BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC ĐÀ NẴNG

LÊ VIỆT ANH

ỨNG DỤNG MÔ HÌNH MAPREDUCE XÂY DỰNG HỆ THỐNG ĐÁNH GIÁ ĐỘ TƯƠNG ĐỒNG VĂN BẢN

Chuyên ngành: KHOA HỌC MÁY TÍNH
Mã số: 60.48.01.01

TÓM TẮT LUẬN VĂN THẠC SĨ KỸ THUẬT

Đà Nẵng – Năm 2016

Công trình được hoàn thành tại

ĐẠI HỌC ĐÀ NẴNG

Người hướng dẫn khoa học: **PGS.TS. Nguyễn Tấn Khôi**

Phản biện 1: TS. Ninh Khánh Duy

Phản biện 2: PGS. TS. Lê Mạnh Thạnh

Luận văn đã được bảo vệ trước Hội đồng chấm Luận văn tốt nghiệp thạc sĩ Khoa học máy tính họp tại Đại học Đà Nẵng vào ngày 25 tháng 07 năm 2016

Có thể tìm hiểu luận văn tại:

Trung tâm Thông tin - Học liệu, Đại học Đà Nẵng

MỞ ĐẦU

1. Lý do chọn đề tài

Theo thống kê, vào năm 2000 chỉ mới có một phần tư lượng thông tin lưu trữ trên toàn thế giới ở dạng kỹ thuật số. Ba phần tư còn lại được người ta lưu trên giấy tờ, phim, và các phương tiện analog khác. Nhưng do lượng dữ liệu kỹ thuật số bùng nổ quá nhanh, cứ 3 năm lại tăng gấp đôi, cục diện trên nhanh chóng thay đổi. Ngày nay, chỉ dưới 2% tổng lượng thông tin chưa được chuyển sang lưu trữ ở dạng kỹ thuật số.

Những loại dữ liệu này được sử dụng vào những chức năng đặc biệt nhờ vào sự hỗ trợ của những bộ nhớ máy tính có chi phí rất thấp, những bộ xử lý cực mạnh. Dữ liệu lớn cho phép chúng ta tiến hành các thử nghiệm nhanh hơn, khám phá nhiều thông tin hơn. Việc phân tích, đánh giá, xử lý tìm kiếm, tổng hợp dữ liệu lớn một cách nhanh chóng trở nên cần thiết trong thời đại mà dữ liệu đang bùng nổ. Những lợi thế ấy cần được phục vụ cho tiến trình sáng tạo, và nhiều khi sự sáng tạo sẽ đem lại kết quả bất ngờ mà không dữ liệu nào có thể tiên đoán, vì nó chưa từng có trước đấy .

Hiện nay kỹ thuật xử lý song song, phân tán đang được nghiên cứu và ứng dụng rộng rãi, đáng kể nhất là mô hình MapReduce - nền tảng lập trình giúp Google, Yahoo, Facebook, và các hãng khác cho phép xử lý khối dữ liệu khổng lồ (hàng petabytes) của họ. Mô hình MapReduce đưa ra quy trình giúp xử lý tập hợp dữ liệu siêu lớn đặt tại các máy tính phân tán, có thể xử lý được dữ liệu không cấu trúc (dữ liệu lưu trữ dạng tệp tin hệ thống) và dữ liệu cấu trúc (dữ liệu quan hệ 2 chiều). Ở dạng đơn giản nhất, MapReduce chia việc xử lý thành nhiều khối công việc nhỏ, phân tán khắp liên cung gồm các nút tính toán, rồi thu thập các kết quả. MapReduce định nghĩa dữ liệu

dưới dạng cặp khóa/giá trị, hỗ trợ xử lý song song có khả năng mở rộng cao.

Việc tìm kiếm, phát hiện sự tương đồng, giống nhau về nội dung các văn bản trong kho văn bản khổng lồ là vấn đề quan trọng trong công tác kiểm tra, ngăn chặn việc sao chép nội dung các công trình nghiên cứu hay phục vụ cho việc tra cứu các tài liệu học tập, tìm kiếm tài liệu cùng chủ đề. Nhưng với khối lượng lớn như vậy là quá khó cho máy tính đơn lẻ trong khi yêu cầu của người dùng là thời gian thực, bằng cách kết hợp sự vượt trội trong tính toán của hệ thống máy tính song song để giải quyết vấn đề này.

Bài toán tìm kiếm và đánh giá độ tương đồng về nội dung giữa hai văn bản có nhiều ứng dụng thực tiễn như trong giáo dục, nghiên cứu khoa học, sinh học, tìm kiếm, …

Xuất phát từ nhu cầu thực tiễn, tôi đã nghiên cứu và thực hiện đề tài luận văn cao học:

"*Ứng dụng mô hình MapReduce xây dựng hệ thống đánh giá độ tương đồng văn bản*".

2. Mục tiêu đề tài

Mục tiêu chính

Nghiên cứu xây dựng hệ thống xử lý, đánh giá độ tương đồng giữa một tài liệu với các tài liệu khác trong cơ sở dữ liệu dựa trên mô hình MapReduce.

Mục tiêu cụ thể

- Nghiên cứu và phân tích về các mô hình triển khai, mô hình dịch vụ trong cơ sở dữ liệu phân tán

- Nghiên cứu về cơ sở dữ liệu phân tán, ưu và nhược điểm của hệ cơ sở dữ liệu phân tán, phạm vi áp dụng

- Tìm hiểu về hệ thống quản lý tập tin phân tán (HDFS), kiến trúc của HDFS, cách thức lưu trữ

- Tìm hiểu mô hình lập trình Map Reduce, Hadoop

- Xây dựng ứng dụng so khớp, tìm độ tương đồng của một tài liệu với các tài liệu khác trong cơ sở dữ liệu lớn dựa vào mô hình MapReduce

3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu

- Mô hình lập trình phân tán MapReduce

- Nền tảng Hadoop, cơ sở dữ liệu phân tán

- Thư viện lập chỉ mục ngôn ngữ Apache Lucene

Phạm vi nghiên cứu

- Đánh giá độ tương đồng giữa các văn bản sử dụng phương pháp thống kê tần số xuất hiện của các từ trong văn bản dựa vào xử lý phân tán

- Mô hình Map Reduce trong việc xử lý tài liệu trong kho dữ liệu lớn

4. Phương pháp nghiên cứu

Phương pháp lý thuyết

- Khảo sát, phân tích và tổng hợp các nhu cầu ứng dụng mô hình MapReduce.

