Hệ thống cơ sở dữ liệu NoSQL: khảo sát và hướng dẫn quyết định

Tóm tắt sơ lược:

Ngày nay, nhiều dữ liệu được tạo ra và tiêu thụ ở quy mô chưa từng có. Điều này đã dẫn đến việc tìm ra các phương pháp mới cho khả năng mở rộng quản lý dữ liệu được gộp lại dưới dạng thuật ngữ hệ thống CSDL NoSQL để xử lí khối lượng dữ liệu ngày càng tăng và tải yêu cầu. Tuy nhiên, ở thời điểm hiện tại thì tính đa dạng và tính đồng nhất của hệ thống đang có cản trở việc cung cấp đầy đủ thông tin lựa chọn kho lưu trữ dữ liệu phù hợp cho 1 ứng dụng nào đó nhất định hoặc được dùng để đăng kí. Do đó, từ bài viết này ta có thể thấy 1 cái nhìn tổng quan từ trên xuống về trường: thay vì đối chiếu các chi tiết về việc triển khai cụ thể của các đại diện cá nhân, thì chúng tôi đề xuất về 1 mô hình về phân loại so sánh liên quan đến chức năng và phi chức năng yêu cầu đối với kỹ thuật và thuật toán được sử dụng trong cơ sở dữ liệu SQL. Toolbox(hộp công cụ) NoSQL này cho phép chúng tôi rút ra 1 loạt quyết định đơn giản để giúp các học viên và nhà nghiên cứu lọc các ứng cử viên hệ thống tiềm năng dựa trên ứng dụng trung tâm theo yêu cầu.

_ Các từ khoá trong NoSQL bao gồm:

+ Data Management: Quản lý dữ liệu
+ Scalability: Khả năng mở rộng
+ Data Models: Dữ liệu mô hình
+ Replication: Nhân rộng
+ Sharding: Phân mảnh

1. Giới thiệu:

Các hệ quản trị CSDL truyền thống (RDBMSs) cung cấp được nhiều cơ chế mạnh mẽ để lưu trữ và truy vấn dữ liệu có cấu trúc dưới sự thống nhất và giao dịch mạnh mẽ, đảm bảo và 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 trở lại đây thì lượng dữ liệu hữu ích trong 1 số lĩnh vực ứng dụng đã trở nên rộng lớn đến mức không thể được lưu trữ hoặc xử lý bằng các giải pháp csdl truyền thống. Nọi dung được dành cho người dùng tạo trong mạng xã hội hoặc dữ liệu

được truy xuất từ các mạng cảm biến lớn chỉ là 2 ví dụ căn bản về điều này, hiện tượng này thường được gọi là dữ liệu lớn. Một lớp hệ thống lưu trữ dữ liệu mới có thể đối phó với Big Data được gộp lại dưới dạng thuật ngữ csdl NoSQL, nhiều trong số đó cung cấp khả năng mở rộng về chiều ngang và tính sẵn sang cao hơn csdl quan hệ bằng cách hi sinh khả năng truy vấn và đảm bảo tính thống nhất. Những sự đánh đổi này là then chốt cho tính toán hướng dịch vụ hoặc là các mô hình dịch vụ, vì bất kì dịch vụ trạng thái nào chỉ có thể có khả năng mở rộng và chịu lỗi như kho lưu trữ dữ liệu cơ bản của nó. Có rất nhiều hệ thống sử dụng csdl NoSQL và rất khó để theo dõi xem nó xuất sắc ở đâu, thất bại ở đâu hoặc thậm chí nơi ở chúng khác nhau, vì chi tiết triển khai thay đổi nhanh chóng và bộ tính năng phát triển theo thời gian. Do đó, trong bài viết này chúng tôi nhằm mục đích cung cấp 1 cái nhìn tổng quan về bối cảnh NoSQL bằng cách thảo luận về các khai niệm được sử dụng hơn là các đặc điểm của hệ thống và khám phá được các yêu cầu thường đặt ra cho các hệ thống csdl NoSQL, các kỹ thuật được sử dụng để đáp ứng các yêu cầu này và sự đánh đổi trong quá trình thực hiện. Bên cạnh đó, chúng tôi tập trung vào các từ khoá-giá trị, tài liệu, cột rộng vì các danh mục NoQuery này bao gồm các quyết định thiết ké và kỹ thuật phù hợp nhất trong không gian dữ liệu có thể mở rộng ban quản lý.

