1. **Giới thiệu về HDFS**

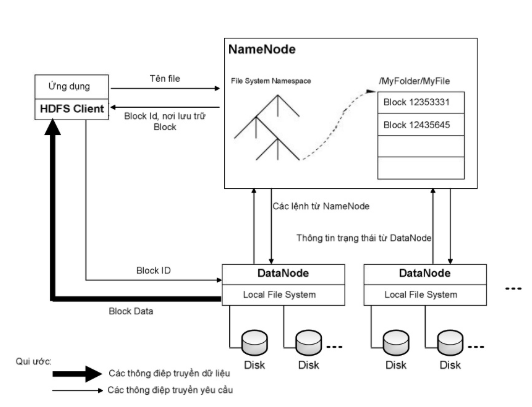
Khi kích thước tập dữ liệu vượt quá khả năng lưu trữ của một máy tính, tất yếu sẽ dẫn đến nhu cầu phân chia dữ liệu lên trên nhiều máy tính. Các hệ thống tập tin quản lý việc lưu trữ dữ liệu trên một mạng nhiều máy tính gọi là hệ thống tập tin phân tán Do hoạt động trên môi trường liên mạng, nên các hệ thống tập tin phân tán phức tạp hơn nhiều so với một hệ thống file cục bộ. Ví dụ, như một hệ thống file phân tán phải quản lý được tình trạng hoạt động của các server tham giai vào hệ thống file.

Hadoop mang đến cho chúng ta hệ thống tập tin phân tán HDFS (Hadoop Distributed File System) với nỗ lực tạo ra một nền tảng lữu trữ dữ liệu đạp ứng cho một khối lượng dữ liệu lớn và chi phí rẻ.

HDFS được sinh ra nhằm giải quyết 3 vấn đề chính:

* Vấn đề 1: các lỗi phần cứng thường xuyên xảy ra. Hệ thống HDFS sẽ chạy trên các cluster với hàng trăm thậm chí hàng nghìn node. Các node này được xây dựng nên từ các phần cứng thông thường, giá rẻ, tỉ lệ lỗi cao. Chất lượng và số lượng của các thành phần phần cứng như vậy sẽ tất yếu dẫn đến tỷ lệ lỗi cao. Chất lượng và số lượng của các thành phần phần cứng như vậy sẽ tất yếu dẫn đến tỷ lệ xảy ra lỗi trên clsster sẽ cao. Các vấn đề có thể điểm qua như lỗi của ứng dụng, lỗi của hệ điều hành, lỗi đĩa cứng, bộ nhớ, lỗi của các thiết bị kết nối, lỗi mạng và lỗi nguồn điện,… Vì thế, khả năng phát hiện lỗi, chống chịu lỗi và tự động phục hồi phải được tích hợp vào trong hệ thống HDFS.
* Vấn đề 2: kích thước file sẽ lớn hơn so với các chuẩn truyền thống, các file có kích thước hàng GB sẽ trở nên phổ biến. Khi làm việc trên các tập dữ liệu với kích thước nhiều TB, ít khi nào người ta lại chọn việc quản lý hàng tỷ file có kích thước hàng KB, thậm chí nếu hệ thống có thể hỗ trợ. Như vậy, việc phân chia tập dữ liệu thành một số lượng ít file có kích thước lớn sẽ là tối ưu hơn. Hai tác dụng to lớn của điều này có thể thấy là giảm thời gian truy xuất dữ liệu và đơn giản hóa việc quản lý các tập tin.
* Vấn đề 3: HDFS không phải là một hệ thống file dành cho các mục đích chung. HDFS được thiết kế dành cho các ứng dụng dạng xử lý khối. Do đó, các file trên HDFS một khi được tạo ra, ghi dữ liệu và đóng lại thì không thể bị chỉnh sửa được nữa. Điều này làm đơn giản hóa đảm bảo tính nhất quán của dữ liệu và cho phép truy cập dữ liệu với thông lượng cao.

1. **Kiến trúc HDFS**



Kiến trúc HDFS

HDFS được xây dựng dựa trên cấu trúc cây phân cấp, 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 sẽ có một blockID để nhận diện. Các block của cùng một file (trừ block cuối cùng) sẽ có cùng kích thước và kích thước này được gọi là block size của file đó. Mỗi block của file sẽ được lưu trữ thành ra nhiều bản sao (replica) khác nhau vì mục đích an toàn dữ liệu.

HDFS có một kiến trúc master/slave. Trên một cluster chạy HDFS, có 2 loại node là Namenode và Datanode. Một cluster có duy nhất một Namenode và có một hay nhiều Datanode.

