Trường Đại Học Sư Phạm Kỹ Thuật TP. Hồ Chí Minh

Khoa Công Nghệ Thông Tin

BÁO CÁO

ĐIỆN TOÁN ĐÁM MÂY

ĐỀ TÀI:

TÌM HIỂU VỀ APACHE HADOOP VÀ VIẾT ỨNG DỤNG DEMO

Nhóm sinh viên thực hiện:

Phan Thành Đạt **18133006**

Nguyễn Anh Triều **18133058**

GVHD: TS. Huỳnh Xuân Phụng

Tp. Hồ Chí Minh-05/01/2020

ĐIỂM SỐ

TIÊU CHÍ	NỘI DUNG	TRÌNH BÀY	TỔNG
ĐIỂM			

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Đánh giá: …….

……

……

……

……

……

……

Giảng viên hướng dẫn

(ký và ghi rõ họ tên)

PHUNG

Huỳnh Xuân Phụng

LỜI CẢM ƠN

Để hoàn thành tốt đề tài và bài báo cáo này, chúng em xin gửi lời cảm ơn chân thành đến giảng viên Huỳnh Xuân Phụng đã hỗ trợ chúng em tận tình trong suốt quá trình làm báo cáo, chúng em cảm thấy thầy thật sự tận tình với sinh viên, luôn giúp chúng em giải quyết mọi thắc mắc và lỗi trong quá trình làm đề tài. Đề tài và bài báo cáo được chúng em thực hiện trong khoảng thời gian ngắn, sẽ còn nhiều thiếu sót về mặt kỹ thuật và kiến thức, em rất mong thầy có thể bỏ qua và có thể cho chúng em những đóng góp ý kiến tốt để định hướng được tốt hơn cho các đề tài sắp tới và cũng như sau này. Một lần nữa nhóm em xin cảm ơn thầy và chúc thầy thật nhiều niềm vui và hạnh phúc.

TP.HCM, ngày 01 tháng 01 năm 2021

Nhóm sinh viên thực hiện

MỤC LỤC

Danh mục các hình

Danh mục các bảng

Chương 1: Tổng quan chương trình

1. Giới thiệu chung

1.1. Tìm hiểu đề tài

1.1.1. Lí do chọn đề tài

Trong thời đại công nghệ thông tin hiện nay, dữ liệu ở khắp mọi nơi, không ngành nghề nào cũng dùng đến dữ liệu, vì vậy dữ liệu ngày một càng lớn và đa dạng các hệ thống cũ xử lý không được ổn định, vì vậy cần có những hệ thống mới để xử lý nguồn dữ liệu lớn, và Hadoop ra đời để đáp ứng được điều này, nó giúp việc quản lý và xử lý dữ liệu tốt hơn. Chúng em cảm thấy đề tài này là phù hợp với chúng em và thỏa được niềm đam mê tìm hiểu về dữ liệu, nhóm chúng em xin chọn đề tài này để thực hiện.

1.1.2. Mô tả đề tài

Nắm rõ được cái khái niệm liên quan đến hệ thống Hadoop:

- Tìm hiểu về các khái niệm trong Hadoop.

- Tìm hiểu hệ thống HDFS.

- Tìm hiểu về Map-Reduce.

- Tìm hiểu về Yarn.

- Cài đặt hệ thống hadoop.

- Cài đặt phần mềm SH kết nối với Node trong cụm.

Viết các hàm hỗ trợ bằng ngôn ngữ java để đếm được số lần đạt học bổng của sinh viên Trường Đại học Sư phạm Kỹ Thuật TP.HCM đạt được trong 2 năm gần đây.

1.2. Yêu cầu đề tài

1.2.1. Yêu cầu kỹ thuật

- Hiểu được các thành phần trong hadoop.

- Hiểu được các chức năng của hệ thống HDFS và Map-Reduce.

- Cài đặt được hệ thống hadoop.

- Cài đặt SSH và share Key cho Master và Slave kết nối qua lại với nhau.

1.2.2. Công cụ và công nghệ sử dụng

- Sử dụng hệ điều hành Ubuntu desktop.

- Sử dụng hệ thống Apache Hadoop 2.7.7.

- Sử dụng phần mềm Intellij để viết ứng dụng.

