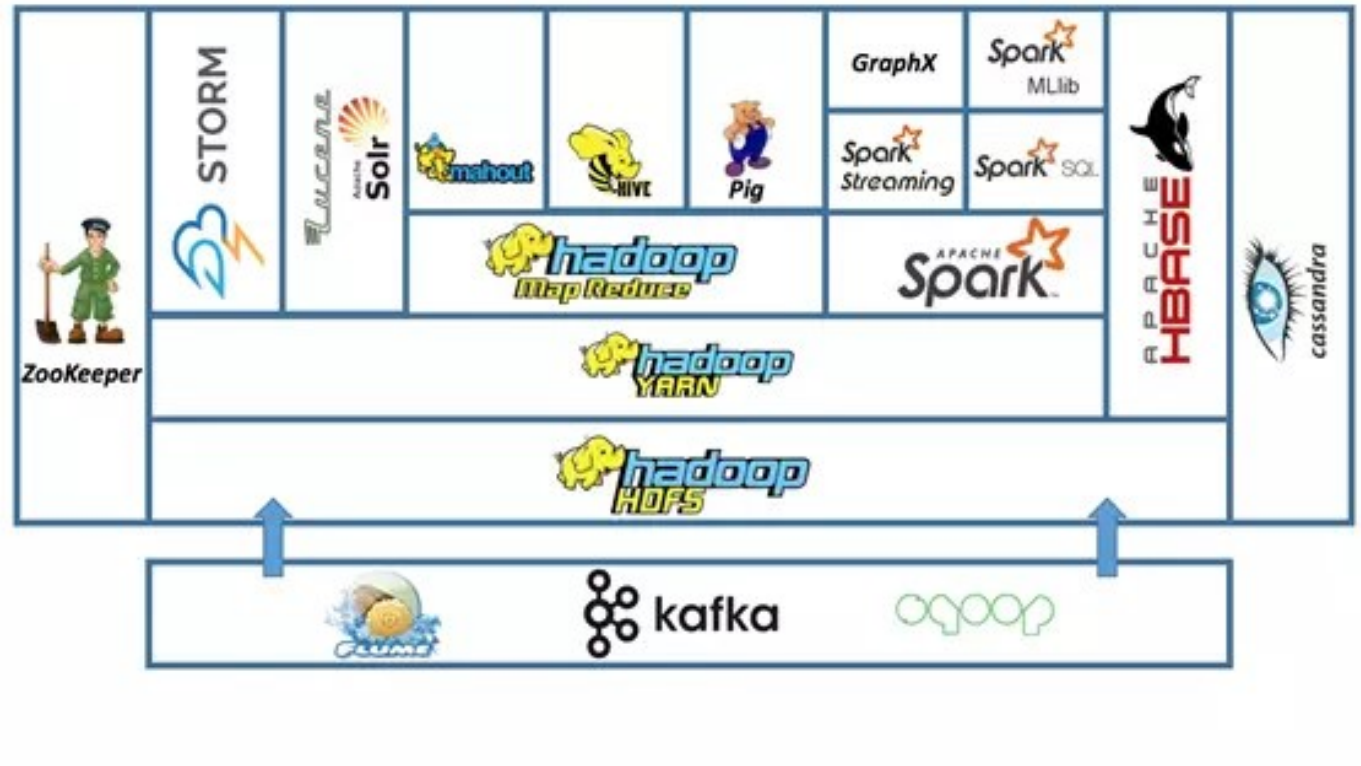
CHƯƠNG 3.
HDFS và Object Storage

Nội dung

- Kiến trúc HDFS và Google File System
- Cụm Hadoop
- Cấu trúc liên kết mạng & Hadoop
- Object Storage
- Các công cụ phổ biến

Kiến trúc HDFS và Google File System

❖ Hệ sinh thái Hadoop



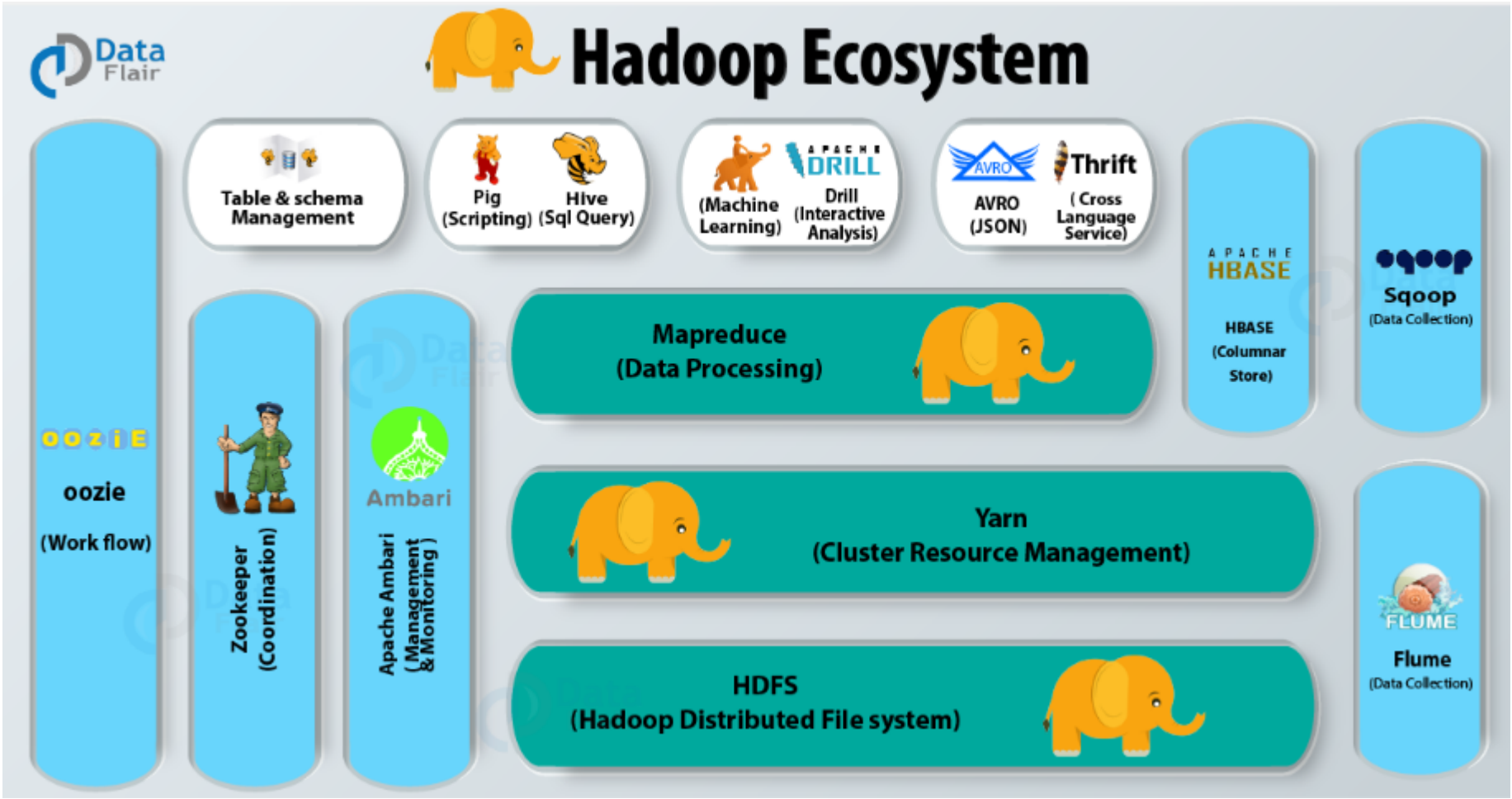
Kiến trúc HDFS và Google File System

❖ Hệ sinh thái Hadoop

- **Hadoop** được xây dựng dựa trên ba phần chính là **Hadoop Distributes FileSystem (HDFS), YARN** và **MapReduce.**
- **Hadoop** phân chia dữ liệu thành các khối và lưu chúng trên nhiều máy tính, giúp tăng khả năng mở rộng, độ tin cậy và hiệu năng.
- **Hadoop** phân tán dữ liệu trên nhiều nút máy tính và xử lý nó *song song* trên các nút, giúp giảm thời gian xử lý và tăng hiệu suất

