Họ và tên: Đặng Ngọc Quân

MSV: 21000699

**BÁO CÁO THỰC HÀNH**

**TÌM HIỂU VỀ HADOOP**

1. **Khái niệm Hadoop**

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

1. **Các thành phần chính của Hadoop**

* Hadoop Common: Là các thư viện và tiện ích cần thiết của Java để các module khác sử dụng
* YARN: Quản lý tài nguyên cúa các hệ thống lưu trữ dữ liệu và chạy phân tích
* HDFS: Là hệ thống file phân tán cung cấp truy cập thông lượng cao cho ứng dụng khai thác dữ liệu
* Map Reduce: Là hệ thống dựa trên YARN dùng để xử lý song song các tập dữ liệu lớn

1. **Cách hoạt động của Hadoop**
   1. Giai đoạn phân tán dữ liệu (Data Distribution):

Dữ liệu được chia thành các phân vùng nhỏ hơn và được phân tán trên nhiều máy chủ khác nhau để tối ưu hiệu xuất xử lý

* 1. Giai đoạn xử lý dữ liệu (Data Processing):

Hadoop sử dụng MapReduce để phân tích và xử lý dữ liệu phân tán. Trong giai đoạn này, các tác vụ được chia thành 2 phần:

* + 1. Map: các dữ liệu được xử lý và chuyển thành cặp key-value
    2. Reduce: các cặp key-value được tổng hợp và xử lý để đưa ra kết quả cuối cùng
  1. Giai đoạn đưa ra kết quả (Data Presentation):

Kết quả xử lý được trình bày cho người dùng thông qua giao diện hoặc được lưu trữ vào cơ sở dữ liệu để sử dụng sau này

* 1. Giai đoạn quản lý dữ liệu (Data Management):

Hadoop hỗ trợ quản lý dữ liệu phân tán bằng cách lưu trữ dữ liệu trên nhiều máy chủ và đảm bảo dữ liệu được sao lưu và phục hồi trong trường hợp có lỗi xảy ra

* 1. Giai đoạn quản lý tài nguyên (Resource Management):

Hadoop sử dụng Hadoop YARN (Yet Another Resource Negotiator) để quản lý và phân phối tài nguyên giữa các tác vụ xử lý dữ liệu trên nhiều máy chủ. YARN giup tối ưu hóa việc sử dụng tài nguyên và đảm bảo tính sẵn sàng và hiệu quả của Hadoop

1. **Ứng dụng của Hadoop**
   1. Xử lý và phân tích dữ liệu lớn: hadoop được sử dụng để xử lý và phân tích các tập dữ liệu lớn như logs, dữ liệu đo lường cảm biến, dữ liệu từ trang web, dữ liệu từ các mạng xã hội, dữ liệu từ các thiết bị di động, dữ liệu tài chính,..Các công ty như Facebook, Twitter, Yahoo, Amazon đều sử dụng Hadoop để xử lý và phân tích dữ liệu lớn của họ
   2. Lưu trữ dữ liệu lớn: Hadoop cũng được sử dụng như 1 nền tảng lưu trữ dữ liệu lớn. Ví dụ, công ty thương mại điện tử Alibaba sử dụng Hadoop để lưu trữ dữ liệu của hàng triệu khách hàng và triệu giao dịch mỗi ngày
   3. Xử lý hình ảnh và video
   4. Phân tích và dự đoán thị trường tài chính
   5. Phát hiện gian lận và an ninh mạng
2. **Ưu điểm của Hadoop**
   1. Xử lý các tập dữ liệu lớn lên đến hàng petabyte
   2. Có khả năng mở rộng dễ dàng, cho phép người dùng thêm nhiều nút vào hệ thống để xử lý các tập dữ liệu lớn hơn
   3. Mã nguồn mở và miễn phí, cho phép các công ty sử dụng nó mà không cần phải chi tiêu nhiều tiền cho giấy phép phần mềm
   4. Có khả năng tự phục hồi trong trường hợp một nút bị lỗi hoặc không hoạt động, giúp đảm bảo hệ thống luôn sẵn sàng
   5. Cung cấp 1 số công cụ phân tích dữ liệu như Hive, Pig, Hbase giúp người dùng phân tích dữ liệu nhanh chóng và dễ dàng
3. **Nhược điểm của Hadoop:**
   1. Không phù hợp cho việc xử lý các tập dữ liệu nhỏ vì nó sử dụng một lượng lớn bộ nhớ để xử lý tập dữ liệu này
   2. Không phù hợp cho các ứng dụng thời gian thực
   3. Yêu cầu nhiều tài nguyên phần cứng để hoạt động tốt, bao gồm bộ nhớ RAM, bộ vi xử lý và ổ cứng
   4. Khó khăn trong việc tích hợp với các hệ thống khác, đặc biệt là các hệ thống có sẵn
4. **HDFS**
   1. Định nghĩa

