BIG DATA VÀ GIẢI PHÁP LƯU TRỮ DỮ LIỆU

Nguyễn Mậu Hân

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

Email: nmhan2009@gmail.com

Ngày nhận bài: 29/3/2018; ngày hoàn thành phản biện: 26/4/2018; ngày duyệt đăng: 8/6/2018

TÓM TẮT

Big Data được biết như là một giải pháp lý tưởng để xử lý các dữ liệu có cấu trúc, dữ liệu bán cấu trúc hay thậm chí là phi cấu trúc như weblogs, mạng xã hội, e-mail, các dữ liệu cảm biến và các bức ảnh mà có thể được khai thác nhằm tìm ra những thông tin hữu ích. Vấn đề đặt ra là giải pháp nào cho bài toán lưu trữ đối với loại dữ liệu này. Bài báo này đề xuất một giải pháp lưu trữ cho các trung tâm dữ liệu, nơi đang tiếp xúc hằng ngày các loại Big Data, trong khi các phương pháp lưu trữ truyền thống đã bộc lộ nhiều khiếm khuyết.

Từ khóa: Big Data, Dữ liệu có cấu trúc, Haddop, MapReduce.

1. GIỚI THIỆU

"Big data" là thuật ngữ dùng để chỉ một tập hợp dữ liệu rất lớn và rất phức tạp đến nỗi những công cụ xử lí dữ liệu truyền thống không thể nào đảm đương được. Theo IBM [1], big data được hiểu một cách chung nhất là "mỗi ngày, trên toàn thế giới, chúng ta tạo ra 2.5 tỷ Gigabytes dữ liệu. Và dữ liệu này tương ứng với dữ liệu mà chúng ta thu được trong 2 năm trước. Dữ liệu ngày càng thu được ở khắp mọi nơi bao gồm: các mạng cảm biến thu thập thông tin, các bài viết trên các trang mạng xã hội, các bức ảnh kỹ thuật số, video clip, các thanh toán mua hàng trực tuyến và các tín hiệu GPS từ điện thoại". Năm 2014, công ty phân tích dữ liệu Gartner [3] đưa ra khái niệm mới có thể chấp nhận được về mô hình "5Vs" của Big Data. Đó là, 5 đặc trưng của Big Data: tăng về lượng (volume), tăng về vận tốc (velocity) , tăng về chủng loại (variety), tăng về độ chính xác (veracity) và tăng về giá trị thông tin (value). Hiểu một cách đơn giản, đó chính là sự phát triển không ngừng của khối lượng dữ liệu cần lưu trữ, cách thức để xử lý dữ liệu với tốc độ cao, tính đa dạng dữ liệu (variety): Theo IBM [7], chỉ có 20% dữ liệu thu được là có cấu trúc nhưng thực tế là 80% dữ liệu trên thế giới đều là dạng phi cấu trúc hoặc bán cấu trúc. Ngoài ra còn phải quản lý được các dữ liệu mới được tạo ra và các dữ liệu được cập nhật, độ chính xác của xử lý và giá trị thông tin được lưu trữ.

Tất cả các quan điểm trên đều hướng tới việc trả lời cho câu hỏi: Big Data là vấn đề gì và tại sao chúng ta cần phải nghiên cứu và tìm hiểu nó. Vấn đề này được các nhà cung cấp dịch vụ, các trung tâm tích hợp dữ liệu nghiên cứu, tìm hiểu phương pháp tốt nhất để lưu trữ loại dữ liệu với 5 yếu tố trên. Nhìn chung, có bốn lợi ích chính mà Big data có thể mang lại đó là: cắt giảm chi phí, giảm thời gian tìm kiếm thông tin, tăng thời gian phát triển và tối ưu hóa sản phẩm, đồng thời hỗ trợ con người đưa ra những quyết định đúng đắn và hợp lý. Trong phạm vi bài báo này chúng tôi chỉ nghiên cứu về giải pháp lưu trữ của Big data, một công cụ không thể thiếu trong cuộc cách mạng 4.0 này.