- Nghiên cứu các mô hình lập trình song song, phân tán.

- Phân tích, lựa chọn công nghệ.

Phương pháp thực nghiệm

- Triển khai mô hình MapReduce, cấu hình Hadoop.

- Xây dựng giải thuật so trùng, phân tích các mục tiêu và yêu cầu của bài toán tìm kiếm các văn bản có độ tương đồng.

- Xây dựng giải thuật mô phỏng trên hệ thống đã được cài đặt.

5. Ý nghĩa khoa học và thực tiễn

Ý nghĩa khoa học

- Ứng dụng công nghệ xử lý phân tán trong việc quản lý, tìm kiếm và lưu trữ dữ liệu với quy mô dữ liệu lớn, đáp ứng được nhu cầu về quản lý dữ liệu ngày càng mở rộng của xã hội.

- Nâng cao chất lượng, hiệu quả kinh tế, thời gian, công sức trong việc quản lý, khai thác dữ liệu phân tán lớn, phục vụ cho mục đích cụ thể của người dùng.

Ý nghĩa thực tiễn

- Ứng dụng mô hình lập trình phân tán MapReduce và các công cụ hỗ trợ giúp giải quyết các bài toán xử lý trên tập dữ liệu lớn

- Xây dựng ứng dụng phục vụ tìm kiếm các văn bản có độ tương đồng

- Ứng dụng mô hình lập trình MapReduce trong nhiều giải thuật tìm kiếm khác trên các tập dữ liệu lớn.

6. Bố cục của luận văn

Mở đầu: Đặt vấn đề về tính cần thiết của đề tài, mục đích, phạm vi nghiên cứu của đề tài, phương pháp nghiên cứu và bố cục của luận văn.

Chương 1: Tổng quan đề tài.

Chương 2: Xử lý phân tán MapReduce.

Chương 3: Thiết kế và xây dựng hệ thống.

CHƯƠNG 1
TỔNG QUAN ĐỀ TÀI

1.1. HỆ THỐNG PHÂN TÁN

1.1.1. Khái niệm hệ thống phân tán

Khái niệm về hệ thống phân tán có thể hiểu theo nhiều mức độ khác nhau, đơn giản nhất là việc trao đổi thông tin giữa các chương trình sử dụng cơ chế đường ống, cho đến mức độ cao hơn là sự trao đổi thông tin giữa các chương trình trên mạng. Nhìn chung việc xây dựng các ứng dụng phân tán phức tạp hơn nhiều so với các ứng đụng tập trung. Trong nhiều trường hợp, việc bắt buộc phải xây dựng các ứng dụng phân tán thường vì những lý do sau:

- Yêu cầu tính toán phân tán : ứng dụng chạy trên nhiều máy tính khác nhau nhằm tận dụng khả năng tính toán song song hoặc nhằm mục đích sử dụng khả năng tính toán của các máy tính chuyên dụng.

- Yêu cầu xử lý phân tán : ứng dụng chạy trên nhiều máy tính khác nhau mới có để đáp ứng được yêu cầu bài toán bài toán. Trong trường hợp này, đối tượng được phục vụ, được giải quyết trong bài toán thường ở xa đối tượng phục vụ, đối tượng giải quyết.

- Yêu cầu tính chính xác và an toàn dữ liệu: Yêu cầu này liên quan tới các hệ thống cần phải đảm bảo tính chính xác cao và an toàn dữ liệu lớn. Các hệ thống này thường được thiết kế để có thể đáp ứng được yêu cầu tính toán ngay cả khi có sự cố xảy ra, điều này được thực hiện bằng cách tăng số lần tính toán cho cùng một nhiệm vụ, tăng số ứng đụng chạy dự phòng nhằm mục đích kịp thời phát hiện và xử lý lỗi.

- Chia sẻ tài nguyên: Các ứng dụng chạy trên mạng thực hiện trao đổi thông tin với nhau. Các ứng dụng này có thể chia sẻ tài nguyên của mình cho các ứng dụng khác, hoặc sử dụng tài nguyên được chia sẻ với một quyền hạn được định nghĩa. Một số ứng dụng phải chạy trên nhiều máy tính vì dữ liệu được đặt phân tán trên mạng liên quan đến quyền quản lý và quyền sở hữu dữ liệu: cho phép truy nhập và sử dụng theo quyền được cho phép.

1.1.2. Mô hình xử lý phân tán mở

Việc phát triển các ứng dụng trong hệ thống phân tán thường gặp một số khó khăn sau:

- Khác hiệt chủng loại: Trên mạng có nhiều loại máy tính và thiết bị mạng của nhiều nhà sản xuất khác nhau và các máy tính được cài đặt các hệ điều hành khác nhau.

- Không tương thích giao diện liên kết: Các nhóm phát triển hệ thống phân tán hoàn toàn có thể tự qui định giao diện liên kết giữa các hệ thống với nhau, do đó có thể sẽ gặp khó khăn khi liên kết với hệ thống phân tán do nhóm khác phát triển.

- Khó tích hợp: Các phần mềm thường được phát triển trên các hệ điều hành khác nhau, các ngôn ngữ khác nhau, do nhiều đơn vị độc lập phát triển, sử dụng cơ sở dữ liệu khác nhau. Do vậy, để tích hợp được các phần mềm này thường đòi hỏi nhiều thời gian, nhận lực, và đôi khi, khả năng tích hợp là không thể.