Kiến trúc HDFS và Google File System

❖ Hệ sinh thái Hadoop



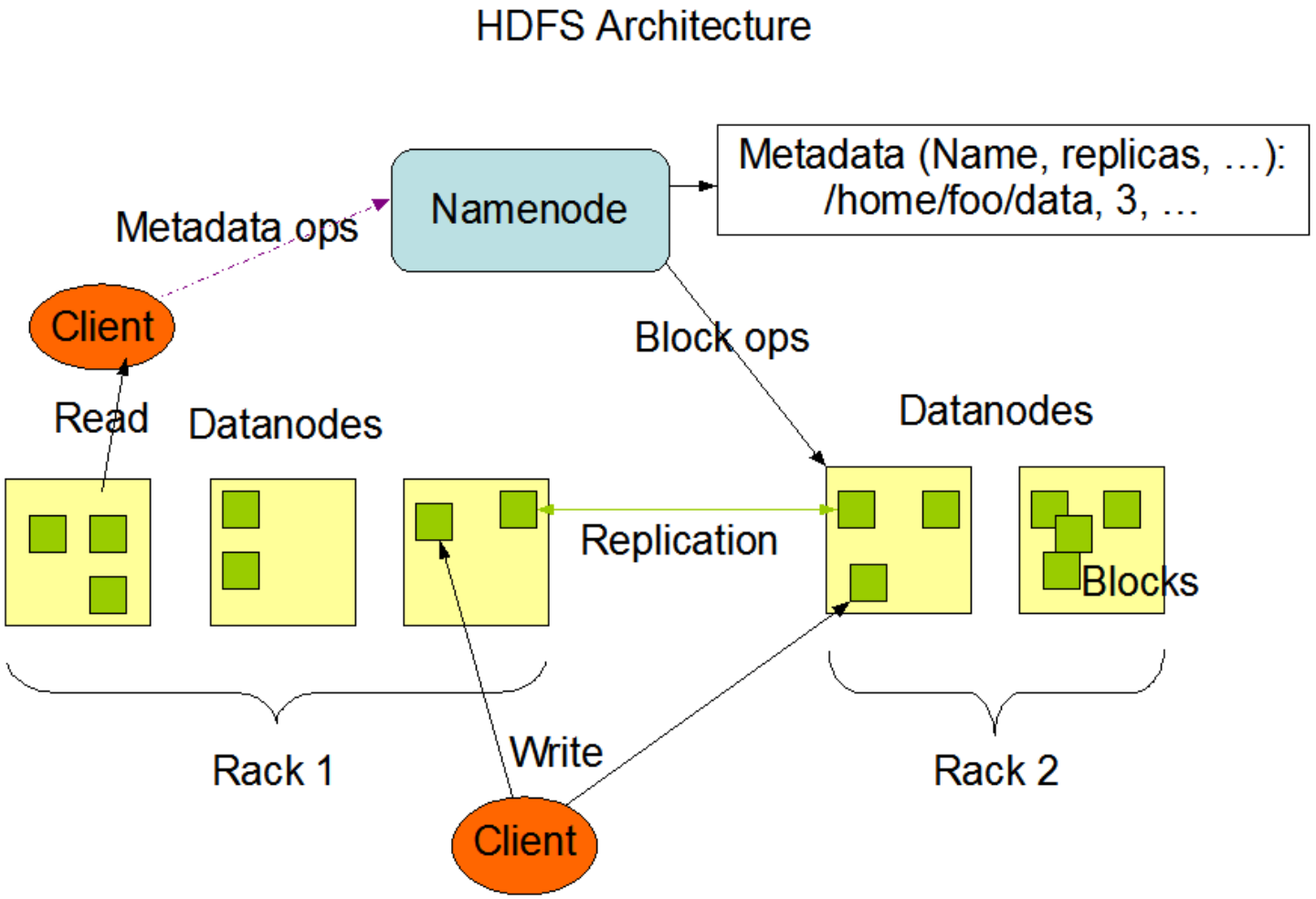
Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

- **Hadoop Distributes FileSystem (HDFS)** là một hệ thống lưu trữ dữ liệu phân tán được thiết kế lưu trữ và quản lý các tệp dữ liệu lớn trên các cụm máy tính phân tán trong một mạng **Hadoop**.
- Sử dụng kiến trúc **Master/Slave**, Với các tính năng như sao lưu dữ liệu, phục hồi dữ liệu, xử lý song song và truy cập dữ liệu từ xa, **HDFS** là một phần quan trọng của việc xử lý dữ liệu của **Hadoop**.

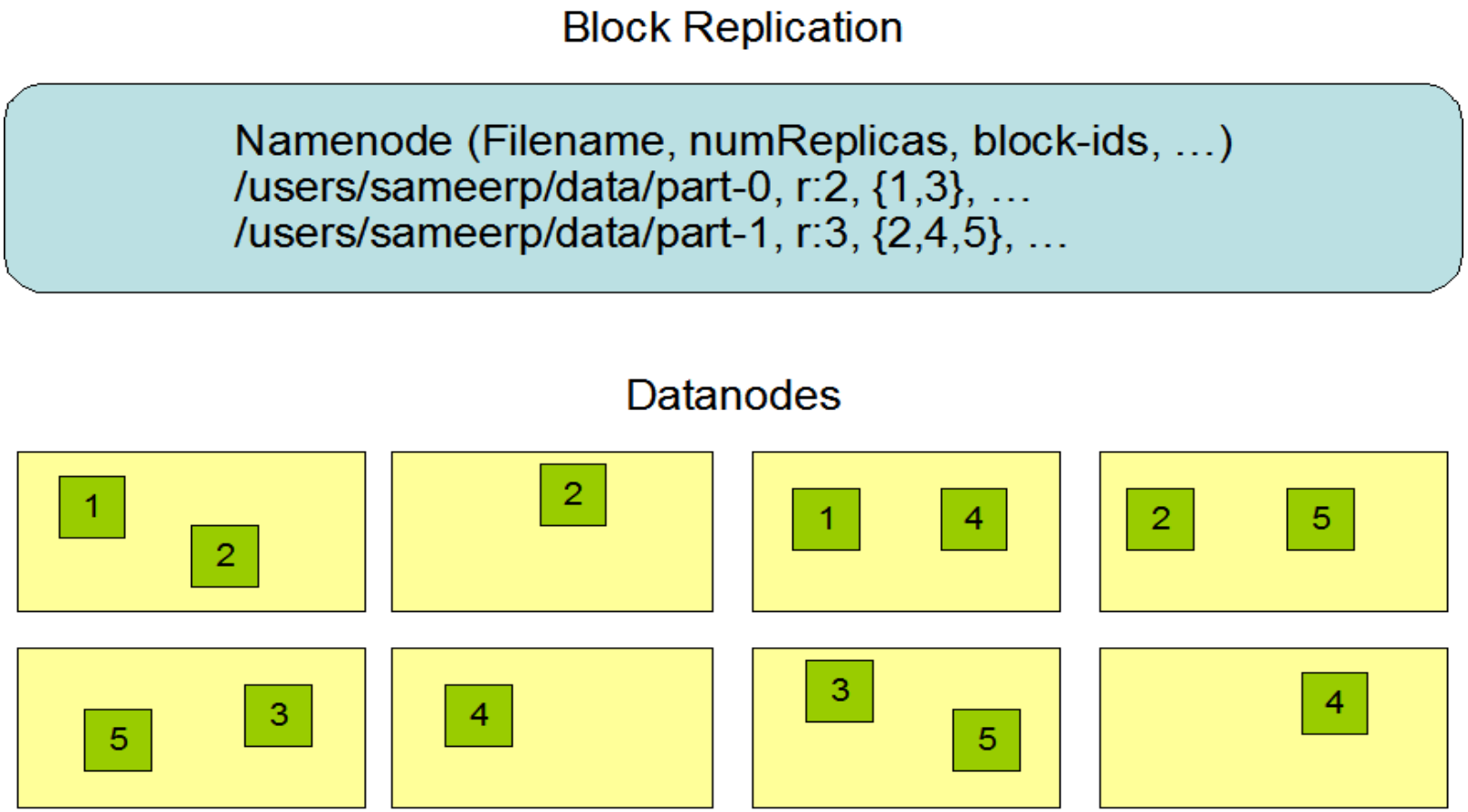
Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS



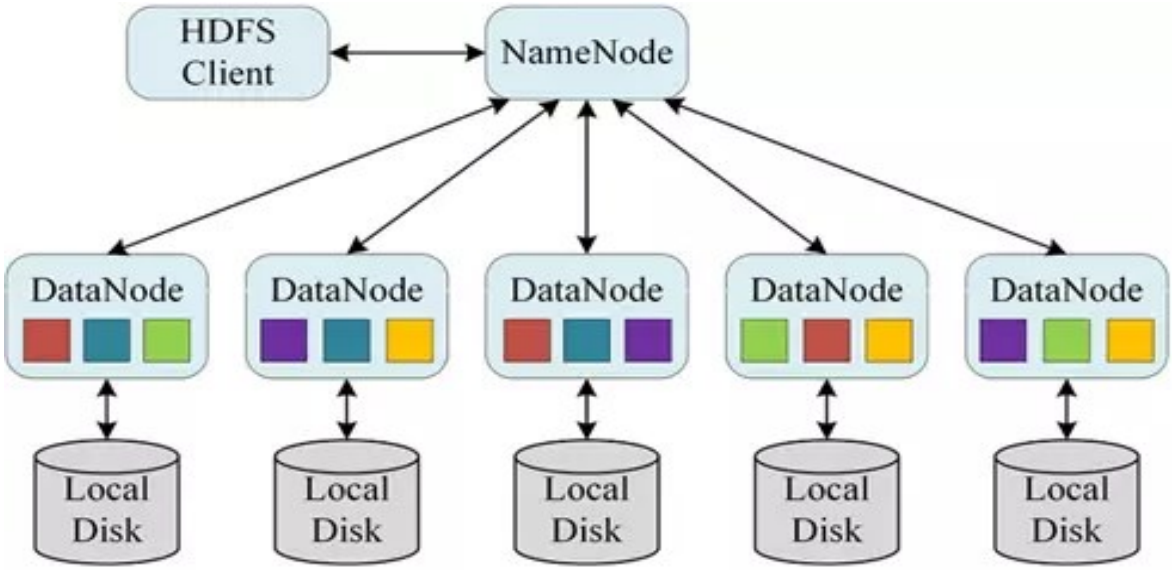
Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