Chương 2: Kế hoạch thực hiện

1. Kế hoạch

Table 1: Kế hoạch theo tuần

Tuần	Công việc
9	Tìm hiểu về đề tài và lên kế hoạch thực hiện đề tài
10	Tìm hiểu các khái niệm cơ bản trong Hadoop và Cài đặt 1 cụm Hadoop 2 Node
11	Củng cố kiến thức về hệ thống HDFS
12	Củng cố kiến thức về hệ thống Map-Reduce
13	Củng cố kiến thức về hệ thống Yarn
14	Củng cố kiến thức về hệ thống Common, viết chương trình ứng dụng Demo
15	Kiểm thử và viết báo cáo

2. Phân công công việc

Table 2:Phân công công việc & đóng góp của mỗi sinh viên

STT	Tên sinh viên	Mô tả công việc	Đóng góp
1	Phan Thành Đạt	- Xây dựng kế hoạch - Tìm hiểu và Apache Hadoop - Viết hàm reduce, chuẩn bị dữ liệu - Chuẩn bị báo cáo - Chuẩn bị video	50%
2	Nguyễn Anh Triều	- Xây dựng kế hoạch - Tìm hiểu về ApacheHadoop - Viết hàm map, runner - Chuẩn bị powerpoint - Chuẩn bị nội dung video	50%

Chương 3: Apache Hadoop

1. Lý thuyết chung

- Apache Hadoop là phần mềm mã nguồn mở (software framework) được xây dựng bởi Doug khi làm việc tại Nutch, để giải quyết việc quản lý dự liệu tập trung, chính xác hơn là để đáp ứng việc tìm kiếm dữ liệu trên website. Phần mềm được đặt tên dự theo con voi đồ chơi của con trai tác giả Doug. Hadoop trở thành một trong những công nghệ phổ biến nhất cho việc lưu trữ dữ liệu bao gồm có cấu trúc, bán cấu trúc và phi cấu trúc.

- Hadoop được bắt nguồn từ các bài báo MapReduce của Google và Google File System (GFS). Hiện nay Hadoop được phát hành theo Giấy phép Apache 2.0.

-Nguồn gốc của Hadoop đến từ các bài viết File System Google được xuất bản vào tháng 10 năm 2003. Bài viết này là một bài nghiên cứu được sản sinh ra từ Google – MapReduce: Xử lý dữ liệu đơn giản trên các cluster lớn. Bắt đầu phát triển trên dự án Apache Nutch, nhưng đã được chuyển qua dự án con Hadoop mới trong tháng 1 năm 2006. Doug Cutting đã làm việc tại Yahoo! vào thời điểm đó, đặt tên Hadoop theo tên của con voi đồ chơi của con trai mình.

- Hadoop 0.1.0 được phát hành vào tháng 4 năm 2006 và tiếp tục phát triển bởi nhiều người đóng góp đến dự án Apache Hadoop. Hadoop frameword gồm hai layer chính: Hadoop Distributed File System (HDFS), Map-Reduce.

2. Tìm hiểu về Apache Hadoop

2.1. Khái niệm

Hadoop là một Apache framework mã nguồn mở cho phép phát triển các ứng dụng phân tán (distributed processing) để lưu trữ và quản lý các tập dữ liệu lớn. Hadoop được viết bằng Java tuy nhiên vẫn hỗ trợ C++, Python, Perl bằng cơ chế streaming.

Hadoop có một cấu trúc liên kết master-slave. Trong cấu trúc này, chúng ta có một node master và nhiều node slave. Chức năng của node master là gán một tác vụ cho các node slave khác nhau và quản lý tài nguyên. Các node slave là máy tính thực tế có thể không mạnh lắm.

2.2. Lý do sử dụng Hadoop

- Không cần phần cứng đặc biệt để chạy Hadoop.
- Tính bảo mật dữ liệu cao
- Khi 1 node lỗi, nền tảng Hadoop tự động chuyển sang node khác.
- Hadoop được xây dựng với tiêu chí xử lý dữ liệu có cấu trúc và không cấu trúc.

2.3. Hoạt động của Hadoop

Giai đoạn 1

Một user hay một ứng dụng có thể submit một job lên Hadoop (hadoop job client) với yêu cầu xử lý cùng các thông tin cơ bản:

- Nơi lưu (location) dữ liệu input, output trên hệ thống dữ liệu phân tán.
- Các java class ở định dạng jar chứa các dòng lệnh thực thi các hàm map và reduce.

- Các thiết lập cụ thể liên quan đến job thông qua các thông số truyền vào.

Giai đoạn 2

Hadoop job client submit job (file jar, file thực thi) và các thiết lập cho JobTracker. Sau đó, master sẽ phân phối tác vụ đến các máy slave để theo dõi và quản lý tiến trình các máy này, đồng thời cung cấp thông tin về tình trạng và chẩn đoán liên quan đến job-client.

Giai đoạn 3

TaskTrackers trên các node khác nhau thực thi tác vụ MapReduce và trả về kết quả output được lưu trong hệ thống file.

Khi "chạy Hadoop" có nghĩa là chạy một tập các trình nền – daemon, hoặc các chương trình thường trú, trên các máy chủ khác nhau trên mạng của bạn. Những trình nền có vai trò cụ thể, một số chỉ tồn tại trên một máy chủ, một số có thể tồn tại trên nhiều máy chủ.

