Phần cứng (memory – CPU - IO)

**Memory**

Giá RAM giảm khiến các dbs được thiết kế xoay quanh việc sử dụng bộ nhớ trở nên phổ biến. Nó cũng nhanh hơn 25 lần so với SSD. Các hoạt động phụ thuộc nhiều vào bộ nhớ bao gồm:

* Aggregation
* Index Traversing
* Write Operations
* Query Engine (to retrieve query results)
* Connections (~1MB per established connection)

Nói chung, MongoDB càng có nhiều bộ nhớ thì hiệu năng của nó sẽ càng tốt.

**CPU**

Được sử dụng bởi tất cả các ứng dụng, cơ sở dữ liệu chỉ là một trong số đó. Hai yếu tố chính của MongoDB liên quan đến CPU:

* Storage Engine
* Concurrency Model

Nếu có các hoạt động không chặn (ví dụ: ghi tài liệu đồng thời hoặc trả lời yêu cầu truy vấn - đọc), MongoDB hoạt động tốt hơn khi có nhiều tài nguyên CPU hơn.

Các hoạt động yêu cầu tính khả dụng của chu kỳ CPU:

* Page Compression
* Data Calculation
* Aggrgation Framework Operations
* Map Reduce

Không phải tất cả các thao tác đọc và ghi đều là các thao tác không chặn, chẳng hạn như việc ghi liên tục vào cùng một tài liệu, nhiều CPU sẽ không giúp ích cho hiệu suất vì công việc này không thể thực hiện song song vì tất cả chúng đều ảnh hưởng đến cùng một tài liệu. Do đó phải chặn tất cả các thao tác ghi khác trên tài liệu đó

**IOPS**

MongoDB sử dụng đĩa để lưu trữ dữ liệu. IOPS: Hoạt động đầu vào/đầu ra mỗi giây do máy chủ cung cấp. Tốc độ này càng nhanh thì mongo có thể đọc/ghi dữ liệu càng nhanh. Loại đĩa sẽ ảnh hưởng lớn đến hiệu suất MongoDB.

Khía cạnh quan trọng của MongoDB là nhu cầu ghi vào nhiều đĩa khác nhau. Điều này phân phối tải IO của các cơ sở dữ liệu, chỉ mục, tệp ghi nhật ký và khóa khác nhau -> tối ưu hóa hiệu suất.

**Network**

MongoDB là cơ sở dữ liệu phân tán có tính sẵn sàng cao, do đó việc triển khai cũng phụ thuộc vào phần cứng mạng. Băng thông cho mạng nhanh hơn và lớn hơn -> hiệu suất tốt hơn.

* Các ứng dụng tiếp cận cơ sở dữ liệu bằng cách thiết lập kết nối tới máy chủ nơi phiên bản MongoDB đang chạy.
* Đạt được tính sẵn sàng cao nhờ các bộ cụm bản sao.
* Chia tỷ lệ theo chiều ngang đạt được với cụm sharding.

Cách kết nối các máy chủ khác nhau đang giữ các nodes of cluster khác nhau sẽ ảnh hưởng đến hiệu suất. Những cân nhắc khác bao gồm:

* Các loại switch mạng
* Load balancer
* Tường lửa
* Các cluster nodes cách nhau bao xa (trên các trung tâm dữ liệu hoặc khu vực khác nhau)
* Các loại kết nối giữa các trung tâm dữ liệu (tức là độ trễ - không thể nhanh hơn tốc độ ánh sáng)

Ứng dụng, trong khi phát ra lệnh có thể thiết lập:

* Write Concern
* Read Concern
* Read Preference

Hiệu suất trên Cụm (Replica Cluster or Shard Cluster)

Bây giờ, về replica sets của bạn, hãy để tôi củng cố một thông điệp xung quanh vấn đề này. Vui lòng sử dụng chúng trong môi trường sản xuất. Tính sẵn sàng cao là chìa khóa để đảm bảo rằng dịch vụ của bạn không bị ảnh hưởng bởi bất kỳ lỗi hệ thống tiềm ẩn và có thể xảy ra nào. Việc có sẵn một bản sao là cực kỳ quan trọng.

Ngoài mục đích chính là cung cấp tính sẵn sàng cao, trong trường hợp node bị lỗi, chúng tôi vẫn sẽ có sẵn dịch vụ do các node còn lại cung cấp, nhưng các bộ bản sao cũng có thể cung cấp một số chức năng khác, như giảm tải dữ liệu nhất quán ở bụng cho các nút phụ , ưu tiên khối lượng công việc chính của bạn cho khối lượng công việc vận hành hoặc có khối lượng công việc cụ thể với cấu hình chỉ mục mục tiêu trên các nút phụ.