Namenode đóng vai trò là master, chịu trách nhiệm và duy trì thông tin về cấu trúc cây phân cấp các file, thư mục của hệ thống file và các metada khác của hệ thống file. Cụ thể, các metadata mà Namenode lưu trữ gồm có:

* File System Namespace: là hình ảnh cây thư mục của hệ thống file tại một thời điểm nào đó. File System namespace thể hiện tất cả các file, thư mục có trên hệ thống file và quan hệ giữa chúng.
* Thông tin để ánh xạ từ tên file ra thành danh sách các block: với mỗi file, ta có một danh sách có thứ tự các block của file đó, mỗi block đại diện bởi block id.
* Nơi lưu trữ các block: các block được đại diện bởi một block id. Với mỗi block ta có một danh sách các DataNode lưu trữ các bản sao của block đó.

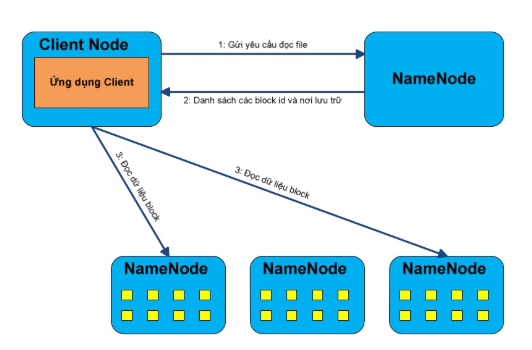
Namenode sẽ chịu trách nhiệm điều phối các thao tác truy cập (đọc/ghi dữ liệu) của client lên hệ thống HDFS. Và tất nhiên, do các Datanode là nơi thật sự lưu trữ các block của các file trên HDFS, nên chúng sẽ là nơi trực tiếp đáp ứng các thao tác truy cập này. Chẳng hạn như khi client của hệ thống muốn đọc 1 file trên hệ thống HDFS, client này sẽ thực hiện một request (thông qua RPC) đén Namenode để lấy các metadata của file cần đọc. Từ medata này nó sẽ biết được danh sách các block của file và vị trí của các Datanode chứa các bản sao của từng block. Client sẽ truy cập vào các Datanode để thực hiện các request đọc các block.

Namenode thực hiện nhiệm vụ của nó thông qua một daemon tên namenode chạy trên port 8021. Mỗi Datanode server sẽ chạy một daemon datanode trên port 8022.

Định kỳ, mỗi Datanode sẽ báo cáo cho NameNode biết về danh sách tất cả các block mà nó đang lưu trữ, Namenode sẽ dựa vào những thông tin này để cập nhật các metadata trong nó. Cứ sau mỗi lần cập nhật lại như vậy, metadata trên namenode sẽ đạt được tình trạng thống nhất với dữ liệu trên các Datanode. Toàn bộ trạng thái của metadata khi đang ở tình trạng thống nhất này sẽ được gọi là một checkpoint. Metadata ở trạng thái checkpoint sẽ được dùng để nhân bản metadata dùng cho mục đích phục hồi lại NameNode nếu Namenode bị lỗi.

1. **Quá trình đọc/ghi**

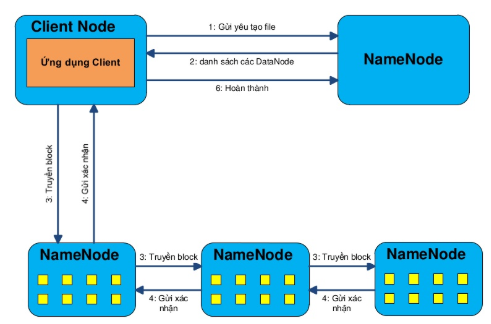
**Quá trình đọc:**



* Bước 1: client sẽ mở file cần đọc bằng cách gửi yêu cầu đọc file đến Namenode. Sau đó, Namenode sẽ thực hiện một số kiểm tra xem file được yêu cầu đọc có tồn tại không hoặc file cần đọc có đang ở trạng thái khỏe mạnh không. Nếu mọi thứ đều ổn, 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.
* Bước 2: client sẽ mở các kết nối tới Datanode, thực hiện một RPC để yêu cầu nhận block cần đọc và đóng kết nối với Datanode. Lưu ý: với mỗi block ta có thể có nhiều DataNode lưu trữ các bản sao của block đó. Client sẽ chỉ đọc bản sao của block từ DataNode gần nhất. Việc tính khoảng cách giữa 2 node trên cluster sẽ được trình bày sau.
* Bước 3: Client sẽ thực hiện việc đọc các block lặp đi lặp lại cho đến khi block cuối cùng của file được đọc xong. Quá trình client đọc dữ liệu từ HDFS sẽ transparent với người dùng hoặc chương trình của ứng dujgn client, người dùng sẽ dùng 1 tập API của Hadoop để tương tác với HDFS, các API này che giấu đi quá trình liên lạc với Namenode và kết nối các DataNode để nhận dữ liệu.