2.4. Giải quyết vấn đề

- Xử lý và làm việc khối lượng dữ liệu khổng lồ tính bằng Petabyte.
- Xử lý trong môi trường phân tán, dữ liệu lưu trữ ở nhiều phần cứng khác nhau, yêu cầu xử lý đồng bộ.
- Xử lý các lỗi xuất hiện thường xuyên.

2.5. Các thành phần chính trong Hadoop

Hadoop gồm 4 thành phần chính:

+ Map-Reduce (Distributed Computation)
+ HDFS (Distributed Storage)
+ Yarn Framwork

+ Common Utilities

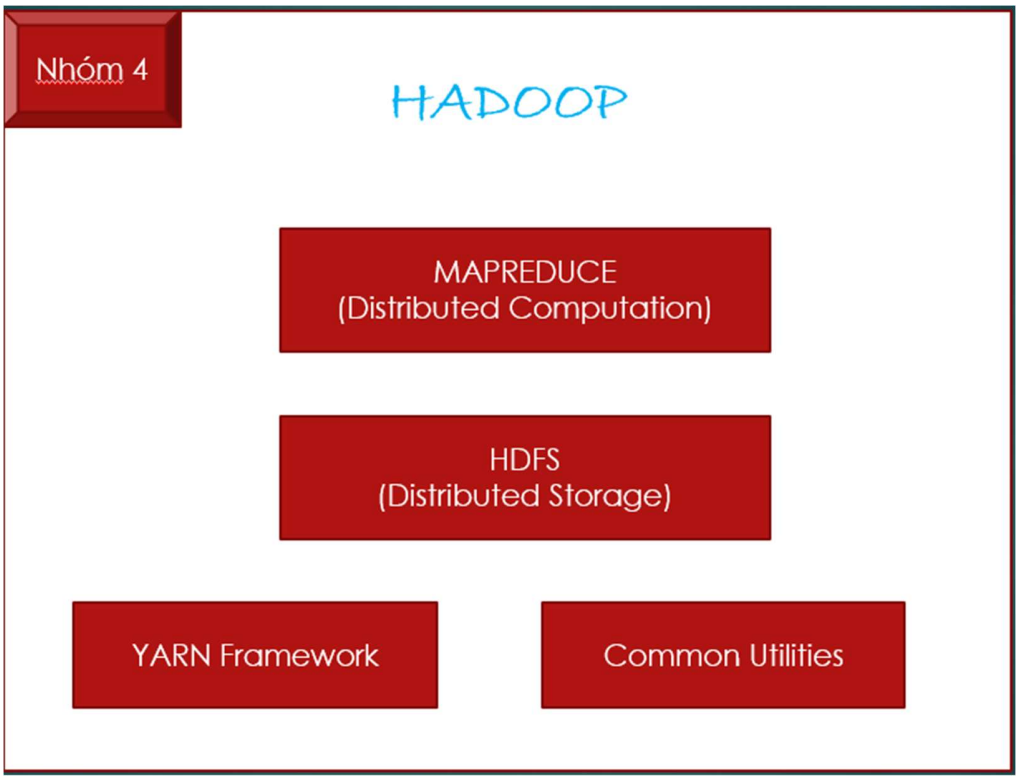

Figure 1: Mô hình Hadoop

2.5.1. Hadoop Distributed File System (HDFS)

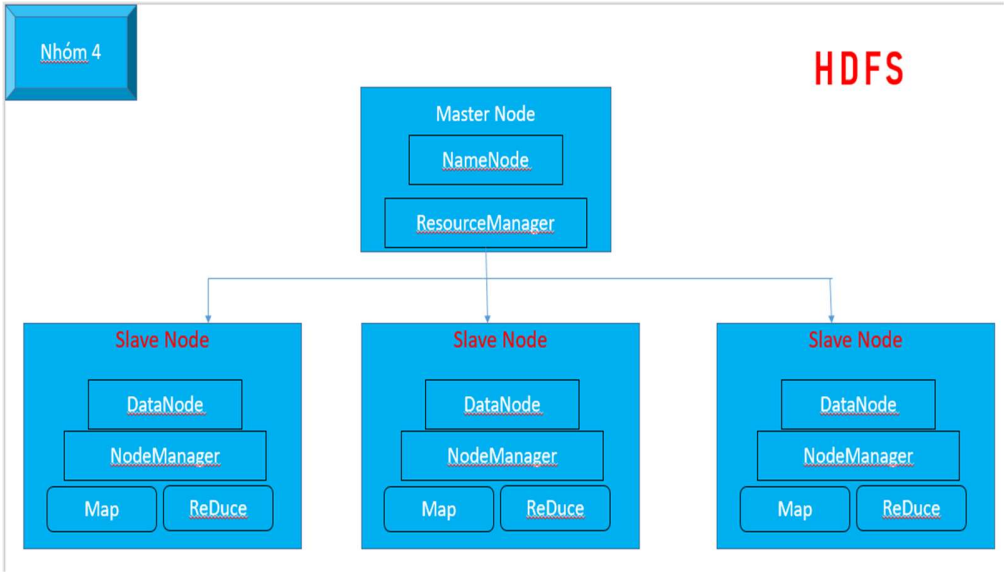

Figure 2: Mô hình HDFS

Đây là hệ thống file phân tán cung cấp truy cập thông lượng cao cho ứng dụng khai thác dữ liệu. Hadoop Distributed File System (HDFS) là hệ thống tập tin ảo. Khi chúng ta di chuyển 1 tập tin trên HDFS, nó tự động chia thành nhiều mảnh nhỏ. Các đoạn nhỏ của tập tin sẽ được nhân rộng và lưu trữ trên nhiều máy chủ khác để tăng sức chịu lỗi và tính sẵn sàng cao.

HDFS sử dụng kiến trúc master/slave, trong đó master gồm một NameNode để quản lý hệ thống file metadata và một hay nhiều slave DataNodes để lưu trữ dữ liệu thực tại.

NameNode chạy trên máy chủ Master, có tác vụ quản lý Namespace và điều chỉnh truy cập tệp của client. NameNode có tác vụ chính là đóng, mở và đổi tên cho

các tập tin, thư mục. Ngoài ra, nó còn điều chỉnh cho các truy cập đến hệ thống tập tin.

DataNode chạy trên các nút Slave. có tác vụ lưu trữ business data thực tế. DataNode có tác vụ là ghi, đọc vào hệ thống tập tin; nó còn có nhiệm vụ tạo, xóa, nhân rộng các dữ liệu dựa trên chỉ dẫn của NameNode.

Một tập tin với định dạng HDFS được chia thành nhiều block và những block này được lưu trữ trong một tập các DataNodes, kích thước 1 block thông thường là 128MB, kích thước này có thể thay đổi được bằng việc cấu hình.

Ưu điểm nổi bật của HDFS:

- HDFS cho phép dữ liệu có thể phân tán: Điều này có thể hiểu như sau: Nếu như có một cụm Hadoop mà trong đó bao gồm 60 máy tính thì chỉ cần đưa một file dữ liệu vào HDFS. Khi đó, thì file sẽ tự động được chia nhỏ thành nhiều phần rồi được lưu trữ ở 60 máy tính đó.
- HDFS cho phép tính toán và phân tán song song: Thay vì chỉ sử dụng một máy để xử lý công việc, thì với HDFS có thể để các máy hoạt động song song để xử lý chung một công việc để tiết kiệm thời gian.
- HDFS cho phép nhân bản các file: Đặc điểm này sẽ giúp đề phòng được các trường hợp một máy tính trong cụm Hadoop phát sinh sự cố thì dữ liệu sẽ được backup lại mà không bị mất.
- HDFS có thể mở rộng theo chiều dọc: Lúc này, sẽ có nhiệm vụ nâng cấp cho các hệ thống bằng cách tăng cấu hình cho máy tính lên. Tính năng này còn được gọi là Scale Up, Vertical scaling.
- HDFS sở hữu khả năng mở rộng hệ thống theo chiều ngang: Đặc điểm này có nghĩa rằng, không cần phải nâng cấp cho phần cứng mà chỉ cần mua thêm một chiếc máy tính mới để chia sẻ với chiếc máy hiện tại là được.

- Được thiết kế đặc biệt cho các ứng dụng xử lý dạng khối**.** Các file khi được tạo ra trên hệ thống HDFS đều sẽ được ghi, đóng lại và không thể chỉnh sửa được nữa. Điều này sẽ giúp bảo mật, đảm bảo tính nhất quán cho các tập tin của dữ liệu.

HDFS giúp giải quyết những vấn đề:

- Có 3 vấn đề mà khi sử dụng HDFS bạn đều có thể giải quyết một cách triệt để là:

- Lỗi phần cứng thường xuyên xảy ra

- HDFS được tích hợp với hệ thống phát hiện ra lỗi, tự động khôi phục lỗi và chống chịu lỗi. Các tính năng này sẽ giúp bạn giảm rủi ro xuống mức thấp nhất khi một hệ thống phát sinh lỗi phần cứng quá bất ngờ.

- Phân chia tập dữ liệu thành tập dữ liệu có dung lượng ít hơn

Quá trình hoạt động của NameNode và DataNode như sau:

NameNode: Có trách nhiệm điều phối cho các thao tác truy cập của client với hệ thống HDFS. Bởi vì các DataNode là nơi lưu trữ thật sự các block của các file trên HDFS nên chúng là nơi đáp ứng các truy cập này. NameNode sẽ thực hiện nhiệm vụ của nó thông qua daemon tên namemode chạy trên port 8021.