Hadoop Distributed File System (HDFS) là hệ thống lưu trữ dữ liệu được Hadoop sử dụng. Hệ thống này có chức năng cung cấp khả năng truy cập với hiệu suất cao đến với các dữ liệu nằm trên các cụm của Hadoop. Đây là nơi cung cấp phương tiện đáng tin cậy để quản lý dữ liệu lớn và hỗ trợ các ứng dụng liên quan đến phân tích dữ liệu lớn.

* 1. Ứng dụng của HDFS
     1. ***Quản lý các tập dữ liệu lớn(Manage large datasets)***: Tổ chức và quản lý các tập dữ liệu lớn là một vấn đề khó giải quyết. HDFS được sử dụng để quản lý các ứng dụng phải xử lý các tập dữ liệu lớn. Nó sẽ hỗ trợ chia các tệp dữ liệu hàng trăm Terabyte thành những tệp dữ liệu có dung lượng ít hơn. Quá trình phân chia đều thực hiện trên hệ thống máy tính. Điều này giúp cho việc quản lý dữ liệu trở nên dễ dàng hơn, đồng thời giúp hệ thống giảm thiểu được thời gian truy xuất dữ liệu và đơn giản hóa quá trình quản lý cho tệp tin
     2. ***Phát hiện lỗi(Detecting faults)***: HDFS được tích hợp công nghệ để quét và phát hiện lỗi một cách nhanh chóng và hiệu quả. Các tính năng này giúp giảm rủi ro xuống mức thấp nhất khi hệ thống bất ngờ phát sinh lỗi phần cứng
     3. ***Hiệu quả phần cứng(Hardware efficiency)***: Khi có các tệp dữ liệu lớn, nó có thể giảm lưu lượng mạng và tăng tốc độ xử lý
     4. ***Được thiết kế dành cho các ứng dụng dạng xử lý khối***: Các file khi được tạo ra trên hệ thống HDFS đều sẽ được ghi, đóng lại và không thể chỉnh sửa được nữa.
  2. Kiến trúc của HDFS

Một cụm HDFS cluster bao gồm 1 NameNode và nhiều DataNode, trong đó NameNode đóng vai trò là master, còn DataNode đóng vai trò là slave

* + NameNode thực hiện các nhiệm vụ:
    - Lưu trữ metadata của dữ liệu thực tế(tên, đường dẫn, blocks id, cấu hình datanode, vị trí blocks,…)
    - Quản lý không gian tên của hệ thống file(ánh xạ các file name với các blocks, ánh xạ các block vào các datanode)
    - Quản lý cấu hình của cụm
    - Chỉ định công việc cho datanode.
  + DataNode:
    - Lưu trữ dữ liệu thực tế.
    - Trực tiếp thực hiện và xử lý công việc(đọc/ ghi dữ liệu)
  + Secondary NameNode: Là 1 node phụ chạy cùng với NamdeNode. Nó như một trợ lý đắc lực của NameNode, có nhiệm vụ:
    - Thường xuyên đọc các file, các metadata được lưu trữ trên RAM của datanote và ghi vào ổ cứng.
    - Liên tục kiểm tra tính chính xác của các tệp tin lưu trên các DataNode.
  + Rack: Theo thứ tự giảm dần từ cao xuống thấp Rack > Node > Block. Rack là một cụm DataNode cùng một đầu mạng, bao gồm các máy vật lý(Tương đương 1 server hay 1 node) cùng kết nối chung.
  + Block: Là một đơn vị lưu trữ của HDFS, các dữ liệu được đưa vào HDFS sẽ được chia thành các block có kích thước cố định ( nếu không cấu hình thì mặc định là 128MB)
  1. Cách hoạt động
     1. Quá trình đọc file