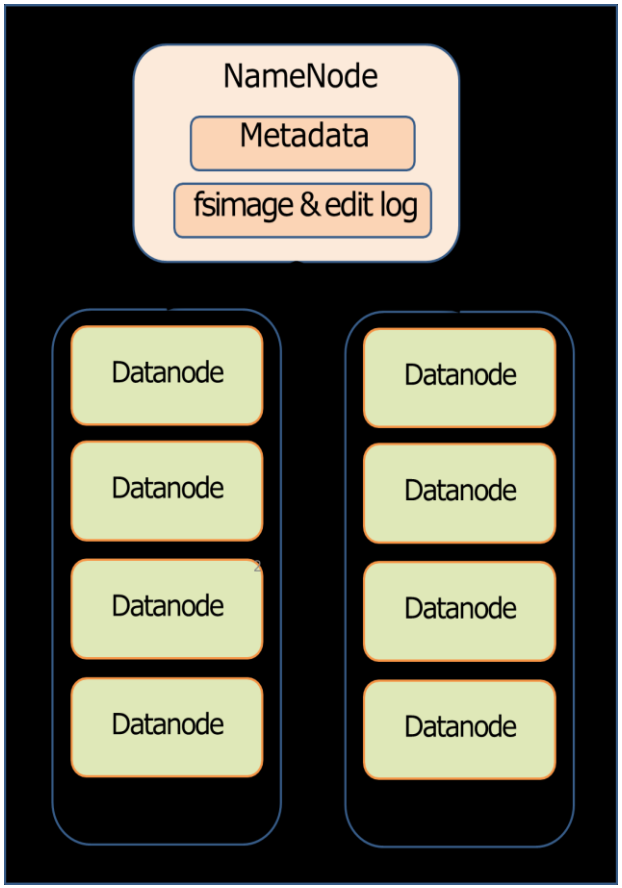


HDFS được thiết kế để chịu được sự cố và có thể xử lý các tệp dữ liệu có kích thước rất lớn. **HDFS** lưu trữ dữ liệu dưới dạng các khối, mỗi khối có kích thước mặc định là **128MB**, nhưng có thể tùy chỉnh kích thước theo nhu cầu sử dụng. Dữ liệu được phân tán trên nhiều nút máy tính trong một mạng **Hadoop**, giúp tăng tốc độ truy xuất dữ liệu và giảm thời gian xử lý.

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

- Mỗi **Cluster** có một **NameNode** và một/nhiều **DataNode**
- **NameNode** đóng vai trò là master, chịu trách nhiệm duy trì thông tin về cấu trúc cây phân cấp các file, thư mục của hệ thống file và các **metadata** khác của hệ thống file
- **DataNode** chịu trách nhiệm lưu trữ và truy xuất file theo yêu cầu của người dùng, **DataNode** đồng thời thực hiện tạo, xoá và nhân rộng các block theo yêu cầu của **NameNode**
- **FsImage** lưu thông tin về hệ thống tập tin, **Edit Log** lưu lại thao tác xử lý trong hệ thống là cấu trúc dữ liệu trung tâm của toàn bộ HDFS, nếu những file này bị lỗi thì có thể dẫn đến toàn bộ dữ hệ thống HDFS không hoạt động được.

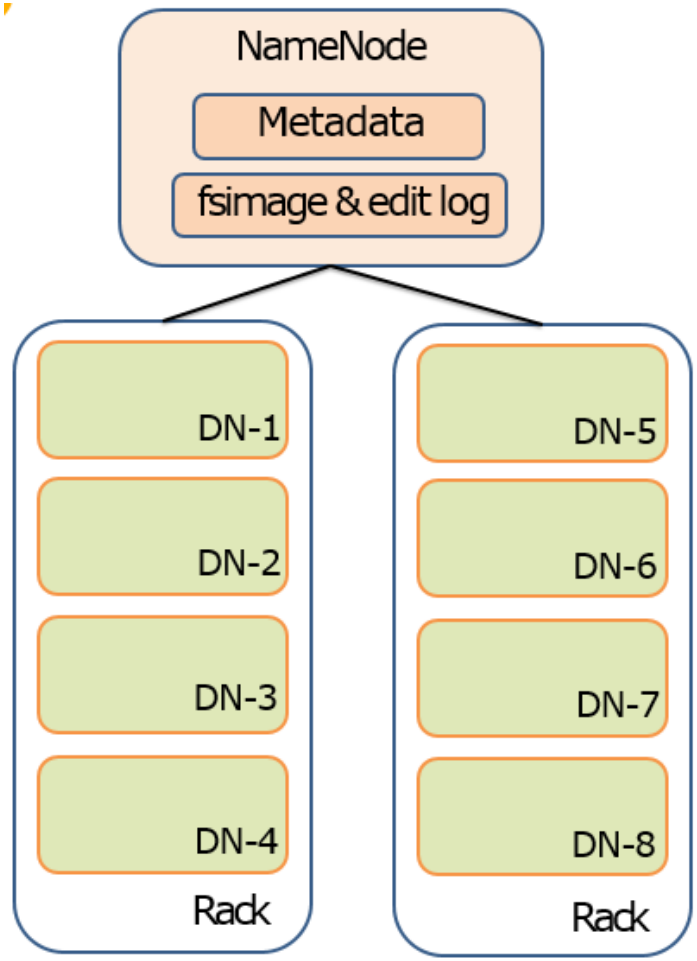


Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình ghi file

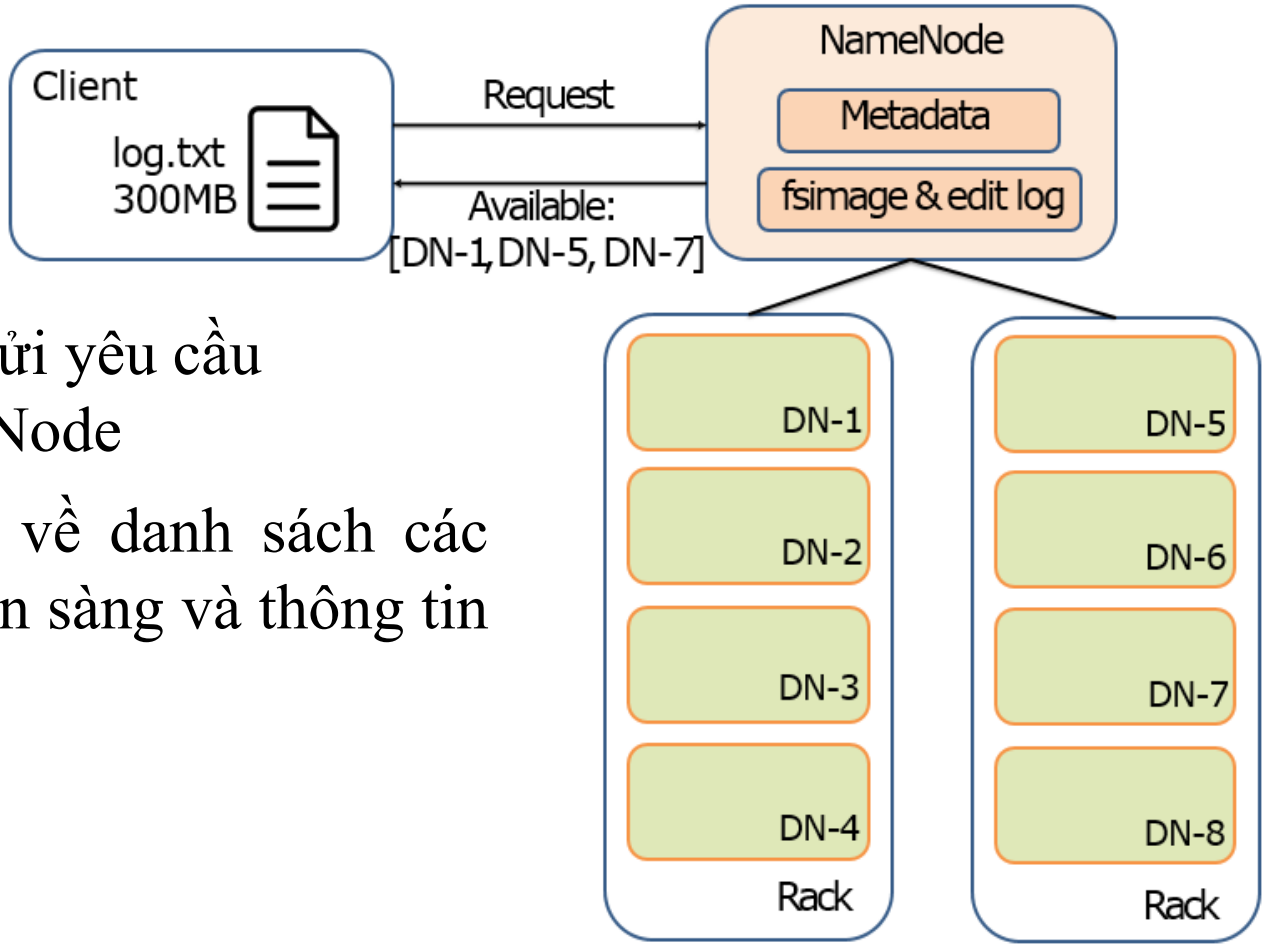
- Giả sử hệ thống HDFS có cấu hình:
  - –Block size: 128MB
  - –Replication: 3
- Cần ghi 1 file dung lượng 300MB



