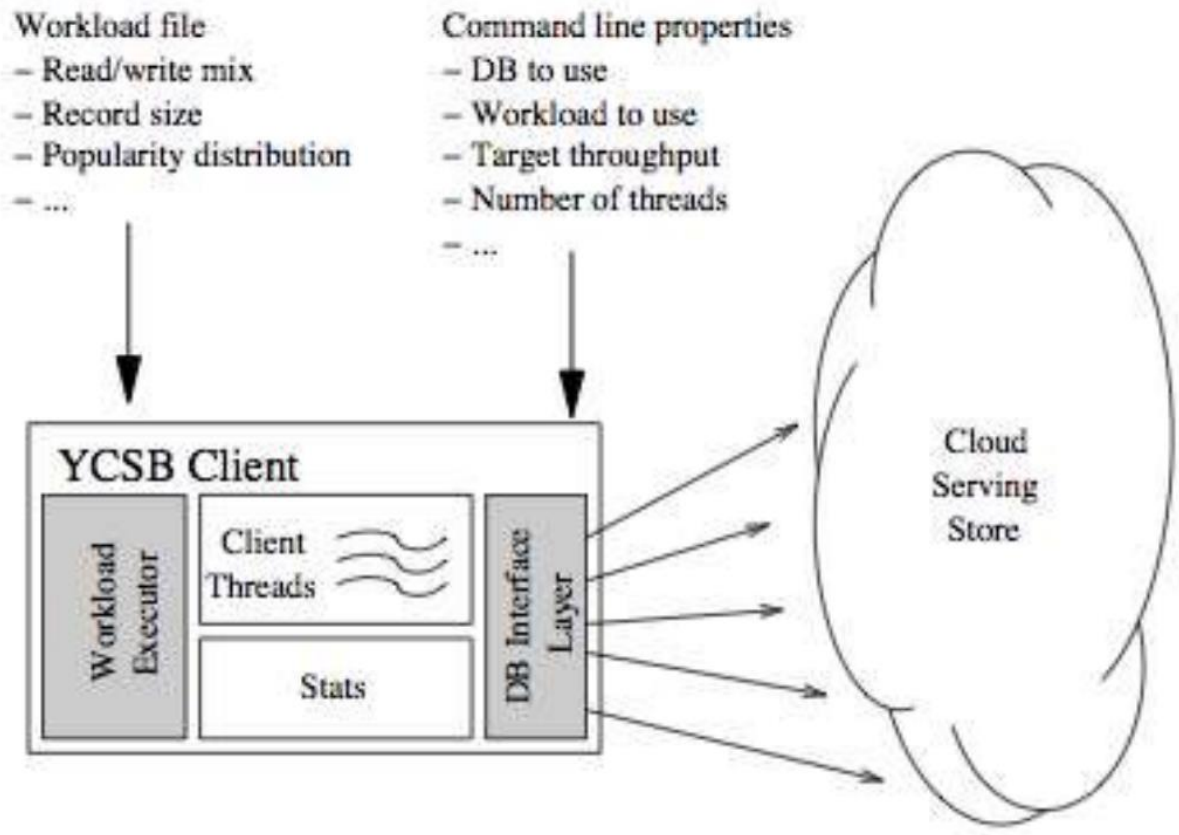
5. Khảo sát các bài báo thảo luận về hiệu suất

Trong chương này, chúng tôi cung cấp một cuộc khảo sát ngắn về một số bài viết liên quan đến NoSQL/NewSQL so sánh hiệu năng cơ sở dữ liệu

5.1. “Yahoo! Cloud Serving Benchmark”

Trước tiên, chúng tôi phải cung cấp một mô tả ngắn gọn về bộ đo điểm chuẩn YCSB được sử dụng trong các thí nghiệm được mô tả trong tất cả các bài báo được xem xét trong phần này của bài báo. YCSB từ viết tắt của Yahoo! Điểm chuẩn phục vụ đám mây. Hệ thống YCSB đã được phát triển bởi một nhóm các nhà phát triển vào năm 2010 và mục tiêu chính của nó là đưa ra một giải pháp có thể mở rộng và khuôn khổ chung để đánh giá các cửa hàng khóa-giá trị.

