Quản trị hệ cơ sở dữ liệu và tối ưu Hóa hiệu suất

Tổng quan

1. giới thiệu về tối ưu hóa cSDL
2. Các nguyên tắc cơ bản của tối ưu hóa csdl

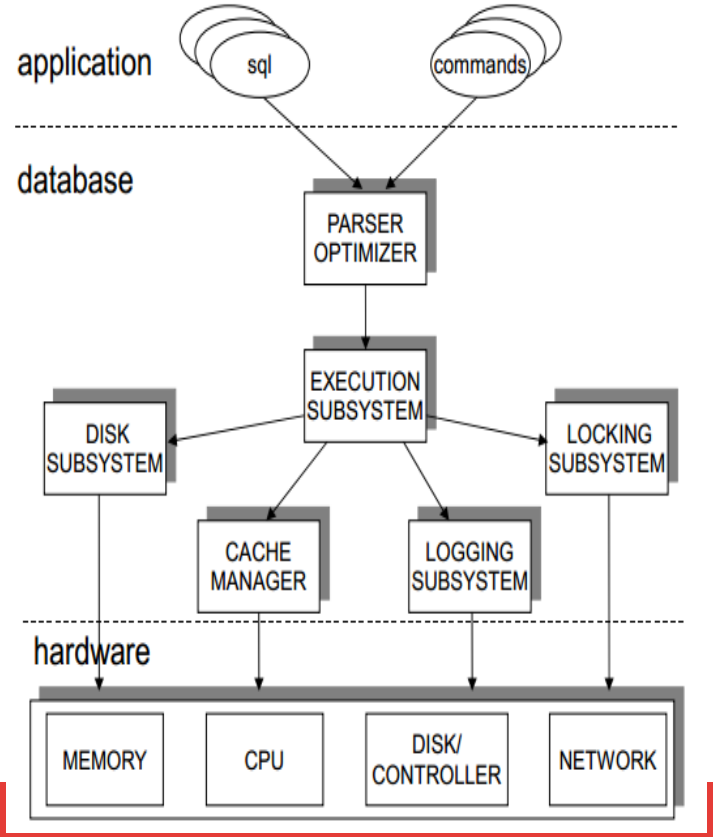
GIỚI THIỆU VỀ TỐI ƯU CSDL
1. Thế nào là tối ưu CSDL

- Giúp các ứng dụng có CSDL chạy nhanh hơn
  - Nhanh hơn tức là có thông lượng cao và có thời gian hồi đáp thấp.
  - Một sự cải thiện dù chỉ 5% cũng là đáng kể.
- Các tham số nào nên xem xét cho việc tinh chỉnh
  - Tất cả các tham số giúp đạt được mục tiêu tinh chỉnh
  - VD: tận dụng tối đa bộ nhớ, viết được các câu truy xuất tốt, tránh những tính toán không cần thiết, tránh tình trạng thắt cổ chai,…
- Nhược điểm: luôn có giá tiền tương ứng với những lợi ích đạt được
- Ưu điểm: đôi khi giá sẽ rất thấp và hiệu quả đạt được sẽ rất cao, vd: tránh được tình trạng thắt cổ chai hoặc câu truy xuất chạy cả giờ một cách không cần thiết

Tại sao tối ưu CSDL lại khó?

- Tối ưu câu truy xuất sau:

SELECT * FROM R WHERE R.a>5



- Các định nghĩa cần quan tâm của hệ quản trị CSDL thương mại
  - Giúp hiểu cấu trúc cơ bản của hệ quản trị CSDL
  - Giúp ra những quyết định tinh chỉnh chuẩn xác hơn
- Nguyên tắc tối ưu
  - Các nguyên tắc tối ưu ảnh hưởng đến hệ thống như thế nào?
- Các bước tối ưu khi gặp lỗi
  - Chẩn đoán (đang bị vấn đề ở đâu?)
  - Đưa ra các hệ giả thuyết:
    - Nguyên nhân của vấn đề là gì?
    - Áp dụng các nguyên tắc tối ưu để đề ra cách sửa
  - Kiểm nghiệm giả thuyết (làm thử)

Điều kiện tiên quyết

- Kĩ năng lập trình
- Kiến thức cấu trúc dữ liệu và giải thuật
- Kiến thức cơ bản hệ quản trị CSDL

Sự khác nhau giữa học phần thiết kế và quản trị CSDL(DMPT) và học phần CSDL(DS)

