Tạp chí Khoa học Trường Đại học Cần Thơ
website: sj.ctu.edu.vn

ĐIỆN TOÁN ĐÁM MÂY VÀ BÀI TOÁN XỬ LÍ DỮ LIỆU LỚN THEO MÔ HÌNH ÁNH XẠ - RÚT GỌN

Trần Cao Đệ[1]

[1] *Khoa Công nghệ Thông tin & Truyền thông, Trường Đại học Cần Thơ*

Thông tin chung:
Ngày nhận: 06/03/2013
Ngày chấp nhận: 19/08/2013

Title:
Cloud computing and big data processing problem with map - reduce model

Từ khóa:
Điện toán đám mây, dữ liệu lớn, ánh xạ - rút gọn, Hadoop

Keywords:
Cloud computing, big data, map - reduce, Hadoop

ABSTRACT

Today, we are living in the information age in which the information is explosive growth in exponent rate. Some leading company in information technology as Google, Yahoo, Amazon, Microsoft, Facebook, Twitter, etc. have challenged with huge amount of data. This growth has demanded new strategies for processing and analyzing data. Cloud computing has been developed and Hadoop-MapReduce has become a powerful computation model addressing these problems. This model provides a programming framework for text processing applications that have ability to process quickly a large amount of data due to the parallel in a large computer cluster. This article provides an overview of large data processing problem on cloud computing platforms, such as architecture and components of Hadoop, HDFS (Hadoop Distributed File System), MapReduce model and its various applications.

TÓM TẮT

Ngày nay, chúng ta đang sống trong thời đại thông tin, với sự tăng trưởng bùng nổ thông tin theo cấp số nhân. Những công ty hàng đầu về công nghệ thông tin như Google, Yahoo, Amazon, Microsoft, Facebook, Twitter… đối mặt với một khối lượng dữ liệu khổng lồ. Sự tăng trưởng này đòi hỏi các chiến lược mới để xử lý và phân tích dữ liệu. Điện toán đám mây được phát triển và Hadoop-MapReduce đang là một mô hình tính toán mạnh mẽ để giải quyết cho những vấn đề này. Mô hình này đưa ra một khung lập trình cho các ứng dụng xử lí văn bản có khả năng xử lí nhanh chóng một lượng lớn dữ liệu nhờ vào sự song song trong một cụm lớn máy tính. Bài viết này cung cấp một cái nhìn tổng quan về vấn đề xử lí dữ liệu lớn trên nền tảng tính toán đám mây, như là, kiến trúc và thành phần của Hadoop, HDFS (Hadoop distributed File System), Mô hình MapReduce và các ứng dụng khác nhau của nó.

1 GIỚI THIỆU

Theo định nghĩa của NIST(National Institute of Standards and Technology), điện toán đám mây là một mô hình cho phép thuận tiện, truy cập mạng theo yêu cầu đến một nơi chứa các nguồn tài nguyên tính toán có thể chia sẻ và cấu hình được (ví dụ: mạng, máy chủ, lưu trữ, ứng dụng và dịch vụ), ở đó chúng có thể được cung cấp và phát hành nhanh chóng với nỗ lực quản lý hoặc tương tác với nhà cung cấp tối thiểu. Điện toán đám mây đôi khi còn được coi là thế hệ internet mới. Tự điển mở Wikipedia định nghĩa: "Điện toán đám mây là việc sử dụng các tài nguyên máy tính (phần cứng và phần mềm) có sẵn từ xa và truy cập được qua mạng (thường là Internet)". Điện toán đám mây cung cấp 3 mô hình dịch vụ cơ bản: dịch vụ phần cứng (IaaS), dịch vụ hạ tầng (PaaS) và dịch vụ phần mềm (SaaS). Với một số đặc trưng chính : thuê bao theo yêu cầu, nhiều thuê bao,

dùng bao nhiêu trả bấy nhiêu. Về mặt kỹ thuật, đám mây là một tập hợp tài nguyên tính toán rộng lớn và cung cấp các dịch vụ như đã nói trên.

Về mặt lịch sử, John McCarthy từ năm 1960 đã dự đoán rằng "một ngày nào đó tính toán có thể là một dịch vụ công cộng". Điện toán đám mây hiện thực ra đời nhờ vào sự phát triển của các công ty lớn như Google, Amazon với các trung tâm dữ liệu khổng lồ kết nối bằng mạng và Internet. Điện toán đám mây có lẽ không phải để phát triển riêng cho xử lí dữ liệu lớn nhưng nó là một công nghệ mới đầy hứa hẹn cho bài toán dữ liệu lớn. Trong lời dẫn của Hội thảo quốc tế về điện toán đám mây và dữ liệu lớn năm 2012 (http://www.cloudserviceresearch.com/bdc2012/) có đoạn viết "Gần đây, điện toán đám mây đã nổi lên như một công nghệ đầy hứa hẹn đối với việc xử lý dữ liệu lớn."