YCSB cho phép tạo khối lượng công việc tổng hợp được định nghĩa là khối lượng công việc có thể định cấu hình phân phối các hoạt động CRUD (tạo, đọc, cập nhật và xóa) trên một tập hợp các bản ghi. Các bản ghi có một số trường được xác định trước và được lập chỉ mục logic bằng một khóa. Đây mô hình dữ liệu chung có thể dễ dàng được ánh xạ tới một khóa-giá trị cụ thể hoặc dữ liệu dựa trên cột người mẫu.



Thiết kế cấp cao của YCSB

Lý do cho sự phổ biến của YCSB là nó bao gồm một trình tạo dữ liệu, một khối lượng công việc trình tạo, cũng như trình điều khiển cho một số cửa hàng khóa-giá trị, một số trong đó cũng được sử dụng trong đánh giá này.

5.2. “So sánh giữa một số cơ sở dữ liệu NoSQL với nhận xét và ghi chú”

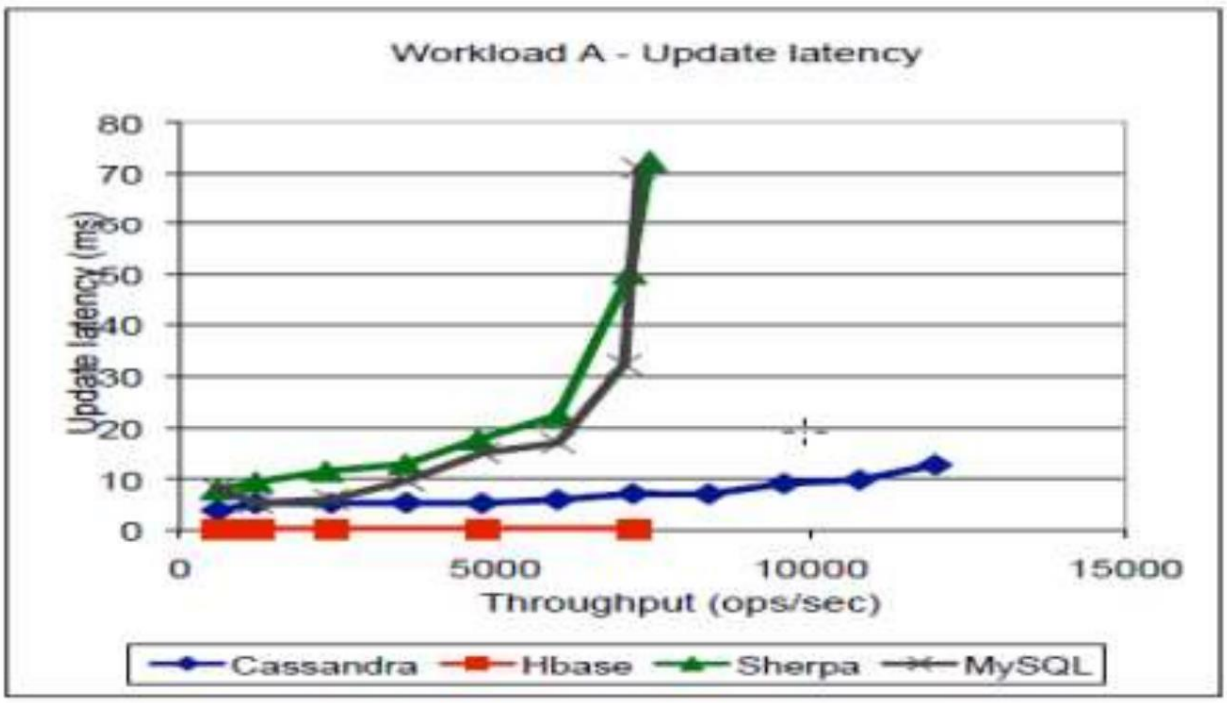
Các tác giả của bài viết đó đang cố gắng bình luận về các NoSQL khác nhau (Không chỉ Ngôn ngữ truy vấn có cấu trúc) và để so sánh giữa chúng. Ban đầu họ cung cấp định nghĩa cho thuật ngữ NoSQL, ACID và BASE và chỉ ra một số khác biệt giữa cách tiếp cận NoSQL và RDBMS truyền thống. Hơn nữa họ đang cố gắng để xác định "những gì để so sánh" và đưa ra phân loại, tương tự như cái được sử dụng trong bài viết này, nhưng hơi khác một chút:

