XỬ LÝ DỮ LIỆU BÁN CẤU TRÚC TRÊN MÔI TRƯỜNG HADOOP

Nguyễn Mậu Hân

Khoa Công nghệ Thông tin, Trường Đại học Khoa học, Đại học Huế

Email: nmhan@hueuni.edu.vn

Ngày nhận bài: 01/12/2022; ngày hoàn thành phản biện: 26/12/2022; ngày duyệt đăng: 26/12/2022

TÓM TẮT

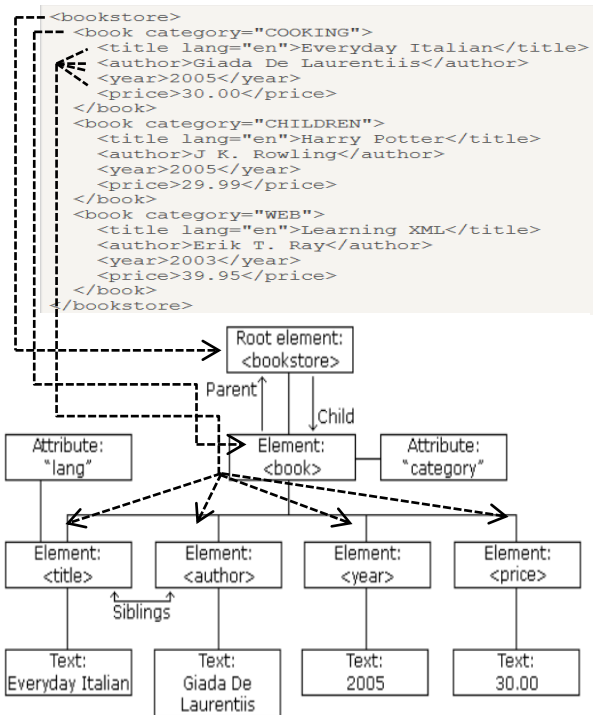
Dữ liệu bán cấu trúc thường được biểu diễn ở định dạng XML. Mặc dù hiện nay đã có một số hệ quản trị có khả năng tổ chức, lưu trữ và xử lý dữ liệu bán cấu trúc một cách có hiệu quả nhưng đối với các tập dữ liệu có kích thước lớn thì các hệ thống này lại bộc lộ nhiều hạn chế. Khi bộ nhớ thứ cấp được sử dụng để lưu trữ một lượng lớn dữ liệu nếu không có chỉ mục phù hợp nào được sử dụng thì thời gian thực hiện của các truy vấn phức tạp sẽ tăng lên đáng kể. Hiện nay, Hadoop với công cụ MapReduce đã thành công trong việc xử lý loại dữ liệu có cấu trúc với kích thước tập tin khá lớn và thời gian xử lý nhanh. Tuy nhiên, đối với dữ liệu bán cấu trúc như XML thì ít được bàn đến. Trong bài báo này chúng tôi đề xuất mô hình xử lý dữ liệu bán cấu trúc trên nền tảng Hadoop với công cụ MapReduce bằng cách chuyển tài liệu XML thành cấu trúc phân cấp XML Tree.

Từ khóa: Hadoop, MapReduce, dữ liệu bán cấu trúc, XML

1. MỞ ĐẦU

Dữ liệu có cấu trúc được mô tả trên một lược đồ xác định, cho phép mô tả rõ ràng thông tin về mỗi đối tượng nào đó trong mô hình dữ liệu quan hệ. Tuy nhiên, dữ liệu trên môi trường quan hệ lại tỏ ra không mấy thuận lợi khi được sử dụng để trao đổi thông tin giữa các hệ thống trong môi trường web. Trong trường hợp này dữ liệu XML được sử dụng như là cách để loại bỏ các hạn chế này của dữ liệu quan hệ. XML là một định dạng bán cấu trúc phân cấp được biểu diễn dưới cấu trúc cây phân cấp. Nghĩa là, chúng ta luôn có thể chuyển một tài liệu XML bất kỳ thành một cây phân cấp (XML Tree) gồm 7 loại node khác nhau với thẻ gốc của tài liệu XML là nút gốc của cây phân cấp XML Tree. Các trình phân tích XML (XML Parser) [1] có thể di chuyển trên các nút của XML Tree để đọc và truy xuất thông tin. Một lợi ích quan trọng của XML là nó không nhất thiết phải cần một lược đồ xác định để lưu trữ thông tin trong một cơ sở dữ liệu theo mô hình quan hệ. Các tài liệu XML là những tập tin văn bản với định dạng plain text được tạo ra không phải với mục đích để hiển thị (display) dữ liệu như

