**Tìm hiểu về Hadoop Distributed File System (HDFS)**

**1. Định nghĩa**

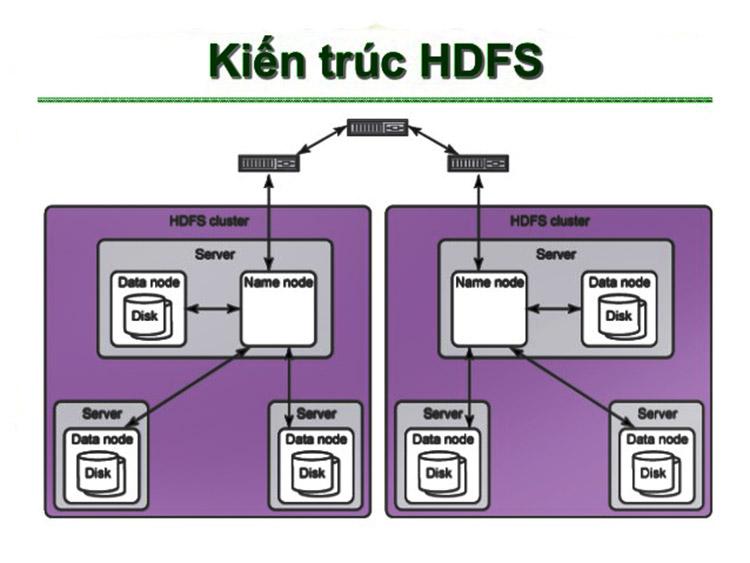
Hadoop Distributed File System (HDFS) là hệ thống lưu trữ dữ liệu được Hadoop sử dụng. Hệ thống này có chức năng cung cấp khả năng truy cập với hiệu suất cao đến với các dữ liệu nằm trên các cụm của Hadoop. Đây là nơi cung cấp phương tiện đáng tin cậy để quản lý dữ liệu lớn và hỗ trợ các ứng dụng liên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