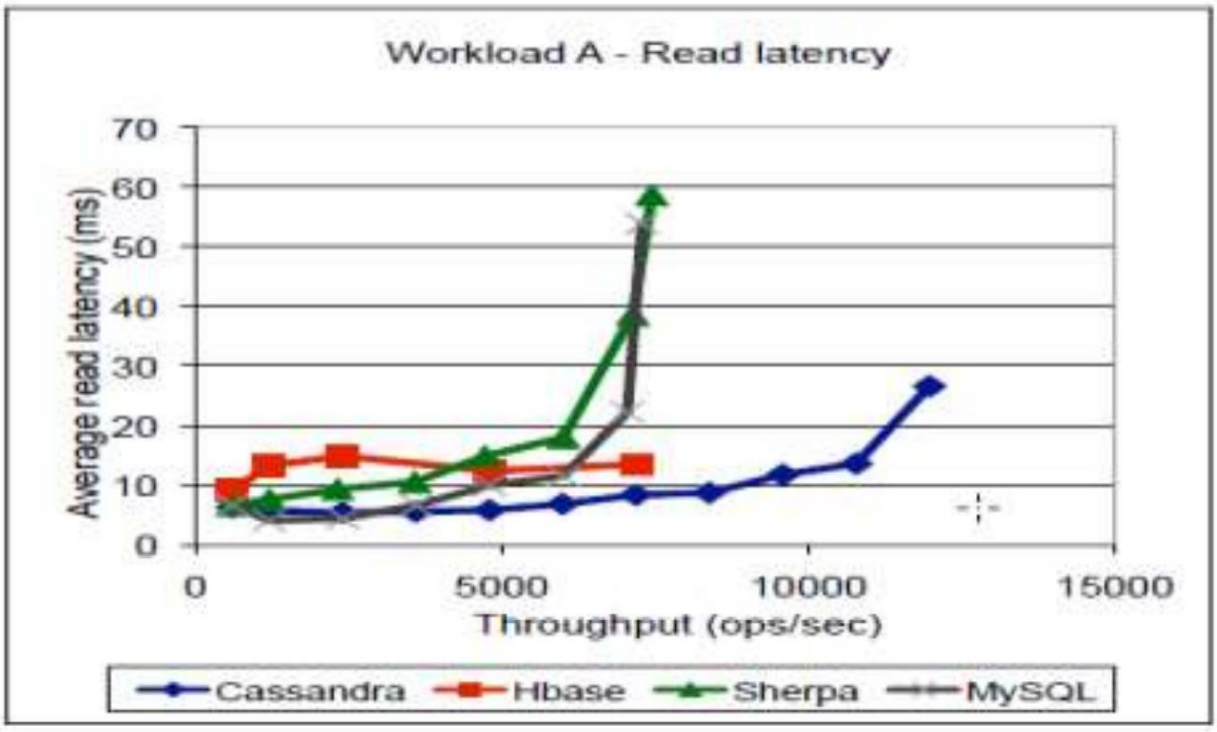
- Wide Column
- Document Store
- Tuple Store
- Eventually Consistent Key Value Store
- Graph Databases

Hơn nữa, họ đề xuất hai cách tiếp cận để so sánh - một là định tính và một là định lượng.