Khái niệm dữ liệu lớn (big data) chưa có một định nghĩa thống nhất nhưng nó ám chỉ các tập dữ liệu có dung lượng rất lớn mà đơn vị đo thường là cỡ terabyte trở lên. Theo tự điển mở Wikipedia, dữ liệu lớn thông thường là các tập dữ liệu mà kích cỡ của nó vượt quá khả năng thu thập, quản lý và xử lí trong thời gian chấp nhận được. Vì vậy, khái niệm dữ liệu lớn không chỉ là các tập dữ liệu lớn về dung lượng mà còn bao hàm cả về công nghệ lưu trữ, tính toán, tìm kiếm trên đó. Ví dụ, HP cho rằng: "doanh nghiệp đang chìm trong thông tin - có quá nhiều dữ liệu mà không có cách hiệu quả để xử lí nó. Đám mây của HP là đám mây có tính co giãn được, cung cấp nền tảng lưu trữ và lập chỉ mục cho tập dữ liệu hàng trăm petabyte và có thể truy vấn được hầu như tức thời" (https://www.hpcloud.com/solution/big-data-hp-cloud). Sách trắng của IDG năm 2011 viết rằng " dữ liệu lớn đang là thách thức không ngừng gia tăng mà các tổ chức phải đối mặt khi họ gặp các nguồn dữ liệu hay thông tin tăng lên rất nhanh". Làm sao để có thể lưu trữ và xử lí một khối lượng dữ liệu lớn và ngày càng lớn hơn theo cấp số nhân? Giải pháp truyền thống là mua phần cứng lớn hơn (CPU, RAM, đĩa cứng lớn). Giải pháp này ngày càng kém hiệu quả, đòi hỏi đầu tư mới liên tục. Mặt khác, đây là giải pháp đối diện với bế tắc vì dữ liệu có dung lượng lớn hơn khả năng lưu trữ và xử lí của bất kỳ một máy tính nào, chẳng hạn như qui mô dữ liệu của Google. Điện toán đám mây cung cấp một giải pháp lưu trữ trong cụm máy tính có khả năng co giãn được và có thể xử lí được. Mô hình hiện tại có thể xử lí dữ liệu song song phân tán trong một cụm lớn máy kết nối với nhau đó là ánh xạ - rút gọn (Map - Reduce). Có thể Map - Reduce chẳng liên quan gì đến điện toán đám mây, nhưng Hadoop là nền tảng công nghệ cho phép thiết lập cụm máy tính, che dấu mọi gánh nặng về song song hay giao tiếp mạng giữa các máy tính nên nó có thể coi là một giải pháp đám mây cho bài toán dữ liệu lớn. Giải pháp này đang được quan tâm vì tính dễ sử dụng, khả năng mở rộng, dễ chuyển đổi và sao lưu dự phòng.

Mô hình MapReduce cung cấp khung lập trình với chức năng chính là xử lý dữ liệu theo chức năng cơ bản: Ánh xạ (map) và rút gọn (reduce). Việc ánh xạ được thực hiện cục bộ, song song trên các máy trạm khác nhau. Nguyên tắc làm việc: các máy trạm thực hiện xử lí các khối dữ liệu đầu vào theo một kích thước (cố định) nào đó và xuất ra các giá trị trung gian theo dạng cặp dữ liệu <khóa K, giá trị V>. Các cặp này tạo thành một tập hợp các cặp, không có thứ tự và khóa không duy nhất. Tiếp đến, tập các cặp này sẽ được rút gọn thành cặp <K,V'> trong đó khóa là duy nhất.

Theo kiến trúc Hadoop, mô hình MapReduce được cài đặt bao gồm một bộ theo dõi công việc (Job Tracker) hay còn gọi là chủ (Master) và các bộ xử lí nhiệm vụ (Task Trackers hay Workers) để thực thi một công việc (Hadoop Job).