DataNode: DataNode server sẽ chạy một daemon datanode trên port 8022, theo định kỳ thì mỗi DataNode sẽ có nhiệm vụ báo cáo cho Namenode biết được danh sách tất cả các block mà nó đang lưu trữ để NameNode có thể dựa vào nó để cập nhật lại các metadata trong nó.

Sau mỗi lần cập nhật thì metadata trên NameNode đều sẽ đạt được các tình trạng thống nhất dữ liệu trên các DataNode. Toàn bộ trạng thái của metadata trên NameNode sẽ đạt được sự thống nhất với các dữ liệu ở trên DataNode. Tất cả các trạng thái của metadata ngay khi đang ở tình trạng hệ thống này sẽ được gọi là checkpoint. Mỗi một metadata ở trạng thái checkpoint đều sẽ được sử dụng cho mục đích nhân bản metadata với mục đích phục hồi lại NameNode nếu như NameNode xuất hiện lỗi.

Đọc file trên HDFS: Khi các máy con client gửi yêu cầu đọc đến NameNode, khi đó NameNode nhận được phải sẽ thực hiện các tác vụ để kiểm tra xem file có tồn tại không, file có bị lỗi hoặc bị nhiễm virus không? Nếu như file không có vấn đề thì NameNode sẽ gửi các danh sách của các Block của file cùng với địa chỉ của các DataNode. Sau đó, hệ thống sẽ mở kết nối với DataNode rồi thực hiện chức năng RPC để nhận được các dữ liệu cần đọc, rồi đóng kết nối với DataNode còn lại. Khi đó, các client đọc các block của file liên tục và lặp lại cho đến block cuối của file. Tiếp theo, lập trình viên sẽ sử dụng một tập tin API của Hadoop để có thể tương tác trực tiếp được với HDFS. Những tập API sẽ có chức năng giấu đi các NameNode để giúp kết nối với các DataNode để nhận được dữ liệu.

Ghi file trên HDFS: Đầu tiên, client sẽ gửi yêu cầu đến NameNode tạo một file entry lên File System Namespace. File mới được tạo sẽ rỗng, tức chưa có một block nào. Sau đó, NameNode sẽ quyết định danh sách các DataNode sẽ chứa các bản sao của file cần gì và gửi lại cho client Client sẽ chia file cần gì ra thành các block, và với mỗi block client sẽ đóng gói thành một packet. Lưu ý là mỗi block sẽ được lưu ra thành nhiều bản sao trên các DataNode khác nhau (tuỳ vào chỉ số độ nhân bản của file).
Client gửi packet cho DataNode thứ nhất, DataNode thứ nhất sau khi nhận được packet sẽ tiến hành lưu lại bản sao thứ nhất của block. Tiếp theo DataNode thứ nhất sẽ gửi packet này cho DataNode thứ hai để lưu ra bản sao thứ hai của block.

Tương tự
DataNode thứ hai sẽ gửi packet cho DataNode thứ ba. Cứ như vậy, các DataNode cũng lưu các bản sao của một block sẽ hình thành một ống dẫn dữ liệu data pipe. Sau khi DataNode cuối cùng nhận thành được packet, nó sẽ gửi lại cho DataNode thứ hai một gói xác nhận rằng đã lưu thành công . Và gói thứ hai lại gửi gói xác nhận tình trạng thành công của hai DataNode về DataNode thứ nhất.
Client sẽ nhận được các báo cáo xác nhận từ DataNode thứ nhất cho tình trạng thành công của tất cả DataNode trên data pipe.

Nếu có bất kỳ một DataNode nào bị lỗi trong quá trình ghi dữ liệu, client sẽ tiến hành xác nhận lại các DataNode đã lưu thành công bản sao của block và thực hiện một hành vi ghi lại block lên trên DataNode bị lỗi.
Sau khi tất cả các block của file đều đã đươc ghi lên các DataNode, client sẽ thực hiên một thông điệp báo cho NameNode nhằm cập nhật lại danh sách các block của file vừa tạo. Thông tin Mapping từ Block ID sang danh sách các DataNode lưu trữ sẽ được NameNode tự động cập nhật bằng các định kỳ các DataNode sẽ gửi báo cáo cho NameNode danh sách các block mà nó quản lý.

Lưu ý: NameNode và DataNode đều là phần mềm được thiết kế nhằm mục đích chạy trên máy chủ và chúng được viết bằng Java.