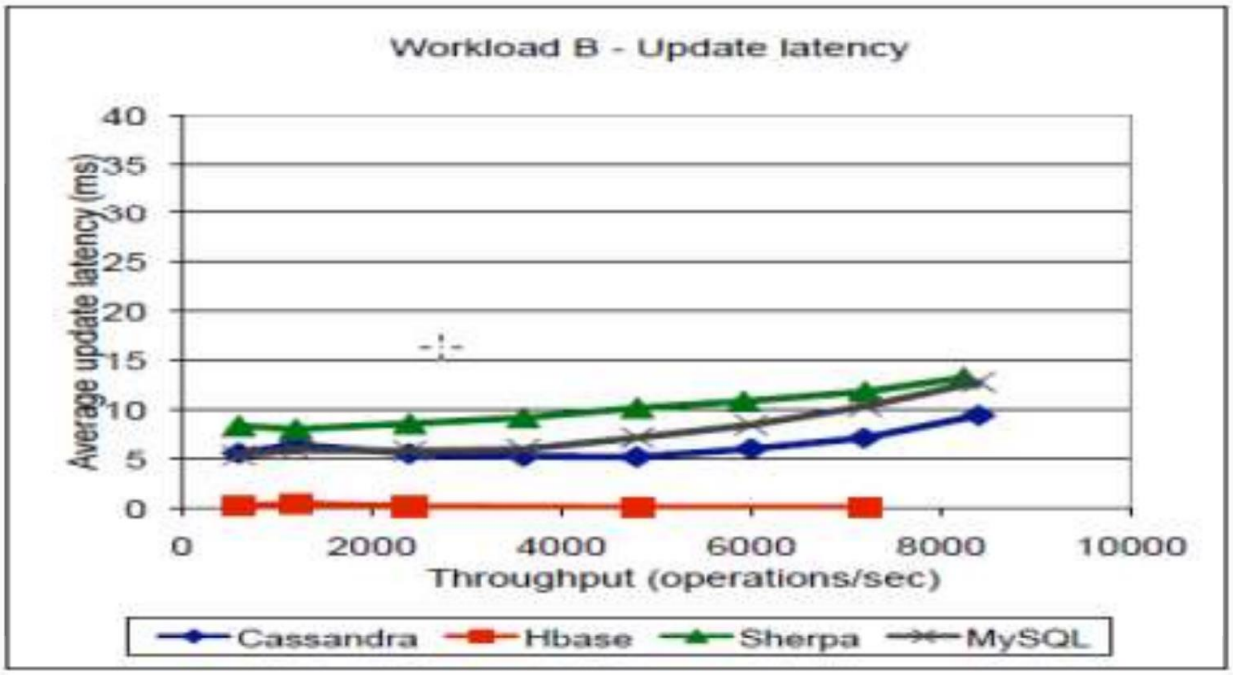
- Đánh giá ĐỊNH TÍNH được thực hiện bằng cách so sánh các tính năng có sẵn cho cơ sở dữ liệu NoSQL được tính đến.
- Đối với các tiêu chí đánh giá ĐỊNH LƯỢNG, họ đã sử dụng hai bộ khác nhau, một bộ liên quan đến kích thước và một liên quan đến hiệu suất.

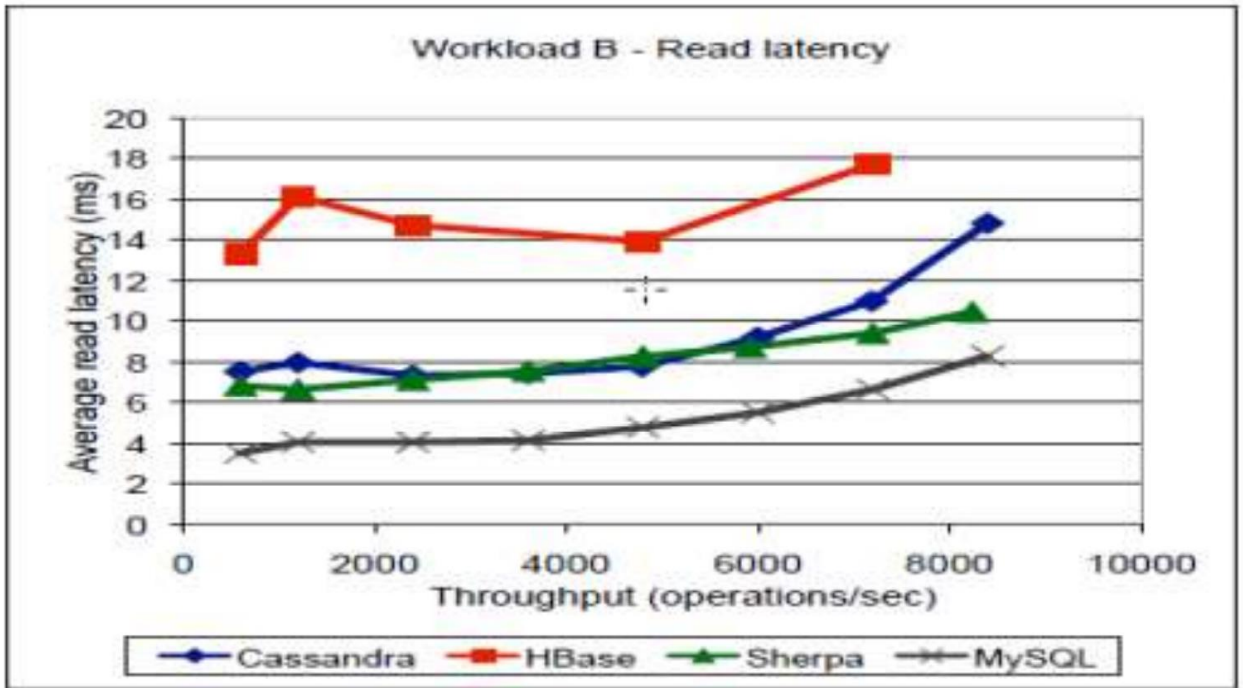
Các phép đo hiệu suất được liệt kê trong một vài đoạn cuối và được bao gồm trong các số liệu được lấy từ đang mô tả điểm chuẩn dựa trên phòng thí nghiệm sử dụng YCSB (Yahoo! Cloud Server Benchmark) làm công cụ đo lường. Điểm chính xác đã được chạy trên 120 triệu bản ghi có kích thước nhỏ (1kB), 6 nút và tương đương 0,12 TB cài đặt của ba sản phẩm. Hình dưới cho thấy kết quả của thí nghiệm.