- Cùng xem xét các vấn đề giống nhau nhưng ở các góc nhìn khác nhau.
- Yêu cầu về tính cụ thể của thuật toán và các hoạt động đằng sau nó:
  - DS: Cách thức hoạt động chi tiết của B-tree
  - DBPT: B-tree có hiệu quả như thế nào và tại sao?
- Lý thuyết và thực hành:
  - DB: thực hiện các phép toán trên giấy
  - DBPT: thử nghiệm, so sánh, đánh giá các phép toán trên thiết bị thật, giải thích các kết quả.
- Tính toàn cục:
  - DB: chỉ tập trung vào các vấn đề một cách biệt lập
  - DBPT: tập trung vào sự tương tác giữa các thành phần và hệ thống
- Có sự trùng lặp và những khái niệm quan trong sẽ được xem xét lại

Tối ưu trong lý thuyết và thực nghiệm

- Các nhà thực nghiêm: áp dụng quy tắc ngón tay cái
  - VD: "Không bao giờ sử dụng các hàm tập hợp (vd như AVG) khi thực hiện các giao dịch cần hiệu suất cao (thời gian đáp ứng nhanh)"
  - Vấn đề: lạm dụng quy tắc ngón tay cái có thể gây lỗi, vd: AVG sẽ lỗi nếu có quá nhiều các bộ dữ liệu được truy xuất.
- Các nhà lý luận: đưa ra các mô hình và các giải pháp đảm bảo
  - VD: đặc điểm thời gian chạy của các thuật toán với các tham số truyền vào khác nhau.
  - Vấn đề: các cách tiếp cận phức tạp không được dùng trong thực tế kể từ lúc chúng dừng tại các giả thiết không thể thực hiện được.

Tối ưu trong lý thuyết và thực nghiệm

- Người tối ưu cơ sở dữ liệu: phải hiểu rõ và áp dụng các nguyên tắc

  VD:

  - Hiểu rõ: vấn đề không phải AVG mà là việc phải xử lý một khối lượng lớn dữ liệu (cách mà AVG thường làm)
  - Nguyên tắc: không quét khối lượng lớn dữ liệu cùng lúc.
  - Hiểu rõ để áp dụng đúng nguyên tắc cho các trường hợp thích hợp.

5 nguyên tắc cơ bản của tối ưu hóa:

1. Think globally; Fix locally
2. Phân vùng những nơi bị thắt nút cổ chai
3. Khởi tạo với chi phí cao, tiến hành với chi phí thấp
4. Hiểu rõ hoạt động của server
5. Be prepared for trade-offs

Think globally; Fix locally

- Người tối ưu phải như một bác sĩ giỏi
  - Think globally: xác định vấn đề
  - Fix locally: giảm tối đa sự can thiệp với các phần khác trong hệ thống (giảm phản ứng phụ)

  VD: bộ nhớ ngoài đang quá tải, phải làm sao?
  - Giải pháp 1: mua thêm bộ nhớ (local thinking)
    - ❑ Sự quá tải của bộ nhớ là một triệu chứng
    - ❑ Global thinking: sự quá tải của bộ nhớ được tạo ra ở đâu?
      - Thiếu chỉ số trong những câu truy vấn thường xuyên (thêm chỉ số)
      - Bộ nhớ tạm của CSDL quá nhỏ(tang bộ nhớ tạm)
      - Ghi và truy xuất dữ liệu trong bộ nhớ quá nhiều (ghi sang bộ nhớ khác)
    - ❑ Giải quyết vấn đề rẻ hơn và hiệu quả hơn nhiều là chống lại các

Think globally; Fix locally

- Giải pháp 2: tăng tốc truy vấn có thời gian chạy lâu nhất
  - Những truy vấn chậm nhất có thể không thường xuyên và thường chiếm khoảng 1% thời gian chạy.
  - Tăng tốc những câu truy vấn quan trọng
- Giải pháp 3: tăng tốc truy vấn chiếm thành phần lớn nhất trong thời gian chạy
  - Truy vấn mà làm chậm hệ thống có thể không cần thiết.
  - Nói với người lập trình ứng dụng: truy vấn này có cần thiết không? Có thể dùng cách khác đơn giản hơn mà vẫn đạt được kết quả không?

➢ **Kết luận: quan sát hệ thống khi bạn xác định vấn đề (think globally) và giải quyết vấn đề nơi no phát sinh (Fix locally)**

Phân vùng những nơi bị thắt nút cổ chai

- Thế nào là "nút cổ chai"?
  - Hiếm khi tất cả các phần của hệ thống làm việc cùng lúc
  - Thường có một phần giới hạn trong hiệu suất tổng thể của hệ thống
  - "Nút cổ chai" là phần giới hạn của hệ thống
- VD: tắc đường trên đường cao tốc
  - Có thể do 1 đoạn đường hẹp hoặc đường giao nhau
  - "Nút cổ chai": đoạn đường có tỷ lệ xe cao nhất trên một làn.