HTML, mà chức năng chính của nó là chuyển tải (transport) và lưu trữ (store) dữ liệu nhưng cũng không loại trừ trường hợp nếu cần thì vẫn có thể hiển thị dữ liệu được. XML cho phép nhập dữ liệu với các nội dung hỗn hợp. Chẳng hạn, có thể trộn thêm thông tin bất kỳ vào trong một đoạn văn bản trong một tài liệu XML nào đó. XML cho phép chia sẻ và sử dụng thông tin phân tán trên các hệ thống khác nhau và hỗ trợ người dùng thông qua khả năng tạo nội dung động, phát triển ứng dụng và tích hợp trên nhiều qui mô khác nhau. XML được sử dụng để tạo cấu trúc dữ liệu: dữ liệu này có thể là bảng công tác, sổ địa chỉ, các tham số cấu hình, giao dịch tài chính, vẽ kỹ thuật, ... XML là một hệ thống bao gồm các luật dùng để thiết kế các format cho văn bản, giúp tạo cấu trúc cho dữ liệu [1]. XML không phải là ngôn ngữ lập trình và người học cũng không cần phải là một lập trình viên để có thể học và sử dụng nó thành thạo. Nó giúp cho các ứng dụng dễ dàng tạo dữ liệu, đọc dữ liệu và làm cho cấu trúc dữ liệu trở nên rõ ràng dễ hiểu. Một thuận lợi của khuôn dạng văn bản là cho phép người ta, nếu cần thiết, có thể xem dữ liệu mà không cần phải có chương trình đã tạo ra dữ liệu đó. Nói cách khác, chúng ta có thể đọc nó với bất kì bộ soạn thảo văn bản nào. Các khuôn dạng văn bản cũng cho phép tìm lỗi dễ dàng hơn trong các ứng dụng. Do đó, không thể nói rằng thông tin bán cấu trúc yếu hơn thông tin có cấu trúc, nhưng thông tin bán cấu trúc có một định nghĩa dữ liệu phức tạp hơn so với trong các dữ liệu có cấu trúc với các quan hệ cổ điển.



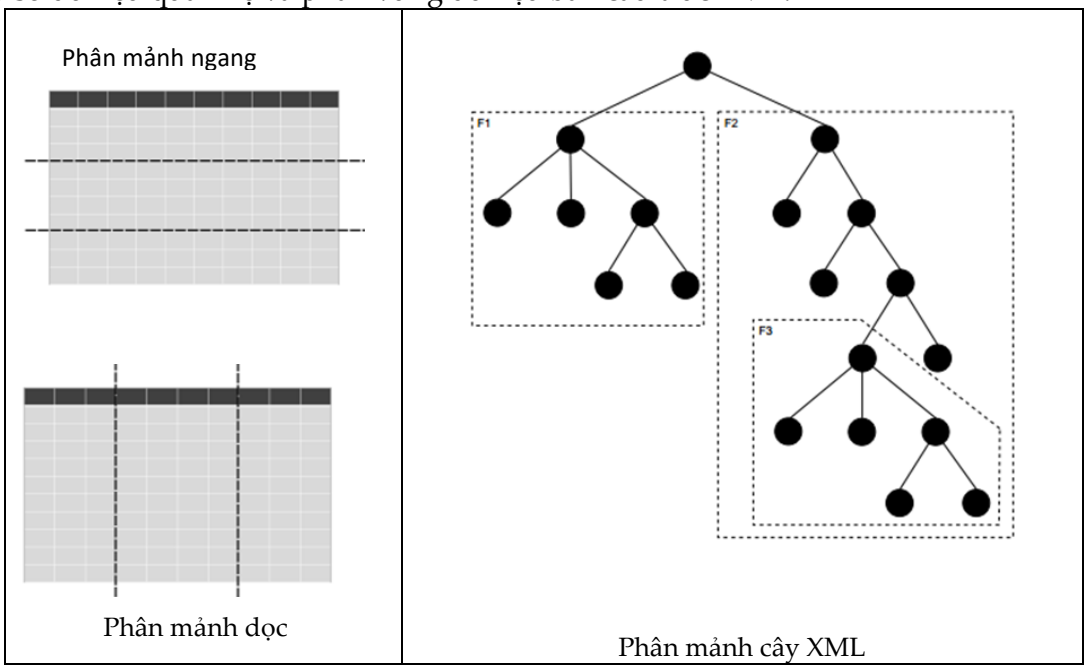
Hình 1.1: Một tài liệu XML bất kỳ được chuyển thành một cây phân cấp XML Tree