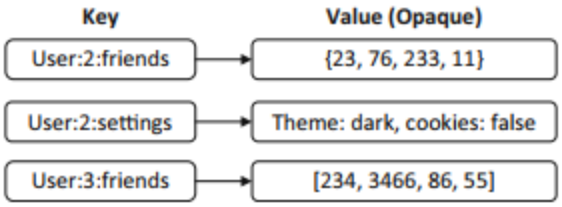
In sect. 2

Chúng tôi mô tả level cao hơn phở biến nhất phương pháp tiếp cận để phân loại các hệ thống csdl NoSQL hoặc bằng mô hình dữ liệu của họ và cho vào kho khoá-giá trị, tài liệu cửa hang và cửa hàng cột rộng hoặc bằng sự cân bằng giữa an toàn và độ sống trong thiết kế của họ(CAP và PACELC). Sau đó, chúng tôi khảo sát kỹ thuật thường được sử dụng chi tiết hơn và thảo luận mô hình của chúng tôi về các yêu cầu và kỹ thuật có liên quan như thế nào

In Sect. 3

Trước khi chúng tôi cung cấp tổng quan về các hệ thống csdl nổi bật bằng cách áp dụng mô hình của chúng tôi cho chúng. 1 mô hình quyết định đơn giuản và trừu tượng để hạn chế sự lựa chọn của các hệ thống NoSQL thích hợp dựa trên các yêu cầu của ứng dụng kết thúc bài báo.

Fig 1. Cửa hang khoá-giá trị cung cấp khả năng lưu trữ và truy xuất tuỳ ý hiệu quả giá trị



2. Phân loại hệ thống cấp cao:

Để trừu tượng hoá từ các chi tiết thực hiện của cá nhân hệ thống NoSQL, tiêu chí phân loại cấp cao có thể được sử dụng có thể được sử dụng để nhóm các cửa hàng dữ liệu tương tự thành các danh mục. Trong phần này, chúng tôi tìm hiểu về 2 cách sử dụng nổi bật nhất: Mô hình dữ liệu và các lớp định lý(CAP)

2.1 Các mô hình dữ liệu khác nhau:

Sự khác biệt được sử dụng phổ biến nhất giữa NoSQL csdl là cách chúng lưu trữ và cho phép truy cập dữ liệu. Mỗi hệ thống được đề cập cập trong 1 bài báo này có thể được phân loại thành kho lưu trữ giá trị chính, kho lưu trữ tài liệu hoặc kho lưu trữ nhiều cột.

2.1.1 Cửa hang khoá-giá trị:

Kho lưu trữ khoá-giá trị bao gồm 1 tập hợp các cặp khoá giá trị và phím độc đáo. Do cấu trúc đơn giản này nó chỉ hỗ trợ nhận và đưa các hoạt động. vì bản chất được lưu trữ là trong suốt với csdl, các kho lưu trữ khoá-giá trị thuần tuý không hỡ trợ các hoạt động ngoài CRUD đơn giản (tạo, đọc, cập nhật, xoá bỏ). Do đó các cửa hang khoá giá trị thường được gọi là không có sơ đồ: bất kỳ giả định nào về cấu trúc của dữ liệu được lưu trữ, được mã hoá hoàn toàn trong logic ứng dụng(lược đồ đọc 31) và không được xác định rõ rang thông qua 1 ngôn ngữ định nghĩa dữ liệu (Schema-on write).

Ưu điểm rõ rang ở mô hình này là nằm ở tính đơn giản của nó. Sự trừu tượng hoá rất đơn giản giúp dễ dàng phân vùng và truy vấn dữ liệu, để hệ thống csdl có thể đạt được độ trễ thấp cũng như thông lượng cao. Tuy nhiên, nếu 1 ứng dụng yêu cầu các hoạt động phức tạp hơn ví dụ: truy vấn phạm vi, mô hình dữ liệu này không đủ mạnh. Hình1, minh hoạ cách dữ liệu và cài đặt tài khoản người dùng có thể được lưu trữ trong kho giá trị khoá. Vì các truy vấn phực tạp hơn so với tra cứu đơn giản không được hỗ trợ, dữ liệu phải được phân tích không hiệu quả trong mã ứng dụng để trích xuất thông tin như dữ liệu cookie có được hỗ trợ hay không(cookie: sai).