- Tính rủi ro sản phẩm cao : Xây dựng và triển khai các hệ thống phân tán thường chịu nhiều rủi ro do không tuân thủ các chuẩn được khuyến nghị trước, hoặc bản thân các khuyên nghị không đồng nhất. Vì vậy, giá thành phát triển cũng bị kéo theo. Thời gian phát

triển và khả năng bảo hành bảo trì thường lớn hơn các hệ thống độc lập.

1.1.3. Các đặc trưng của hệ phân tán

a. Chia sẻ tài nguyên

b. Tính mở (Openess)

c. Tính tương tranh (Concurrency)

d. Tính chịu lỗi (Fault tolerance)

e. Tính trong suốt (Transparency)

f. Khả năng thay đổi quy mô (Saclability)

1.1.4. Các mô hình hệ thống phán tán

Tuỳ theo yêu cầu sử dụng và khả năng phát triển, có thể xây dựng hệ thống phân tán theo các mức độ sau:

- Mô hình truyền tập tin
- Mô hình khách chủ thuần tuý (Client/Server)
- Mô hình ngang hàng (Pear-To-Pear)

a. Mô hình truyền tập tin

Mô hình truyền tập tin là mô hình cổ điển nhất, về cơ bản các chương trình trên các máy tính cùng sử dụng một giao thức để trao đổi thông tin với nhau dưới dạng truyền tập tin.

b. Mô hình Client/Server

Mô hình Client/Server đang được áp dụng phổ biến vì các chương trình sử dụng cơ chế trao đổi tin báo để liên lạc với nhau, như vậy mềm dẻo hơn mô hình truyền tập tin. Thực chất mô hình này xuất phát từ phương pháp gọi thủ tục từ xa RPC, hiện nay mô hình Client/Server đã phát triển theo hướng sử dụng các đối tượng phân tán.

Trong lĩnh vực công nghệ thông tin, mô hình Client/Server được hiểu là hình thức trao đổi thông tin giữa các tiến trình cung cấp

dịch vụ (Server) và tiến trình sử dụng dịch vụ (Client). Trong mô hình này, Client yêu cầu các dịch vụ đã được cài đặt trên Server, Server xử lý yêu cầu và trả về kết quả cho Client.

c. Mô hình ngang hàng

Mô hình ngang hàng tương tự như mô hình Client/Server, nhưng các tiến trình tương tác có thể là Client, Server hoặc đồng thời là Client và Server.

1.1.5. Ưu nhược điểm của hệ thống phân tán

a. Ưu điểm

- Tính sẵn sàng cao: Hệ thống là phân tán nhằm mục đích đáp ứng nhu cầu xử lý phân tán. Vì vậy, có thể tăng tính sẵn sàng của hệ thống nhờ tăng độ dự phòng phần cứng, phần mềm trong hệ thống, hoặc dựa trên việc lưu trữ dữ liệu phân tán tại nhiều điểm để dự phòng lỗi.

- Tăng hiệu năng xử lý: Hệ thống phân tán có thể giải quyết được nhu cầu về tính toán song song, phân tải hệ thống, đồng thòi, các yêu cầu truy cập dữ liệu thường xuyên cũng nhanh hơn do dữ liệu được lưu trữ gần.

- Chia sẻ tài nguyên: Hệ thống phân tán có thể thay đổi quy mô. Do đó có thể kết nối với các mạng bên ngoài để chia sẻ tài nguyên, và sử dụng tài nguyên được chia sẻ.

- Nhất quán trong sử dụng: Các ứng dụng trong một hệ thống phân tán đều nhất quán về giao diện, về chuẩn kết nối dữ liệu, về giao thức truyền thông. Do vậy sẽ giúp người sử dụng hệ thống dễ dàng nắm bắt các module trong hệ thống.

b. Nhược điểm

- Khó thiết kế và xây dựng: Hệ thống phân tán thường rất phức tạp. Để một hệ thống phân tán có thể hoạt động được, người ta phải

tính toán đến nhiều góc độ lý thuyết và thực tiễn như: Giao thức truyền thông, thuật toán tối ưu truy xuất dữ liệu, bảo mật và an toàn dữ liệu tại các điểm phân tán và trên đường truyền,... Do vậy, hệ phân tán khó thiết kế và xây dựng.

- Khó bảo hành bảo trì: Hệ thống phân tán có các đối tượng thành phần trong hệ thống đặt tại nhiều điểm khác nhau nên khi bảo hành bảo trì thường khó khăn hơn các hệ thống độc lập hoặc các hộ thống tập trung.

1.2. XỬ LÝ PHÂN TÁN

1.3. PHƯƠNG PHÁP TÍNH ĐỘ TƯƠNG ĐỒNG GIỮA CÁC VĂN BẢN

Độ tương đồng ngữ nghĩa giữa các câu đóng một vai trò quan trọng trong các nghiên cứu về xử lý văn bản. Nó được sử dụng như là một tiêu chuẩn của trích chọn thông tin nhằm tìm ra những tri thức ẩn trong các cơ sở dữ liệu văn bản hay trên các kho dữ liệu trực tuyến. Hiện nay tồn tại một số phương pháp tính độ tương đồng giữa các câu, điển hình là các phương pháp dựa trên tính toán thống kê và các phương pháp dựa trên quan hệ ngữ nghĩa giữa tập các từ trong các câu đó.

1.3.1. Phương pháp thống kê

a. Độ tương đồng Cosine

b. Độ tương đồng dựa vào khoảng cách Euclide

1.3.2. Phương pháp dựa trên quan hệ ngữ nghĩa giữa các từ:

Một số hướng tiếp cận phân tích cấu trúc ngữ pháp, sử dụng mạng ngữ nghĩa đối với từ như Wordnet corpus hoặc Brown corpus... Các phương pháp này xử lý chậm hơn, tốn nhiều chi phí hơn nhưng xét về mặt ngữ nghĩa thì độ tương đồng chính xác cao hơn phương pháp thống kê.

1.4. CÁC KẾT QUẢ NGHIÊN CỨU LIÊN QUAN

1.5. HƯỚNG NGHIÊN CỨU CỦA ĐỀ TÀI

Đánh giá độ tương đồng giữa các văn bản sử dụng phương pháp thống kê tần số xuất hiện của các từ trong văn bản dựa vào xử lý phân tán.