2. PHÂN MẢNH DỮ LIỆU TRONG MÔI TRƯỜNG PHÂN TÁN

Trong cơ sở dữ liệu quan hệ, việc phân rã một quan hệ thành nhiều mảnh, mỗi mảnh được xử lý như một đơn vị, sẽ cho phép thực hiện nhiều giao dịch đồng thời. Mặt khác, khung nhìn hoặc đơn vị truy xuất của các ứng dụng thường không phải là toàn bộ quan hệ mà là một phần quan hệ chứa những thông tin cần thiết. Ngoài ra, việc phân mảnh các quan hệ sẽ cho phép thực hiện song song một câu truy vấn bằng cách chia nó ra thành một tập các câu vấn tin con hoạt tác trên các mảnh [2]. Vì thế việc

phân mảnh sẽ làm tăng mức độ hoạt động đồng thời và như thế làm tăng lưu lượng hoạt động của hệ thống. Thể hiện của các quan hệ là các bảng, vì thế vấn đề đặt ra là tìm cách để chia một bảng thành nhiều bảng nhỏ hơn thỏa một số ràng buộc cho trước, điều này được gọi là phân mảnh quan hệ. Có hai cách phân mảnh một quan hệ: phân mảnh ngang (horizontal fragmentation) và phân mảnh dọc (vertical fragmentation). Phân mảnh ngang được thực hiện bằng một phép toán chọn trên các quan hệ của một lược đồ CSDL. Còn phân mảnh dọc được thực hiện bằng một phép chọn các thuộc tính của một quan hệ. Tuy nhiên phân mảnh ngang được sử dụng phổ biến hơn. Ngoài ra trong thực tế, người ta còn sử dụng phân mảnh hỗn hợp (hibrid fragmentation) tức là kết hợp cả phân mảnh ngang và phân mảnh dọc [2]. Tất nhiên quá trình phân mảnh phải được gắn liền với vấn đề cấp phát dữ liệu và bài toán cụ thể như thế nào. Vì vậy, vấn đề chúng ta bàn luận dưới đây chỉ mang tính chất minh họa cho việc phân mảnh quan hệ.

Đối với dữ liệu bán cấu trúc như XML thì việc phân mảnh phức tạp hơn do nó khó phân mảnh hơn. Cấu trúc XML không phải là thông thường theo mặc định mà tùy thuộc vào thông tin mang theo, một cây XML có thể có cấu trúc hướng văn bản, có thể tập trung vào một tài liệu nào đó cũng có thể là một định hướng dữ liệu có cấu trúc hoặc một tập tin XML chỉ tập trung vào dữ liệu. Do đó, việc phân mảnh XML phải sử dụng thuật toán phân vùng tùy thuộc vào cấu trúc của cây. Hình 2.1, mô tả phân vùng cơ sở dữ liệu quan hệ và phân vùng dữ liệu bán cấu trúc XML.



Hình 2.1. Phân mảnh dữ liệu có cấu trúc và dữ liệu bán cấu trúc

2.1 Giải pháp Hadoop/MapReduce

Nền tảng Hadoop cùng với mô hình MapReduce đã áp dụng thành công trong việc xử lý, truy vấn các file dữ liệu quan hệ có kích thươc lớn với hàng trăm terabytes (như BigData). Trong khi đó, cùng với file dữ liệu này SQL Server lại tỏ ra khá hạn chế về khả năng và tốc độ xử lý. Việc xử lý dữ liệu bán cấu trúc và phi cấu trúc trên môi trường Hadoop hiện nay ít được bàn tới, mặc dù một số hệ quản trị CSDL cũng có đề cập đến vấn đề này. Do mô hình dữ liệu quan hệ không thể lưu trữ các loại dữ liệu bán cấu trúc và không cấu trúc nên việc xử lý dữ liệu loại này trong Hadoop/MapReduce chủ yếu được thể hiện ở phần NoSQL (cơ sở dữ liệu theo cột, cặp khóa-giá trị) [4].