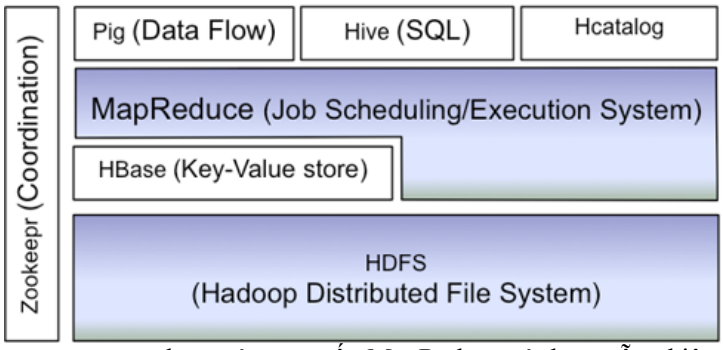
Job Tracker nhận nhiệm vụ từ người dùng (user) và phân chia thành các nhiệm vụ và chia cho các Task Tracker, kiểm soát quá trình thực hiện của các Task Tracker. Sau khi các Task Tracker hoàn thành nhiệm vụ thì Job Tracker báo cáo hoàn thành công việc. Mỗi Task Tracker được ấn định trước một số các ánh xạ và rút gọn (map và reduce) để chỉ ra khả năng xử lí của nó tại một thời điểm. Mô hình tính toán này dựa trên hệ thống quản lí tập tin phân tán HDFS (Hadoop Distributed File System), đó là hệ thống tin cậy và khả năng chịu lỗi cao nhằm quản lí lưu trữ và thực hiện sao chép dữ liệu đầu vào, đầu ra của một công việc Hadoop.

Cần lưu ý rằng những gì chúng ta nói đến về dữ liệu đầu vào đầu ra là trong ngữ cảnh dữ liệu lớn (Big data), vượt quá khả năng lưu trữ và xử lí của bất kỳ một hệ quản trị CSDL hay một máy tính đơn lẻ nào. Dữ liệu lớn có thể có cấu trúc,

bán cấu trúc hay phi cấu trúc, thông thường là phi cấu trúc. Dữ liệu đó cần có một hệ thống quản lí lưu trữ mới đó là HDFS.

2 HADOOP

Hadoop làm việc trên nguyên tắc xử lí theo lô dựa trên một cụm máy tính gọi là các *nút* (nodes). Các nút được cung cấp nền tảng phục vụ các hoạt động phân tích dữ liệu lớn theo mô hình *ánh xạ-rút gọn* (Map-reduce) cho dữ liệu phi cấu trúc trên hệ thống tập tin phân tán. Hadoop là nền tảng chính đóng góp cho thành công của bộ tìm kiếm Yahoo. Hadoop bao gồm nhiều thành phần cung cấp chức năng tính toán song song và phân tán. Kiến trúc tổng quát của Hadoop được cho trong Hình 1.



Hình 1: Kiến trúc của Hadoop [1]

Lõi (core) Hadoop: bao gồm một tập hợp của các thành phần và các giao diện cung cấp chức năng truy cập vào các hệ thống tập tin phân tán và vào ra tổng quát. Các thành phần cốt lõi cũng cung cấp sự tối ưu hóa dựa vào phân nhóm địa lý của máy chủ nhằm giảm thiểu lưu lượng mạng giữa các máy chủ trong các cụm tính toán.

Hadoop-MapReduce: là một mô hình lập trình và khung phát triển phần mềm cho phép viết các ứng dụng nhanh chóng xử lý một lượng lớn dữ liệu song song dựa vào một cụm lớn các máy tính gọi là các nút tính toán (node) dựa trên kiến trúc của nền tảng Hadoop. MapReduce sử dụng HDFS để truy cập vào các khối (phân đoạn tập tin) và lưu trữ kết quả rút gọn.

Hệ thống tập tin phân tán (Hadoop Distributed File System-HDFS) là hệ thống lưu trữ chính được sử dụng bởi các ứng dụng Hadoop. HDFS, như tên gọi của nó, một hệ thống tập tin phân tán cung cấp truy cập thông lượng cao vào dữ liệu của ứng dụng, tạo ra nhiều bản sao của khối dữ liệu và phân phối chúng trên các nút tính toán trong một cụm để cho phép tính toán song song, đáng tin cậy và nhanh chóng.

HBase: là một cơ sở dữ liệu phân tán theo cột. HBase sử dụng HDFS cho việc lưu trữ cơ bản của nó. Nó ánh xạ dữ liệu HDFS vào một cơ sở dữ liệu có cấu trúc giống và cung cấp các giao diện lập trình được cho Java (Java API) truy cập vào CSDL này. Nó hỗ trợ hàng loạt kiểu tính toán sử dụng các truy vấn MapReduce và đọc ngẫu nhiên. HBase thường được sử dụng trong Hadoop khi có truy cập đọc/ ghi ngẫu nhiên, thời gian thực. Mục tiêu của nó là lưu trữ các bảng rất lớn đang chạy trên cụm thiết bị phần cứng.