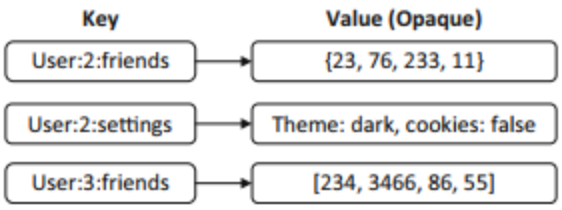
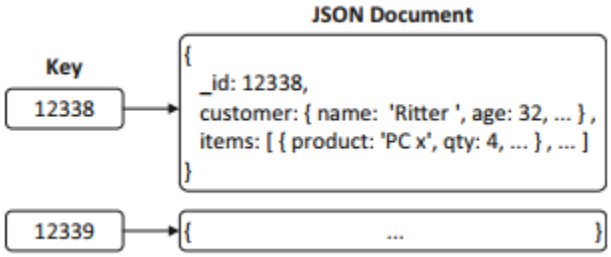


Fig. 1 Key-value stores offer efficient storage and retrieval of arbitrary values

2.1.2 Stories tài liệu:

Kho lưu trữ tài liệu là kho lưu trữ khoá-giá trị hạn chế các giá trị sang các định dạng bán cấu trúc như tài liệu JSON1. Đây là hạn chế so với cửa hang khoá-giá trị mang lại lợi ích lớn trong việc linh hoạt truy cập dữ liệu. Nó không chỉ có thể tìm nạp toàn bộ tài liệu theo ID của nó, nhưng cũng chỉ để lấy các phần của tài liệu ví dụ: tuổi của 1 khách hang, và để truy vấn như tổng hợp, truy vấn theo ví dụ hocw thậm chí toàn văn tìm kiếm. (hình 2)



Hình 2: Các cửa hàng tìa liệu nhận được cấu trúc bên trong của tài liệu được lưu trữ thực thể và do đó có thể hỗ trợ các truy vấn

2.1.3 Cửa hàng cột rộng:

Các cửa hàng có cột rộng kế thừa tên của chúng từ hình ảnh thường được sử dụng để giải thích mô hình dữ liệu cơ bản: 1 bảng quan hệ với nhiều cột thưa thớt. Tuy nhiên, về mặt kỹ thuật, cửa hàng nhiều cột gần với cửa hàng đa cấp phân tán hơn bản đồ được sắp xếp: các khoá cấp đầu tiên được xác định bởi các hàng mà bản thân chúng bao gồm các cặp khoá_giá trị. Các khoá cấp 1 được gọi là khoá hàng, Khoá cấp 2 được gọi là cột phím. Lược đồ lưu trữ này làm cho các bảng có nhiều tuỳ ý cột khả thi, bởi vì không có khoá cột nào mà không có giá trị tương ứng. Chính vì thế, 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 vào cái gọi là họ cột để được định vị các cột trên đĩa thường được truy cập cùng nhau. Trên đĩa, cột

rộng các cửa hàng không sắp xếp tất cả dữ liệu từ mỗi hàng mà thay vào đó các giá trị của cùng 1 họ cột và từ cùng 1 hàng. Do đó, 1 thực thể(hoặc 1 hàng) không thể được truy xuất bởi 1 tra cứu như trong kho tài liệu nhưng phải nối chúng lại với nhau.

3. Techniques: (Kĩ thuật)

Mỗi cơ sở dữ liệu thành công được thiết kế cho một lớp ứng dụng, hoặc để đạt được sự kết hợp cụ thể của các thuộc tính hệ thống mong muốn. Lý do đơn giản là vì có nhiều dữ liệu khác nhau là nó không thể cho bất kì hệ thống nào đạt được tất cả các thuộc tính mong muốn cùng một lúc. Cơ sở dữ liệu SQL truyền thống như là PostgreSQL đã được xây dựng để cung cấp các gói chức năng đầy đủ: mô hình dữ liệu rất linh hoạt, khả năng truy vấn phức tạp bao gồm tham gia, ràng buộc toàn vẹn toàn cầu và giao dịch bảo đảm. Ở đầu kia của phổ quang thiết kế, các cửa hàng có giá trị cốt lõi như Dynamo cho rằng với thang dữ liệu và dung lượng yêu cầu và đồng thời cung cấp lượng đọc và ghi cao cũng như độ trễ thấp, nhưng hầu như không có bất kì chức năng nào từ việc tra cứu đơn giản.