2. BIG DATA VÀ CÔNG CỤ XỬ LÝ

Kỹ thuật xử lý dữ liệu trong Big data chủ yếu là NoSQL (cơ sở dữ liệu theo cột, cặp khóa-giá trị) [4], bởi vì mô hình dữ liệu quan hệ không thể đáp ứng được. Tuy nhiên trong thực tế nhiều công ty lớn cũng đã đưa ra các công cụ khác nhau để xử lý Big data.

2.1 Giải pháp Hadoop/MapReduce

Năm 2004, Google công bố kiến trúc của hệ thống file phân tán GFS (Google File System) và công cụ MapReduce. Từ đó Hadoop, một Framework, cùng với GFS và MapReduce được ra đời bởi Dough Cutting.

2.1.1 Hadoop

Apache Hadoop [7] là một framework dùng để chạy những ứng dụng trên 1 cluster lớn được xây dựng trên những phần cứng thông thường. Hadoop hiện thực mô hình Map/Reduce, đây là mô hình phân tán song song, ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn khác nhau (phân tán), và các phần này sẽ được chạy song song trên nhiều node khác nhau. 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 các máy tính có cấu hình cao với khả năng tính toán, lưu trữ và truy xuất mạnh mẽ. Do vậy, chi phí triển khai MapReduce sẽ rẻ hơn. MapReduce làm đơn giản hoá các giải thuật tính toán phân tán

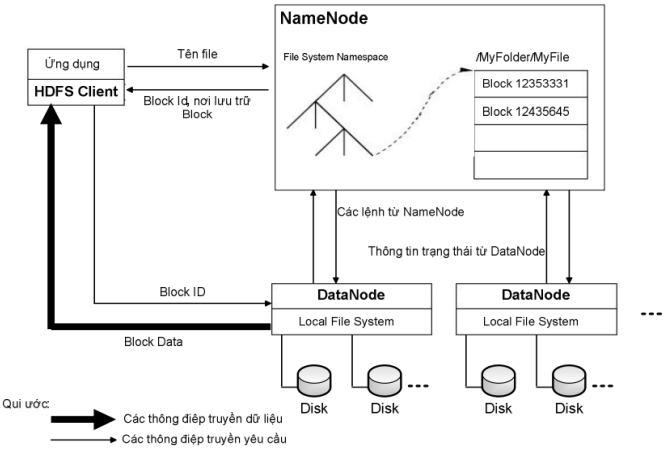
bằng cách chỉ cần cung cấp hai hàm Map và Reduce cùng với một số thành phần xử lý dữ liệu đầu vào.

Hàm Map: Người dùng đưa một cặp dữ liệu (key, value) làm input cho hàm Map, và tùy vào mục đích của người dùng mà hàm Map sẽ trả ra danh sách các cặp dữ liệu (intermediate key, value).

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 dự 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, …

2.2.2 Kiến trúc Hadoop File System (HDFS)

Giống như các hệ thống file khác, HDFS duy trì một cấu trúc cây phân cấp các file [6], 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 1. Kiến trúc của HDFS