Độ trễ của khối lượng công việc A





Độ trễ của khối lượng công việc B

Cuối cùng, kết luận mà các tác giả đưa ra là mặc dù SQL và Cơ sở dữ liệu NoSQL có một số tính năng được chia sẻ, nhưng chúng không giống nhau về hành vi. Do đó, chúng không thể được sử dụng thay thế cho nhau để giải quyết bất kỳ loại vấn đề nào và cho một vấn đề cụ thể nhất định, người ta sẽ xem xét cả hai loại và chọn giữa chúng.

5.3. ”Cơ sở dữ liệu NoSQL: MongoDB so với Cassandra”

Bài viết này chủ yếu tập trung vào việc so sánh hai NoSQL phổ biến nhất cơ sở dữ liệu: MongoDB và Cassandra. Trong chương công việc liên quan, các tác giả liên quan đến cùng một bài báo nổi tiếng mô tả hoạt động của YCCB và họ chỉ ra rằng Brian F. Cooper và cộng sự. đã phân tích cơ sở dữ liệu NoSQL và hiệu suất cơ sở dữ liệu MySQL bằng YCSB điểm chuẩn bằng độ trễ liên quan với số lượng hoạt động mỗi giây. trong họ paper họ đang ưu tiên các thông số thực thi khác nhau.

Hơn nữa, bài báo cung cấp định nghĩa cho các thuật ngữ NoSQL, BASE và định nghĩa phân loại chính xác giống như chúng tôi sử dụng trong bài báo cuối cùng này.

Bài viết cũng đưa ra một mô tả ngắn gọn cho MongoDB và Cassandra và tóm tắt cả hai dữ liệu lưu trữ các tính năng như sau:

	MongoDB	Cassandra
Development language	C++	Java
Storage Type	BSON files	Column
Protocol	TCP/IP	TCP/IP
Transactions	No	Local
Concurrency	Instant update	MVCC
Locks	Yes	Yes
Triggers	No	Yes
Replication	Master-Slave	Multi-Master
CAP theorem	Consistency, Partition tolerance	Partition tolerance, High Availability
Operating Systems	Linux / Mac OS / Windows	Linux / Mac OS / Windows
Data storage	Disc	Disc
Characteristics	Retains some SQL properties such as query and index	A cross between BigTable and Dynamo. High availability
Areas of use	CMS system, comment storage	Banking, finance, logging

Các tính năng của MongoDB và Cassandra

Bài báo mô tả các thử nghiệm được thực hiện bằng YCCB (Yahoo! Cloud Điểm chuẩn phục vụ). YCSB có một ứng dụng khách bao gồm hai phần: trình tạo khối lượng công việc và tập hợp các tình huống. Các kịch bản, được gọi là khối lượng công việc, là sự kết hợp của đọc, thao tác ghi và cập nhật được thực hiện trên các bản ghi được chọn ngẫu nhiên. khối lượng công việc đã được sử dụng trong các thí nghiệm được mô tả trong bài viết sau đây:

- Workload A: Cập nhật khối lượng công việc nặng. Khối lượng công việc này có sự kết hợp của 50/50 lần đọc và cập nhật.