Trong phần này, chúng ta nhấn mạnh không gian thiết kế của phân tán hệ thống cơ sở dữ liệu, tập trung vào việc nhân rộng, quản lý lưu trữ và xử lý truy vấn. Chúng ta khảo sát các kĩ thuật có sẵn và thảo luận một cách mà chúng có liên quan đến các thuộc tính chức năng và phi chức năng khác nhau của các hệ thống quản lí dữ liệu. Để minh họa những kĩ thuật nào phù hợp để đạt được những hệ thống nào, chúng ta cung cấp hộp công cụ NoSQL (Hình 4) ở đây mỗi kĩ thuật được kết nối với các thuộc tính chức năng và phi chức năng nó cho phép.

3.1.Sharding: (Phân vùng)

Một số hệ thống cơ sở dữ liệu quan hệ phân tán như Oracle RAC hoặc IBM DB2 dựa trên đĩa chia sẻ kiến trúc nơi mà 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). Như vậy, những hệ thống đó cung cấp dữ liệu phù hợp, nhưng cũng vốn dĩ khó mở rộng quy mô. Ngược lại, các hệ thống cơ sở dữ liệu (NoSQL) trọng tâm trong bài viết này được xây dựng dựa trên kiến trúc không chia sẽ, có nghĩa là mỗi hệ thống bao gồm nhiều máy chủ với bộ nhớ riêng và đĩa riêng được

kết nối thông qua một mạng lưới. Do đó, khả năng mở rộng cao về thông lượng và khối dữ liệu đạt được bằng cách phân vùng dữ liệu trên các nút khác nhau trong hệ thống.

Có ba kĩ thuật phân phối cơ bản: phân vùng phạm vi, phân vùng băm và phân vùng nhóm thực thể. Để làm cho quét hiệu quả, dữ liệu có thể được phân vùng thành phạm vi giá trị được sắp xếp theo thứ tự và liền kề bằng cách chia nhỏ phạm vi. Tuy nhiên, cách tiếp cận này đòi hỏi một số phối hợp thông qua một bậc thầy quản lí các bài tập. Để đảm bảo sự co giãn, 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 phân vùng được hỗ trợ bởi các cửa hàng cột rộng như BigTable, Hbase hoặc Hypertable và kho lưu trữ tài liệu, ví dụ : MongoDB, RethinkDB, Espresso và DocumentDB. Một cách khác để phân vùng dữ liệu trên nhiều máy là phân vùng băm trong đó mọi dữ liệu được gán cho một phân vùng máy chủ theo một số giá trị băm được xây dựng 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 làm cho quá trình quét không khả thi. Phân vùng băm được sử dụng trong của hàng khóa-giá trị và cũng có sẵn trong một số cột rộng ở các cửa hàng như Cassandra hoặc Azure.

Máy chủ cứng chịu trách nhiệm về bản ghi có thể xác định asserveried = hash (id) mod servers. Tuy nhiên, sơ 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 số lượng máy chủ phân vùng. Do đó, không thể sử dụng trong các hệ thống đàn hooig như Dynamo, Riak hoặc Cassandra, cho phép thêm tài nguyên được thêm vào theo yêu cầu và một lần nữa được gỡ bỏ khi cần thiết. Để tăng tính linh hoạt, các hệ thống đàn hồi thường sử dụng hàm băm nhất quán khi các bản ghi không được gán trực tiếp đến các máy chủ, mà thay vào đó là các phân vùng logic mà sau đó được phân bố trên tất cả máy chủ phân vùng. Như vậy, chỉ một phần nhỏ của dữ liệu phải được chỉ định lại khi có thay đổi trong hệ thống cấu trúc liên kết. Ví dụ, một hệ thống đàn hồi có thể được thu nhỏ bằng cách giảm tải tất cả các phân vùng hợp lí cư trú trên một máy chủ cụ thể này sang các máy chủ khác và sau đó tắt máy hiện không hoạt động cỗ máy. Để biết chi tiết về cách sử dụng hàm băm nhất quán trong các hệ thống