Lưu ý: trong quá trình một client đọc 1 file trên HDFS, client sẽ trực tiếp kết nối với Datanode để lấy dữ liệu chứ không cần thực hiện gián tiếp qua Namenode. Điều này sẽ làm giảm đi rất nhiều việc trao đổi dữ liệu giữa client NameNode, khối lượng luân chuyển dữ liệu sẽ được trải đều khắp cluster, tình trạng nghẽn cổ chai sẽ không xảy ra. Do đó, cluster chạy HDFS có thể đáp ứng đồng thời nhiều client cùng thao tác tại một thời điểm.

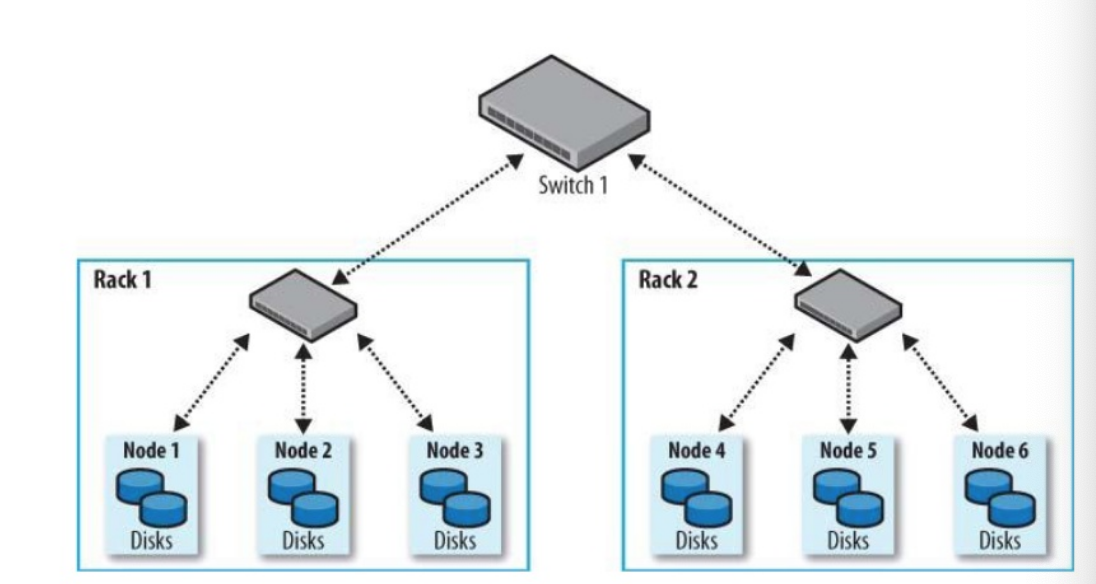
**Quá trình ghi:**



* Bước 1: 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.
* Bước 2: Client sẽ chia file cần gì ra thành các block, và với mỗi block client sẽ đóng gói thành một packet. Lưu ý, mỗi block sẽ được lưu ra thành nhiều bản sao trên các DataNode khác nhau (tuy vào chỉ số độ nhân bản của file).
* Bước 3: Client gửi góc tin cho DataNode thứ nhất, DataNode thứ nhất sau khi nhận được gói tin sẽ tiến hành lưu lại bản sao thứ nhất của block. Tiếp theo DataNode thứ nhất sẽ gửi gói tin này cho Datanode thứ 2 để lưu ra bản sao thứ 2 của block, cứ như vậy cho đến Datanode cuối cùng.
* Bước 4: Datanode cuối cùng sẽ gửi lại cho các Datanode trước đó xác nhận tình trạng thành công. Nếu có bất ký 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 hành vi ghi lại block bị lỗi trên DataNode.
* Bước 5: Sau khi tất cả các block của file đều được ghi lên các Datanode, client sẽ thực hiện một thông điệp báo cho NameNode nhằm cập nhật lại danh sách các vlock của file vừa tạo. Thông tin Mapping từ Block Id sang danh sách các Data Node 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.