- Workload B: Đọc phần lớn khối lượng công việc. Khối lượng công việc này có tỉ lệ đọc/cập nhật là 95/5
- Workload C: Chỉ đọc. Khối lượng công việc này được đọc 100%.
- Workload D: Đọc khối lượng công việc mới nhất. Trong khối lượng công việc này, các bản ghi mới được chèn vào, và các bản ghi được chèn gần đây nhất là phổ biến nhất.
- Workload E: Phạm vi ngắn. Trong khối lượng công việc này, phạm vi bản ghi ngắn được truy vấn, thay vì các bản ghi riêng lẻ
- Workload F: Đọc-sửa đổi-ghi. Trong khối lượng công việc này, khách hàng sẽ đọc một bản ghi, sửa đổi nó và viết lại các thay đổi. Bởi vì trọng tâm của chúng tôi là cập nhật và đọc hoạt động, khối lượng công việc D và E sẽ không được sử dụng. Thay vào đó, để hiểu rõ hơn cập nhật và hiệu suất đọc, hai khối lượng công việc bổ sung đã được xác định:
- Workload G: Cập nhật phần lớn khối lượng công việc. Khối lượng công việc này có 5/95 lượt đọc/cập nhật pha trộn
- Workload H: Chỉ cập nhật. Khối lượng công việc này là bản cập nhật 100%.

Cuối cùng, sau khi thực hiện nhiều thử nghiệm bằng YCCB và khối lượng công việc nêu trên tác giả đi đến kết luận sau:

- Với việc tăng kích thước dữ liệu, MongoDB bắt đầu giảm hiệu suất
- Cassandra trở nên nhanh hơn khi làm việc với lượng dữ liệu tăng lên
- Cassandra nhanh hơn MongoDB, cung cấp thời gian thực hiện độc lập thấp hơn kích thước cơ sở dữ liệu được sử dụng trong đánh giá của chúng tôi.

Do đó, khi phân tích tổng thể hóa ra rằng Cassandra cho thấy kết quả tốt nhất đối với hầu hết tất cả kịch bản

5.4. “Giải quyết các thách thức dữ liệu lớn cho hiệu suất ứng dụng doanh nghiệp quản lý”

Bài viết này cung cấp một đánh giá hiệu suất toàn diện của sáu hiện đại (mởnguồn) lưu trữ dữ liệu: Apache Cassandra, Apache HBase, Project Voldemort, Redis, VoltDB và MySQL. Họ đã đánh giá các hệ thống này với dữ liệu và khối lượng công việc có thể được tìm thấy trong giám sát hiệu suất ứng dụng, cũng như quảng cáo trực tuyến, giám sát năng lượng và nhiều trường hợp sử dụng khác.

Ban đầu, bài viết cung cấp nền tảng về các hệ thống doanh nghiệp quy mô lớn và APM. Các hệ thống quy mô lớn không đồng nhất và có nhiều phụ thuộc lẫn nhau khiến quản lý của họ một nhiệm vụ rất phức tạp.

Các công cụ Quản lý hiệu suất ứng dụng (APM), cung cấp chế độ xem phức tạp hơn trên hệ thống được giám sát, mục đích của chúng là cung cấp cho quản trị viên một cái nhìn trực tuyến về sức khỏe hệ thống. Những công cụ này giúp các ứng dụng truy xuất thông tin về thời gian phản hồi, cũng như về tỷ lệ lỗi, sử dụng tài nguyên, v.v.

Sau đó, bài báo chỉ ra rằng APM có các yêu cầu tương tự với nền tảng Web hiện tại hệ thống thông tin như yêu cầu nhất quán yếu hơn, địa lý phân phối và xử lý không đồng bộ. Do sự giống nhau này của bộ lưu trữ APM yêu cầu đối với các yêu cầu của các ứng dụng hệ thống thông tin Web, hiển nhiên 67 ứng cử viên cho các hệ thống lưu trữ APM mới là các kho lưu trữ khóa-giá trị và các công cụ phái sinh của chúng (NoSQL và NewSQL).

Trong chương tiếp theo, các tác giả cung cấp một mô tả kỹ lưỡng hơn về trường hợp sử dụng và thách thức dữ liệu lớn như sau: trong một hệ thống phân tán cao, rất khó để xác định nguyên nhân gốc rễ của sự suy giảm hiệu suất, đặc biệt là vì nó thường không gắn liền với một thành phần duy nhất, nhưng với một sự tương tác cụ thể của các thành phần. Hệ thống các thành phần không đồng nhất và không có cơ sở mã thống nhất và thường truy cập vào toàn bộ mã nguồn là không thể. Vì vậy, một phân tích chuyên sâu về các thành phần hoặc việc tích hợp một cơ sở hạ tầng hồ sơ là không thể. Để vượt qua thử thách này, hệ thống quản lý hiệu suất ứng dụng (APM) đã được phát triển.