NoSQL. Nhóm phân vùng thực thể là sơ đồ phân vùng dữ liệu với mục tiêu cho phép các giao dịch một phân vùng trên cùng một vị trí dữ liệu. Các phân vùng được gọi là nhóm thực thể và được ứng dụng khái bán rõ ràng (ví dụ: trong G-Store và MegaStore hoặc bắt nguồn từ các mẫu truy cập của giao dịch (ví dụ: trong Relational Cloud và Cloud SQL Server). Nếu một giao dịch truy cập dữ liệu kéo dài nhiều hơn một nhóm, quyeenfn sở hữu dữ liệu có thể được chuyển giao giữa các nhóm thực thể hoặc người quản lí giao dịch phải dự phòng đắt hơn giao thức giao dịch đa nút.

3.2.Replication: (Sự tái tạo)

Về mặt CAP, các RDBMS thông thường là các hệ thống CA chạy ở chế độ một máy chủ: Toàn bộ hệ thống trở thành không khả dụng khi máy bị lỗi. Và vì vậy những người vận hành hệ thống đảm bảo tính toàn vẹn và tính sẵn sàng của dữ liệu thông qua chi phí đắt đỏ, nhưng phần cứng cao cấp đáng tin cậy. Ngược lại, các hệ thống NoSQL như Dynamo, BigTable hoặc Cassandra được thiết kế cho dữ liệu và khối lượng yêu cầu không thể được xử lí bởi một máy duy nhất và do đó chúng chạy trên các cụm bao gồm hàng ngàn máy chủ. Bởi thất bại là điều không thể tránh khỏi và sẽ xảy ra thường xuyên trong bất kì hệ thống phân tán quy mô lớn nào, phần mềm phải đối phó với chúng hàng ngày. Năm 2019, Jeff Dean, đồng nghiệp của Google đã tuyên bố rằng một điển hình cụm mới tại Google gặp phải hàng nghìn ổ cứng lỗi, 1000 lỗi máy đơn, 20 lỗi giá đỡ và một số phân vùng mạng do dự kiến và không mong muốn hoàn cảnh năm đầu tiên của nó một mình. Nhiều trường hợp gần đây phân vùng mạng và sự cố ngừng hoạt động trong các trung tâm dữ liệu đám mây lớn đã được báo cáo. Bản sao cho phép hệ thống để duy trì sự sẵn có và độ bền khi đối mặt với lỗi như vậy. Nhưng lưu trữ cùng một bản ghi trên các máy khác nhau (máy chủ bản sao) trong cụm giới thiệu vấn đề về đồng bộ hóa giữa chúng và do đó là sựu đánh đổi giữa tính nhất quán trên một mặt và độ trễ và tính khả dụng trên cái khác. Đè xuất phân loại hai lớp các chiến lược sao chép khác nhau tùy theo thời điểm cập nhật được truyền đến các bản sao và nơi các bản cập nhật được chấp nhận. Có hai lựa chọm khả thi ở cấp 1: háo hức sao chép (đồng bộ) lan truyền các thay đổi sắp tới một cách đồng bộ tới tất cả các bản sao trước khi cam kết có thể được trả về máy khách, trong khi sao chép lười (không đồng bộ) áp dụng chỉ thay đổi tại baen sao nhận và chuyển chúng vào không