Sử dụng thư viện nguồn mở Apache Lucene để lập chỉ mục các văn bản trong kho dữ liệu, tìm kiếm trọng số của từ trong kho dữ liệu.

Sử dụng MapReduce framework để chia chỏ quá trình đánh chỉ mục cho kho dữ liệu lớn và tính toán độ tương đồng của văn bản nhập vào so với kho dữ liệu.

1.6. KẾT CHƯƠNG

Chương này giới thiệu khái niệm hệ thống phân tán, các mô hình xử lý phân tán, đặc trưng của hệ phân tán, mô hình hệ thống phân tán, xử lý dữ liệu phân tán, các phương pháp tính độ tương đồng giữa các văn bản. Ngoài ra còn đưa ra các nghiên cứu đã có và hướng nghiên cứu riêng của đề tài.

CHƯƠNG 2
XỬ LÝ PHÂN TÁN MAPREDUCE

Chương này tìm hiểu, phân tích các thành phần cấu tạo và cài đặt mô hình xử lý phân tán MapReduce. Tìm hiểu thư viện nguồn mở ApacheLucene trong việc lập chỉ mục và tìm kiếm nội dung văn bản.

2.1. MÔ HÌNH MAPREDUCE

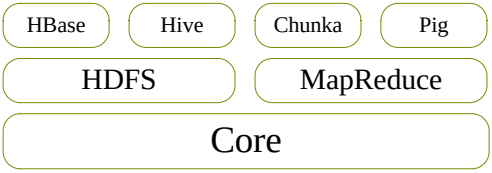
2.1.1. Cơ chế xử lý

2.1.2. Hàm Map

2.1.3.Hàm Reduce

2.2. NỀN TẢNG HADOOP

2.2.1. Giới thiệu Hadoop



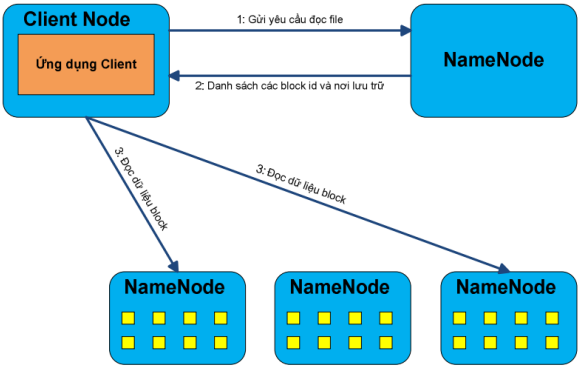
Hình 2.4. Kiến trúc Hadoop

2.2.2. Hệ thống tập tin phân tán HDFS

a. Kiến trúc HDFS

b. Quá trình đọc file trên HDFS

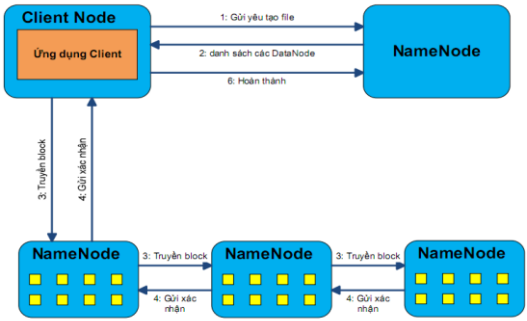
Hình sau miêu tả rõ quá trình client đọc một tập tin trên HDFS.



Hình 2.6. Quá trình client đọc một tập tin trên HDFS

c. Ghi file trên HDFS

Tiếp theo, ta sẽ khảo sát quá trình client tạo một tập tin trên HDFS và ghi dữ liệu lên tập tin đó. Hình sau mô tả quá trình tương tác giữa client lên hệ thống HDFS.



Hình 2.4. Quá trình client ghi một tập tin trên HDFS

2.2.3. Kiến trúc MapReduce Engine

a. Các thành phần của MapReduce Engine

Client Program: Chương trình HadoopMapReduce mà client đang sử dụng và tiến hành chạy một MapReduce Job. JobTracker: Tiếp nhận job và đảm nhận vai trò điều phối job này, nó có vai trò như bộ não của Hadoop MapReduce. Sau đó, nó chia nhỏ job thành các task, tiếp theo sẽ lên lịch phân công các task (map task, reduce task) này đến các tasktracker để thực hiện. Kèm theo vai trò của mình, JobTracker cũng có cấu trúc dữ liệu riêng của mình để sử dụng cho mục đích lưu trữ, ví dụ như nó sẽ lưu lại tiến độ tổng thể của từng job, lưu lại trang thái của các TaskTracker để thuận tiện cho thao tác lên lịch phân công task, lưu lại địa chỉ lưu trữ của các output của các TaskTracker thực hiện maptask trả về. TaskTracker: Đơn giản nó chỉ tiếp nhận maptask hay reducetask từ JobTracker để sau đó thực hiện. Và để giữ liên lạc với JobTracker, Hadoop Mapreduce cung cấp cơ chế gửi heartbeat từ TaskTracker đến JobTracker cho các nhu cầu như thông báo tiến độ của task do TaskTracker đó thực hiện, thông báo trạng thái hiện hành của nó (idle, in-progress, completed). HDFS:** là hệ thống file phân tán

được dùng cho việc chia sẻ các file dùng trong cả quá trình xử lý một job giữa các thành phần trên với nhau.

b. Cơ chế hoạt động

Đầu tiên chương trình client sẽ yêu cầu thực hiện job và kèm theo là dữ liệu input tới JobTracker. JobTracker sau khi tiếp nhận job này, nó sẽ thông báo ngược về chương trình client tình trạng tiếp nhận job. Khi chương trình client nhận được thông báo nếu tình trạng tiếp nhận hợp lệ thì nó sẽ tiến hành phân rã input này thành các split (khi dùng HDFS thì kích thước một split thường bằng với kích thước của một đơn vị Block trên HDFS) và các split này sẽ được ghi xuống HDFS. Sau đó chương trình client sẽ gửi thông báo đã sẵn sàng để JobTracker biết rằng việc chuẩn bị dữ liệu đã thành công và hãy tiến hành thực hiện job.