Mặt khác của các hệ thống phân tán trong MongoDB, với mục đích mở rộng theo chiều ngang, là Cụm phân đoạn của chúng tôi.

Hiệu suất trên cụm

Trong Shard Cluster, chúng tôi sẽ có Mongos, chịu trách nhiệm định tuyến các yêu cầu ứng dụng khách của chúng tôi đến các node được chỉ định.

Chúng ta sẽ có các config servers. Các nút này chịu trách nhiệm giữ ánh xạ Shard Cluster của chúng tôi, nơi dữ liệu nằm ở mỗi thời điểm, đồng thời cũng là cấu hình chung của Cụm phân đoạn của riêng nó.

Và cuối cùng, chúng ta có Shard Nodes. Nút Shard chịu trách nhiệm lưu giữ dữ liệu ứng dụng. Databases, collections, indexes sẽ nằm trong các thành viên này và chính tại đây, tất cả workload sẽ được thực hiện.

Bản thân các nút Shard là các bộ replica sets

Hiệu suất trên cụm

Trước Sharding

˗ Sharding là một giải pháp mở rộng quy mô theo chiều ngang;

˗ Chúng ta đã đạt đến giới hạn của việc mở rộng quy mô theo chiều dọc chưa?

˗ Bạn cần hiểu dữ liệu của bạn phát triển như thế nào và dữ liệu của bạn được truy cập như thế nào;

˗ Sharding hoạt động bằng cách xác định phạm vi dựa trên khóa – shardkey của chúng tôi;

˗ Điều quan trọng là phải có được shard key tốt

Hiệu suất trên cụm

˗ Hai kiểu đọc:

• Scatter Gather (thu thập rải rác)

• Routed Queries (truy vấn định tuyến rải rác)

When

• Nếu chúng tôi không sử dụng shard key, chúng tôi sẽ thực hiện các truy vấn Scatter Gather

• Nếu chúng tôi đang sử dụng shard key, chúng tôi sẽ thực hiện Truy vấn Routed Queries

Hiệu suất trên cụm

˗ cân nhắc before sharding

˗ Latency

˗ Scattered gather and routed queries

˗ Sorting, limit & skip

Hiệu suất trên cụm

Tăng hiệu suất ghi với sharding

˗ Chia tỷ lệ theo chiều dọc và chiều ngang;

˗ Phân đoạn các quy tắc chính;

˗ Viết hàng loạt.

Index

- Sử dụng ESR (Equality, Sort, Range)

˗ Tạo chỉ mục để hỗ trợ truy vấn của bạn

˗ Sử dụng chỉ mục để sắp xếp truy vấn

˗ Đảm bảo các chỉ mục phù hợp với RAM

˗ Tạo truy vấn đảm bảo tính chọn lọc

Index

• "equality" đề cập đến sự trùng khớp chính xác trên một giá trị duy nhất.

• Đặt các trường yêu cầu kết quả khớp chính xác đầu tiên trong chỉ mục của bạn.

• Một chỉ mục có thể có nhiều khóa cho các truy vấn có kết quả khớp chính xác. Các khóa chỉ mục cho các kết quả khớp bằng nhau có thể xuất hiện theo bất kỳ thứ tự nào. Tuy nhiên, để đáp ứng sự so khớp bình đẳng về chỉ mục, tất cả các khóa chỉ mục cho kết quả khớp chính xác phải xuất hiện trước bất kỳ trường chỉ mục nào khác.

• Sự trùng khớp chính xác phải được chọn lọc. Để giảm số lượng khóa chỉ mục được quét, hãy đảm bảo kiểm tra equality loại bỏ ít nhất 90% tài liệu trùng khớp có thể có.

Index

• "Sắp xếp" xác định thứ tự kết quả. Sắp xếp tuân theo các kết quả khớp đẳng thức vì các kết quả khớp đẳng thức làm giảm số lượng tài liệu cần được sắp xếp.

• Một chỉ mục có thể hỗ trợ các thao tác sắp xếp khi các trường truy vấn là tập hợp con của các khóa chỉ mục. Các thao tác sắp xếp trên một tập hợp con của các khóa chỉ mục chỉ được hỗ trợ nếu truy vấn bao gồm các điều kiện bằng nhau cho tất cả các khóa tiền tố đứng trước các khóa sắp xếp.