2.1.1 Hadoop

Apache Hadoop [4] là một framework dùng để chạy những ứng dụng trên một cluster lớn được xây dựng trên những phần cứng thông thường. Thư viện phần mềm Hadoop là một khuôn mẫu cho phép xử lý phân tán các bộ dữ liệu lớn trên các nhóm máy tính sử dụng các mô hình lập trình đơn giản. Nó được thiết kế để mở rộng từ một máy chủ duy nhất sang hàng ngàn máy khác, mỗi máy cung cấp tính toán và lưu trữ cục bộ. Hadoop thực hiện mô hình MapReduce, đây là mô hình phân tán song song, ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn dữ liệu khác nhau (phân tán), và các phần này sẽ được thực hiện trên đồng thời trên nhiều node khác nhau (song song). Bên cạnh đó, Hadoop cung cấp một hệ thống file phân tán (HDFS) cho phép lưu trữ dữ liệu lên trên nhiều node. Cả Map/Reduce và HDFS đều được thiết kế sao cho framework sẽ tự động quản lý được các lỗi, các hư hỏng về phần cứng của các node.

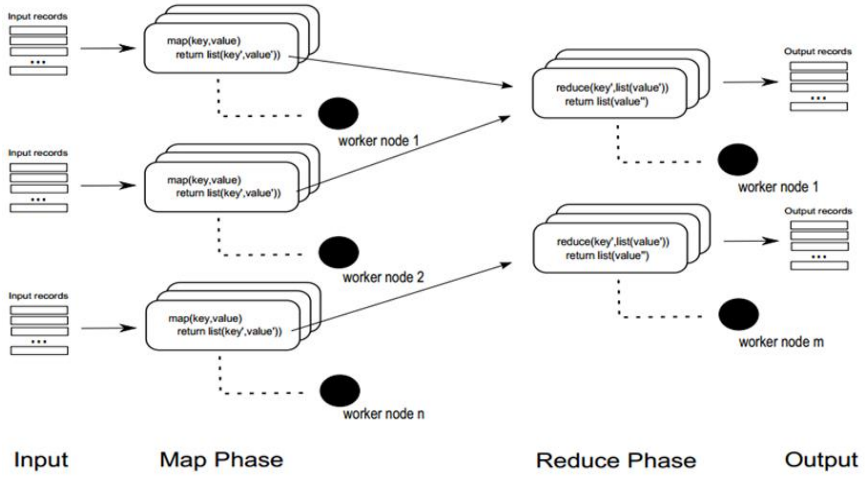
2.1.2 MapReduce

Có thể hiểu một cách đơn giản, MapReduce phân chia các công việc xử lý thành nhiều khối công việc nhỏ, phân tán khắp các nút tính toán (giai đoạn Map), rồi thu hồi các kết quả (giai đoạn Reduce). MapReduce có thể chạy trên các phần cứng thông thường, không đòi hỏi các server chạy MapReduce phải là những máy tính có cấu hình cao với khả năng tính toán, lưu trữ và truy xuất mạnh mẽ. Do đó, chi phí triển khai MapReduce sẽ rẻ hơn các mô hình tương tự khác. MapReduce làm đơn giản hoá các giải thuật tính toán phân tán bằng cách chỉ cần cung cấp hai hàm Map và Reduce cùng với một số thành phần xử lý dữ liệu đầu vào theo yêu cầu [4].

a. **Hàm Map**: Nhận dữ liệu đầu vào cặp dữ liệu (key, value) làm input cho hàm Map, và tùy vào mục đích của người dùng mà hàm Map sẽ trả về danh sách các cặp dữ liệu trung gian (*intermediate key*, *value*). Dữ liệu này sẽ là input cho hàm Reduce [4].

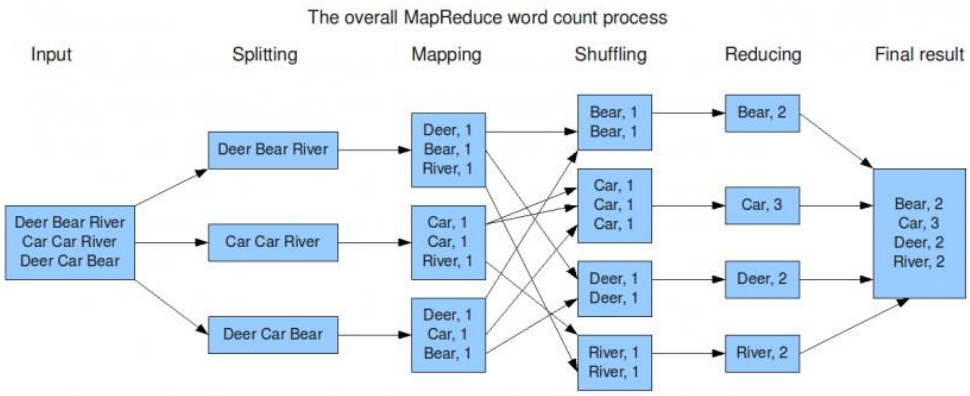
b. **Hàm Reduce**: Hệ thống sẽ gom nhóm tất cả *value* theo *intermediate key* từ các *output* của hàm Map, để tạo thành tập các cặp dữ liệu với cấu trúc là (*key*, *tập các value cùng key*). Dữ liệu *input* của hàm Reduce là từng cặp dữ liệu được gom nhóm ở trên và sau khi thực hiện xử lý nó sẽ trả ra cặp dữ liệu (*key*, *value*) *output* cuối cùng cho người