3. BIGTABLE VÀ GIẢI PHÁP LƯU TRỮ

3.1 Giới thiệu về Bigtable

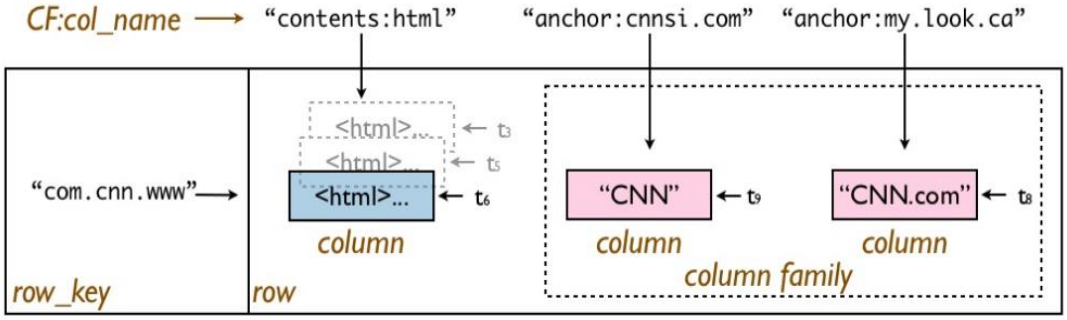
Bigtable là một hệ thống lưu trữ phân tán dùng để quản lý dữ liệu có cấu trúc được thiết kế để co giản trong phạm vi rất lớn. Những cụm Bigtable được sử dụng với nhóm hàng nghìn server, và lưu trữ tới vài trăm terabyte dữ liệu. Bigtable không hỗ trợ mô hình dữ liệu quan hệ. Thay vào đó, nó cung cấp các ứng dụng client với một mô hình dữ liệu đơn giản có hỗ trợ điều khiển động đối với kiến trúc và định dạng dữ liệu. Bigtable cho phép các ứng dụng client suy ra những đặc tính vị trí của dữ liệu được mô tả trong kho lưu trữ bên dưới. Dữ liệu được đánh chỉ mục theo tên hàng và cột có thể là các xâu bất kì. Bigtable cũng coi dữ liệu như là các xâu không diễn dịch (uninterpreted), mặc dù các ứng dụng client thường sắp xếp những dạng khác nhau

của dữ liệu có cấu trúc và bán cấu trúc vào những xâu này. Client có thể điều khiển vị trí của dữ liệu của họ thông qua những lựa chọn cẩn thận ở lược đồ.

3.2 Mô hình dữ liệu

Một Bigtable là một bản đồ phân tán, đa chiều, ổn định [5]. Bản đồ này được đánh chỉ mục bởi một khóa hàng (row), khóa cột (column), và một nhãn thời gian (timestamp). Mỗi giá trị trong bản đồ là một mảng dữ liệu không diễn dịch (uninterpreted): (row:string, column:string, time:int64) →string

Ví dụ về lưu trữ trang "cnn.com": Tên hàng là địa chỉ URL, họ cột "contents:" chứa nội dung trang, họ cột "anchor" chứa văn bản của bất kì liên kết nào tới trang web. Trang cnn được 2 trang tham chiếu tới, do đó hàng chứa các cột có tên là anchor:cnnsi.com và anchor:my.look.ca. Mỗi ô anchor có nhiều phiên bản, cột "contents:" có 3 phiên bản với nhãn thời gian là t_3, t_5, t_6. Giả sử rằng chúng ta muốn giữ một bản sao của một tập hợp lớn các trang web và thông tin liên quan mà có thể được sử dụng bởi nhiều dự án khác nhau; chúng ta gọi những bảng này là Webtable. Trong Webtable, chúng ta sử dụng địa chỉ URL như là các khóa hàng, các bộ phận khác nhau của trang web như là tên cột, và lưu trữ nội dung trang Web vào CONTENTS, và cột dưới nhãn thời gian khi chúng được lấy ra.