Khi nhận được thông báo từ chương trình client, JobTracker sẽ đưa job này vào một stack mà ở đó lưu các job mà các chương trình client yêu cầu thực hiện. Tại một thời điểm JobTracker chỉ được thực hiện một job. Sau khi một job hoàn thành hay bị block, JobTracker sẽ lấy job khác trong stack này (FIFO) ra thực hiện. Trong cấu trúc dữ liệu của mình, JobTrack có một job scheduler với nhiệm vụ lấy vị trí các split (từ HDFS do chương trình client tạo), sau đó nó sẽ tạo một danh sách các task để thực thi. Với từng split thì nó sẽ tạo một maptask để thực thi, mặc nhiên số lượng maptask bằng với số lượng split. Còn đối với reduce task, số lượng reduce task được xác định bởi chương trình client. Bên cạnh đó, JobTracker còn lưu trữ thông tin trạng thái và tiến độ của tất cả các task.

Ngay khi JobTracker khởi tạo các thông tin cần thiết để chạy job, thì bên cạnh đó các TaskTracker trong hệ thống sẽ gửi các heartbeat đến JobTracker. Hadoop cung cấp cho các TaskTracker cơ

chế gửi heartbeat đến JobTracker theo chu kỳ thời gian nào đó, thông tin bên trong heartbeat này cho phép JobTrack biết được TaskTracker này có thể thực thi task hay không. Nếu TaskTracker còn thực thi được thì JobTracker sẽ cấp task và vị trí split tương ứng đến TaskTracker này để thực hiện. Tại sao ở đây ta lại nói TaskTracker còn có thể thực thi task hay không. Điều này được lý giải là do một Tasktracker có thể cùng một lúc chạy nhiều map task và reduce task một cách đồng bộ, số lượng các task này dựa trên số lượng core, số lượng bộ nhớ Ram và kích thước heap bên trong TaskTracker này.

Khi một TaskTracker nhận thực thi maptask, kèm theo đó là vị trí của input split trên HDFS. Sau đó, nó sẽ nạp dữ liệu của split từ HDFS vào bộ nhớ, rồi dựa vào kiểu format của dữ liệu input do chương trình client chọn thì nó sẽ parse split này để phát sinh ra tập các record, và record này có 2 trường: key và value. Cho ví dụ, với kiểu input format là text, thì tasktracker sẽ cho phát sinh ra tập các record với key là offset đầu tiên của dòng (offset toàn cục), và value là các ký tự của một dòng. Với tập các record này, tasktracker sẽ chạy vòng lặp để lấy từng record làm input cho hàm map để trả ra out là dữ liệu gồm intermediate key và value. Dữ liệu output của hàm map sẽ ghi xuống bộ nhớ chính, và chúng sẽ được sắp xếp trước ngay bên trong bộ nhớ chính.

Trước khi ghi xuống local disk, các dữ liệu output này sẽ được phân chia vào các partition (region) dựa vào hàm partition, từng partition này sẽ ứng với dữ liệu input của reduce task sau này. Và ngay bên trong từng partition, dữ liệu sẽ được sắp xếp (sort) tăng dần theo intermediate key, và nếu chương trình client có sử dụng hàm combine thì hàm này sẽ xử lý dữ liệu trên từng partition đã sắp xếp

rồi. Sau khi thực hiện thành công maptask thì dữ liệu output sẽ là các partition được ghi trên local, ngay lúc đó TaskTracker sẽ gửi trạng thái completed của maptask và danh sách các vị trí của các partition output trên localdisk của nó đến JobTracker.

2.3. THƯ VIỆN LUCENE

Apache Lucene là một thư viện mã nguồn mở, được phát triển bởi Dough Cutting. Thư viện này cung cấp các hàm cơ bản hỗ trợ cho việc đánh chỉ mục và tìm kiếm, mục tiêu phát triển nó thành một thư viện tìm kiếm tài liệu hoàn chỉnh, cho phép các nhà phát triển ứng dụng dễ dàng tích hợp chức năng tìm kiếm vào hệ thống của mình.

Lucene là một thư viện tìm kiếm thông tin có khả năng xử lý và khả năng mở rộng ở mức cao, cho phép chúng ta có thể tích hợp vào các ứng dụng. Lucene là một dự án mã nguồn mở và nguyên thuỷ được phát triển bằng ngôn ngữ Java, ngày nay Lucene được phát triển bằng nhiều ngôn ngữ khác nhau như Delphi, Perl, C#, C++, Python, Ruby và PHP…

2.3.1. Tìm kiếm với dữ liệu cấu trúc và phi cấu trúc

Với việc dữ liệu do con người tạo ra ngày càng phong phú, nhu cầu tìm kiếm thông tin ngày càng bức thiết và đa dạng, dẫn đến sự ra đời của rất nhiều công cụ hỗ trợ tìm kiếm.

Trong lĩnh vực tìm kiếm, người ta tạm chia dữ liệu thành hai loại chính, dữ liệu có cấu trúc (structured data) và dữ liệu phi cấu trúc (unstructured data). Dữ liệu có cấu trúc thường dùng để chỉ dữ liệu lưu trữ trong các hệ quản trị cơ sở dữ liệu quan hệ như MS SQL server hay MySQL, trong đó các thực thể và các thuộc tính được định nghĩa sẵn.

2.3.2. Tìm kiếm toàn văn bản

Thư viện Lucene cung cấp các hàm cơ bản hỗ trợ cho việc đánh chỉ mục và tìm kiếm. Để có thể sử dụng Lucene,cần phải có sẵn dữ liệu. Dữ liệu có thể là tập hợp các tập tin dạng PDF, Word hay là các trang web HTML, hoặc là dữ liệu lưu trong các hệ quản trị CSDL như MS SQL Server hay MySQL. Dùng Lucene, bạn có thể tiến hành đánh chỉ mục trước trên dữ liệu hiện có để sau này có thể thực hiện thao tác tìm kiếm, giảm thời gian chờ đợi khi tìm kiếm.