dùng. Cho đến nay, Hadoop đã trở thành giải pháp nguồn mở hàng đầu hỗ trợ mô hình MapReduce. Hadoop được viết bằng Java, tuy nhiên hỗ trợ phát triển MapReduce trên nhiều ngôn ngữ khác ngoài Java như C++, Pearl, Python, …



Hình 2.2. Kiến trúc của mô hình MapReduce

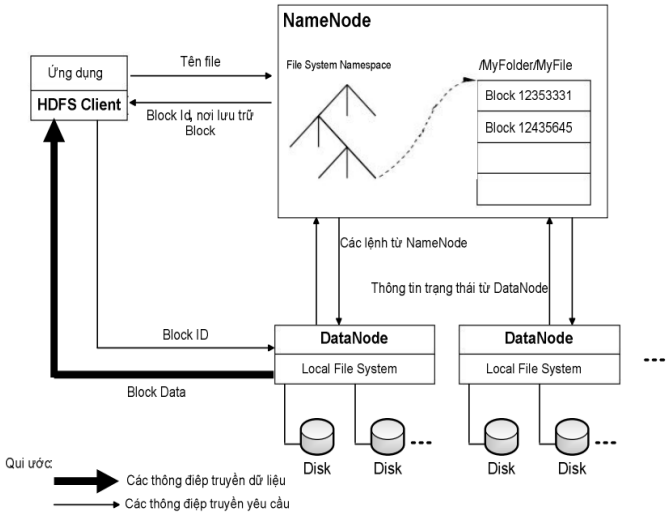
c. Một ví dụ minh họa Đếm các từ giống nhau trong một xâu cho trước theo mô hình MapReduce:



Hình 2.3: Đếm các từ giống nhau trong một xâu cho trước theo mô hình MapReduce

2.1.3 Kiến trúc Hadoop File System (HDFS)

Giống như các hệ thống file khác, HDFS duy trì một cấu trúc cây phân cấp các file [5], thư mục mà các file sẽ đóng vai trò là các node lá. Trong HDFS, mỗi file sẽ được chia ra làm một hay nhiều block và mỗi block này sẽ có một block ID để nhận diện. Mỗi block của file sẽ được lưu trữ thành ra nhiều bản sao khác nhau vì mục đích an toàn dữ liệu.



Hình 2.3. Kiến trúc của HDFS

2.2 XPath và XQuery

2.2.1 XPath

Để rút trích thông tin từ một tài liệu XML các chương trình ứng dụng (gọi chung là XML Parser) phải có cách di chuyển bên trong tài liệu đó, lấy ra giá trị của thẻ và thuộc tính theo yêu cầu. XML Path Language, gọi tắt là XPath, là một ngôn ngữ được thiết kế với mục đích giúp cho các ứng dụng có thể di chuyển trên các nút bên trong của một cây XML và truy xuất giá trị trên các nút của cây này. XPath đóng một vai trò quan trọng trong việc trao đổi dữ liệu giữa các máy tính hay giữa các chương trình ứng dụng vì nó cho phép lựa chọn hay sàng lọc ra những thông tin cần thiết để trao đổi hay hiển thị. XPath xem mỗi tài liệu XML lả một cây với thẻ gốc là nút gốc cùng với 7 loại nút khác nhau, đó là: element, attribute, text, namespace, processing-instruction, comment, and document (root). Chúng ta có thể sử dụng biểu thức đường dẫn của XPath để chỉ định đường đi đến vị trí một nút nào trên XML Tree hoặc trả về một hay nhiều nút thỏa mãn điều kiện yêu cầu.