Hình 2. Ví dụ về lưu trữ một trang web

3.2.1 Hàng

Các khóa hàng là các xâu bất kì (dung lượng có thể lên tới 64KB). Tất cả các hoạt động đọc hay ghi dữ liệu bên dưới một khóa hàng đơn đều là "nguyên tử", một giải pháp thiết kế có thể làm cho các ứng dụng khách thấy dễ dàng hơn khi suy luận về nguyên lý của hệ thống khi xảy ra cập nhật đồng thời lên cùng một hàng. Bigtable bảo trì dữ liệu theo thứ tự từ điển bởi khóa hàng. Mỗi một dãy hàng được gọi là bảng phụ (tablet), bảng phụ là đơn vị của phân tán và cân bằng tải. Việc đọc các dãy hàng ngắn có hiệu quả và yêu cầu giao tiếp với chỉ một số lượng nhỏ các máy. Client có thể khai thác thuộc tính này bằng cách chọn những khóa hàng của họ vì thế họ có được những vị trí tốt cho việc truy cập dữ liệu. Ví dụ, trong Webtable, các trang trong cùng tên miền được nhóm vào các hàng kề nhau bằng cách đảo ngược các thành phần trong địa chỉ URL. Ví dụ, chúng ta lưu dữ liệu cho địa chỉ maps.google.com/index.html bằng

khóa com.google.maps/index.html. Lưu trữ các trang có tên miền giống nhau gần nhau giúp cho các host và phân tích tên miền được hiệu quả hơn.

3.2.2 Họ cột

Các khóa cột được nhóm vào một bảng được gọi là "họ" cột, tạo thành các khối cơ bản của kiểm soát truy xuất. Tất cả dữ liệu được lưu trong một "họ" cột thường có chung kiểu (do chúng ta nén dữ liệu trong cùng một họ đồng thời với nhau). Một "họ" cột phải được tạo ra trước khi dữ liệu được lưu trữ tại một cột nào đó trong họ. Sau khi một họ được tạo, mọi khóa cột bên trong họ đó đều có thể sử dụng. Số họ cột trong một bảng không nhiều (nhiều nhất là hàng trăm), và những họ này hiếm khi thay đổi trong quá trình hoạt động. Ngược lại, một bảng có số cột không giới hạn. Một khóa cột được đặt tên dựa theo cú pháp "tên_họ:tính_chất". Ví dụ về họ cột cho Webtable là LANGUAGE, nó lưu trữ ngôn ngữ mà trang web đó được viết. Chúng ta chỉ sử dụng một khóa cột cho họ LANGUAGE, và nó lưu trữ định danh của ngôn ngữ của mỗi trang web. Một họ cột cũng rất hữu dụng cho bảng này là ANCHOR; mỗi cột trong họ đại diện cho một anchor đơn lẻ. Phần tính chất là tên của trang liên quan, nội dung ô là kết nối văn bản.

Điều khiển truy xuất cùng với đĩa và tính toán bộ nhớ được thực hiện tại mức họ cột. Trong ví dụ Webtable, bộ điều khiển cho phép chúng ta quản lý vài loại ứng dụng khác nhau: một vài trong số chúng dùng để tạo mới dữ liệu cơ bản, một vài để đọc dữ liệu cơ bản và tạo ra các họ cột từ đó, và một vài thì chỉ cho phép xem dữ liệu đang tồn tại.

3.2.3 Nhãn thời gian

Mỗi ô trong Bigtable có thể chứa nhiều phiên bản của cùng một dữ liệu, những phiên bản này được đánh chỉ mục bởi nhãn thời gian. Nhãn thời gian là các số nguyên 64 bit. Chúng có thể được chỉ định bởi Bigtable, trong trường hợp chúng mô tả thời gian thực tới từng micro giây, hoặc được chỉ định bởi các ứng dụng người dùng. Ứng dụng nào cần tránh các xung đột phải tự sinh ra nhãn thời gian duy nhất của riêng chúng. Các phiên bản khác của một ô được lưu trữ theo thứ tự giảm dần của nhãn thời gian, nhờ đó phiên bản mới nhất có thể được đọc trước. Để cho việc quản lý các phiên bản dữ liệu được dễ dàng hơn, cho phép hỗ trợ hai môi trường trên các họ cột. Phía client có thể chỉ định một số n nào đó phiên bản cuối cùng được giữ lại, hoặc chỉ giữ lại những phiên bản đủ mới (ví dụ, chỉ giữ lại giá trị được ghi trong vòng 7 ngày trở lại).