2.5.2. Hadoop MapReduce

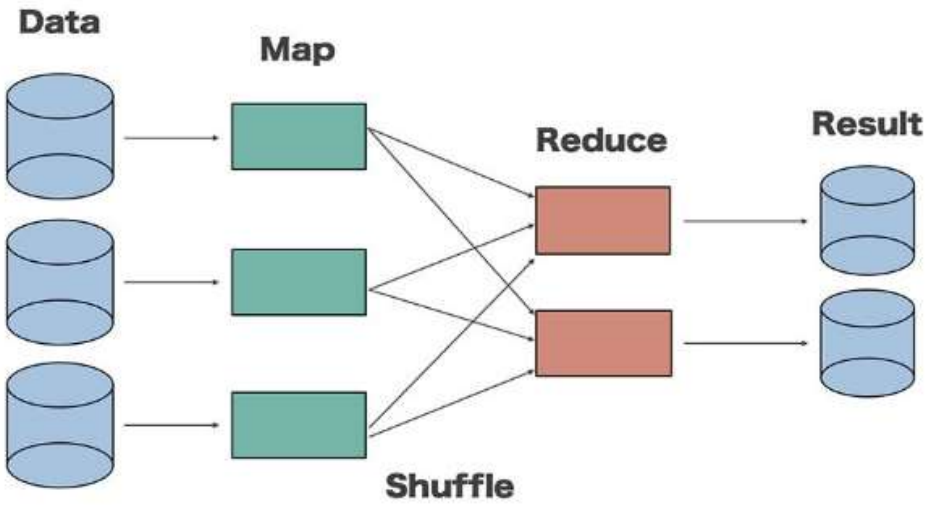

Figure 3: Mô hình MapReduce

Mapreduce là một mô hình lập trình, thực hiện quá tình xử lý tập dữ liệu lớn.

❖ Gồm 2 phần chính: Map và Reduce

Hàm Map : Các xử lý một cặp (key, value) để sinh ra một cặp (keyI, valueI) - key và value trung gian. Dữ liệu này input vào hàm Reduce.

Hàm Reduce: Tiếp nhận các(keyl,valuel) và trộn các cặp (keyl,valuel) trung gian, lấy ra các valuel có cùng keyl.

Đây là hệ thống dựa trên YARN dùng để xử lý song song các tập dữ liệu lớn. Là cách chia một vấn đề dữ liệu lớn hơn thành các đoạn nhỏ hơn và phân tán nó trên

nhiều máy chủ. Mỗi máy chủ có 1 tập tài nguyên riêng và máy chủ xử lý dữ liệu trên cục bộ. Khi máy chủ xử lý xong dữ liệu, chúng sẽ gởi trở về máy chủ chính.

MapReduce gồm một single master (máy chủ) JobTracker và các slave (máy trạm) TaskTracker trên mỗi cluster-node. Master có nhiệm vụ quản lý tài nguyên, theo dõi quá trình tiêu thụ tài nguyên và lập lịch quản lý các tác vụ trên các máy trạm, theo dõi chúng và thực thi lại các tác vụ bị lỗi. Những máy slave TaskTracker thực thi các tác vụ được master chỉ định và cung cấp thông tin trạng thái tác vụ (task-status) để master theo dõi. JobTracker là một điểm yếu của Hadoop Mapreduce.

Mapreduce được ưa chuộng sử dụng như vậy bởi nó sở hữu nhiều ưu điểm vượt trội như sau:

- MapReduce có khả năng xử lý dễ dàng mọi bài toán có lượng dữ liệu lớn nhờ khả năng tác vụ phân tích và tính toán phức tạp. Nó có thể xử lý nhanh chóng cho ra kết quả dễ dàng chỉ trong khoảng thời gian ngắn.
- Mapreduce có khả năng chạy song song trên các máy có sự phân tán khác nhau. Với khả năng hoạt động độc lập kết hợp phân tán, xử lý các lỗi kỹ thuật để mang lại nhiều hiệu quả cho toàn hệ thống.
- MapRedue có khả năng thực hiện trên nhiều nguồn ngôn ngữ lập trình khác nhau như: Java, C/ C++, Python, Perl, Ruby,… tương ứng với nó là những thư viện hỗ trợ.
- Các ứng dụng MapReduce dần hướng đến quan tâm nhiều hơn cho việc phát hiện các mã độc để có thể xử lý chúng. Nhờ vậy, hệ thống mới có thể vận hành trơn tru và được bảo mật nhất.

Các bước hoạt động của MapReduce

- Bước 1: Tiến hành chuẩn bị các dữ liệu đầu vào để cho Map() có thể xử lý.
- Bước 2: Lập trình viên thực thi các mã Map() để xử lý.

- Bước 3: Tiến hành trộn lẫn các dữ liệu được xuất ra bởi Map() vào trong Reduce Processor
- Bước 4: Tiến hành thực thi tiếp mã Reduce() để có thể xử lý tiếp các dữ liệu cần thiết.
- Bước 5: Thực hiện tạo các dữ liệu xuất ra cuối cùng.