đồng bộ. ưu điểm lớn của sao chép háo hức là tính nhất quán guiuwac các bản sao, nhưng nó phải trả giá bằng độ trễ ghi cao hơn do phải đợi các bản sao khác và tính khả dụng bị suy giảm. Sao chép lười biếng nhanh hơn, bởi vì nó cho phép các bản sao phân kì kết quả là, dữ liệu cũ có thể được phục vụ. Ở tầng thứ hai, một lần nữa có thể có hai cách tiếp cận khác nhau: hoặc là chủ-tớ lược dồ (bản sao chính) được theo đuổi khi những thay đổi chỉ có thể được chấp nhận bởi một bản sao (chính) hoặc trong một bản cập nhật phương pháp tiếp cận mọi nơi (đa chủ) mọi bản sao đều có thể chấp nhận viết. Trong các giao thức chủ tớ điều khiển đồng thời không phức tạp hơn trong một hệ thoongd phân tán không có bản sao, nhưng toàn bộ bản sao sẽ không khả dụng ngay khi chủ thất bại. Các giao thức đa chủ yêu cầu phức tạp cơ chế phòng ngừa hoặc phát hiện và hòa giải của những thay đổi trái ngược nhau. Các kĩ thuật thường được sử dụng cho các mục đích là lập phiên bản, đồng hồ vector, bàn luận và đọc sửa chữa và hội tụ hoặc giao hoán kiểu dữ liệu.

Về cơ bản, tất cả bốn kết hợp của phân loại hai tầng đều có thể thực hiện được. Các hệ thống quan hệ phân tán thường thực hiện sao chép chủ-tớ háo hức để duy trì tính nhất quán mạnh mẽ. Háo hức cập nhật bất cứ nơi nào sao chép như ví dụ đặc trưng trong Google's Megastore phải chịu chi phí liên lạc lớn do đồng bộ hóa tạo ra và có thể gây ra bế tắc phân tán rất tốn kém để phát hiện. Các hệ thống cơ sở dữ liệu NoSQL thường dựa vào sao chép lười biếng, kết hợp với chủ-tớ (các hệ thống CP, ví dụ: HBase và MongoDB) hoặc phương pháp cập nhật mọi nơi (các hệ thống AP, ví dụ: Dynamo và Cassandra). Nhiều hệ thống NoSQL để máy khách lựa chọn giữa độ trễ và tính nhất quán, tức 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 nhất quán chắc chắn (bởi phần lớn các bản sao hoặc chính) để ngăn dữ liệu cũ.

Một khía cạnh của bản sao không được đề cập trong sơ đồ hai lớp 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. Nhưng ngoài khả năng không có sẵn tạm thời, việc đặt các bản sao gần đó còn có nguy cơ mất tất cả các bản sao cùng một lúc

trong một kịch bản thảm họa. Một kỹ thuật thay thế để giảm độ trễ được sử dụng trong Orestes, trong đó dữ liệu được lưu vào bộ nhớ đệm 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. Háo hức sao chép địa lý, như được triển khai trong Google's Megastore, Spanner, MDCC và Mencius đạt được sự nhất quán mạnh mẽ với chi phí là độ trễ ghi cao hơn (thường là 100 mili giây đến 600 mili giây). Với sao chép địa lý lười biếng như trong Dynamo, PNUTS, Walter, COPS, Cassandra và BigTable, những thay đổi gần đây có thể bị mất, nhưng hệ thống hoạt động tốt hơn và vẫn khả dụng trong quá trình phân vùng. Charron-Bost et al và Öszu và Valduriez cung cấp một cuộc thảo luận toàn diện về sao chép cơ sở dữ liệu.

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 các 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). 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 rất khác nhau của bộ lưu trữ RAM, SSD và HDD cũng như các chiến lược khác nhau để tận dụng điểm mạnh của chúng (quản lý lưu trữ) là một lý do cho sự đa dạng của cơ sở dữ liệu NoSQL. Quản lý lưu trữ có khía cạnh không gian (nơi lưu trữ dữ liệu) và khía cạnh 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 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 "sự kết thúc của một kỷ nguyên kiến trúc", Stonebraker 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 các giao dịch

– khai thác gỗ (11,9%), để đảm bảo độ bền khi đối mặt với sự cố

– tối ưu hóa được mã hóa thủ công (16,2 %)