Diagram

Description automatically generated

* + - 1. ***Bước 1:*** Người dùng sẽ mở file cần đọc bằng cách gửi yêu cầu đọc đến NameNode
      2. ***Bước 2:*** Sau đó NamdeNode sẽ thực hiện một số kiểm tra xem file được yêu cầu đọc có tồn tại không hoặc file cần đọc có đang bị lỗi không. Nếu mọi thứ đều ổn, DistributedFileSystem gọi name để lấy vị trí của block đầu tiên. NameNode sẽ gửi danh sách các block (đại diện bởi block id) của file cùng với địa chỉ các DataNode chứa các bản sao của block này
      3. ***Bước 3:*** Sau khi nhận được địa chỉ các NameNode, một đối tượng FSDataInputStream được trả về cho người dùng. FSDataInputStream chứa DFSInputStream
      4. ***Bước 4:*** Khi người dùng yêu cầu đọc file, DFSInputStream kết nối với DataNode gần nhất để đọc block đầu tiên của file. Việc đọc các block sẽ được thực hiện lặp đi lặp lại cho đến khi block cuối cùng của file được đọc xong
      5. ***Bước 5:*** Sau khi đọc xong, DFSInputStream ngắt kết nối và xác định DataNode cho block tiếp theo. Khi DFSInputStream đọc file, nếu có lỗi xảy ra nó sẽ chuyển sang DataNode khác gần nhất có chứa block đó
      6. ***Bước 6:*** Kết thúc đọc file.
    1. Quá trình ghi file