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình ghi file

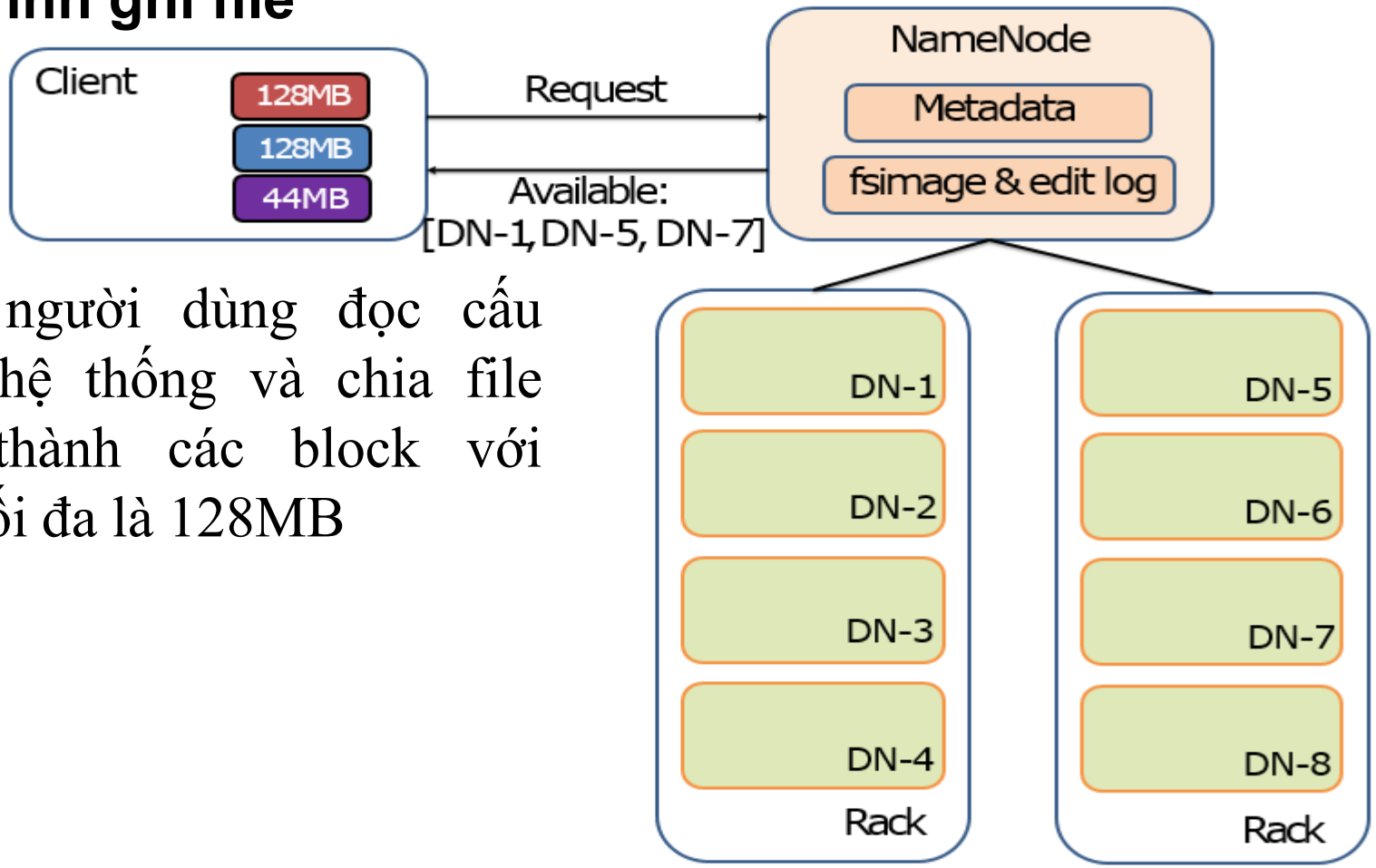


- Phía người dùng gửi yêu cầu ghi file đến NameNode
- NameNode sẽ trả về danh sách các Datanode đang sẵn sàng và thông tin hệ thống

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình ghi file

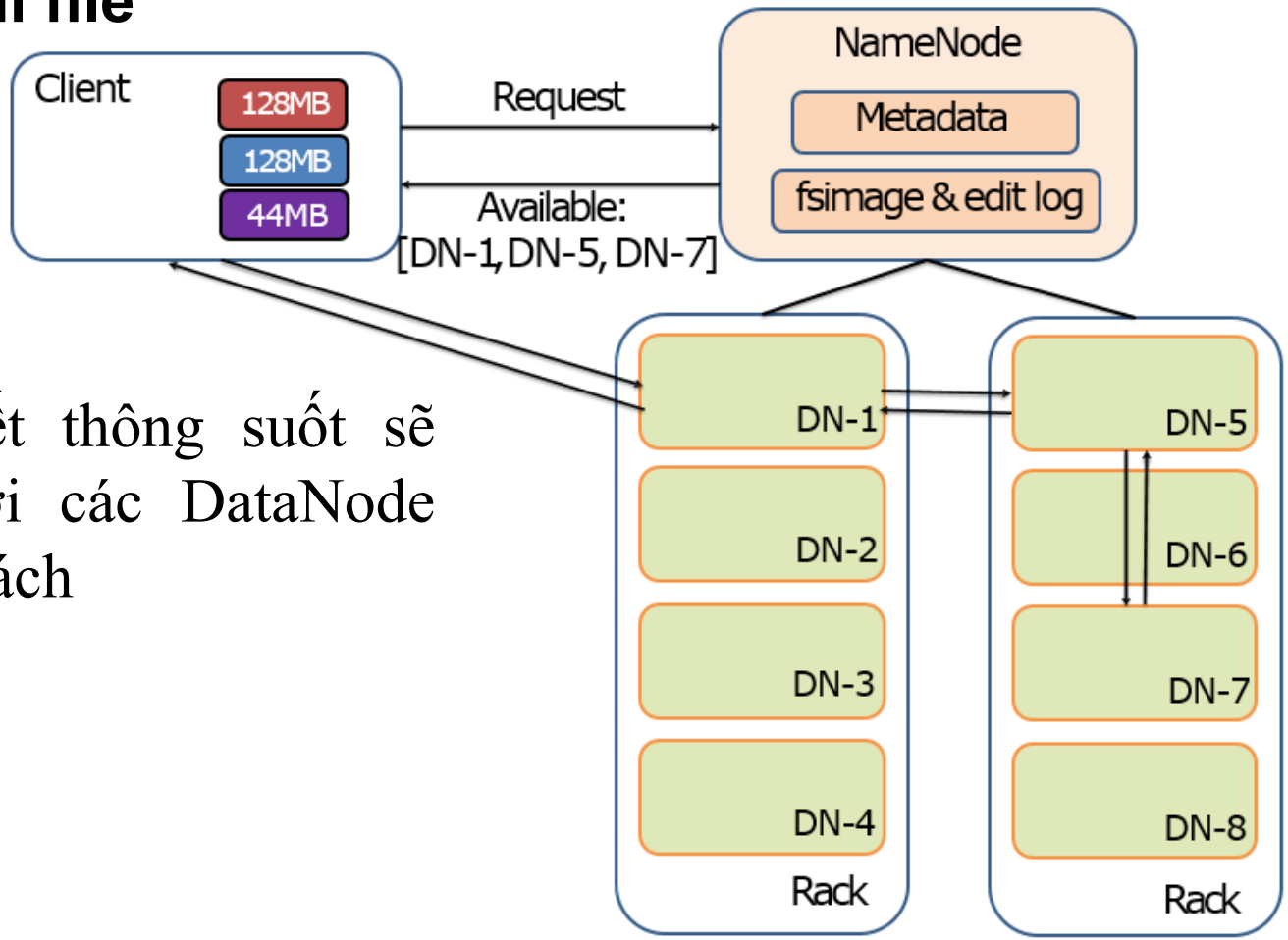


- Phía người dùng đọc cấu hình hệ thống và chia file gốc thành các block với size tối đa là 128MB