3.3.4 Giao diện lập trình ứng dụng API

Bigtable API cung cấp chức năng cho việc tạo và xóa các bảng và các họ cột. Nó cũng cung cấp chức năng để chuyển cụm(cluster), bảng, và siêu dữ liệu họ cột.

Các ứng dụng client có thể ghi và xóa giá trị, tìm kiếm giá trị từ các hàng riêng lẻ, hoặc lặp lại một nhóm dữ liệu trong một bảng. Dưới đây là một đoạn mã của C++ sử

dụng hàm *RowMutation* để thực hiện một chuỗi cập nhật và gọi hàm *Apply* thực hiện một sự thay đổi nguyên tử đến Webtable: thêm 1 anchor vào www.cnn.com và xóa một anchor khác đi.

```
Table *T = OpenorDie("/bigtable/web/webtable"); // Open a table
RowMutation r1 (T, "com.cnn.www"); // Write a new anchor
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com"); // Delete an old anchor
Operation op;
Apply( &op, &r1);
```

Đoạn mã dưới đây cho thấy hàm *Scanner* của C++ được sử dụng để lặp lại tất cả các anchor trong 1 hàng. Client có thể lặp lại trên nhiều họ cột, và có vài cơ chế định ra giới hạn số hàng, cột, nhãn thời gian tạo ra bởi 1 bộ scan. Ví dụ, chúng ta có thể hạn chế bộ scan chỉ tạo ra những anchor có cột phù hợp với biểu thức anchor.*.cnn.com, hoặc chỉ tạo ra những anchor mà nhãn thời gian trong vòng 10 ngày trở lại.

```
Scanner scanner(T);
ScanStream *stream;
 stream= scanner.FetchColumnFamily("anchor");
 stream-> SetReturnAllVersions();
scanner.Lookup("com.cnn.www");
for (; !stream->Done();
stream->next())
{
printf ("%s %s %11d %s \n, scanner.Rowname(), stream->Columnname(),
stream->MicroTimestamp(), stream->Value());
 }
```

Bigtable cũng có thể sử dụng với MapReduce, dùng để chạy các tính toán song song phát triển bởi Google.

3.4 Xây dựng các khối

Bigtable được xây dựng trên các phần khác nhau của cơ sở hạ tầng của Google. Bigtable sử dụng hệ thống file phân tán Google để lưu trữ bản ghi và file dữ liệu. Một cụm Bigtable hoạt động trong một nhóm các máy được chia sẻ, các máy này chạy nhiều ứng dụng phân tán khác nhau, và các tiến trình Bigtable thường chia sẻ máy tính với tiến trình từ các ứng dụng khác. Bigtable phụ thuộc vào hệ thống quản lý cụm trong việc lên lịch công việc, quản lý tài nguyên khi chia sẻ, giải quyết sự cố, và kiểm tra trạng thái của máy. Định dạng file Google SStable được sử dụng để lưu trữ dữ liệu Bigtable. Một SSTable cung cấp một bản đồ liên tục, và thứ tự không đổi từ các khóa tới các giá trị, nơi mà cả khóa và giá trị đều là các xâu bất kì. Các phép toán được cung cấp để tìm kiếm giá trị liên quan đến khóa được chỉ rõ, và để lặp lại tất cả các cặp khóa/giá trị trong một dãi khóa được chỉ ra. Hơn nữa, mỗi Sstable mang một chuỗi các block (mỗi block có kích thước 64KB, có thể điều chỉnh được). Một chỉ mục block (lưu tại cuối của Sstable) được sử dụng để định vị block; chỉ mục được tải vào bộ nhớ khi

SStable được mở. Bigtable dựa vào một dịch vụ khóa phân tán có tính sẵn sàng cao gọi là Chubby. Một dịch vụ Chubby bao gồm 5 mô hình hoạt động, một trong số chúng được chọn làm chủ và đáp ứng các yêu cầu. Dịch vụ này chỉ "sống" khi phần lớn các mô hình đang chạy và có giao tiếp với các mô hình khác. Bigtable sử dụng Chubby để: bảo đảm chỉ có duy nhất một mô hình chủ tại mọi thời điểm; để lưu trữ vị trí khởi động của dữ liệu Bigtable để lưu trữ thông tin lược đồ Bigtable và để lưu trữ danh sách điều khiển truy xuất.