1. **Nhận biết cấu trúc topology của mạng**

Trong bối cảnh xử lý dữ liệu với kích thước lớn qua môi trường mạng, việc nhận biết ra giới hạn về băng thông giữa các node là một yếu tố quan trọng để Hadoop đưa ra các quyết định trong việc phân bố dữ liệu và phân tán tính toán. Ý tưởng đo băng thông giữa 2 node có ẻ hượp lý, tuy nhiên làm được điều này là khó khăn (vì việc đo băng thông mạng cần được thực hiện trong một môi trường yên tĩnh, tức tại thời điểm đo thì không được xảy ra việc trao đổi dữ liệu qua mạng). Vì vậy, Hadoop đã sử dụng cấu trúc tôplogy mạng của cluster để định lượng khả năng truyền tải dữ liệu giữa 2 node bất kỳ.



Hadoop nhận biết cấu trúc topology mạng cluster qua một cấu trúc cây phân cấp. Cấu trúc này sẽ giúp Hadoop nhận biết được khoảng cách giữa 2 node trên cluster. Trên cấu trúc phân cấp này, các bridge sẽ đóng vai trò là các node trong để phân chia mạng ra thành các mạng con, 2 node có cùng một node cha hay cùng nằm trên một mạng con thì được xem như là nằm trên cùng một rack.

Hadoop đưa ra một khái nhiệm là địa chỉ mạng để xác định vị trí tương đối của các node. Địa chỉ mạng của một node bất kỳ sẽ là đường dẫn từ node gốc đến node xác định. Ví dụ địa chỉ mạng của Node 1 của hình bên trên sẽ là Switch 1 – Rack 1 – Node 1.

Hadoop sẽ tính toán khoảng cách giữa 2 node bất kỳ đơn giản bằng tổng khoảng cách của 2 node đến node cha chung gần nhất. ví dụ như hình trên, khoảng cách giữa Node 1 và Node 2 là 2, khoảng cách giữa Node 1 và Node 4 là 4.

Hadoop đưa ra một số giả định sau đây về rack:

* Băng thông giảm dần theo thứ tự sau đây:
  + Các tiền trình trên cùng một node.
  + Các node khác nhau trên cùng một rack.
  + Các node không cùng nằm trên 1 rack.
* Hai node có khoảng cách càng gần nhau thì có băng thông giữa 2 node đó càng lớn. Giả định này khẳng định lại giả định đầu tiên.
* Khả năng hai node nằm trên cùng một rack cùng bị lỗi là coa hơn so với hai node nằm trên 2 rack khác nhau. Điều này sẽ được ứng dụng khi NameNode thực hiện sắp đặt các bản sao cho một block xuống các DataNode.

1. **Sắp xếp các bản sao của các block lên các DataNode**

Khi chọn DataNode để lưu trữ, NameNode sẽ chọn dựa trên 3 yếu tố sau:

* Độ tin cậy
* Băng thông đọc
* Băng thông ghi dữ liệu

DataNode sẽ được chọn theo chiến lược như sau, bản sao đầu tiên của một block dữ liệu sẽ được đặt trên cùng node với client – nếu chương trình client ghi dữ liệu cũng thuộc cluster, ngược lại, NameNode sẽ chọn ngẫu nhiên một DataNode. Bản sao thứ 2 sẽ được đặt trên một DataNode ngẫu nhiên nằm trên rack khác với node lưu bản sao đầu tiên. Bản sao thứ 3, sẽ được đặt trên một DataNode nằm cùng rack với node lưu bản sao thứ 2. Các bản sao xa hơn được đặt trên các DataNod được chọn ngẫu nhiên.

1. **Cân bằng cluster**

Theo thời gian sự phân bố của các block dữ liệu trên DataNode có thể trở nên mất cân đối, một số node lưu trữ quá nhiều block dữ liệu trong khi một số node khác lại ít hơn. Một cluster bị mất cân bằng có thể ảnh hưởng tới sự tối ưu hoá MapReduce và sẽ tạo áp lực lên các DataNode lưu trữ quá nhiều block dữ liệu.

Để tránh tình trạng này, Hadoop có một chương trình tên là balancer – chương tình này sẽ chạy như là một daemon trên NameNode sẽ thực hiện việc cân bằng lại cluster. Việc khởi động hay mở. chương trình này sẽ độc lập với HDFS (tức khi HDFS chạy, ta có thể tự do tắt hay mở chương trình này), tuy nhiên nó vẫn là một thành phần trên HDFS. Balancer sẽ định kỳ thực hiện phân tán lại các bản sao của block dữ liệu bằng cách di chuyển nó từ các DataNode đã quá tải sang nhưng DataNode còn trống mà vẫn đảm bảo các chiến lược sắp xếp bản sao của các block lên các DataNode.