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình ghi file

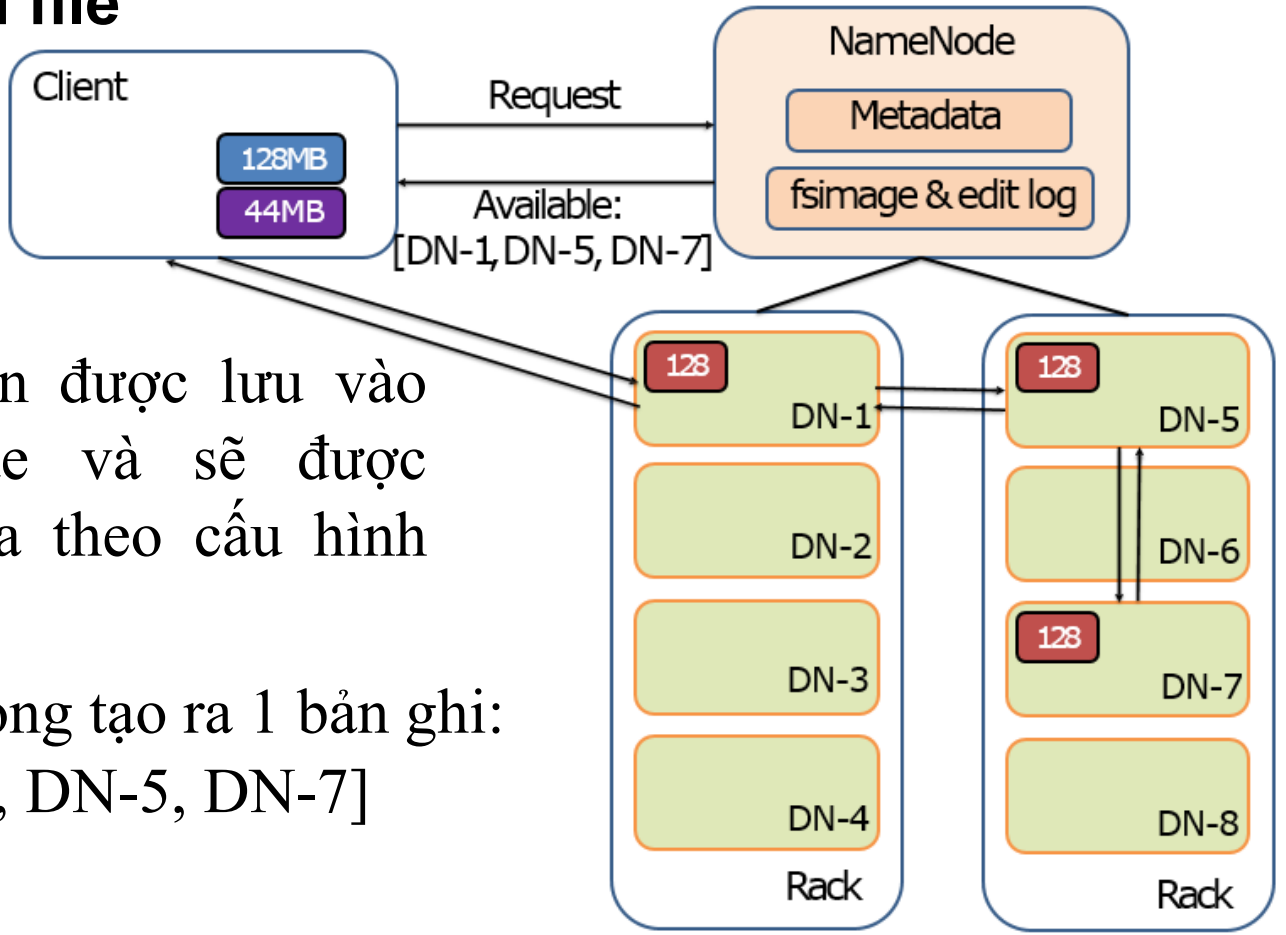


- Một liên kết thông suốt sẽ được tạo tới các DataNode trong danh sách

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình ghi file

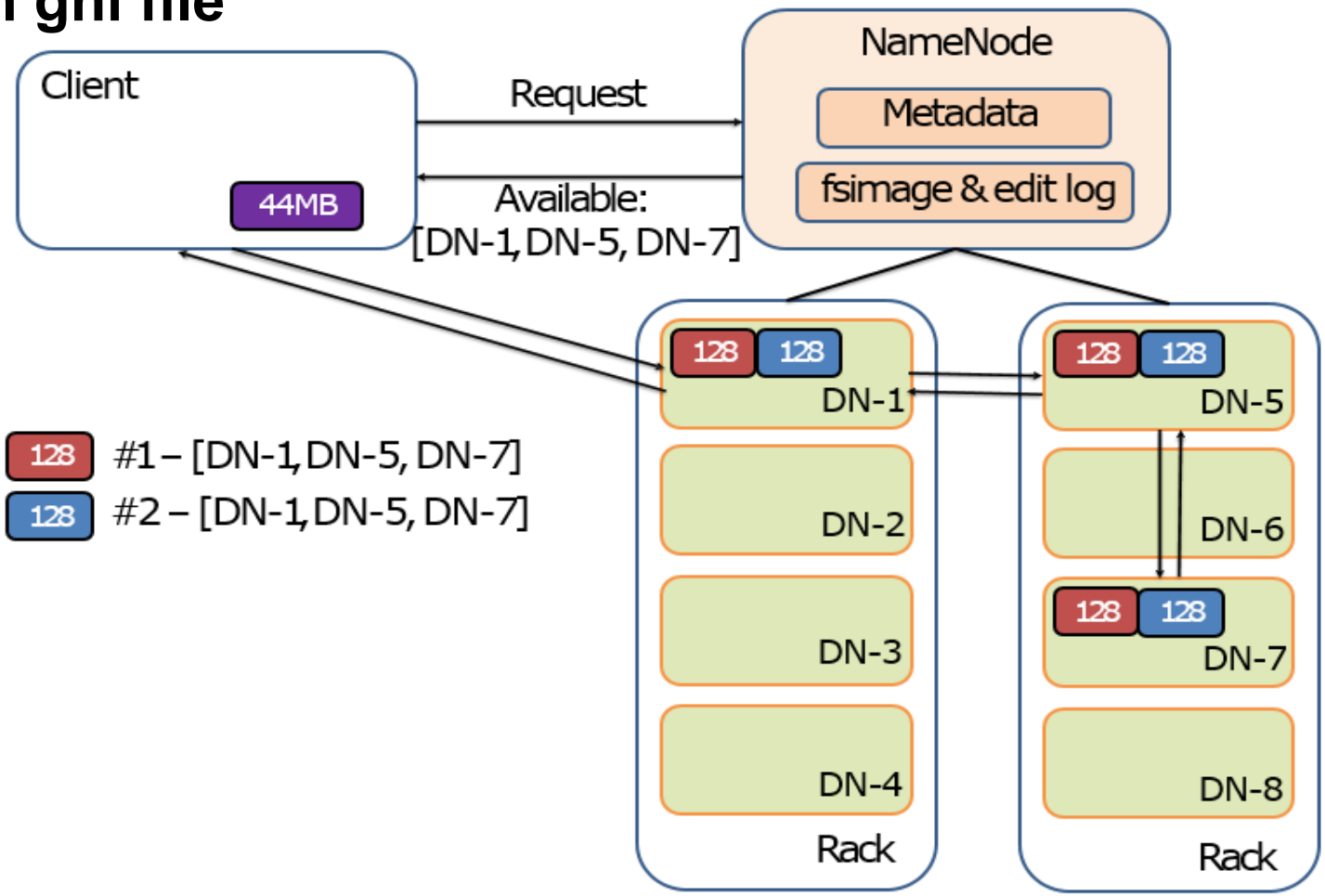


- Các block lần được lưu vào các Datanode và sẽ được nhân bản dựa theo cấu hình Replication
- Sau khi lưu xong tạo ra 1 bản ghi:
  #1 – [DN-1, DN-5, DN-7]