2.2.2 XQuery

XQuery là một ngôn ngữ để truy vấn dữ liệu XML. Có thể nói rằng, XQuery truy vấn dữ liệu XML giống như SQL truy vấn dữ liệu quan hệ. XQuery được xây dựng trên biểu thức đường dẫn XPath và được hầu hết các hệ thống cơ sở dữ liệu hỗ trợ. XQuery cho phép sử dụng các biểu thức đường dẫn XPath để trích xuất dữ liệu từ tài liệu XML. Dữ liệu đó có thể là một giá trị hoặc là một cấu trúc cây con của cây XML gốc. XQuery sử dụng các biểu thức đường dẫn XPath, liên quan đến các từ khóa FLWOR (For, Let, Where, Order by và Return), để tách và lấy dữ liệu từ cây XML trong hầu hết các trường hợp [1]. Cấu trúc ngữ pháp của ngôn ngữ XQuery dựa trên cấu trúc cây của chính tài liệu XML. Ngoài ra, XQuery có thể được sử dụng để trích lọc thông

tin để sử dụng trong các Web Service, tạo ra các báo cáo tóm tắt, chuyển đổi một tài liệu XML thành một tài liệu HTML và tìm kiếm tài liệu Web cho thông tin liên quan.

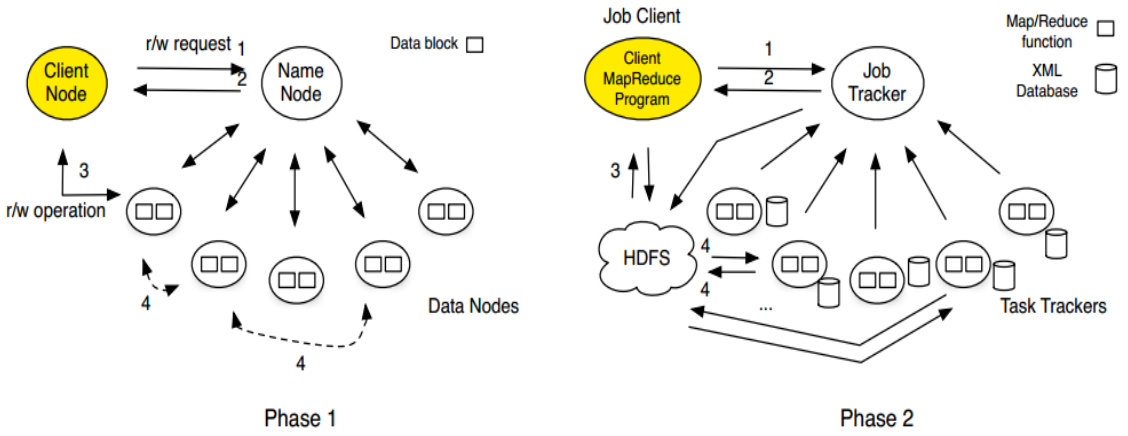
XPath và XQuery giống nhau về một số mặt. Thậm chí XPath là một phần trọn vẹn của XQuery. Cả hai ngôn ngữ cho phép chọn những mẫu dữ liệu từ một tài liệu XML hoặc một kho lưu trữ tài liệu XML. Trong truy vấn dữ liệu XPath mang lại sự đơn giản còn XQuery cung cấp thêm sức mạnh và tính linh hoạt.

3. TRUY VẤN DỮ LIỆU XML TRÊN NỀN TẢNG HADOOP VÀ MAPREDUCE

3.1 Hadoop và XML

Có ba phương pháp chính trong môi trường phân tán để thực hiện việc phân phối dữ liệu đến các nút của hệ thống mạng: *thứ nhất*, cách tiếp cận tập trung với nút chính chịu trách nhiệm điều phối các nút con của mạng; *thứ hai*, cách tiếp cận phi tập trung, trong trường hợp này tất cả các nút của mạng đều bình đẳng như nhau và chịu trách nhiệm chuyển tiếp các yêu cầu của hệ thống; và cách tiếp cận *thứ ba*, kết hợp với hai phương pháp trước, nghĩa là, trong hệ thống có nhiều mạng con, mỗi mạng con được điều phối bởi một nút chính; các nút chính này được kết nối với một mạng peer to-peer. Mỗi cách tiếp cận đều có điểm mạnh và điểm yếu riêng, tùy thuộc vào kiến trúc mạng hiện có để sử dụng phương pháp nào cho thích hợp.