Điều này thúc đẩy rằng có thể mong đợi những cải tiến hiệu suất lớn nếu RAM được sử dụng làm bộ nhớ chính (cơ sở dữ liệu trong bộ nhớ). 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 sao chép trên n nút máy chủ trong bộ nhớ để bảo vệ chống lại n - 1 lỗi nút đơn (ví dụ: HStore, VoltDB) hoặc bằng cách đăng nhập vào bộ lưu trữ lâu 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 chuyển sang bộ lưu trữ liên tục. Để tránh độ trễ quay của ổ cứng phát sinh bằng cách ghi nhật ký từng hoạt động riêng lẻ, các lần xóa 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 flash 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 – toàn bộ khối phải được xóa trước khi ghi đè bất kỳ trang trong khối đó và (3) chu

kỳ chương trình/xóa hạn chế". Do đó, quản lý lưu trữ của hệ thống cơ sở dữ liệu không được coi SSD và HDD là RAM liên tục, chậm hơn một chút, do ghi ngẫu nhiên vào ổ SSD chậm hơn khoảng 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 đến 100 lần so với truy cập tuần tự. Do đó, việc ghi nhật ký phù hợp với điểm mạnh của SSD và HDD, cả hai đều cung cấp thông lượng cao hơn đáng kể cho ghi tuần tự.

Đối với cơ sở dữ liệu trong bộ nhớ, mẫu truy cập cập nhật tại chỗ là lý tưởng: Nó đơn giản hóa việc triển khai và ghi ngẫu nhiên vào RAM về cơ bản nhanh như ghi tuần tự, với những khác biệt nhỏ bị ẩn bởi đường ống dẫn 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 tốc độ 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 phù hợp với 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 lợi 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 hóa kiến trúc cơ sở dữ liệu, 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 nào chúng bị loại bỏ. Ngoài ra, tính năng đọc trước (đọc suy đoán) và ghi sau (đệm ghi) được thực hiện minh bạch với bộ đệm hệ điều hành thiếu sự phức tạp vì chúng dựa trên logic hệ thống tệp thay vì truy vấn cơ sở dữ liệu.

Lưu trữ chỉ nối thêm (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 tệp có cấu trúc nhật ký đã có một lịch sử nghiên cứu lâu dài, nhưng I/O chỉ nối thêm mới chỉ được phổ biến gần đây cho cơ sở dữ liệu bằng

việc BigTable sử dụng các cây Hợp nhất có cấu trúc nhật ký (LSM) bao gồm một bộ đệm trong bộ nhớ, một nhật ký liên tục. và các tệp 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, Bitcask, RethinkDB, WiredTiger, RocksDB, InfluxDB, TokuDB). 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 cấu trúc chỉ mục phù hợp được cập nhật vĩnh viễn dưới dạng cấu trúc dữ liệu sao chép khi ghi (COW) (ví dụ: cây B-COW 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.

Trong các môi trường ảo hóa như các đám mây cơ sở hạ tầng dưới dạng dịch vụ, nhiều đặc điểm được thảo luận của lớp lưu trữ bên dưới bị ẩn đi.

3.3.Query processing: (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. Truy vấn bộ lọc 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 dưới dạng quét toàn bộ bảng được lọc. Đối với cơ sở dữ liệu được phân vùng băm, điều này ngụ ý một mẫu thu thập phân tán trong đó mỗi phân vùng thực hiện quá trình 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 tránh sự thiếu hiệu quả của các lần quét O(n), chỉ có các mục phụ có thể được sử dụng. Đây có thể là các chỉ mục phụ cục bộ được quản lý trong mỗi phân vùng hoặc các

chỉ mục phụ toàn cầu lập chỉ mục dữ liệu trên tất cả các phân vùng. Vì bản thân chỉ mục toàn cầu phải được phân 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 theo 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 về tên cần được đánh giá ở 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 vấn là nhiệm vụ tối ưu hóa kế hoạch truy vấn để giảm thiểu chi phí thực hiện. Đối với các tập hợp và liên kết, lập kế hoạch truy vấn là 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 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 về xử lý truy vấn phân tán tập trung vào khối lượng công việc OLAP (xử lý phân tích trực tuyến) 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, vẫn còn là một thách thức nghiên cứu mở để khái quát hóa phần lớn các kỹ thuật tối ưu hóa truy vấn có thể áp dụng, đặc biệt là trong bối cảnh cơ sở dữ liệu tài liệu.

