Sắp xếp ngoại

I. Đặt vấn đề

1. Đang có những thuật toán sắp xếp nào?

Chúng ta đang có những thuật toán sắp xếp từng học như bubble sort, selection sort, quick sort, heap sort,… với độ phức tạp khác nhau. Thường độ phức tạp các thuật toán hay dùng là O(nlogn).

2. Tại sao chúng ta cần thuật toán sắp xếp ngoại?

Với lượng dữ liệu khổng lồ cần sắp xếp (lớn vượt qua khả năng xử lý của RAM), ta cần một phương pháp xử lý dữ liệu mới.

Giả sử máy tính của chúng ta chỉ có 4GB RAM, nhưng chúng ta cần xử lý lượng dữ liệu 100GB. Với những thuật toán thường thấy với độ phức tạp O(nlogn), chúng ta không thể tải toàn bộ dữ liệu lên RAM để xử lý (hoặc không dư tiền để mua thêm mấy thanh RAM để xử lý). Thì chúng ta sử dụng những tài nguyên khác, rẻ và nhiều hơn thế, đó là ổ cứng.

3. Vì sao gọi là “sắp xếp ngoại”?

Vì phương pháp xử lý dữ liệu khổng lồ cần sắp xếp thông qua sự hỗ trợ của bộ nhớ ngoài (thường là ổ đĩa). Chúng ta sẽ chia nhỏ lượng dữ liệu, xử lý, lưu vào các file tạm trên ổ đĩa. Sau đó trộn các file kết quả thành một file cuối cùng.

II. Mô tả thuật toán

\* Hình trong mô tả là 2-ways merge sort (simple external sort). Chỉ dùng hai file nhỏ để thực hiện. Với phương pháp này không đảm bảo về luồng file lẫn tốc độ xử lý.

Giới thiệu về phương pháp cải tiến hơn Multiway merge. Thay vì chỉ dùng 2 file thì chia nhỏ hơn trong phạm vi bộ nhớ ngoài cho phép.

Bước 1: Đọc dữ liệu đầu vào từng phần nhỏ vừa với khả năng xử lý của RAM tải lên RAM.

Bước 2: Thực hiện sắp xếp phần dữ liệu mới tả lên RAM. (Có thể dùng bất kì thuật toán nào mà mình giỏi, độ phức tạp tối ưu như quicksort)

Bước 3: Ghi dữ liệu đã sắp xếp vào bộ nhớ ngoài thành những file nhỏ gọi là runs. Lặp lại các bước trên cho đến khi xử lý hết dữ liệu đầu vào. Ta sẽ có được n/k file nhỏ.

Bước 4: Dùng K-way merge sort để trộn các dữ liệu đó thành một file đã được sắp xếp.

\* Giới thiệu về kỹ thuật K-way merge sort:

Input: k danh sách đầu vào đã được sắp xếp ở từng danh sách

Output: Một danh sách chứa tất cả các phần tử của k danh sách trên và đã được sắp xếp.

Thực hiện một vòng lặp đến khi tất cả các danh sách đều rỗng thì dừng lại với 2 bước trong mỗi vòng lặp như sau:

Tìm phần tử nhỏ nhất trong số các phần tử đầu tiên của các danh sách. (vì các phần tử đầu tiên của mỗi danh sách là phần tử nhỏ nhất trong mỗi danh sách, nên phần tử tìm được sẽ là phần tử nhỏ nhất đang nằm trong k danh sách.)

Đẩy phần tử tìm được vào danh sách kết quả, và xóa nó khỏi danh sách hiện tại của nó. Quay lại thực hiện bước 1.

\* Cải tiến với Replacement Selection

Sử dụng heap

M: elements in main memory

Ưu điểm:

- Hiệu quả cho dữ liệu có kích thước lớn

- Tối ưu hơn cho dữ liệu co nhiều trục

- Có thể sử dụng để sắp xếp dữ liệu có nhiều trục

Nhược điểm:

- Có thể sử dụng nhiều bộ nhớ

- Có thể phức tạp hơn

\* Một số phương pháp khác

Các phương pháp khác nhau ở bước 4 là kỹ thuật trộn các file nhỏ đã sắp xếp lại với nhau. Các phương pháp có độ tối ưu về thời gian thực thi khác nhau. Một số phương pháp hay thấy là:

- Phương pháp trộn runs

- Phương pháp trộn tự nhiên

- Phương pháp trộn đa lối cân bằng

- Phương pháp trộn đa pha

\* Một vài kỹ thuật giúp tối ưu bài toán hơn

- Song song hóa

- Cải thiện tốc độ phần cứng

- Sử dụng thuật toán sắp xếp (ở bước 2) tối ưu hơn (ví dụ radix sort thay vì quick sort)\

III. Độ phức tạp

Tương tự như độ phức tạp của thuật toán sắp xếp nội được sử dụng

Tuy nhiên, thời gian thực tế có thể cao hơn do việc đọc và ghi đĩa

IV. So sánh

External Sort là thuật toán sắp xếp ngoại

Sắp xếp nội là thuật toán sắp xếp nội

External Sort chỉ được sử dụng khi dữ liệu quá lớn để lưu trữ trong bộ nhớ chính

Sắp xếp nội có thể được sử dụng cho dữ liệu có kích thước bất kỳ

V. Ứng dụng

- Sắp xếp dữ liệu có kích thước lớn như: dữ liệu thống kê, dữ liệu hình ảnh, dữ liệu video

- Sắp xếp, tìm kiếm trên những dataset lớn

- Quản lý các log lớn, truy vấn các database

- Các công ty, tập đoàn có nguồn dữ liệu khổng lồ cần xử lý

Kích thước của bộ nhớ đệm ảnh hưởng đến hiệu suất của External Sort theo hai cách:

Tăng tốc độ sắp xếp: Khi kích thước của bộ nhớ đệm tăng, thuật toán sắp xếp sẽ có thể đọc và ghi dữ liệu nhiều hơn trong mỗi lần truy cập đĩa. Điều này sẽ giúp tăng tốc độ sắp xếp.

Giảm số lượng truy cập đĩa: Khi kích thước của bộ nhớ đệm tăng, thuật toán sắp xếp sẽ cần ít truy cập đĩa hơn để sắp xếp dữ liệu. Điều này sẽ giúp giảm thời gian thực thi của thuật toán sắp xếp.