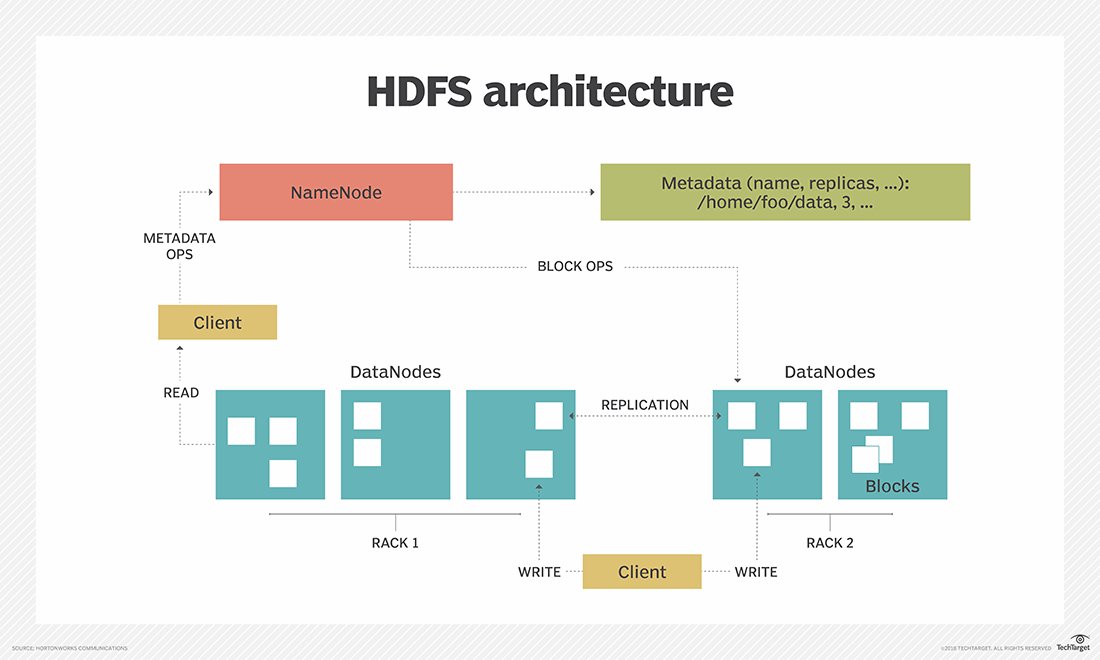
**2. Mục đích của HDFS**

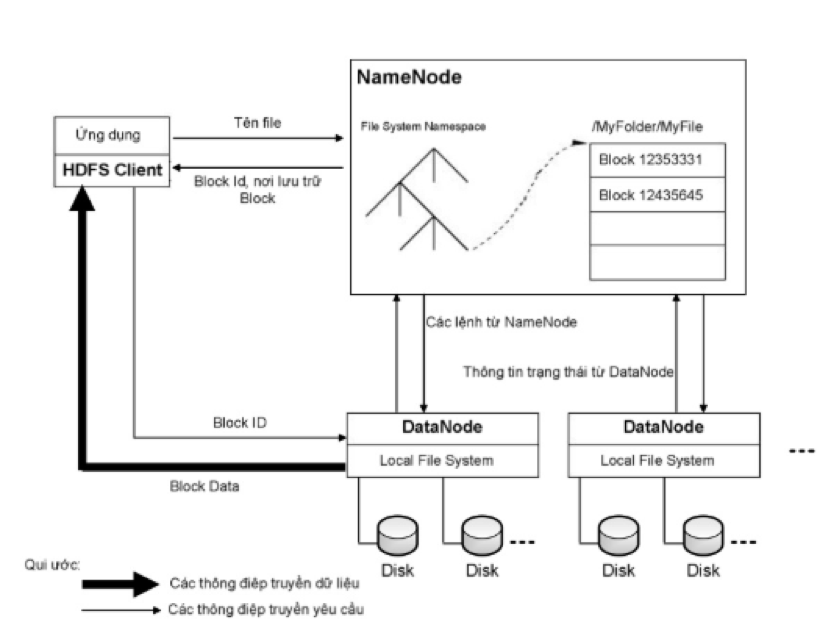
* ***Quản lý các tập dữ liệu lớn(Manage large datasets)***: Tổ chức và quản lý các tập dữ liệu lớn là một vấn đề khó giải quyết. HDFS được sử dụng để quản lý các ứng dụng phải xử lý các tập dữ liệu lớn. Nó sẽ hỗ trợ chia các tệp dữu liệu hàng trăm Terabyte thành những tệp dữ liệu có dung lượng ít hơn. 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.
* ***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.
* ***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
* ***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.

**3. Kiến trúc của HDFS**

Với HDFS, dữ liệu được ghi trên 1 máy chủ và có thể đọc lại nhiều lần sau đó tại bất cứ máy chủ khác trong cụm HDFS. HDFS bao gồm 1 NameNode chính và nhiều DataNode kết nối lại tạo thành một cụm (cluster).





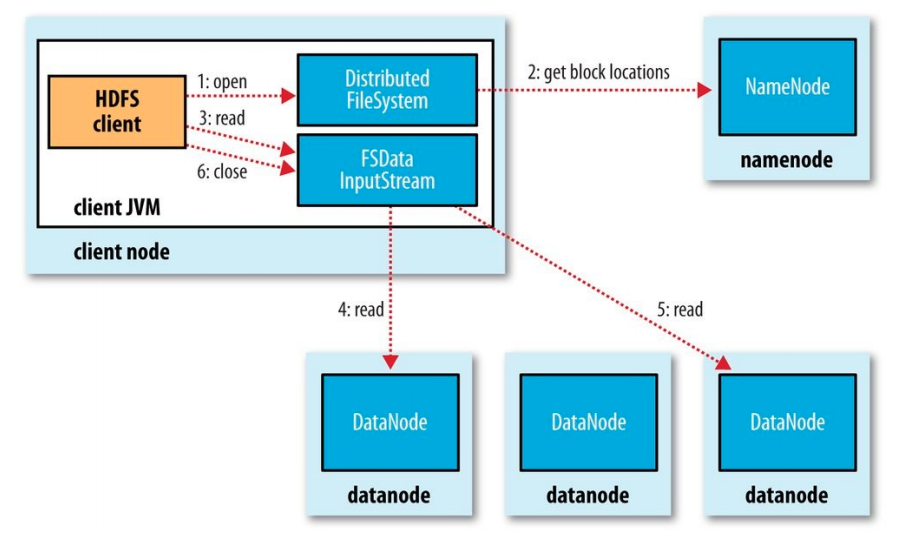
* ***NameNode:*** HDFS chỉ bao gồm duy nhất 1 NameNode được gọi là master node thực hiện các nhiệm vụ:
  + Lưu trữ metadata của dữ liệu thực tế(tên, đường dẫn, blocks id, cấu hình datanode, vị trí blocks,…).
  + Quản lý không gian tên của hệ thống file(ánh xạ các file name với các blocks, ánh xạ các 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).
* ***Metadata:*** Được NameNode lưu trữ, gồm có:
  + *File System NameSpace:* Là hình ảnh cây thư mục của hệ thống file tại một thời điểm nào đó. Nostheer hiện tất cả các file, thư mục có trên hệ thống file và quan hệ giữa chúng.
  + *Thông tin để ánh xạ từ file name 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 lock id.
  + *Nơi lưu trữ các block:* Các block đưuọc đại diện bởi một block id. Với mỗi block ta có một danh sách các DataNode lưu trữ các bản sao của block đó.