Luồng dữ liệu nền tảng của Mapreduce

- Input Reader
- Map Function
- Partition Function
- Compare Function
- Reduce Function
- Output Writer

2.5.3. Hadoop YARN

YARN (Yet-Another-Resource-Negotiator) là một framework hỗ trợ phát triển ứng dụng phân tán, YARN cung cấp daemons và APIs cần thiết cho việc phát triển ứng dụng phân tán, đồng thời xử lý và lập lịch sử dụng tài nguyên tính toán (CPU hay memory) cũng như giám sát quá trình thực thi các ứng dụng đó.

Bên trong YARN, chúng ta có hai trình quản lý ResourceManager và NodeManage

+ ResourceManager: Quản lý toàn bộ tài nguyên tính toán của cluster.

+ NodeManger: Giám sát việc sử dụng tài nguyên của container và báo cáo với ResourceManger. Các tài nguyên ở đây là CPU, memory, disk, network, ...

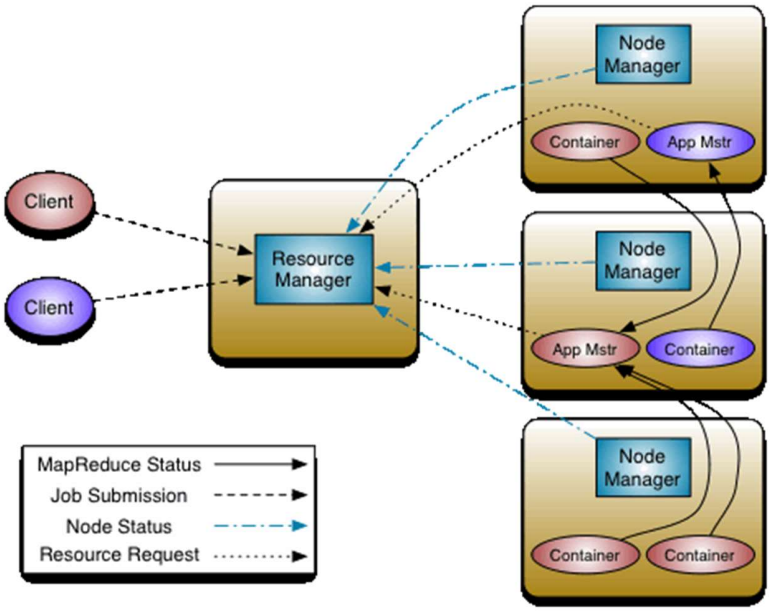

Figure 4: Mô hình Yarn

Quá trình 1 ứng dụng chạy trên YARN được mô tả bằng sơ đồ trên qua các bước sau:

Client giao 1 task cho Resource Manager

Resource Manager tính toán tài nguyên cần thiết theo yêu cầu của ứng dụng và tạo 1 App Master (App Mstr). Application Master được chuyển đến chạy 1 một node tính toán. Application Master sẽ liên lạc với các NodeManager ở các node khác để ra yêu cầu công việc cho node này.

Node Manager nhận yêu cầu và chạy các task trên container

Các thông tin trạng thái thay vì được gửi đến JobTracker sẽ được gửi đến App Master.

ResourceManger có hai thành phần quan trọng đó là Scheduler và ApplicationManager

Scheduler có trách nhiệm phân bổ tài nguyên cho các ứng dụng khác nhau. Đây là Scheduler thuần túy vì nó không thực hiện theo dõi trạng thái cho ứng dụng. Nó cũng không sắp xếp lại các tác vụ bị lỗi do lỗi phần cứng hoặc phần mềm. Bộ lập lịch phân bổ các tài nguyên dựa trên các yêu cầu của ứng dụng

ApplicationManager có chức năng sau:

- Chấp nhận nộp công việc.
- Đàm phán container đầu tiên để thực thi ApplicationMaster. Một nơi chứa kết hợp các yếu tố như CPU, bộ nhớ, đĩa và mạng.
- Khởi động lại container ApplicationMaster khi không thành công.

Chúng ta có thể mở rộng YARN ngoài một vài nghìn node thông qua tính năng YARN Federation. Tính năng này cho phép chúng ta buộc nhiều cụm YARN thành một cụm lớn. Điều này cho phép sử dụng các cụm độc lập, ghép lại với nhau cho một job rất lớn

❖ Tài nguyên

Ở YARN, tài nguyên không đơn thuần là CPU cores nữa mà bao gồm cả bộ nhớ, DiskIO, và GPUs... Một ứng dụng khi khởi động sẽ yêu cầu

- o Priority: đặc quyền
- o Bộ nhớ MB: chương trình ứng dụng dự định sẽ sử dụng bao nhiêu MB bộ nhớ
- o CPU: ứng dụng dự định sử dụng bao nhiêu cores.
- o Số lượng containers.