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

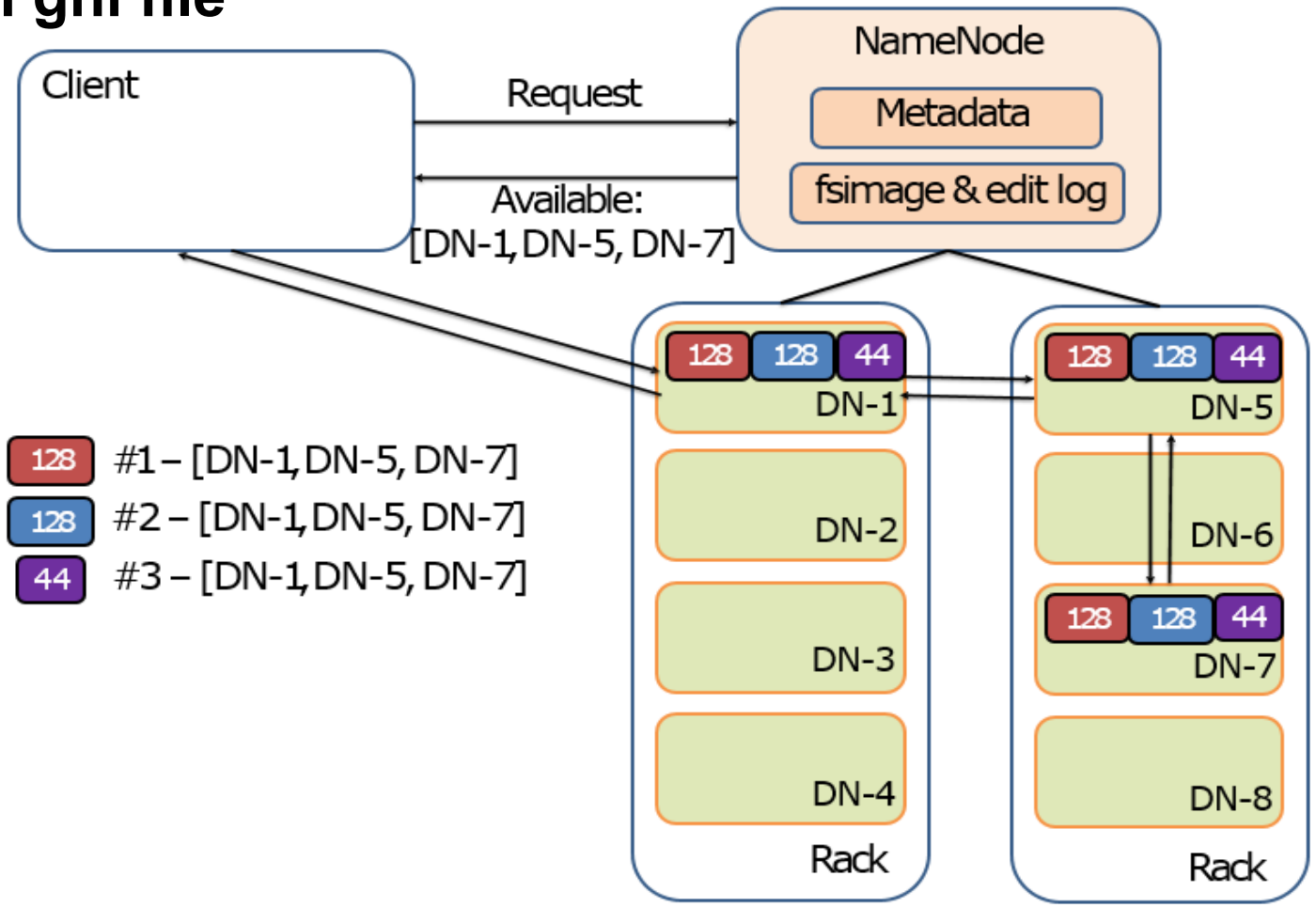
Quá trình ghi file



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình ghi file

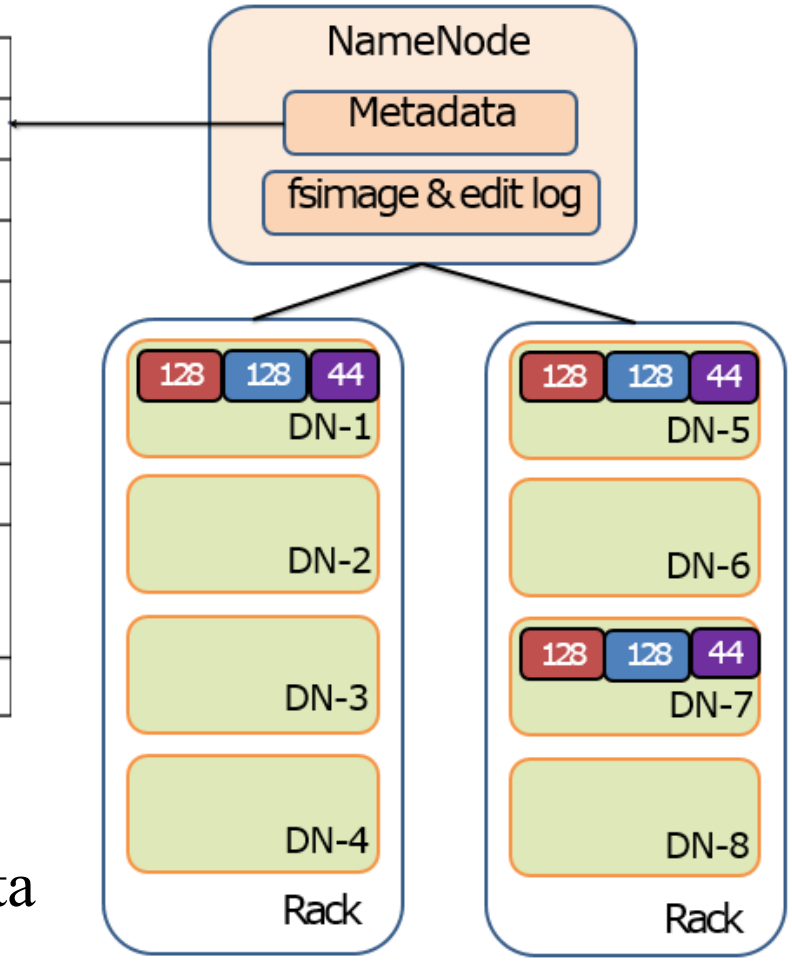


Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình ghi file

File name	log.txt
File size	300MB
No. of blocks	3
Block IDs	[#1, #2, #3]
User	hduser
Permission	-rw--r--r
Replication	3
Block size	128MB
Block Locations	#1 - [DN-1, DN-5, DN-7] #2 - [DN-1, DN-5, DN-7] #3 - [DN-1, DN-5, DN-7]
...	...



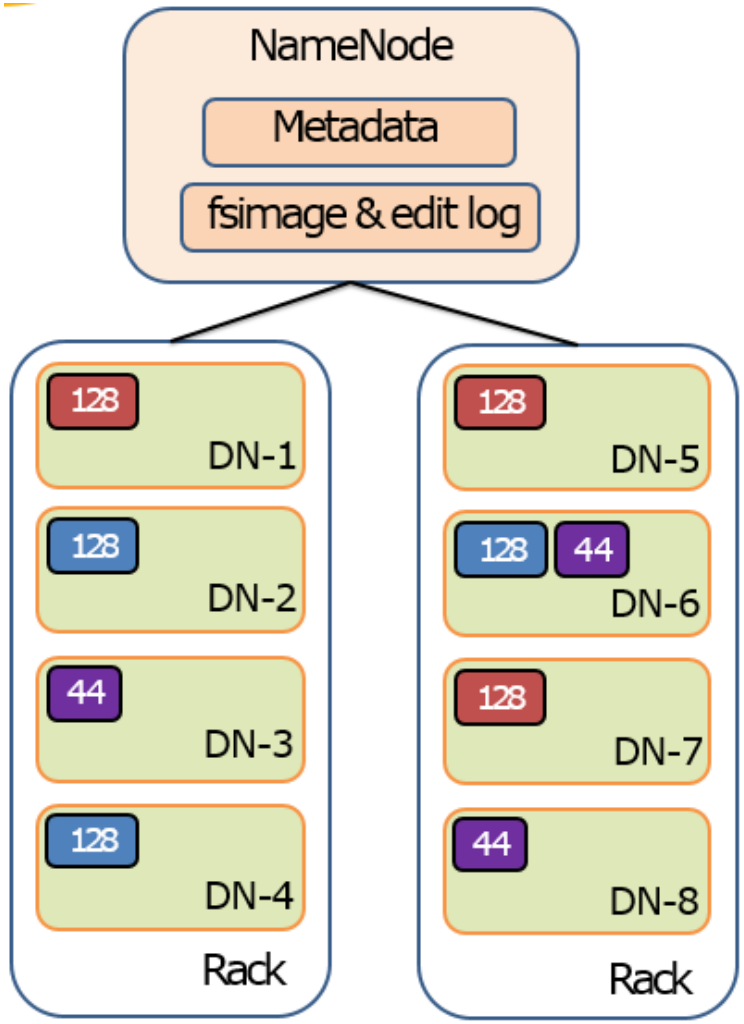
- Sau khi lưu hết các block, NameNode lưu 1 bản metadata