Ở đây Hadoop MapReduce bao gồm hệ thống tập tin phân tán (HDFS) chịu trách nhiệm phân phối dữ liệu thông qua một nút chính đến các nút dữ liệu. Nút chính này chịu trách nhiệm điều phối các yêu cầu và quản lý tài nguyên trên mỗi nút dữ liệu. Hơn nữa, nút chính không chịu trách nhiệm về chuyển dữ liệu đến các nút dữ liệu từ một máy khách. Nhiệm vụ của nó là xác định vị trí các nút dữ liệu có trách nhiệm và cung cấp vị trí nút dữ liệu cho chương trình khách HDFS. Sau đó chương trình khách sẽ phân phối dữ liệu đến cho các nút dữ liệu. Như vậy, nút chính chỉ chịu trách nhiệm điều phối các yêu cầu, quản lý tính khả dụng của bộ nhớ và để lưu trữ ảnh chụp nhanh trong trường hợp lỗi chính. Trong trường hợp nút chính gặp sự cố thì một nút chính mới sẽ được khởi tạo với dữ liệu từ các ảnh chụp nhanh cuối cùng. Theo mặc định, dữ liệu phân tán được tổ chức theo kích thước khối là 64 MB. MapReduce chịu trách nhiệm thực hiện tính toán song song trên một số nút dữ liệu có chứa dữ liệu được phân phối. Hàm *map* nhận các bản ghi từ nút dữ liệu dưới dạng các cặp khóa/giá trị và kết quả tính toán được viết dưới dạng kết quả trung gian cho đầu ra và được phân phối và nhân rộng thông qua HDFS. Hàm *reduce* nhận tất cả các kết quả trung gian được liên kết với một khóa và thực hiện tổng hợp kết quả, được ghi vào đầu ra cuối cùng đồng thời được phân tán và nhân bản như mong muốn. Hình ảnh dưới đây mô tả quá trình bàn luận ở trên.

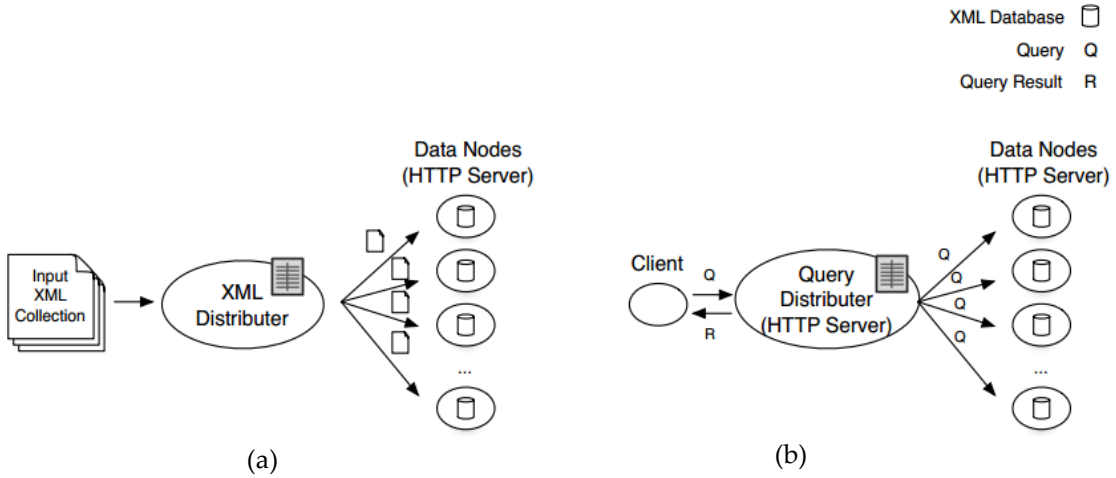


***Hình 2.4*:** Phase 1: Các tệp đầu vào XML, qua HDFS, sẽ được phân tán đến các nút dữ liệu. Phase 2: Gộp các tệp XML được phân tán vào CSDL XML.

Trong phase 1, các tập tin đầu vào XML sẽ được phân tán đến các nút dữ liệu. NameNode chịu trách nhiệm định vị các khối rảnh rỗi trên các nút dữ liệu và gửi các vị trí khối đến nút khách hàng, được gắn nhãn bằng các mũi tên 1 và 2. Sau đó, Client Node sẽ ghi đầu vào của nó đến NameNode tương ứng, được mô tả như mũi tên 3. Cuối cùng, khối được ghi sẽ được nhân bản thông qua Data Node, xem mũi tên 4.

Trong phase 2, MapReduce được khởi động cho mỗi tệp đầu vào. Mỗi Client Node gửi yêu cầu MapReduce của mình đến bộ theo dõi công việc Job Tracker, xem mũi tên 1, Job Tracker định vị vị trí dữ liệu trong HDFS, nơi mà kết quả của hàm *map* và *reduce* được lưu trữ.