Pig: là ngôn ngữ xử lí dòng dữ liệu. Apache Pig là một nền tảng cho việc phân tích dữ liệu lớn bao gồm một ngôn ngữ cấp cao để diễn tả các chương trình phân tích dữ liệu. Đặc điểm chính của chương trình Pig là cấu trúc của chúng có thể được song song hóa cho phép nó xử lý các tập hợp dữ liệu rất lớn, cú pháp đơn giản. Các tính năng xây dựng sẵn (built-in functionality) cung cấp một mức độ trừu trượng để cho phát triển các công việc Hadoop nhanh hơn và dễ dàng hơn để viết hơn so với MapReduce truyền thống.

Zookeeper: là một công cụ cấu hình cụm (cluster) và quản lý sự dãy hóa (serialization) rất hữu ích để xây dựng các cụm lớn các nút của Hadoop, dịch vụ hiệu năng cao cho các ứng dụng phân tán. Nó tập trung vào các dịch vụ như quản lí thông tin cấu hình, đặt tên, đồng bộ hóa phân tán cũng và các dịch vụ nhóm.

Hive: là một kho dữ liệu cơ sở hạ tầng được xây dựng trên Hadoop. Hive cung cấp các công cụ để cho phép tóm tắt dữ liệu, truy vấn không chuẩn (ad-hoc) và phân tích các bộ dữ liệu lớn được lưu trữ trong các tập tin Hadoop. Nó cung cấp một cơ chế để định cấu trúc cho loại dữ liệu này và cung cấp một ngôn ngữ truy vấn đơn giản gọi là Hive

QL, dựa trên SQL, cho phép người sử dụng quen thuộc với SQL để truy vấn dữ liệu này.

Chukwa: công cụ mã nguồn mở, được sử dụng để quản lí các cụm lớn máy chủ phân tán. Nó là một hệ thống thu thập dữ liệu để theo dõi các hệ thống phân tán lớn. Chukwa bao gồm một bộ công cụ linh hoạt và mạnh mẽ cho hiển thị, theo dõi và phân tích kết quả để sử dụng tốt nhất các dữ liệu thu thập được.

HCatalog: là một lớp quản lý lưu trữ cho Hadoop cho phép người dùng sử dụng với các công cụ xử lý dữ liệu khác nhau. Bảng HCatalog trình bày cho người dùng một khung nhìn kiểu quan hệ cho dữ liệu trong hệ thống tập tin phân tán Hadoop (HDFS) và đảm bảo rằng người dùng không cần phải quan tâm về nơi lưu trữ hoặc định dạng của dữ liệu được lưu trữ.

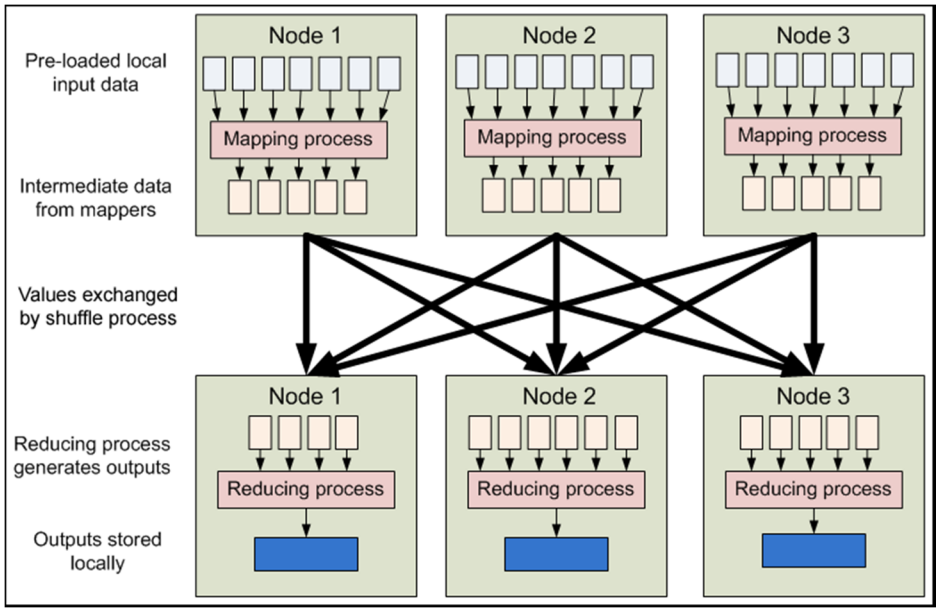
3 HỆ THỐNG TẬP TIN PHÂN TÁN (HDFS)