2.4. ĐÁNH CHỈ MỤC ĐỐI TƯỢNG VĂN BẢN

2.4.1. Thành phần tạo chỉ mục

Bao gồm các phần chức năng xử lý tạo chỉ mục, từ văn bản đầu vào để cho ra kết quả là một tập chỉ mục. Lucene chỉ hỗ trợ trên văn bản sau khi được tách nội dung ở dạng ký tự thuần, nó cho phép lập chỉ mục trên từng trường thông tin của văn bản và cho phép thiết lập hệ số cho từng trường thông tin để nâng cao vai trò lúc tìm kiếm.

2.4.2. Thành phần tìm kiếm

Bao gồm các phần chức năng cho xử lý tìm kiếm, từ yêu cầu của người dùng, thông qua biên dịch và so khớp để lấy về kết quả tốt nhất. Lucene hỗ trợ nhiều loại truy vấn thuận tiện cho người sử dụng, nó cho phép tìm theo trường thông tin hay các thiết lập nâng cao như sắp xếp kết quả, giới hạn thời gian hoặc số lượng kết quả, phân trang…

2.5. KẾT CHƯƠNG

Chương này phân tích các thành phần cấu tạo và cài đặt mô hình xử lý phân tán MapReduce, nghiên cứu và đề xuất giải pháp lập chỉ mục, tìm kiếm văn bản bằng thư viện nguồn mở Apache Lucene làm tiền đề để xây dựng hệ thống trong chương tiếp theo.

CHƯƠNG 3
THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

Chương này đưa ra giải thuật so khớp trên mô hình MapReduce và minh họa, xây dựng hệ thống kiểm tra độ tương đồng văn bản sử dụng mô hình MapReduce, chạy thử nghiệm triển khai chương trình, tổng kết kết quả thử nghiệm và đưa ra nhận xét, đánh giá so sánh với các chương trình khác tương tự.

3.1. PHÁT BIỂU BÀI TOÁN

Xây dựng được một chương trình nhằm mục đích kiểm tra độ tương đồng giữa một tài liệu thử và kho dữ liệu lớn, kết quả đưa ra được tài liệu tương đồng với tài liệu cần kiểm tra dựa trên tính toán, xử lý song song thông qua mô hình MapReducenhằm tối ưu về mặt thời gian tìm kiếm.

3.2. XÂY DỰNG MÔ HÌNH

Kho dữ liệu được lập chỉ mục, người dùng đưa tài liệu cần kiểm tra vào thông qua giao diện hệ thống để kiểm tra trọng số, tổng hợp trọng số và trả về kết quả văn bản tương đồng cao nhất.

3.3. GIẢI THUẬT SO KHỚP TRÊN MAPREDUCE

3.3.1. Cách thức xác định độ tương đồng của cặp văn bản

Để xác định độ tương đồng của cặp văn bản, hệ thống xây dựng thông qua 2 bước như sau:

Bước 1: Lập chỉ mục ngược cho từ (Indexing)

Xây dựng một chỉ số đảo ngược tiêu chuẩn (Frakes và Baeza - Yates, 1992), trong đó mỗi từ t được liên kết với một danh sách các *docid* cho tài liệu có chứa nó và trọng số liên quan đến từ t.

Lập bản đồ trên tất cả các tài liệu, tính trọng số cho mỗi từ t trong các tài liệu, với mỗi "thuật ngữ" như là giá trị key trong

mapper, và một bộ bao gồm trọng số docid và t đóng vai trò là value trong hàm mapper chạy MapReduce tự động xử lý các nhóm của các bộ dữ liệu t, tiếp tục qua hàm reduce, do đó tạo ra tập postings(t).

Bước 2: Tính trọng số văn bản trong kho văn bản

Khi văn bản cần kiểm tra được nhập vào sẽ được phân tích ra thành các từ đơn, tiếp theo thành phần tìm kiếm sẽ tìm kiếm từ đó trong các văn bản trong kho văn bản đã lập chỉ mục để lấy trọng số. Cuối cùng các trọng số trong cùng một giá trị key giống nhau sẽ được tính lại để xuất ra kết quả cuối cùng.

3.3.2. Kỹ thuật làm giảm các bộ từ trong các tài liệu

a. *Lọc các từ trùng lặp trong tài liệu dựa vào các hàm trong thư viện Lucene.*

Ví dụ: Trong một câu văn bản:"*Tôi biết tôi có thể bị đau khi cố gắng làm việc đó nên tôi sẽ suy nghĩ thật kỹ trước khi làm*". Từ "tôi" xuất hiện 3 lần trong câu sẽ được lượt bỏ để so sánh lấy trọng số 1 lần trong văn bản cần kiểm tra.Trong quá trình kiểm tra đánh trọng số cho từ thuộc văn bản cần kiểm tra, ta chỉ cần kiểm tra và đánh trọng số cho từ đó 1 lần còn các từ trùng lặp lại các lần tiếp theo sẽ không được đánh trọng số nữa vì việc đánh giá nhiều lần không có nhiều ý nghĩa thực tế đồng thời làm giảm số thao tác cho hệ thống hoạt động nhanh hơn.

b. Đưa các từ về lại dạng nguyên mẫu của từ (từ gốc)

Trong Tiếng Anh, một từ, cụm từ có nhiều thể như tiền tố, hậu tố, thể danh từ, động từ, động từ thể quá khứ…Để tránh cho quá trình lập chỉ mục hiểu lầm là những từ khác nhau làm cho việc lập chỉ mục thêm nhiều hơn, văn bản sẽ được xử lý đưa về dạng nguyên mẫu của từ (từ gốc)

Ví dụ: từ "*harm", "harmful", "harmless",*.....Phải được

chuyển về từ gốc “harm”.

c. Loại bỏ các từ stopword (sử dụng danh sách stopword của Lucene)

Đây là những từ có tần số xuất hiện cao nhưng có trọng số ý nghĩa thấp, không có ý nghĩa nhiều trong quá trình tìm kiếm. Chúng thường đóng vai trò về mặt ngữ pháp trong ngôn ngữ, không chuyển tải được nội dung của tài liệu, ví dụ các mạo từ, giới từ được cắt bỏ để giảm bớt số lượng từ chỉ mục.