Nói chung, dữ liệu APM nói chung tương đối đơn giản. Nó thường bao gồm một số liệu tên, giá trị và dấu thời gian. Liên quan đến lưu trữ, các truy vấn có thể được phân biệt thành hai loại chính:

- Tra cứu giá trị đơn lẻ để truy xuất giá trị mới nhất
- Các bản quét nhỏ để truy xuất thông tin tình trạng hệ thống và để tính toán tổng hợp theo cửa sổ thời gian.

Dựa trên các thuộc tính này, họ xác định điểm chuẩn. Tương tự với các bài báo khác đã khảo sát ở trên trong chương, hệ thống đo điểm chuẩn được sử dụng trong thử nghiệm là YCSB điểm chuẩn. Đối với thí nghiệm, họ đã chọn hai trong số các lớp sau theo với phân loại được trình bày bởi Cartell trong:

- Cửa hàng khóa-giá trị: Dự án Voldemort và Redis
- Cửa hàng bản ghi mở rộng: HBase và Cassandra
- Các cửa hàng quan hệ có thể mở rộng: MySQL Cluster và VoltDB

Tóm tắt tất cả các khối lượng công việc khác nhau được trình bày trong bài báo.

Workload Type	Read	Write	Scan
Workload R	95%	5%	0%
Workload RW	50%	50%	0%
Workload W	1%	99%	0%
Workload RS	47.5%	5%	47.5%
Workload RSW	25%	50%	25%