Một cụm máy tính cài đặt hệ thống HDFS có hai loại nút: *Nút tên* (NameNode), hay còn gọi là *nút chủ* (master), và *nút dữ liệu* (DataNodes), hay còn gọi là *nút tớ* (worker). NameNode quản lý không gian tên hệ thống tập tin. Nó duy trì cây hệ thống tập tin và siêu dữ liệu cho tất cả các tập tin và thư mục trong cây. NameNode nhận biết các DataNode mà trên đó tất cả các khối cho một tập tin được đặt phân tán. Các DataNode lưu trữ và lấy khối khi nó được gọi (bởi người dùng hoặc từ nút chủ). DataNode báo cáo định kỳ cho các NameNode danh sách của khối mà nó đang lưu trữ. NameNode quyết định về sao lưu dữ liệu của 1 khối. Trong một HDFS điển hình, mỗi khối có kích thước là 64MB và thường được nhân làm 3 bản, bản thứ hai sao đĩa cục bộ và bản thứ ba ở trên đĩa từ xa để phòng hờ.

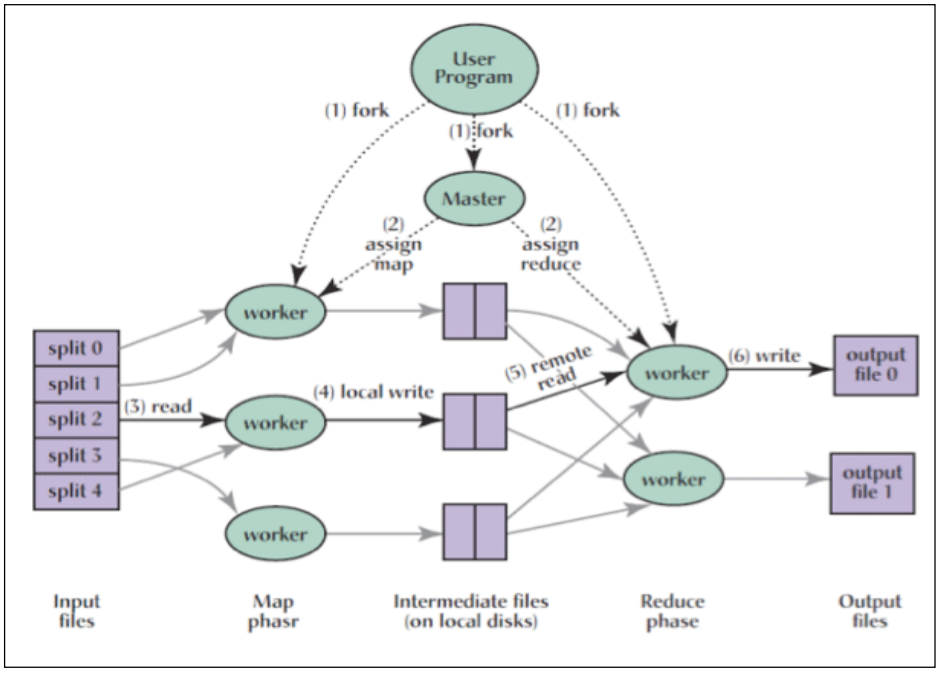
Nguyên tắc hoạt động: Để đọc một tập tin HDFS, các ứng dụng từ máy khách (client) chỉ cần sử dụng một luồng đầu vào (như file input stream trong Java) giống như đọc tập tin trong hệ thống tập tin cục bộ. Tuy nhiên, về xử lí đằng sau hậu trường, luồng đầu vào này được xử lí để lấy dữ liệu từ HDFS. Đầu tiên, NameNode liên lạc để xác định cho phép truy cập. Nếu được phép truy cập, NameNode sẽ xác định danh sách của khối HDFS chứa tập tin và một danh sách các nút dữ liệu lưu trữ mỗi khối và trả về cho máy khách. Tiếp theo, máy khách sẽ mở một kết nối đến DataNode "gần nhất" và yêu cầu một truy cập các khối cụ thể. Kế đến, các khối HDFS được trả về trên cùng một kết nối và chúng được đưa vào ứng dụng. Để ghi dữ liệu HDFS, các ứng dụng xem các tập tin HDFS như là một luồng đầu ra (output stream). Tuy nhiên, bên trong, dòng dữ liệu, trước hết, được phân chia thành các khối có kích thước HDFS (64MB chẳng hạn) và sau đó lại chia thành các gói nhỏ hơn (thường là 64KB) để xếp hàng chờ ghi. Việc này được một tiến trình thực hiện (thread). Một tiến trình thứ hai thực hiện giải phóng hàng, phối hợp với các NameNode để gán định danh (ID) cho khối và phân phối đến các DataNode để lưu trữ. Ngoài ra còn một tiến trình thứ ba quản lý phản hồi từ các NameNode rằng dữ liệu đã được ghi hoàn tất.