Diagram

Description automatically generated

* + - 1. ***Bước 1:*** Người dùng gửi yêu cầu tạo file ở DistributedFileSystem APIs.
      2. ***Bước 2:*** DistributedFileSystem yêu cầu tạo file ở NameNode. NameNode kiểm tra quyền của người dùng vf kiểm tra file mới có tồn tại hay không.
      3. ***Bước 3:*** DistributdSystemFileName trả về FSDataOutPutStream cho người dùng để ghi dữ liệu . FSDataOutPutStream chứa DFSOutputStream dùng để xử lý tương tác với NameNode và DataNode. Khi người dùng ghi dữ liệu, DFSOutputStream chi dữ liệu thành các packet và đẩy nó vào hàng đợi. NameNode sẽ phân bổ các block vào để các DataNode lưu trữ các bản sao
      4. ***Bước 4:*** Các DataNode tạo thành pipeline, số DataNode bằng số bản sao của file. DataNode đầu tiên trong pipeline sẽ được gửi packet tới, DataNode này sẽ chuyển tiếp packet lần lượt tới các DataNode trong pipeline
      5. ***Bước 5:*** DFSOutputStream có Ack Queue để duy trì các packet chưa được xác nhận bởi các DataNode. Packet ra khỏi Ack Queue khi nhận được xác nhận từ tất cả các DataNode.
      6. ***Bước 6:*** Người dùng thực hiện kết thúc ghi dữ liệu, các packet còn lại được đẩy vào pipeline
      7. ***Bước 7:*** Sau khi toàn bộ các packet được ghi vào DataNode, thông báo hoàn thành ghi file.
  1. ***Các tính năng của HDFS***
     1. ***Sao chép dữ liệu:*** Đây là một tính năng đặc biệt quan trọng của HDFS, đó là đảm bảo dữ liệu luôn sẵn sàng và tránh mất mát
     2. ***Tính khả dụng cao:*** Việc sao chép giúp dữ liệu luôn có sẵn và được sử dụng kịp thời ngay cả khi NameNode hoặc DataNode bị lỗi.
     3. ***Khả năng chịu lỗi và độ tin cậy:*** Các tệp dữ liệu được chia nhỏ, sao chép lưu trữ trên các node trong một cụm lớn và thường xuyên được kiểm tra, nếu có lỗi xảy ra sẽ được khôi phục bằng các bản sao  đảm bảo khả năng chịu lỗi và độ tin cậy của dữ liệu
     4. ***Khả năng mở rộng***: HDFS lưu trữ dữ liệu trên các node khác nhau trong cụm. Khi các yêu cầu tăng lên, một cụm có thể mở rộng đến hàng tram node.
     5. ***Thông lượng cao:*** Các tệp tin định dạng HDFS có thể được xử lý song song trên một cụm nút
     6. ***Dữ liệu được xử lý tại chỗ:*** Với HDFS, việc tính toán diễn ra trên các DataNode nơi dữ liệu nằm, thay vì để dữ liệu di chuyển đến nơi có đơn vị tính toán.
  2. ***Ưu điểm***
     1. ***Dữ liệu được phân tán:*** Một ưu điểm rất nổi bật có ở HDFS, đó chính là khả năng phân tán các dữ liệu thành từng mảnh nhỏ. Giả sử, một cụm dữ liệu Hadoop gồm 20 máy tính khác nhau. Đối với các công cụ khác, mất khá nhiều thao tác để cấp thông tin đến đủ 20 máy tính đó. Với HDFS, bạn chỉ cần đưa dữ liệu vào và toàn bộ file. Nó sẽ được chia nhỏ thành từng phần và phân tán lưu trữ trên 20 máy tính khác nhau
     2. ***Dữ liệu được phân chia và nhân bản:*** Dữ liệu qua HDFS được chia nhỏ giúp các tệp dữ liệu thoát khỏi tình trạng quá tải, đơn giản hóa việc quản lý và giảm thời gian truy xuất dữ liệu. HDFS nhân bản tệp dữ liệu giúp người dung có thể xử lý các sự cố phát sinh trong quá trình triển khai công việc. Ví dụ: Khi một node gặp sự cố hoặc lỗi phần cứng thì hệ thống sẽ lấy dữ liệu từ một nơi khác trong cụm. Từ đó, quá trình xử lý vẫn được tiếp tục trong khi dữ liệu được khôi phục
     3. ***Giải quyết các lỗi phần cứng:*** Các lỗi phần cứng xảy ra gây gián đoạn cho quá trình xử lý dữ liệu. Tuy nhiên HDFS được thiết kế với khả năng phát hiện lỗi và tự phục hồi đảm bảo quá trình xử lý dữ liệu được diễn ra liên tục
     4. ***Khả năng tương thích và tính di động:*** HDFS được sử dụng rộng rãi trên tất cả các nền tảng phần cứng. Đồng thời tương thích với một số [hệ điều hành](https://bkhost.vn/blog/he-dieu-hanh-os-operating-system/) phổ biến như Windows, [Linux](https://bkhost.vn/blog/he-dieu-hanh-linux/) và MacOS
     5. ***Tốc độ:*** Do kiến trúc cụm nên HDFS có thể duy trì 2GB dữ liệu mỗi giây. HDFS có tính năng thông lượng cao giúp cắt giảm thời gian xử lý đồng thời giúp người dùng truy cập vào dữ liệu trực tuyến dễ dàng và nhanh chóng. Tính năng dữ liệu được xử lý tại chỗ giúp giảm tắc nghẽn mạng và tăng thông lượng tổng thể của hệ thống
     6. ***Bộ dữ liệu lưu trữ lớn:*** HDFS lưu trữ nhiều tệp dữ liệu có kích thước lớn từ megabyte đến petabyte ở bất kì dạng nào, bao gồm cả dữ liệu có cấu trúc và phi cấu trúc
     7. ***Chi phí tiết kiệm:*** Các DataNodes lưu trữ dữ liệu trên các phần cứng với chi phí hợp lý. Ngoài ra, vì HDFS là mã nguồn mở nên không cần phí cấp phép
     8. ***Dễ dàng nâng cấp theo chiều dọc:*** Hệ thống còn có thể nâng cấp và mở rộng bằng cách tăng cấu hình máy tính. Điều đó giúp không gian lưu trữ trở nên rộng lớn, vô hạn giúp ta không phải lo lắng về vấn đề quá tải với HDFS
     9. ***Tăng cường bảo mật và tính nhất quán cho dữ liệu:*** HDFS được thiết kế dành cho các ứng dụng xử lý dạng khối. Nhờ đó, tính bảo mật và nhất quán trong các tệp dữ liệu thông tin rất cao, rất an toàn.
  3. ***Hạn chế***
     1. ***Vấn đề lưu trữ file nhỏ:*** HDFS sẽ không tốt khi xử lý một lượng lớn các file nhỏ. Mỗi dữ liệu lưu trữ trên HDFS được đại diện bằng một block với kích thước 128MB. Như vậy, nếu lưu trữ một lượng lớn file nhỏ sẽ cần một lượng lớn các block để lưu trữ chúng và mỗi block ta chỉ cần dùng 1 ít dung lượng nên còn thừa rất nhiều dung lượng gây ra sự lãng phí