Để thực hiện các kỹ thuật loại bỏ, làm giảm các bộ từ và khối lương tính toán cho hệ thống bằng cách sử dụng các gói lệnh sau trong thư viện ApacheLucene

Sử dụng kỹ thuật df-cut loại bỏ 1% các “thuật ngữ” có trọng số cao nhất trong 2 văn bản để rút ngắn thời gian tính toán. Sử dụng mô hình MapReduce tính độ tương đồng cho các thuật ngữ còn lại giữa 2 văn bản.

Việc phân tích so sánh, so khớp các tài liệu là tiền đề để giải quyết một loạt các vấn đề như phân nhóm tài liệu, so sánh các chuỗi ADN trong sinh học, vấn đề bản quyền trong các tác phẩm…. Ví dụ, trong PubMed công cụ tìm kiếm PubMed, 1 công cụ truy cập vào tài liệu khoa học đời sống văn học, tính năng duyệt web được thực hiện như một tra cứu đơn giản của những tài liệu tương tự.

Các kho văn bản, tài liệu với kích thước nhỏ ban đầu phù hợp với bộ nhớ và vi xử lý của một máy tính đơn lẻ, nhưng khi kích thước kho văn bản, tài liệu phát triển gấp nhiều lần, cuối cùng nó trở thành quá tải cho việc lưu trữ trên đĩa khi kích thước kho dữ liệu tăng lên, mà tại đó điểm sắp xếp thứ tự tính toán với truy cập đĩa mẫu trở thành một thách thức. Sự mở rộng dữ liệu hơn nữa trong kho văn bản cuối cùng sẽ làm cho việc xử lý tính toán song song là cần thiết, các

khung công việc MapReduce cung cấp một giải pháp tốt đối với những thách thức này. Ở đây mô tả cách tính các cặp văn bản tương đồng trong kho dữ liệu lớn có thể được thực hiện hiệu quả với MapReduce. Tôi thực nghiệm chứng minh rằng loại bỏ tần số cao (và do đó entropy thấp) về kết quả trong khoảng tuyến tính tăng trưởng trong không gian đĩa yêu cầu và thời gian chạy tăng với kích thước kho dữ liệu, các kho dữ liệu có chứa hàng trăm ngàn tài liệu.

Như vậy một tài liệu, văn bản trước khi được lập chỉ mục hay kiểm tra với kho dữ liệu sẽ trải qua các giai đoạn sau:

- Bước 1: Chuẩn hóa từ (chuyển toàn bộ nội dung văn bản về chữ thường)
- Bước 2: Đưa về nguyên mẫu, từ gốc
- Bước 3: Loại bỏ các từ trùng lặp
- Bước 4: Loại bỏ các từ Stopword

3.3.3. Xử lý phân tán MapReduce

3.3.4. Minh họa giải thuật so trùng

3.4. PHÂN TÍCH VÀ THIẾT KẾ CÁC CHỨC NĂNG HỆ THỐNG

3.4.1. Lập chỉ mục cho kho dữ liệu

Người dùng đưa vào đường dẫn (path) cho thư mục cần lập chỉ mục (thư mục có thể chứa các văn bản thuộc nhiều dạng khác nhau như *.doc, *.xls, *.pdf….) Tiến hành lập chỉ mục trước nhằm "chuẩn bị" cho việc đánh giá độ tương đồng văn bản cho văn bản cụ thể.

Nguồn dữ liệu đầu vào với kích thước lớn được chia nhỏ ra để xử lý theo mô hình lập trình MapReduce thành n tiến trình với mỗi tiến trình có kích thước lớn nhất là 64M. Với mỗi tiến trình thực hiện quá trình lập chỉ mục bằng thư viện Lucene nhằm giảm thời gian cho quá trình lập chỉ mục của ứng dụng. Quá trình gọi là quá trình

"Mapping" hình 3.4.

Sau đó kết quả của mỗi quá trình trên được tập hợp lại để xuất ra thành 1 file lập chỉ mục cho thư mục đó, quá trình này ứng với công đoạn "Reducing" hình 3.4 của mô hình MapReduce.

Kết quả của chức năng này ta thu được tập tin lập chỉ mục ngược của thư mục bằng thư viện Lucene áp dụng quá trình MapReduce để giảm tải thời gian thực cho ứng dụng.

Thành phần tạo chỉ mục: bao gồm các chức năng chính như chỉ định dữ liệu lập chỉ mục, thực hiện phân tích tài liệu, tạo chỉ mục và lưu trữ xuống tập chỉ mục, cập nhật chỉ mục trong trường hợp bổ sung hay thay đổi, theo dõi và quản trị nội dung chỉ mục… Thành phần này dựa trên những lớp có sẵn của Lucene, được phát triển một số lớp văn bản thuộc các lĩnh vực cho việc tìm kiếm tập trung và độ chính xác cao hơn.

3.4.2. Xuất ra file trọng số

Người dùng nhập vào đường dẫn (path) hoặc nội dung của file văn bản cần kiểm tra độ tương đồng (có thể là file *.doc, *.xls, *.pdf….).

Nhập vào đường dẫn (path) của "kho dữ liệu" (big data) có nội dung tương ứng theo các lĩnh vực cần kiểm tra.

Ví dụ: khoa học, văn học, tin học, công nghệ,…

Thành phần tìm kiếm: bao gồm các chức năng chính như nhận thông tin truy vấn, biên dịch và tìm kiếm, trình bày kết quả và liên kết đến tài liệu gốc… trong đó các lớp biên dịch truy vấn và tìm kiếm dựa trên các thành phần có sẵn của Lucene.