4 MÔ HÌNH ÁNH XẠ - RÚT GỌN (MAP-REDUCE)

MapReduce là mô hình xử lý dữ liệu (xem Hình 2) với lập trình song song được giới thiệu bởi Google. Trong mô hình này, người sử dụng xây dựng các tính toán qua hai chức năng, ánh xạ (map) và rút gọn (reduce). Trong giai đoạn ánh xạ, MapReduce lấy các dữ liệu đầu vào và đưa cho *bộ ánh xạ* (Mapper). Bộ ánh xạ thực hiện cái gì là tùy theo cài đặt của người lập trình, tuy nhiên về tổng quan thì nó lọc và chuyển đổi đầu vào thành một cái gì đó là sản phẩm để có thể tổng hợp trong giai đoạn rút gọn về sau. Trong giai đoạn rút gọn, *bộ rút gọn* (Reducer) xử lý các kết quả đầu ra từ mapper để đưa đến kết quả cuối cùng. Các thư viện cơ bản MapReduce hỗ trợ tự động song song hóa tính toán và xử lý các vấn đề phức tạp như phân tán dữ liệu, cân bằng tải và khả năng chịu lỗi, khắc phục lỗi (fault tolerance). Hadoop, nền tảng cài đặt MapReduce, nhằm mục đích để song song hóa tính toán trong các cụm lớn máy tính. Mô hình MapReduce có lợi thế là dễ dàng mở rộng quy mô xử lý dữ liệu trên nhiều nút tính toán.



Hình 2: Mô hình Map- Reduce [2]



Hình 3: Ba vai trò chính trong thực hiện Map-Reduce [3]

5 LẬP TRÌNH VỚI MAP-REDUCE

Theo mô hình MapReduce, cơ bản, các lập trình viên chỉ cần viết hai chức năng (hoặc hai hàm): *ánh xạ* và *rút gọn*. Chức năng ánh xạ sẽ nhận dữ liệu đầu vào và sinh ra các cặp <*khóa, giá trị*> trung gian để được tiếp tục xử lý. Chức năng rút gọn kết hợp tất cả các cặp khóa / giá trị trung gian, liên kết theo khóa để tạo ra đầu ra cuối cùng. Về mặt logic chương trình sẽ có ba vai trò chính: *bộ chỉ huy* (master), *bộ ánh xạ* (mapper) và *bộ rút gọn* (reducer). Vai trò của bộ chỉ huy là lập kế hoạch công việc, quản lý công việc. Mô hình MapReduce được xây dựng trên một hệ thống tập tin phân tán (HDFS) cung cấp lưu trữ và truy cập phân tán. Hình 3 cho thấy quá trình thực hiện của MapReduce qua hai giai đoạn: ánh xạ và rút gọn.

Dữ liệu đầu vào được chia thành một tập hợp của các khối. Các bộ ánh xạ (mapper) đọc các khối và xử lí song song trong giai đoạn ánh xạ. Mỗi mapper sẽ xử lý dữ liệu bằng cách phân tích dữ liệu thành các cặp <khóa, giá trị> và sau đó tạo ra các kết quả trung gian được lưu trữ trong hệ thống tập tin cục bộ. Kết quả trung gian sẽ được sắp xếp và kết hợp theo khóa: tất cả các cặp với cùng khóa sẽ được nhóm lại với nhau. Các bộ rút gọn (reducer) sử dụng phương thức gọi từ xa (RPC) để đọc dữ liệu từ các mapper. Người lập trình sẽ viết mã để xác định cách thức rút gọn, tức là tính toán kết quả đầu ra. Cuối cùng, đầu ra sẽ được ghi vào các tập tin trong hệ thống phân tán.

MapReduce được thiết kế để chịu đựng lỗi (Fault tolerance), tức là khả năng tự thân khắc phục lỗi. Đây là tính năng quan trọng vì việc xảy ra lỗi trong một hệ thống phân tán gồm số lượng lớn máy tính không phải là hiếm gặp. Lỗi có thể xảy ra ở máy chủ và máy tớ (hay máy làm việc). Máy chủ sẽ dò (ping) máy tớ, bao gồm bộ ánh xạ và bộ rút gọn, theo định kỳ. Nếu không có phản ứng từ máy tớ trong một khoảng thời gian nhất định, máy tớ được đánh dấu là hỏng. Nhiệm vụ đang thực thi và các nhiệm vụ cần được thực hiện bởi mapper hỏng sẽ được giao lại để mapper khác và thực hiện ngay từ đầu. Nếu bộ rút gọn hỏng thì công việc rút gọn nào đã hoàn thành không cần phải được thực hiện lại bởi vì kết quả đã được lưu trữ trong hệ thống tập tin toàn cục, phân tán. Trường hợp lỗi xảy ra ở máy chủ (chỉ có một máy tính duy nhất nên xác suất hỏng rất nhỏ), MapReduce sẽ thực hiện lại toàn bộ công việc.