3.5 Thực thi

Thực thi Bigtable có ba thành phần chính: một thư viện được kết nối tới mọi client, một máy chủ, và nhiều máy chủ phụ. Máy chủ phụ có thể được thêm hoặc gỡ bỏ động từ một cụm để điều tiết những thay đổi của tải làm việc. Máy chủ chính có trách nhiệm chỉ định các bảng phụ (tablet) vào các máy chủ phụ, phát hiện sự bổ sung cũng như mở rộng của máy chủ phụ, cân bằng tải, và loại bỏ file trong GFS. Thêm vào đó, nó điều khiển những thay đổi lược đồ ví dụ như việc tạo ra các bảng hay các họ cột. Mỗi máy chủ phụ quản lý một tập các bảng phụ. Máy chủ phụ quản lý các yêu cầu đọc và ghi vào các bảng con đã được tải, và chia nhỏ các bảng khi chúng quá lớn. Như với các hệ thống lưu trữ phân tán một máy chủ, dữ liệu khách không được đưa qua máy chủ, client giao tiếp trực tiếp với các máy chủ phụ để đọc và ghi. Bởi client Bigtable không phụ thuộc vào máy chủ về thông tin vị trí các bảng phụ, hầu hết client không bao giờ giao tiếp với máy chủ. Do đó, máy chủ không phải chịu tải lớn. Một cụm Bigtable lưu trữ một số bảng. Mỗi bảng gồm có một tập các bảng phụ, và mỗi bảng phụ mang toàn bộ dữ liệu kết hợp với một dải các hàng. Khởi đầu mỗi bảng chỉ gồm một bảng phụ và khi phát triển, nó tự động chia thành nhiều bảng phụ, với kích thước tiêu chuẩn trong khoảng 100-200Mb.

3.6 Chỉ định bảng phụ

Mỗi bảng phụ được phân vào một máy chủ phụ vào một thời điểm. Máy chủ chính lưu vết các thiết lập của máy chủ phụ đang hoạt động, và sự phân công hiện tại của các bảng phụ tới các máy chủ, bao gồm bảng phụ nào chưa được chỉ định. Khi một bảng phụ không được chỉ định, và một máy chủ phụ có đủ khả năng cho bảng phụ sẵn sàng, máy chủ chính sẽ phân công bảng phụ bằng cách gửi một yêu cầu tải bảng phụ tới máy chủ phụ. Bigtable sử dụng Chubby để lưu vết các máy chủ phụ. Khi một máy chủ phụ khởi động, nó tạo ra, và yêu cầu một khóa dành riêng, một file với tên duy nhất trong thư mục riêng Chubby. Máy chủ chính giám sát thư mục này (gọi là server directory) để phát hiện ra các máy chủ phụ. Một máy chủ phụ ngừng phục vụ nếu nó mất khóa của nó: ví dụ, do việc phân chia mạng làm cho máy chủ mất phiên làm việc Chubby của nó. Một máy chủ phụ sẽ cố gắng giành lại một khóa dành riêng trên file của nó chỉ cần file đó còn tồn tại. Nếu file không còn tồn tại, máy chủ phụ không bao giờ có thể phục vụ trở lại, vì thế nó tự ngừng hoạt động. Bất cứ khi nào một máy chủ

phụ ngừng hoạt động (ví dụ, do hệ thống quản lý cụm gỡ bỏ máy chủ ra khỏi cụm) nó cố gắng giải phóng khóa của nó nhờ đó máy chủ chính có thể chỉ định lại những bảng phụ này nhanh chóng hơn.