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình ghi file

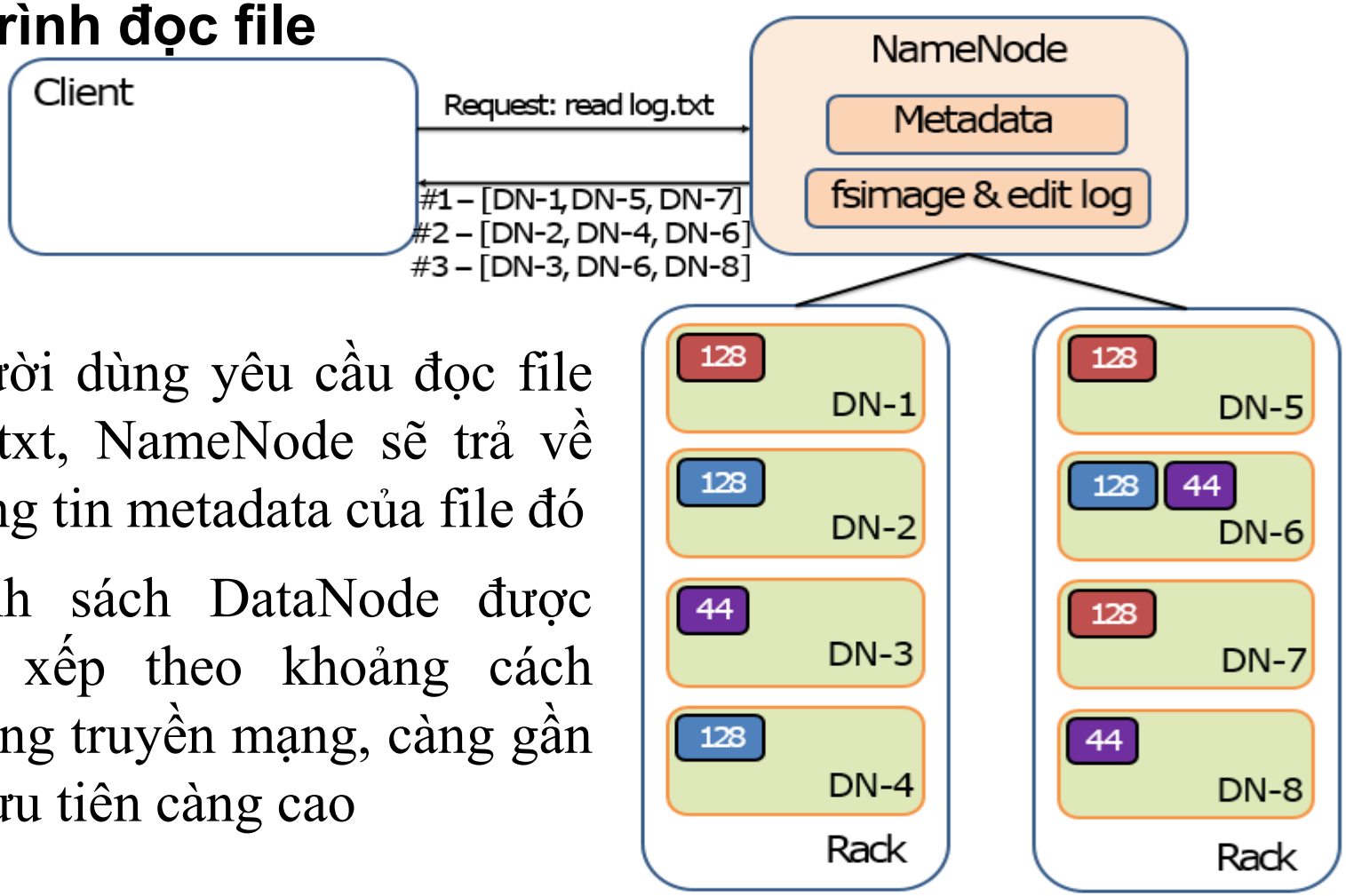
- Trong thực tế để tăng tính sẵn sàng và khả năng chịu lỗi, các block sẽ được lưu ngẫu nhiên trên các DataNode khác nhau



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình đọc file

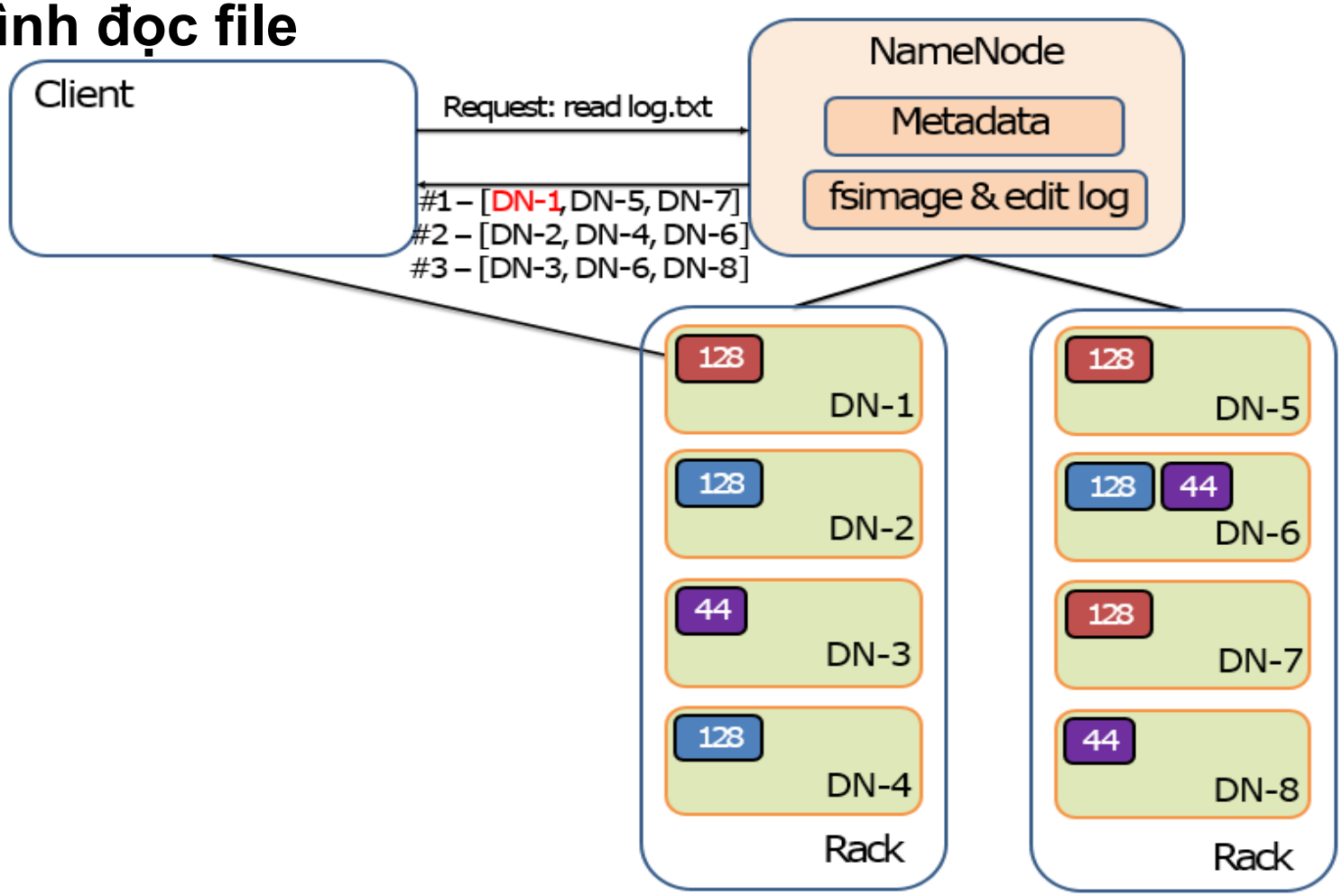


- Người dùng yêu cầu đọc file log.txt, NameNode sẽ trả về thông tin metadata của file đó
- Danh sách DataNode được sắp xếp theo khoảng cách đường truyền mạng, càng gần thì ưu tiên càng cao