6 MỘT VÍ DỤ VỀ LẬP TRÌNH MAP- REDUCE

Xét bài toán thống kê tần suất xuất hiện của các từ trong một tập tin. Chúng ta sẽ giải quyết bài toán này theo mô hình lập trình MapReduce. Xin nhắc lại, MapReduce chỉ là một cách, một mô hình theo đó phân chia một nhiệm vụ lớn thành các nhiệm vụ con rời rạc, có thể được thực hiện song song.

Giải thuật giải quyết bài toán thống kê tần suất nói trên theo mô hình Mapreduce được mô tả như sau:

– Dữ liệu đầu vào được phân chia thành những khối nhỏ lưu trữ phân tán trong cụm máy tính.

– Đối với mỗi khối dữ liệu đầu vào, một bộ ánh xạ (mapper) chạy cho ra kết quả đầu ra là ánh xạ mỗi từ thành cặp <khóa, giá trị> trong đó khóa là từ còn giá trị là tần suất của từ trong khối. Cũng có thể đơn giản là <khóa, 1> trong đó các khóa có thể trùng lại, khóa là một từ trong khối.

– Kết quả đầu ra của tất cả các bộ ánh xạ được sắp xếp và kết tập theo khóa riêng biệt; một bộ sưu tập được tạo ra có chứa tất cả các giá trị tương ứng từ đầu ra của các mapper.

– Tiếp theo, mỗi bộ sưu tập được thực hiện rút gọn theo khóa để tạo ra cặp <khóa, giá trị> trong đó khóa là từ còn giá trị là tần suất của từ đó trong tập tin (tức là trong tất cả các khối). Đây chính là kết quả cuối cùng của hai bước ánh xạ và rút gọn. Đó cũng là kết quả kết tập mong muốn.

Lợi ích của mô hình MapReduce ở đây là rất rõ ràng. Lập trình viên không bận tâm tới lưu trữ vật lý phân tán của tập tin trong HDFS. Chương trình không phụ thuộc vào cách lưu trữ phân tán. Lập trình viên cũng không cần phải quan tâm tới việc tính tải hay khả năng xử lý của các nút (nodes). Tính co giãn của chương trình là tự động do chức năng của Hadoop. Minh họa phần chính của chương trình MapReduce, bao gồm mapper, reducer và thủ tục chính được trong hình 4 trích từ hướng dẫn của Hadoop [2].

Theo cấu trúc chương trình trong Hình 4, bộ ánh xạ được cài trong Mapper, nó đơn giản là ánh xạ mỗi từ với 1 (tức là đếm 1 cho mỗi từ xuất hiện) và sinh ra cặp *<từ, 1>*. Các cặp này sẽ được bộ rút gọn tổng hợp về sau. Ở đây, sẽ có một số thể hiện khác nhau của Mapper chạy trên các máy khác nhau trong cụm máy để thực hiện song song chương trình. Tương tự như vậy, cũng sẽ có nhiều thể hiện khác nhau của bộ rút gọn (Reducer) chạy song song trên các máy tớ. Chúng thực hiện việc tổng hợp (trong bài toán này là tính tổng lần xuất hiện) các cặp *<từ, 1>* để cho ra các cặp <từ, n> với n là tần suất của từ. Cuối cùng, các bộ rút gọn cho ra kết quả cuối cùng là các cặp <từ, n> và ghi kết quả ra tập tin xuất.

```
Public static class MapClass extends MapReduceBase
 implements Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(LongWritable key, Text value,
        OutputCollector<Text, IntWritable> output,
        Reporter reporter) throws IOException {
             String line = value.toString();
             StringTokenizer itr = new StringTokenizer(line);
                  while (itr.hasMoreTokens()) {
                  word.set(itr.nextToken());
                  output.collect(word, one);
                  }
             }
 }

 /**
 * A reducer class that just emits the sum of the input values.
 */
 public static class Reduce extends MapReduceBase
 implements Reducer<Text, IntWritable, Text, IntWritable> {
         public void reduce(Text key, Iterator<IntWritable> values,
         OutputCollector<Text, IntWritable> output,
         Reporter reporter) throws IOException {
              int sum = 0;
              while (values.hasNext()) {
                   sum += values.next().get();
              }
              output.collect(key, new IntWritable(sum));
         }
}

public void run(String inputPath, String outputPath) throws Exception {
          JobConf conf = new JobConf(WordCount.class);
          conf.setJobName("wordcount");

          // the keys are words (strings)
          conf.setOutputKeyClass(Text.class);
          // the values are counts (ints)
          conf.setOutputValueClass(IntWritable.class);

          conf.setMapperClass(MapClass.class);
          conf.setReducerClass(Reduce.class);

          FileInputFormat.addInputPath(conf, new Path(inputPath));
          FileOutputFormat.setOutputPath(conf, new Path(outputPath));
          JobClient.runJob(conf);
}
```