**4. Cách thức hoạt động của HDFS**

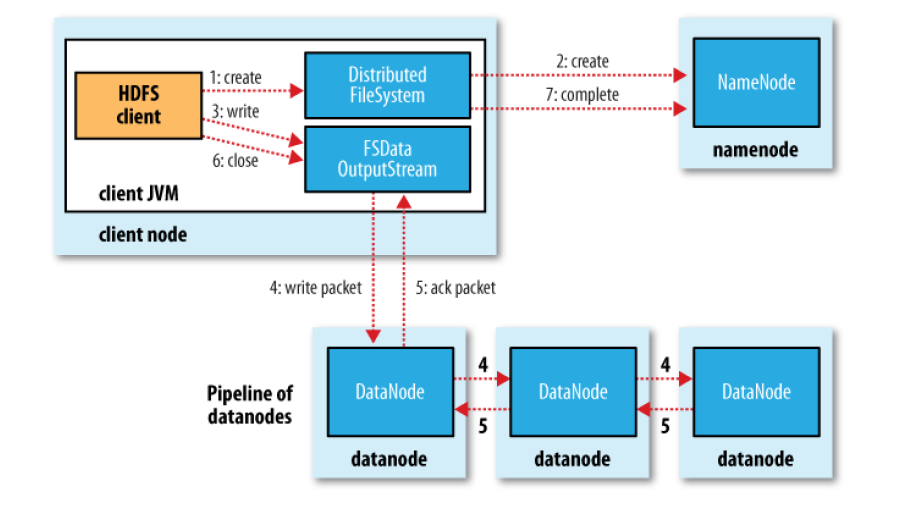
Khi HDFS nhận dữ liệu, nó sẽ chia thành các block riêng biệt rồi phân phối tới các DataNode để lưu trữ. Các block cũng được nhân rộng trên các node cho phép xử lý và khắc phục sự cố một cách nhanh chóng.

NameNode biết DataNode nào chứa block nào và vị trí của các DataNode trong cụm máy, đồng thời, các DataNode cũng thường xuyên liên lạc với NameNode bằng cơ chế Heartbeat.

* ***Cơ chế Heartbeat:*** Heartbeat là cách liên lạc hay là cách để DataNode cho NameNode biết là nó còn sống. Định kì DataNode sẽ gửi một heartbeat về cho NameNode để NameNode biết là DataNode đó còn hoạt động. Nếu DataNode không gửi heartbeat về cho NameNode thì NameNode coi rằng node dó đã hỏng và không thể thực hiện nhiệm vụ được giao. NameNode sẽ phân công nhiệm vụ đó cho một DataNode khác.
* **Quá trình hoạt động của NameNode và DataNode:**
* ***NameNode:*** Có trách nhiệm điều phối cho các thao tác truy cập cua client với hệ thống HDFS (tác vụ đóng, mở, đổi tên tập tin, thư mục,…). 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:*** Có tác vụ ghi, đọc vào hệ thống tập tin; tạo, xóa, nhân rộng các dữ liệu dựa trên chỉ dẫn của NameNode. 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.
* **Quá trình đọc/ ghi file trên HDFS:**
  + **Quá trình đọc file:**

****

* + - ***Bước 1:*** Người dùng sẽ mở file cần đọc bằng cách gửi yêu cầu đọc đến NameNode.
    - ***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.
    - ***Bước 3:*** Sau khi nhận được địa chỉ các NameNode, một đối tượng FSDataInputStream được trả về cho người dùng. FSDataInputStream chứa DFSInputStream.
    - ***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.
    - ***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.
    - ***Bước 6:*** Kết thúc đọc file.
  + **Quá trình ghi file:**

****

* + - ***Bước 1:*** Người dùng gửi yêu cầu tạo file ở DistributedFileSystem APIs.
    - ***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.
    - ***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.
    - ***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.
    - ***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.
    - ***Bước 6:*** Người dùng thực hiện kết thúc ghi dữ liệu, các packet còn lại được đẩy vào pipeline.
    - ***Bước 7:*** Sau khi toàn bộ các packet được ghi vào DataNode, thông báo hoàn thành ghi file.

**3. 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.

**4. Các tính năng nổi bật của HDFS**

* ***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.
* ***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.
* ***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.
* ***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.
* ***Thông lượng cao:*** Các tệp tin định dạng HDFS có thể được xử lý song song trên một cụm nút.
* ***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.

**5. Ưu điểm của HDFS**

* ***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.
* ***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.
* ***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.
* ***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.
* ***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.
* ***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.
* ***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.
* ***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.
* ***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.

**6. Hạn chế của HDFS**

* ***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.