Containers là số lượng tasks có thể được triển khai song song trên 1 node tính toán. Giới hạn tài nguyên sử dụng bởi 1 containers được thực hiện thông qua tính năng cgroups của Linux kernels.

2.5.4. Hadoop Common

Đây là các thư viện và tiện ích cần thiết của Java để các module khác sử dụng. Những thư viện này cung cấp hệ thống file và lớp OS trừu tượng, đồng thời chứa các mã lệnh Java để khởi động Hadoop.

Chương 4: Ứng dụng demo

1. Mô tả ứng dụng

- Viết một ứng dụng Demo trên 1 cụm Hadoop với 1 Node Master và 2 Node Slave.
- Sử dụng HDFS để lưu dữ liệu.
- Viết project CountWord sử dụng Map-Reduce để đếm số lần xuất hiện 1 từ trong dữ liệu, sau đó giải nén projetc về file Jar.
- Chạy projetc trên hệ thống Hadoop.

2. Tính năng chính

- Đếm số lần nhận học bổng của các sinh viên trường Sư Phạm Kỹ Thuật được nhận học bổng khuyến khích học tập trong 2 năm gần đây.
- Hàm Map lấy dữ liệu từ file dữ liệu và trả về các cặp giá trị Key-Value.
- Hàm Reduce sắp xếp lại các từ mà hàm Map trả ra.
- Hàm Worlcount đếm các chuỗi có trong file đầu vào.

3. Các class trong ứng dụng

3.1. Hàm map

Table 3: Map

TT	code	Chức năng

1	package PackageDemo;
	
	import java.io.IOException;
	import java.util.StringTokenizer;
	import org.apache.hadoop.io.IntWritable;
	import org.apache.hadoop.io.LongWritable;
	import org.apache.hadoop.io.Text;
	import
	org.apache.hadoop.mapred.MapReduceBase;
	import org.apache.hadoop.mapred.Mapper;
	import org.apache.hadoop.mapred.OutputCollector;
	import org.apache.hadoop.mapred.Reporter;
	public class map 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();
		Load dữ liệu lên từ file data.txt và trả về dữ liệu là tập các cặp(key,value) Có value là 1 và key là mã số sinh viên

	StringTokenizer tokenizer = new
	StringTokenizer(line);
	while (tokenizer.hasMoreTokens()){
	word.set(tokenizer.nextToken());
	output.collect(word, one);
	}
	}
	
	}	

3.2. Hàm reduce

Table 4: Reduce

TT	code	Chức năng
1	package PackageDemo; import java.io.IOException; import java.util.Iterator; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reducer; import org.apache.hadoop.mapred.Reporter;	Lấy dữ liệu từ hàm Map trả về và trộn các cặp(key,value) và value có cùng mã số sinh viên

	public 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));
	}
	}	

3.3. Hàm runner

Table 5: Runner

TT	Code	Chức năng
1	package PackageDemo; import java.io.IOException; import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.FileInputFormat; import org.apache.hadoop.mapred.FileOutputFormat; import org.apache.hadoop.mapred.JobClient;	Khai báo Job Wordcount và cấu hình các thông số cần cho Job và thực thi Job

```
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class runner {
   public static void main(String[] args) throws
IOException{
      JobConf conf = new JobConf(runner.class);
      conf.setJobName("WordCount");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      conf.setMapperClass(map.class);
      conf.setCombinerClass(reduce.class);
      conf.setReducerClass(reduce.class);
      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);
      FileInputFormat.setInputPaths(conf,new
Path(args[0]));
      FileOutputFormat.setOutputPath(conf,new
Path(args[1]));
      JobClient.runJob(conf);
   }
}
```

Chương 5: Kết luận và hướng phát triển

1. Kết luận

1.1. Ưu điểm

- Đếm chính xác được số lần 1 sinh viên nhận học bổng trong 2 năm gần đây.
- Dữ liệu thực tế.
- Tài liệu rõ ràng và chính xác.

1.2. Nhược điểm

- Trong cụm chỉ có 2 Node Slave.
- Chương trình ứng dụng demo còn đơn giản.
- Chưa thực sự hiểu rõ về các hàm và phương thức trong class Join Tracker.

2. Hướng phát triển

- Phát triển được nhiều Node Slave trong cụm.
- Phát triển thêm ứng dụng có thể xử lý được các vấn đề thực tế và phức tạp hơn.

TÀI LIỆU THAM KHẢO

- https://bitly.com.vn/fhg6wa
- https://bigdataviet.wordpress.com/2015/08/08/hadoop-la-gi/
- https://hadoop.apache.org/
- https://bitly.com.vn/a652u7
- https://bitly.com.vn/9rn7rp
- https://bitly.com.vn/jn7s93