Máy chủ chính có trách nhiệm phát hiện khi một máy chủ phụ không còn phục vụ các bảng phụ của nó, và phân công lại bảng phụ sớm nhất có thể. Để phát hiện khi một máy chủ phụ ngừng phục vụ, máy chủ chính hỏi một cách định kì mỗi máy chủ phụ trạng thái khóa của nó.

Tập các bảng phụ đang tồn tại chỉ thay đổi khi một bảng được tạo ra hay xóa đi, hai bảng phụ đang tồn tại được gộp thành một bảng phụ lớn hơn, hoặc một bảng phụ bị chia thành hai bảng phụ nhỏ hơn. Máy chủ chính có thể lưu vết những thay đổi này. Những bảng phụ bị chia cắt được đối xử đặc biệt khi chúng được khởi tạo bởi một máy chủ phụ. Máy chủ phụ thực thi việc tách bằng cách ghi lại thông tin cho bảng phụ mới trong bảng Metadata. Khi một hoạt động tách được chuyển giao, nó báo cho máy chủ chính. Trong trường hợp thông báo bị mất (do máy chủ chính hoặc phụ lỗi), máy chủ chính phát hiện ra bảng phụ mới bằng cách yêu cầu một máy chủ phụ tải bảng phụ bị tách. Máy chủ phụ báo lại cho máy chủ chính về việc chia tách.

3.7 Phục vụ bảng phụ

Trạng thái liên tục của bảng phụ được lưu tại GFS. Cập nhật được thực thi vào một bản ghi thực thi lưu trữ các bản ghi làm lại. Những lần cập nhật này, những cập nhật gần hơn được lưu trong bộ nhớ đệm được sắp xếp gọi là memtable, những cập nhật cũ hơn được lưu trữ theo trình tự trong Sstable. Để phát hiện ra một bảng phụ, một máy chủ phụ đọc dữ liệu metadata của nó từ bảng Metadata. Dữ liệu metadata này chứa danh sách Sstable bao gồm một bảng phụ và một tập các điểm làm lại, chúng là những con trỏ trỏ vào bất kì bản ghi thực thi nào có thể chứa dữ liệu của bảng phụ. Máy chủ phụ đọc những chỉ số của SSTable vào bộ nhớ và tổ chức lại memtable bằng cách áp dụng tất cả những cập nhật được thực thi từ điểm làm lại. Khi thực hiện ghi trên máy chủ phụ, máy chủ phụ kiểm tra rằng nó được định dạng tốt (well-formed), và người gửi được cho phép thực hiện sự thay đổi. Sự cho phép được thực hiện bằng cách đọc danh sách những người ghi đã được cho phép từ file Chubby. Một thay đổi hợp lệ được viết vào bản ghi thực thi. Nhóm thực thi được sử dụng để tăng thông lượng của nhiều thay đổi nhỏ. Sau khi ghi hoàn tất, nội dung của nó đã được chèn vào memtable. Khi thực hiện đọc trên máy chủ phụ, nó cũng kiểm tra định dạng tốt và quyền hạn tương tự. Một hoạt động đọc hợp lệ được thực thi trên một khung nhìn hợp nhất của chuỗi của SStable và memtable.

3.8 Nén

Do việc thực thi hoạt động ghi, kich thước của memtable tăng lên. Khi kích thước của memtable đạt đến giới hạn, memtable sẽ đóng băng, một memtable mới

được tạo ra, và memtable bị đóng băng được chuyển vào SStable và ghi vào GFS. Bộ xử lý nén nhỏ này có 2 mục đích: nó rút ngắn bộ nhớ sử dụng bởi máy chủ phụ, và giảm lượng dữ liệu được đọc từ bản ghi thực thi trong quá trình hồi phục nếu máy chủ phụ bị lỗi. Hoạt động đọc và ghi sắp tới có thể tiếp diễn khi đang nén.