➔ Giải pháp:

1. Các tài xế phải đi nhanh hơn khi qua đoạn đường đó
2. Làm thêm các làn xe mới
3. Các tài xế tránh đi vào giờ cao điểm

Giải pháp 1 được gọi là local fix

Giải pháp 2 và 3 là phân vùng nơi bị thắt cổ chai

Các chiến lược phân vùng

- Phân vùng trong toán học:
  - Chia một tập hợp thành các phần không giao nhau
  - VD: A={a,b,c,d,e} được chia thành {{a,c},{d},{b,e}} là một cách phân vùng cho của A.
  - Tối ưu CSDL: từng câu truy vấn được thực hiện cũng là đã được phân vùng

➔ Hai chiến lược phân vùng cơ bản:

- Chia các vùng quá tải qua các vùng khác (thêm làn)
- Chia thời gian cho các truy xuất (tránh giờ cao điểm)

- VD1: tài khoản ngân hàng:
  - Một ngân hàng có N chi nhánh
  - Tất cả khách hàng đăng nhập vào tài khoản của họ ở chi nhánh gần nhất
  - Hệ thống trung tâm bị quá tải.
- ➔ Giải pháp: phân vùng
  - Đưa dữ liệu tài khoản của khách hàng tại chi nhánh I vào hệ thống phụ I xử lý.
- VD2: tranh chấp chiếm dụng vùng nhớ trống
  - Vùng nhớ trống: là vùng nhớ chưa được sử dụng
  - Mỗi luồng khi chạy sẽ chiếm dụng một vùng nhớ trống lớn bao gồm các vùng nhớ con và các luồng khác sẽ không thể sử dụng được vùng nhớ này cho đến khi luồng chạy xong
- ➔ Giải pháp: phân chia logic
  - Tạo nhiều vùng nhớ trống nhỏ
  - Mỗi vùng nhớ nhỏ đều chứa con trỏ đến một phần của vùng nhớ lớn
  - Với n vùng nhớ nhỏ, dung lượng cần xử lý của mỗi vùng giảm xuống còn 1/n
  - Phân chia các tài nguyên có thể bị chiếm dụng

- VD3: chiếm dụng và tranh chấp tài nguyên hệ thống với các long and short "online" transaction truy xuất vào cùng một dữ liệu (transaction: giao dịch, tác vụ, một tiến trình được chia nhỏ thành các tài nguyên thực thi độc lập)
  - Chiếm dụng và tranh chấp tài nguyên:
    - Nhiều luồng cùng chiếm dụng 1 tài nguyên (VD: bảng trong DB)
    - Nhiều luồng cùng truy xuất vào 1 vùng dữ liệu trong bộ nhớ

  Long and online transaction:
    - Long transaction (như: tải kho dữ liệu,…) sẽ chiếm dụng nhiều tài nguyên
    - Online transaction ngắn và cần thời gian trả về nhanh
  - Vấn đề:
    - Khóa chết có thể khiến cho các long transaction phải kết thúc.
    - Các online transaction chậm bởi vì nó phải đợi các long transaction kết thúc và giải phóng tài nguyên
  - ➔ Giải pháp: phân vùng thời gian hoặc không gian
    - Phân chia thời gian: chạy long transaction khi mà chỉ có ít online transaction
    - Phân chia không gian: chạy long transaction (nếu là chỉ đọc) trên dữ liệu ngoài trên phần cứng riêng biệt
    - Sắp thứ tự các long transaction để không gây cản trở các transaction khác

Phân vùng những nơi bị thắt nút cổ chai

- Các loại phân vùng:
  - Phân vùng không gian
  - Phân vùng logic
  - Phân vùng thời gian
- Lưu ý: hiệu suất không phải lúc nào cũng được cải thiện

➔ Khi mà gặp nơi bị thắt nút cổ chai:

- Thử tăng tốc khu vực đó (fix locally)
- Nếu không được thì phân vùng

Khởi tạo với chi phí cao, tiến hành với chi phí thấp

- Trong các sản phẩm nhân tạo thì thời gian khởi tạo thường dài:
  - Ô tô: ngắt khẩn cấp, hệ thống đánh lửa
  - Bóng đèn: tuổi thọ phụ thuộc vào số lần bật tắt
  - CSDL.
- Đọc từ bộ nhớ:
  - Chi phí bắt đầu quá trình đọc rất cao
  - Khi bắt đầu đọc, dữ liệu được gửi đi với tốc độ cao
  - VD: đọc 64 KB (128 sectors) chậm hơn ít nhất 2 lần so với đọc 512 bytes (1sectors)

➔ Kết luận:

- Các bảng thường xuyên truy xuất nên được đặt tuần tự trên bộ nhớ
- Các truy vấn thường xuyên vài cột trong một bảng hàng rất nhiều cột thì nên phân đoạn dọc bảng đó

➢ Lưu ý:

- Quét dữ liệu tuần tự trong RAM nhanh hơn nhiều so với đọc cùng dữ liệu đó từ các vị trí khác nhau trong bộ nhớ