Ở đây chúng ta tập trung vào truy vấn dữ liệu bán cấu trúc XML, đặc biệt với mỗi tập tin XML trong một cơ sở dữ liệu XML Việc phân mảnh một số lượng lớn các tập tin XML bao gồm các tập tin XML có kích thước nhỏ được thực hiện tương tự như việc phân mảnh ngang trong cơ sở dữ liệu quan hệ. Vì vậy các kỹ thuật phân mảnh quan hệ đều được áp dụng cho trường hợp này. Hình 2.5 ở dưới cho chúng ta thấy rằng việc phân tán tập các tập tin XML đến các Data Nodes được thực hiện bởi một Master Node (hình a). Sau đó, tại mỗi Client Node bộ phận truy vấn phân tán (Query Distributer) sẽ thực hiện tìm kiếm thông tin theo yêu cầu và lưu trữ trên các Data Node. tương ứng (hình b).



Hình 2.5. a) Phân tán các tệp XML qua Master Node
b) Truy vấn dữ liệu trên các tệp XML con

4. KẾT LUẬN

Việc xử lý dữ liệu có cấu trúc trên nền tảng Hadoop MapReduce đã được ứng dụng trong thực tế trên các tập tin dữ liệu lớn (BigData). Đối với dữ liệu bán cấu trúc như các tài liệu XML thì chúng ta lại gặp rắc rối về vấn đề phân mảnh dữ liệu. Tuy nhiên, nếu chuyển một tài liệu XML sang một cây phân cấp XML thì việc phân mảnh quan hệ có thể thực hiện tương đương với việc phân cây XML thành các cây con và các nhánh để các trình phân tích XML hoạt tác trên các cây con và nhánh này.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Mậu Hân (2019), XML và ứng dụng, NXB Đại học Huế.

[2] Nguyễn Mậu Hân (2012), Cơ sở dữ liệu phân tán, NXB Đại học Huế.

[3] Paul Zikopoulos, Chris Eaton, Tom Deutsch, Dirk Deroos, George Lapis (2018), "Understanding Big Data Analytics for Enterprise Class Hadoop and Streaming Data", McGraw-Hill.

[4] Tom White (2019), "Hadoop: The Definitive Guide", O'Reilly Media, Inc, 3rd Edition.

[5] Jeffrey Dean and Sanjay Ghemawat (2010), "MapReduce: Simplified Data Processing on Large Clusters", Goolge Inc.

[6] https://www.semanticscholar.org/paper/Understanding-big-data-Dessureault/31931aa8d814d45f9cfcd1765aff07e443cca99d

[7] https://thenextweb.com/news/amazons-cto-here-two-definitions-big-data

[8] Big Data IBM, (2014): http://www-01ibmcom/software/data/bigdata/

[9] https://hypecycles.files.wordpress.com/2011/10/nosql-database-498041.pdf

PROCESSING SEMI-STRUCTURED DATA ON HADOOP ENVIRONMENT

Nguyen Mau Han

Faculty of Information Technology, University of Sciences, Hue University

Email: nmhan@hueuni.edu.vn

ABSTRACT

Semi-structured data is usually represented in XML format. Existing database management systems are capable of effectively organizing, storing, and processing semi-structured data, but their limits become apparent when applied to enormous data volumes. When vast volumes of data are stored in secondary memory without an appropriate index, the execution time of complex queries will increase dramatically. Currently, Hadoop with the MapReduce model effectively manages structured data with enormous file sizes and fast processing time. However, semi-structured data such as XML is rarely discussed. In this paper, we propose a semi-structured data processing model on the Hadoop platform using the MapReduce tool. In addition, we propose a semi-structured data processing model on the Hadoop platform with the MapReduce tool by converting XML documents into XML Tree hierarchies.

Keywords: Hadoop, MapReduce, semi-structured data, XML.

Nguyễn Mậu Hân sinh năm 1957 tại Thừa Thiên Huế. Ông tốt nghiệp cử nhân ngành Toán lý thuyết năm 1981 và thạc sĩ chuyên ngành Khoa học máy tính năm 1998. Ông nhận bằng tiến sĩ tại viện Công nghệ thông tin, Hà Nội năm 2003 và được phong hàm Phó Giáo sư năm 2013. Từ năm 1994 đến nay, ông là giảng viên tại khoa Công nghệ thông tin, Trường Đại học Khoa học, Đại học Huế.

Lĩnh vực nghiên cứu: Xử lý song song và phân tán, tính toán lưới và điện toán đám mây.