Hình 4: Trích code mapper, reducer và thủ tục chính (main/driver) chương trình WordCount [2]

Thành phần thứ 3 trong mã chương trình là thủ tục điều khiển (driver), như là thủ tục chính (như hàm main) điều khiển chương trình. Nó khai báo và khởi tạo một công việc Hadoop (Hadoop Job). Thiết lập các đường dẫn vào, ra chương trình để truy cập các tập tin và gọi thực thi gói công việc Hadoop, tức là thực hiện chương trình ánh xạ - rút gọn. Về mặt kỹ thuật, lời gọi JobClient.runJob(conf) sẽ đưa một gói công việc (job) cho MapReduce và chờ đến khi công việc hoàn tất.

Trên đây là mô tả khái quát ở mức cao quá trình thực hiện ánh xạ-rút gọn, chi tiết hơn về cách thức hoạt động của chương trình cài đặt bằng Java trong Hadoop có thể tham khảo trong tài liệu [2].

Nói chung MapReduce làm việc tốt trên tập tin phi cấu trúc như là văn bản. Nó có vẻ không phù hợp lắm với dữ liệu có cấu trúc truyền thống như là cơ sở dữ liệu quan hệ. Một vấn đề đáng quan tâm trong Hadoop liên quan tới việc chia công việc và thực hiện song song trên nhiều máy tớ: nếu có một máy nào đó chậm (đọc đĩa chậm chẳng hạn) thực hiện ánh xạ, thì các máy còn lại đã thực hiện xong việc ánh xạ nhưng vẫn phải chờ cho máy chậm đó hoàn thành công việc.

7 KẾT LUẬN

Tiến bộ công nghệ điện toán đám mây và việc gia tăng sử dụng Internet đang tạo ra những tập dữ liệu rất lớn. Dữ liệu lớn vẫn còn ở giai đoạn sơ khai nhưng cũng đã có những ảnh hưởng sâu sắc đến các công ty công nghệ và cách làm kinh doanh mới. Kích thước của các bộ dữ liệu lớn đòi hỏi cần có cách thức lưu trữ và xử lí mới, phù hợp. Mô hình lập trình MapReduce với Hadoop là một nền tảng cơ bản trong cộng đồng dữ liệu lớn nhờ vào hiệu quả chi phí trên cụm máy tính được thiết lập như là một đám mây điện tử. Tính hiệu quả và dễ dàng sử dụng của nền tảng Hadoop ở chỗ nó cài đặt mô hình ánh xạ - rút gọn chạy song song, liên quan đến nhiều thuật toán phân tích dữ liệu đã được che dấu bên trong. Một trong số đó là hệ thống tập tin phân tán HDFS, có thể chứa một khối lượng rất lớn dữ liệu (tính bằng terabytes hoặc thậm chí petabytes) và cung cấp cơ chế try cập thông lượng cao vào các dữ liệu này. Mô hình này, ngoài việc che dấu sự phức tạp trong tính toán song song, cân bằng tải, phân phối tải,… còn có khả năng cơ bản cung cấp tính mềm dẻo, khả năng chịu đựng lỗi cũng như tính co giãn của hệ thống. Hiện tại đây là mô hình chính để lập trình xử lí dữ liệu lớn theo mô hình đám mây điện tử.

TÀI LIỆU THAM KHẢO

1. http://cto.vmware.com/project-serengeti-theres-a-virtual-elephant-in-my-datacenter/
2. Hadoop Tutorial, http://developer.yahoo.com/hadoop/tutorial/module4.html
3. http://www.cubrid.org/blog/dev-platform/platforms-for-big-data/
4. Thomas A. de Ruiter, A Workload Model for MapReduce, Master Thesis in Computer Science, Parallel and Distributed Systems Group Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, 2nd June 2012.