Mọi bộ nén nhỏ đều tạo ra một SStable mới. Nếu chế độ này không được kiểm tra liên tục, các hoạt động đọc có thể cần phải kết hợp với cập nhật từ một số bất kì của Sstable. Thay vào đó, chúng ta giới hạn số file bằng cách thực thi định kì việc nén gộp (merging compaction) trên nền. Nén gộp đọc nội dung của một vài Sstable và memtable, và ghi ra một SStable mới. SSable và memtable đầu vào có thể được loại bỏ sớm nhất khi việc nén hoàn thành. Nén gộp ghi lại tất cả Sstalbe vào chính xác một SStable được gọi là nén lớn. Sstable tạo ra bởi nén non-major có thể chứa những mục xóa đặc biệt, cấm việc xóa dữ liệu trong SStalbe cũ hơn khi chúng vẫn còn hoạt động. Một bộ nén lớn, mặt khác, tạo ra một SStable không chứa thông tin xóa hay dữ liệu đã bị xóa. Bigtable quay vòng qua tất cả bảng phụ của nó và áp dụng nén lớn một cách đều đặn lên chúng. Nén lớn cho phép Bigtable phục hồi tài nguyên sử dụng bởi dữ liệu đã bị xóa, và cũng cho phép nó để đảm bảo rằng những dữ liệu đã bị xóa biến mất khỏi hệ thống, điều này rất quan trọng để máy chủ lưu trữ những thông tin nhạy cảm.

4. KẾT LUẬN

Mục đích bài viết này là giới thiệu giải pháp lưu trữ các Bigdata và các loại dữ liệu khác, một công việc thường xuyên và quan trọng trong các trung tâm dữ liệu (data centers). Bài báo cũng bàn về cơ chế xử lý cho các Big data và các hoạt động của hai thành phần chính của Hadoop: HDFS và MapReduce để người đọc hiểu được nguyên tắc làm việc trên các Bigtable với nền tảng Hadoop. Việc tìm hiểu các công cụ này sẽ mang lại nhiều lựa chọn giải pháp lưu trữ và cho việc phát triển ứng dụng phân tán với các mục đích khác nhau.

TÀI LIỆU THAM KHẢO

[1] Big Data IBM, (2014): http://www-01.ibm.com/software/data/bigdata/

[2] Danah boyd, Kate Crawford, (2012), Critical questions for big data http://www.tandfonline.com/doi/pdf/10.1080/1369118X.2012.678878

[3] Chandar, T., Griesemer, R., and Redstone, J. Paxos made live (2007), An engineering perspective. In Proc. of PODC (2007).

[4] Christof Strauch, University Hochschule der Medien, Stuttgart (Stuttgart Media University), (2013), "NoSQL Databases"

[5] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber, (2012), "Bigtable: A Distributed Storage System for Structured Data"

[6] Oracle NoSQL Database, (2011), An Oracle White Paper September 2011. Maqsood Alam - Oracle NoSQL Database, "Oracle Platform Technology Solutions".

[7] Understanding Big Data, (2013), Chapter 1, Pages 5, Pages 8 https://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Understanding+Big+Data

[8] http://thenextweb.com/insider/2012/08/23/amazons-cto-here-two-definitions-big-data/

[9] http://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf

BIG DATA AND STRUCTURE STORAGE SOLUTION

Nguyen Mau Han

Faculty of Information Technology, University of Sciences, Hue University

Email: nmhan2009@gmail.com

ABSTRACT

Big Data is known as an ideal solution for processing structured data, semi-structured data or even unstructured data such as weblogs, social networks, e-mail, sensitive data, and images that can be exploited to find useful information. The question is any solution to the problem of storage for this kind of data. This article proposes a storage solution for data centers, which are working on the Big Data daily, while traditional storage methods have exposed many defects.

Keywords: Big Data, , Haddop, MapReduce, structured data.

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.