• Ví dụ: truy vấn bộ sưu tập ô tô, kết quả được sắp xếp theo model

db.cars.find( { nhà sản xuất: "GM" } ).sort( { model: 1 } )

• Để cải thiện hiệu suất truy vấn, hãy tạo chỉ mục trên các trường nhà sản xuất và kiểu máy:

db.cars.createIndex( { nhà sản xuất: 1, model: 1 } )

index

• Sắp xếp chặn (blocking sort) cho biết MongoDB phải sử dụng và xử lý tất cả tài liệu đầu vào để sắp xếp trước khi trả về kết quả. Các loại chặn không chặn các hoạt động đồng thời trên bộ sưu tập hoặc cơ sở dữ liệu.

• Nếu MongoDB không thể sử dụng một hoặc nhiều chỉ mục để lấy thứ tự sắp xếp, MongoDB phải thực hiện thao tác sắp xếp chặn trên dữ liệu.

• MongoDB sử dụng các tệp tạm thời trên đĩa để lưu trữ dữ liệu vượt quá giới hạn bộ nhớ hệ thống 100 megabyte trong khi xử lý thao tác sắp xếp chặn.

• Các thao tác sắp xếp sử dụng chỉ mục thường có hiệu suất tốt hơn so với các thao tác sắp xếp theo khối.

Index

• Bộ lọc "Phạm vi"(Range) quét các trường. Quá trình quét không yêu cầu kết quả khớp chính xác, có nghĩa là các bộ lọc phạm vi

được liên kết lỏng lẻo với các khóa chỉ mục. Để cải thiện hiệu quả truy vấn, hãy làm cho giới hạn phạm vi càng chặt chẽ

nhất có thể và sử dụng các kết quả khớp bằng nhau để hạn chế số lượng tài liệu phải quét.

• Bộ lọc phạm vi giống như sau:

db.cars.find( { giá: { $gte: 15000} } )

db.cars.find( { age: { $lt: 10 } } )

db.cars.find( { trướcAccidents: { $ne: null } } )

index

Đối với các chỉ mục phức hợp, quy tắc ngón tay cái này rất hữu ích trong việc quyết định thứ tự các trường trong chỉ mục:

• Trước tiên, hãy thêm các trường mà truy vấn Bình đẳng đang chạy.

• Các trường tiếp theo được lập chỉ mục sẽ phản ánh thứ tự Sắp xếp của truy vấn.

• Các trường cuối cùng thể hiện Phạm vi dữ liệu được truy cập.

˗ Nếu chúng ta đặt khóa đẳng thức lên hàng đầu, chúng ta sẽ giới hạn dữ liệu chúng ta đang tìm kiếm

˗ Tránh chặn/sắp xếp trong bộ nhớ

Index

Cardinality (Trường tham chiếu)

• Cardinality được định nghĩa là số phần tử duy nhất có trong một tập hợp. Cardinality càng thấp thì các phần tử càng trùng lặp.

• Vì vậy, nếu một tập hợp có 5 phần tử được tạo thành từ các giá trị Boolean thì số lượng phần tử của tập hợp đó sẽ là hai. Vì vậy, tất cả các bộ được tạo từ Booleans sẽ có số lượng tối đa là hai và số lượng tối thiểu là một.

Cardinality tác động như thế nào đến việc lập chỉ mục

• Nếu một trường Boolean được lập chỉ mục thì chỉ mục sẽ không được cải thiện nhiều về mặt hiệu suất.

• Chỉ có Boolean với tỷ lệ chia 50/50. Việc lập chỉ mục sẽ cho phép bạn bỏ qua 50% tài liệu nhưng vẫn quét tuần tự 50% còn lại.

• Nếu có sự phân chia 80/20 giữa đúng và sai thì chỉ mục sẽ gần như vô dụng khi truy vấn phần đúng vì bạn sẽ phải quét tuần tự 80% tài liệu (Nhưng các truy vấn tìm kiếm sai sẽ hưởng lợi từ chỉ số).

• Điều này áp dụng cho bất kỳ trường nào có lượng số thấp, nếu một trường có enum gồm năm giá trị và hàng nghìn tài liệu trong mỗi danh mục, Có thể quan sát thấy hiệu ứng tương tự.

• Các chỉ mục phải được xây dựng cẩn thận trong những điều kiện như thế này. Một tác dụng phụ nữa của việc lập chỉ mục trên các trường như vậy là nó cũng ảnh hưởng đến việc ghi.

Partial index

Chỉ mục một phần

• Chỉ mục một phần chỉ lập chỉ mục các tài liệu trong bộ sưu tập đáp ứng biểu thức lọc được chỉ định. Bằng cách lập chỉ mục một tập hợp con các tài liệu trong một bộ sưu tập, các chỉ mục một phần có yêu cầu lưu trữ thấp hơn và giảm chi phí hiệu suất cho việc tạo và duy trì chỉ mục.