Phân tích trong cơ sở dữ liệu có thể được thực hiện nguyên bản (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 gốc phổ biến được các hệ thống NoSQL đưa ra là MapReduce. 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 lô và vi mô này có thời gian phản hồi cao. Chế độ xem cụ thể hóa là 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 trên các hoạt động thay đổi (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, các đường dẫn phân tích gần thời gian thực thường triển khai Kiến trúc Lambda hoặc Kiến trúc Kappa, kiến trúc trước đây bổ sung cho khung xử lý hàng loạt như Hadoop MapReduce với một bộ xử lý luồng chẳng hạn như Storm và bộ xử lý sau hoàn toàn dựa vào xử lý luồng và hoàn toàn bỏ qua xử lý hàng loạt.

4. Nghiên cứu hệ thống:

Trong phần này, chúng tôi cung cấp một so sánh định tính của một số kho lưu trữ khoá-giá trị, tài liệu và cột nổi bật nhất. Trình bày kết quả dưới dạng so sánh và tham khảo tài liệu của các hệ thống riêng để nắm rõ thông tin chi tiết. Hộp công cụ NoQL được đề xuất (Hình 4, trang 5) là một phương tiện có thể được 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 cho rằng sự phân loại này mô tả đặc điểm tốt của nhiều hệ thống cơ sở dữ liệu và do đó có thể sử dụng để đối chiếu một cách chính xác các hệ thống cơ sở dữ liệu khác nhau : Bảng 1 cho thấy sự so sánh trực tiếp MongoDB, Redis,Hbase, Riak, Cassanda và MySQL trong các cấu hình mặc định tương ứng. Một so sánh chi tiết hơn về các thuộc tính của hệ thống trung tâm được thể hiện trong Bảng 2(xem trang 11).

Phương pháp được sử dụng để xác định các thuộc tính hệ thống cụ thể bao gồm phân tích các tài liệu cụ thể là tài liệu có sẵn công khai về các hệ thống. Hơn nữa, một số thuộc tính phải được đánh giá bằng cách nghiên cứu các cơ sở mã nguồn mở, liên hệ với các nhà phát triển cũng như phân tích tổng hợp các báo cáo và điểm chuẩn của các học viên.

Sự so sánh đã làm rõ cách cơ sở dư liệu SQL và NoSQL được thiết kế để đáp ứng các nhu cầu khác nhau : RDBMSS cung cấp mức độ chức năng khác biệt trong khi các cơ sở dữ liệu NoSQL vượt trội về mặt phi chức năng thông qua quy mô khả năng, tính khả dụng, độ trễ thấp và thông lượng cao. Tuy nhiên, cũng có sự khác biệt lớn giữa các cơ sở dữ liệu NoSQL. Ví dụ, Riak và Cassanda có thể được cấu hình để đáp ứng yêu cầu phi chức năng, nhưng cuối cùng không có nhiều khả năng gì ngoài phân tích dữ liệu. Mặt khác, MongoDB và HBase cung cấp tính nhất quán mạnh mẽ hơn và các khả năng chức năng thông minh hơn như truy vấn quét và chỉ truy vấn MongoDB: lọc, nhưng không duy trì tính khả dụng 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 sự thay đổi khác 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 không đồng bộ nhân rộng.

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 quyết định nhị phân trong Hình 6 các quyết định đánh đổi cho các ứng dụng mẫu và các hệ thống cơ sở dữ liệu có khả năng phù hợp. 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 vấn đề lò xo không gian 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 thêm bởi khối lượng dữ liệu mà chúng xử lý: nếu

bộ nhớ chính của một máy duy nhất có thể chứa tất cả dữ liệu, thì một hệ thống nút đơn như Redis hoặc Memcache có lẽ là lựa chọn tốt nhất, tùy thuộc vào chức năng của nó. (Redis) hoặc 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 đem lại trải nghiệm nổi bật, trong khi các hệ thống như HBase, MongoDB và DynamoDB mang lại sự quyết đoán mạnh mẽ. 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, trước 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 (xử lý giao dịch trực tuyến) phổ biến trên khối lượng dữ liệu lớn vừa phải, RDBMSS 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 lượng, khi được kết hợp với Hadoop.

Tóm lại, chúng tôi tin mô hình từ trên xuống được đề xuất là một hỗ trợ ra quyết định hiệu quả để lọc số lượng lớn các hệ thống cơ sở dữ liệu NoSQL dựa trên các yêu cầu trung tâm. Ngoài ra, Hộp công cụ NoSQL còn cung cấp một 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.