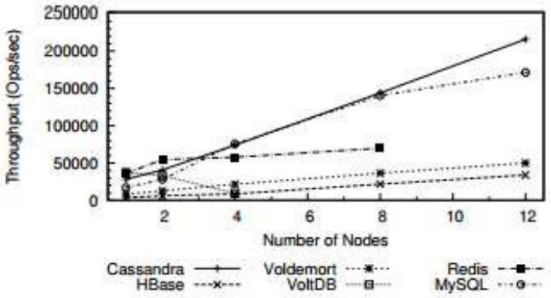


Figure 6: Throughput for Workload RW

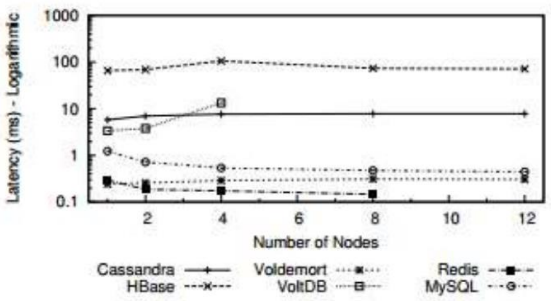


Figure 7: Read latency for Workload RW

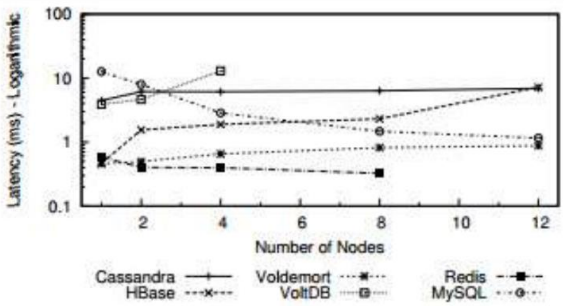


Figure 11: Write latency for Workload W

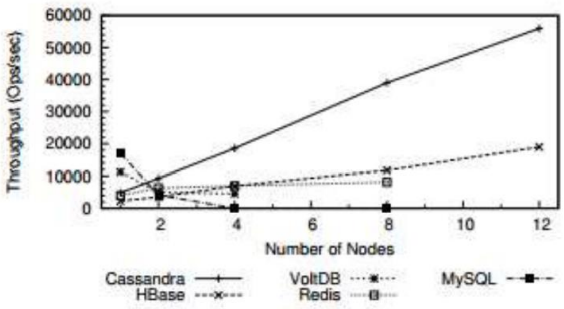


Figure 12: Throughput for Workload RS

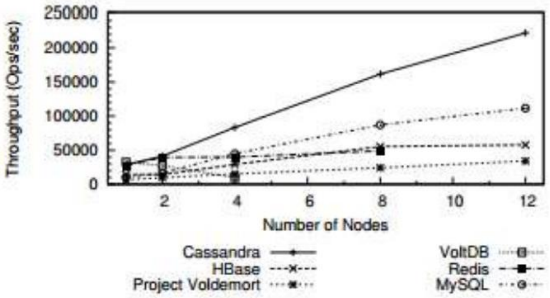


Figure 9: Throughput for Workload W

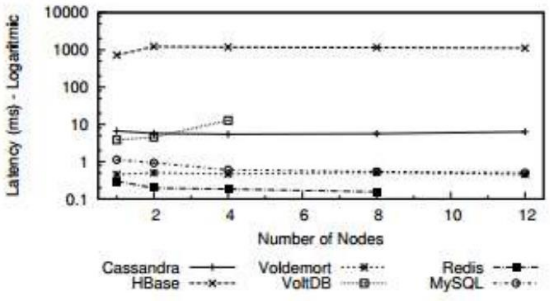


Figure 10: Read latency for Workload W

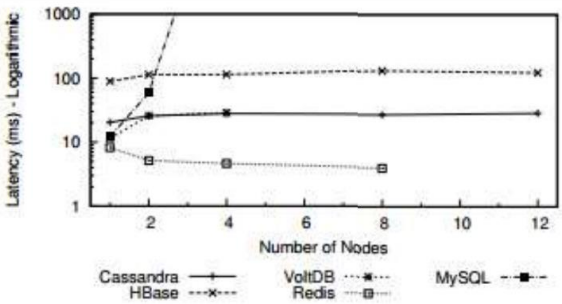


Figure 13: Scan latency for Workload RS

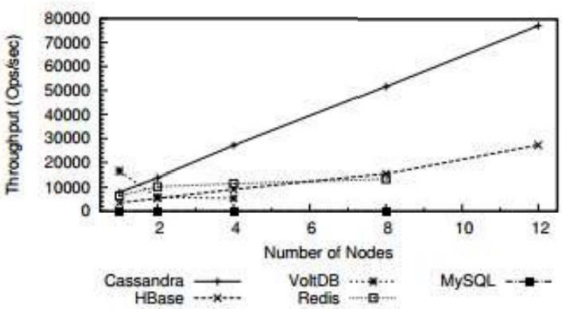


Figure 14: Throughput for Workload RSW

Cuối cùng, bài viết báo cáo chi tiết về kinh nghiệm của họ với các hệ thống này từ một quan điểm của ngành. Họ chỉ ra rằng việc thiết lập Cassandra tương đối dễ dàng, vì có hướng dẫn bắt đầu nhanh có sẵn tại trang web chính thức. Một quan sát khác là khả năng mở rộng tuyến tính cho Cassandra, HBase và Project Voldemort trong hầu hết các các bài kiểm tra. Thông lượng của Cassandra chiếm ưu thế trong tất cả các bài kiểm tra, tuy nhiên, độ trễ của nó ở mức tất cả các bài kiểm tra đặc biệt cao.

Phần lớn cấu hình của Dự án Voldemort rất dễ dàng. Tuy nhiên, trong trái ngược với các hệ thống khác, chúng tôi phải tạo một tệp cấu hình riêng cho từng hệ thống nút. Về hiệu suất - Project Voldemort có độ trễ ổn định thấp hơn nhiều so với độ trễ của Cassandra.

Cấu hình VoltDB hầu hết được lấy cảm hứng từ tài liệu cộng đồng VoltDB đã đề xuất triển khai ứng dụng khách và cấu hình của cửa hàng. Các nhà phát triển VoltDB đã đánh giá tốc độ của VoltDB so với chính Cassandra bằng một cấu hình tương tự, nhưng chỉ có tối đa 3 nút – Tôi đã mô tả thử nghiệm đó trong báo cáo chuyên đề (Phụ lục A). Hiệu suất khôn ngoan quan sát như sau: VoltDB thể hiện thông lượng cao cho một nút, tuy nhiên thiết lập nhiều nút không quy mô tốt.

Điều đáng nói là các kết quả được trình bày trong bài báo cũng có giá trị đối với các vấn đề liên quan. các trường hợp sử dụng, chẳng hạn như tiếp thị quảng cáo trực tuyến, lưu trữ luồng nhấp chuột và cung cấp năng lượng giám sát. Không giống như công việc trước đây, bài báo này tập trung vào thông lượng tối đa mà có thể đạt được bởi các hệ thống. Một trong những kế hoạch tương lai được đề cập trong bài báo là xác định tác động của sao chép và nén đối với thông lượng trong việc sử dụng APM trường hợp.