1. **Map Reduce**
   1. Định nghĩa
      1. Map Reduce là một mô hình lập trình và một phần mềm xử lý, phân tích dữ liệu lớn trên hệ thống phân tán
      2. Được phát triển bởi Google vào năm 2004
      3. Gồm có 2 bước chính là Map và Reduce, trong đó, Map xử lý dữ liệu đầu vào, chuyển nó thành các cặp key-value, còn Reduce thì tổng hợp các cặp key-value có cùng một key và đưa ra kết quả cuối cùng
      4. Map Reduce có khả năng xử lý dữ liệu lớn một cách song song trên các node của một hệ thống phân tán, giúp tăng tốc độ xử lý dữ liệu và giảm thời gian chờ đợi
   2. Các hàm chính
      1. Hàm Map(): có nhiệm vụ nhận Input cho các cặp giá trị/  khóa và output chính là tập những cặp giá trị/khóa trung gian. Sau đó, chỉ cần ghi xuống đĩa cứng và tiến hành thông báo cho các hàm Reduce() để trực tiếp nhận dữ liệu
      2. Hàm Reduce(): có nhiệm vụ tiếp nhận từ khóa trung gian và những giá trị tương ứng với lượng từ khóa đó. Sau đó, tiến hành ghép chúng lại để có thể tạo thành một tập khóa khác nhau. Các cặp khóa/giá trị này thường sẽ thông qua một con trỏ vị trí để đưa vào các hàm reduce. Quá trình này sẽ giúp cho lập trình viên quản lý dễ dàng hơn một lượng danh sách cũng như  phân bổ giá trị sao cho  phù hợp nhất với bộ nhớ hệ thống
      3. Ở giữa Map và Reduce thì còn 1 bước trung gian đó chính là **Shuffle**. Sau khi Map hoàn thành  xong công việc của mình thì Shuffle sẽ làm nhiệm vụ chính là thu thập cũng như tổng hợp từ khóa/giá trị trung gian đã được map sinh ra trước đó rồi chuyển qua cho Reduce tiếp tục xử lý
   3. Ưu điểm
      1. 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
      2. 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
      3. có khả năng thực hiện trên nhiều nguồn ngôn ngữ lập trình khác nhau như: [Java,](https://itnavi.com.vn/blog/java-la-gi-nhung-dieu-can-biet-truoc-khi-lap-trinh-java/)[C/ C++](https://itnavi.com.vn/blog/ngon-ngu-lap-trinh-c-va-c/), [Python](https://itnavi.com.vn/blog/python-la-gi-nhung-dieu-ban-can-biet-de-hoc-ngon-ngu-python/), Perl, Ruby,... tương ứng với nó là những thư viện hỗ trợ
   4. Cách hoạt động
      1. Input: **Dữ liệu đầu vào được lấy từ các nguồn khác nhau như HDFS, cơ sở dữ liệu hoặc các file thông thường**
      2. **Split: Tập tin đầu vào được chia thành các phân đoạn nhỏ hơn (input splits), mỗi phân đoạn có kích thước bằng nhau hoặc gần bằng nhau, được xử lý song song trên nhiều node.**
      3. **Map: Đối với mỗi phân đoạn dữ liệu, Map Function được áp dụng để tạo ra các cặp key-value tương ứng với từng bản ghi. Map Function xử lý dữ liệu phân tán và đồng thời lọc, chọn ra các dữ liệu cần thiết.**
      4. **Suffle and Sort: Các cặp key-value được chia thành các phân vùng dựa trên key, sau đó được gom nhóm và sắp xếp lại.**
      5. **Reduce: Các phân vùng được ghép lại thành một tập hợp dữ liệu lớn hơn. Các cặp key-value sẽ được xử lý bởi các Reducer, ở đây, các Reducer sẽ thực hiện các phép tính, xử lý dữ liệu và tạo kết quả.**
      6. **Output: Kết quả cuôi cùng của toàn bộ quá trình được ghi vào đĩa, file, một bảng trong cơ sở dữ liệu hoặc trả về cho ứng dụng, ...**