❑ Mạng trễ:

- Phí gửi một tin nhắn cũng rất cao
- Chênh lệch chi phí giữa gửi 1 tin nhắn dung lượng lớn và 1 tin nhắn dung lượng nhỏ là không lớn
- VD: gửi 1 gói tin 1 byte đắt như gửi 1 gói tin 1KB

➔ Kết luận:

- Gửi vài gói tin lớn tốt hơn là gửi nhiều gói tin nhỏ

- Chi phí truy vấn:
  - Trước khi câu truy vấn được thực hiện: nó được phân tách => tối ưu => truy nhập vào đường đến dữ liệu cần
  - Chỉ một truy vấn nhỏ cũng có khoảng 10000 giai đoạn thực hiện
- Biên dịch truy vấn:
  - Lưu tạm thời các kết quả phân tách, tối ưu và truy nhập vào đường đến dữ liệu ➔ lân thực hiện tiếp theo sẽ tiết kiệm được chi phí này
  - Lưu tạm thời các câu truy vấn hay được gọi nhưng với tham số truyền vào khác nhau

VD: truy vấn tạo bởi form hỏi khách hàng: chỉ dữ liệu của khách hàng là thay đổi, cấu trúc câu truy vấn không đổi

➔ Kết luân: chỉ biên dịch các truy vấn thường thực hiện

- Chi phí kết nối từ các ngôn ngữ lập trình
  - Ứng dụng viết bởi C++, Java, … gọi tới CSDL
  - Mở kết nối: chi phí khá cao
    - ✓ Tạo kết nối
    - ✓ Xác thực người dung
    - ✓ Kiểm tra các thông số kết nối
- Pool kết nối:
  - Tạo một pool kết nối và giữ nó luôn mở
  - Các yêu cầu kết nối mới sẽ sử dụng kết nối có sẵn của pool
- ➔ Kết luận:
  - SELECT một lần và lặp lại kết quả tốt hơn là lặp lại SELECT nhiều lần
  - Tạo pool kết nối

- Các ý nghĩa của chi phí khởi tạo:
  - Chi phí đọc byte đầu tiên
  - Chi phí gửi byte đầu tiên
  - Chi phí chuẩn bị thực hiện truy vấn
  - Chi phí mở kết nối với CSDL

➔ Kết luận: cố gắng tối giản các bước khởi tạo mà vẫn đạt được kết quả mong muốn

Hiểu rõ hoạt động của server

- Công việc được phân bổ ở đâu?
  - Server
  - Client
- Mỗi hành động đều dựa trên 3 yếu tố:
  - Tài nguyên hiện có của client và server
  - Các dữ liệu cần thiết nằm ở đâu?
  - Có tương tác với màn hình không?

- Tài nguyên hiện có của server và client
  - Nếu server quá tải thì ngắt kết nối với client có chạy ứng dụng
  - Ứng cử viên tốt: phù hợp về tài nguyên
- Tính toán vị trí các dữ liệu cần thiết
  - VD: kết quả trả về của ứng dụng (thông báo màn hình) rằng CSDL đã thay đổi (chèn thêm vào bảng)

  ➔ Giải pháp của client: kiểm tra vòng
    - Định kì truy vấn CSDL để kiểm tra thay đổi
    - Không hiệu quả khi có nhiều câu truy vấn

  ➔ Giải pháp của server: kích hoạt
    - Chỉ "cháy" khi thay đổi xảy ra

  ❑ Khi mà dữ liệu cần thiết trên server thì giải pháp của server là hiều quả hơn

- Các tác vụ sử dụng CSDL có tương tác với màn hình không?
  - Một tương tác với màn hình không nên được thực thi trong một transaction
  - Các transaction tương tác với màn hình thường kéo dài thời gian
  - Giải pháp: chia nhỏ các transaction như sau:
    1. Transaction đầu tiên lấy dữ liệu từ máy chủ
    2. Phiên tương tác tại phía client (ngoài bất cứ transaction nào)
    3. Transaction thứ 2 thay đổi cài đặt trên server

Cân bằng giữa chi phí và hiệu quả

- Để tăng tốc máy tính thì cần:
  - Bộ nhớ chính lớn hơn
  - Bộ nhớ ngoài lớn hơn
  - CPU tốt hơn
  - Hệ điều hành tốt hơn
- Tăng tốc 1 truy vấn có thể làm chậm các truy vấn khác

VD: tham số giúp các câu truy vấn quan trọng nhanh hơn nhưng

- Không gian bộ nhớ cần rộng hơn
- Làm chậm truy vấn INSERT hoặc UPDATE mà không dùng tham số đó.

➔ Bạn muốn cải thiện tốc độ thì bạn sẵn sàng trả bao nhiêu?