Do đặc trưng của các văn bản lưu trữ là được thu thập từ nhiều đơn vị, nhiều nguồn khác nhau và do nhiều người dùng tạo ra, nên tồn tại rất nhiều loại tệp tin văn bản. Về cơ bản, Lucene chỉ hỗ trợ

cho văn bản Text và ngôn ngữ tiếng Anh, do vậy số lượng văn bản lập chỉ mục được là rất ít (chỉ có tệp tin dạng *.txt), nên hệ thống được xây dựng không dùng cơ sở dữ liệu vì hạn chế đối với dữ liệu phi cấu trúc và loại tập tin mà Lucene có thể lập chỉ mục là đa dạng hơn(*.doc, *.xls, *.txt, *.pdf …)

3.4.3. Kết quả kiểm tra

Thực hiện chức năng đánh giá độ tương đồng. Xuất ra tên các file dưới bảng kết quả mà chương trình thực hiện và đánh giá là có độ tương đồng với tài liệu cần kiểm tra, tô nền cho tên văn bản được hệ thống đánh giá độ tương đồng cao nhất.

Trên cơ sở tạo chỉ mục của Lucene, hệ thống còn xuất ra thành các file chỉ mục. Một nội dung văn bản cần kiểm tra khi đưa vào hệ thống xử lý sẽ được tách từ và kiểm tra với file chỉ mục có sẵn này bằng mô hình MapReduce nên cải thiện tốc độ cho kết quả rất nhiều nhờ có sự "chuẩn bị" sẵn và mô hình xử lý phân tán.

3.5. TRIỂN KHAI XÂY DỰNG HỆ THỐNG

3.5.1. Chức năng lập chỉ mục văn bản

3.5.2. Chức năng đánh trọng số cho từ trong văn bản

3.5.3. Chức năng xuất ra kết quả

3.6. KẾT QUẢ TRIỂN KHAI THỰC NGHIỆM

3.6.1. Mô hình triển khai

3.6.2. Các bước xây dựng kho dữ liệu

3.6.3. Kịch bản 1: Tạo văn bản mẫu để so khớp giữa văn bản mẫu và kho dữ liệu

3.6.4. Kịch bản 2: So khớp với kho dữ liệu Đồ án tốt nghiệp Khoa CNTT Đại học Bách Khoa Đà Nẵng

3.6.5. Kịch bản 3: So khớp với các trang tin trên Internet

3.6.6. Nhận xét đánh giá

Luận văn thực hiện so sánh kết quả đánh giá độ tương đồng giữa văn bản và kho cơ sở dữ liệu bằng cách sử dụng:

- Mô hình ứng dụng thư viện Lucene trên máy đơn
- Mô hình MapReduce đã đề xuất trong luận văn

Kết quả cho thấy việc ứng dụng mô hình MapReduce cho tốc độ tốt hơn, xử lý được nhiều dữ liệu trong thời gian ngắn hơn.

Đối với việc xử lý tìm kiếm văn bản có nội dung lớn cho kết quả chính xác hơn vì tính thêm trọng số của những từ ít thông dụng, nằm trong một số ít văn bản. Việc xử lý văn bản Tiếng Việt chưa linh hoạt vì đặc thù riêng của ngôn ngữ gồm các từ ghép, từ láy, từ địa phương, từ đồng âm,… Nhưng khi văn bản đủ lớn thì kết quả trả về đảm bảo độ chính xác. Thời gian thực hiện nhanh hơn nhiều so với kết quả trên một máy tính đơn.

3.7. KẾT CHƯƠNG

Việc tìm kiếm các tài liệu tương đồng là một công việc khá thông dụng trong học tập, nghiên cứu hay trong công tác kiểm tra sự trùng lắp giữa các tài liệu mang tính khoa học, nghiên cứu… Phần này trình bày kết quả xây dựng hệ thống kiểm tra độ tương đồng văn bản sử dụng mô hình MapReduce. Các kết quả thực nghiệm chứng tỏ được ưu điểm của mô hình đề xuất.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả đạt được

Luận văn đã nghiên cứu và ứng dụng mô hình lập trình phân tán MapReduce trên nền tảng Hadoop và thư viện mã nguồn mở ApacheLuceneđể xây dựng hệ thống cho phép kiểm tra độ tương

đồng giữa một văn bản kiểm tra với các văn bản còn lại trong kho tài liệu lớn nhằm đưa ra độ tương đồng giữa các tài liệu này.

Về lý thuyết, luận văn đã đạt được các kết quả có thể như sau:

- Tìm hiểu về mô hình xử lý song song hiệu quả và là xu hướng trong thời đại hiện nay khi tiếp cận xử lý dữ liệu khổng lồ MapReduce.

- Tìm hiểu và ứng dụng thư viện mã nguồn mởApacheLucene, Apache Tika hỗ trợ xử lý, lập chỉ mục nội dung văn bản.

- Xây dựng hệ thống chương trình dựa trên mô hình MapReduceđể kiểm tra độ tương đồng văn bản.

Về ứng dụng, luận văn đã xây dựng được hệ thống kiểm tra độ tương đồng văn bản sử dụng mô hình MapReduce trên cơ sở lý thuyết đã nghiên cứu. Kết quả cũng đã đánh giá hiệu năng của hệ thống và so sánh với các hệ thống khác.

2. Hướng phát triển

- Tối ưu hơn khả năng kiểm tra và đánh trọng số cho văn bản tiếng việt vì một số khác biệt về ngôn ngữ trong tiếng Việt và tiếng Anh như từ ghép, từ đồng nghĩa, từ địa phương…

- Nâng cao hiệu quả trong quá trình lập chỉ mục, giảm thiểu tối đa thời gian chờ đợi của hệ thống,

Kết quả đạt được của luận văn về lý thuyết cũng như chương trình ứng dụng là nền tảng cơ bản để bản thân có thể phát triển hệ thống so sánh văn bản cho nhiều loại dữ liệu như hình ảnh…,theo hướng mở rộng của Lucene như: tìm kiếm theo ngữ nghĩa, tìm kiếm từ xa qua hệ thống tập tinphân tán của Hadoop.