• Ví dụ: thao tác sau đây tạo ra một chỉ mục tổng hợp chỉ lập chỉ mục các tài liệu có trường xếp hạng lớn hơn 5.

db.restaurants.createIndex(

{ cuisine: 1, name: 1 },

{ partialFilterExpression: { rating: { $gt: 5 } } }

)

covered query

• Truy vấn được bảo hiểm là truy vấn có thể được đáp ứng hoàn toàn bằng cách sử dụng chỉ mục và không phải kiểm tra bất kỳ tài liệu nào. Một chỉ mục bao gồm một truy vấn khi áp dụng tất cả những điều sau đây:

• tất cả các trường trong truy vấn đều là một phần của chỉ mục và

• tất cả các trường được trả về trong kết quả đều có cùng chỉ mục.

• không có trường nào trong truy vấn bằng null (tức là {"field" : null} hoặc {"field" : {$eq : null}} ).

• Ví dụ: kho lưu trữ bộ sưu tập có chỉ mục sau trên các trường loại và mục:

db.inventory.createIndex( { type: 1, item: 1 } )

• Chỉ mục này sẽ bao gồm thao tác sau đây truy vấn trên các trường loại và mục và chỉ trả về trường mục:

db.inventory.find( { type: "food", item:/^c/ },{ item: 1, \_id: 0 })

• Để chỉ mục được chỉ định bao gồm truy vấn, tài liệu chiếu phải chỉ định rõ ràng \_id: 0 để loại trừ trường \_id khỏi kết quả vì chỉ mục không bao gồm trường \_id.

• Ví dụ: hãy xem xét một bộ sưu tập dữ liệu người dùng với các tài liệu có dạng sau:

{ \_id: 1, user: { login: "tester" } }

• collections có chỉ mục sau:

{ "user.login": 1 }

• Chỉ mục { "user.login": 1 } sẽ bao gồm truy vấn bên dưới:

db.userdata.find( { "user.login": "tester" }, { "user.login": 1, \_id: 0 } )

˗ Tạo chỉ mục ở nền trước sẽ thực hiện khóa cấp độ collection

˗ Tạo chỉ mục ở chế độ nền giúp khắc phục tình trạng thắt cổ chai nhưng làm giảm hiệu quả của việc truyền chỉ mục

˗ Đề nghị nhà phát triển viết covered query. Loại truy vấn sẽ hoàn toàn hài lòng với một chỉ mục. Vì vậy, không cần kiểm tra tài liệu nào để đáp ứng truy vấn và điều này làm cho truy vấn chạy nhanh hơn rất nhiều. Tất cả các phím chiếu cần được lập chỉ mục

˗ Sử dụng Index để sắp xếp kết quả và tránh chặn việc sắp xếp

˗ Loại bỏ các chỉ mục trùng lặp và không được sử dụng, nó cũng cải thiện thông lượng ổ đĩa và tối ưu hóa bộ nhớ

Hạn chế về bộ nhớ

* Kết quả phải tuân theo giới hạn tài liệu 16 MB -> aggs xuất ra một tài liệu duy nhất, tài liệu này phải < 16 MB. Lưu ý giới hạn không áp dụng cho tài liệu khi chúng chảy qua quy trình, chỉ áp dụng cho kết quả tài liệu cuối cùng.
* Để giảm thiểu giới hạn tài liệu, hãy sử dụng $limitvà $projectđể giảm kích thước tài liệu thu được.
* Tối đa 100 MB RAM có sẵn cho mỗi giai đoạn.
* Để giảm thiểu giới hạn giai đoạn, hãy đảm bảo các giai đoạn lớn nhất có thể sử dụng chỉ mục -> giảm yêu cầu bộ nhớ vì các mục nhập chỉ mục có xu hướng nhỏ hơn tài liệu mà chúng tham chiếu. Ngoài ra, việc sử dụng chỉ mục để sắp xếp giúp giảm đáng kể yêu cầu về bộ nhớ.
* Nếu không thể nhận được mức sử dụng mem < 100 MB, hãy thiết lập allowDiskUseđể agg tràn vào đĩa thay vì thực hiện mọi thứ trong bộ nhớ - nhưng hãy sử dụng như là phương sách cuối cùng vì ổ cứng chậm hơn hàng nghìn lần so với bộ nhớ! Có xu hướng được sử dụng nhiều hơn trong xử lý hàng loạt hơn là thời gian thực.
* Ghi chú allowDiskUsekhông hoạt động với $graphLookup.