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

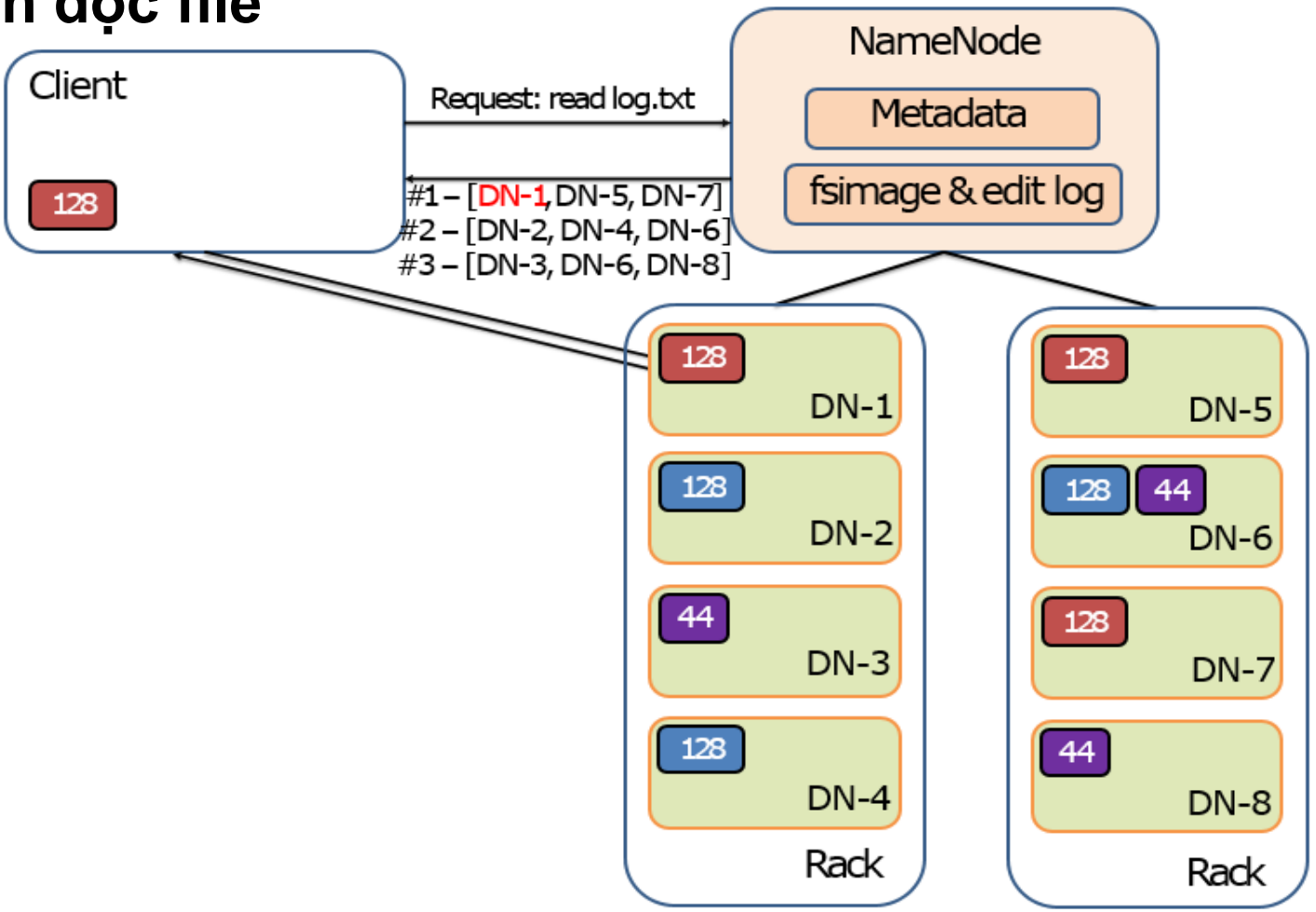
Quá trình đọc file



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

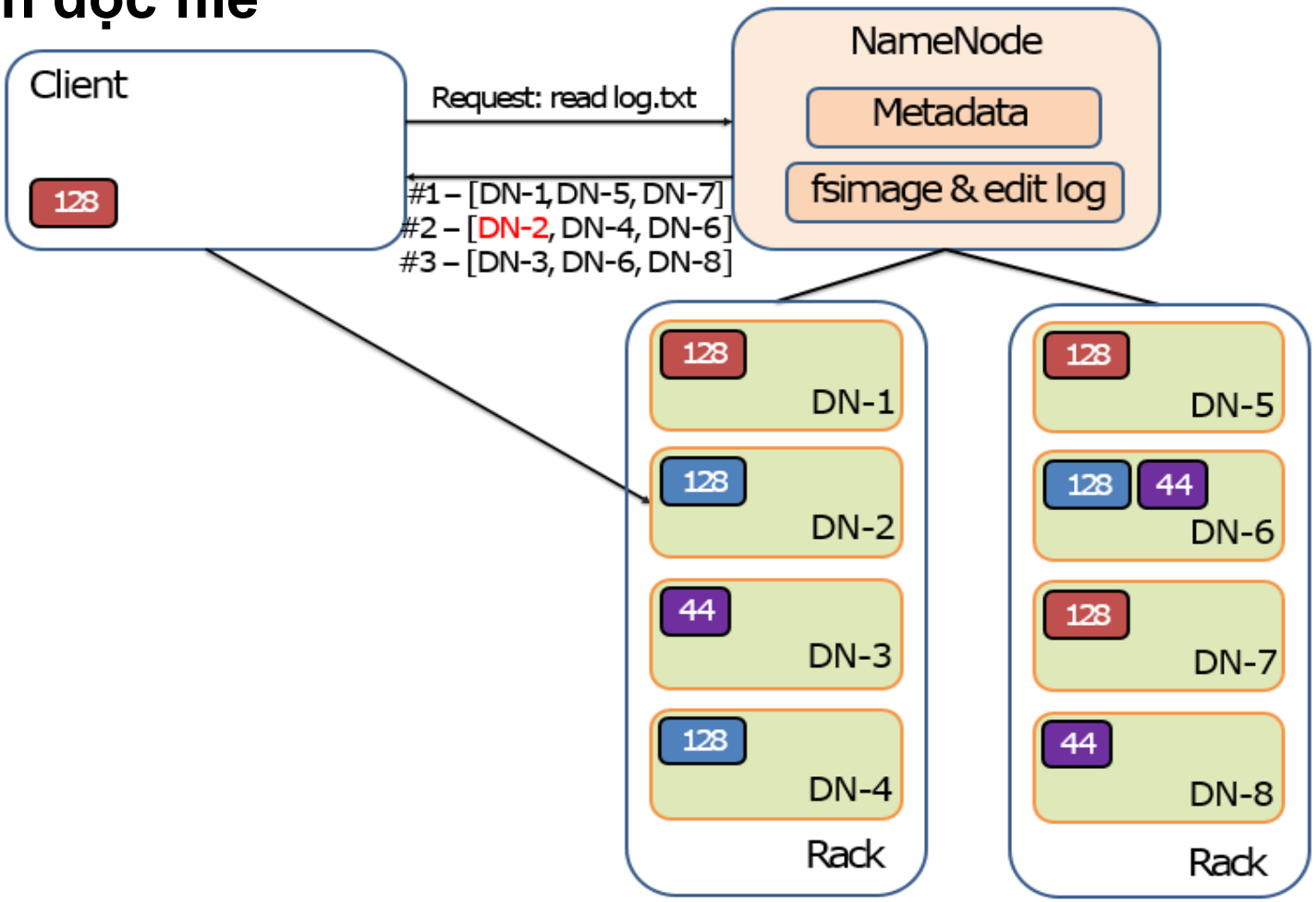
Quá trình đọc file



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

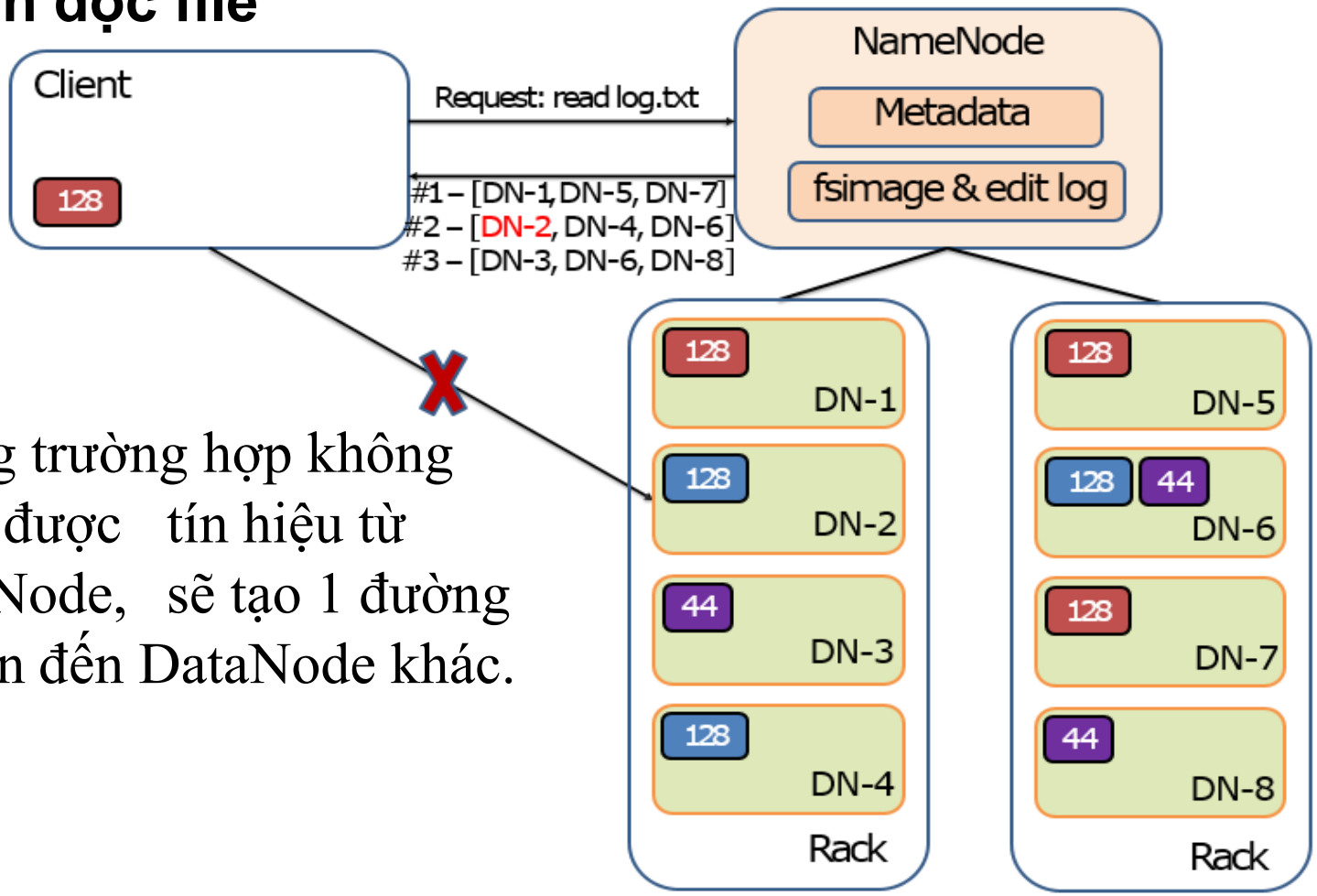
Quá trình đọc file



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình đọc file

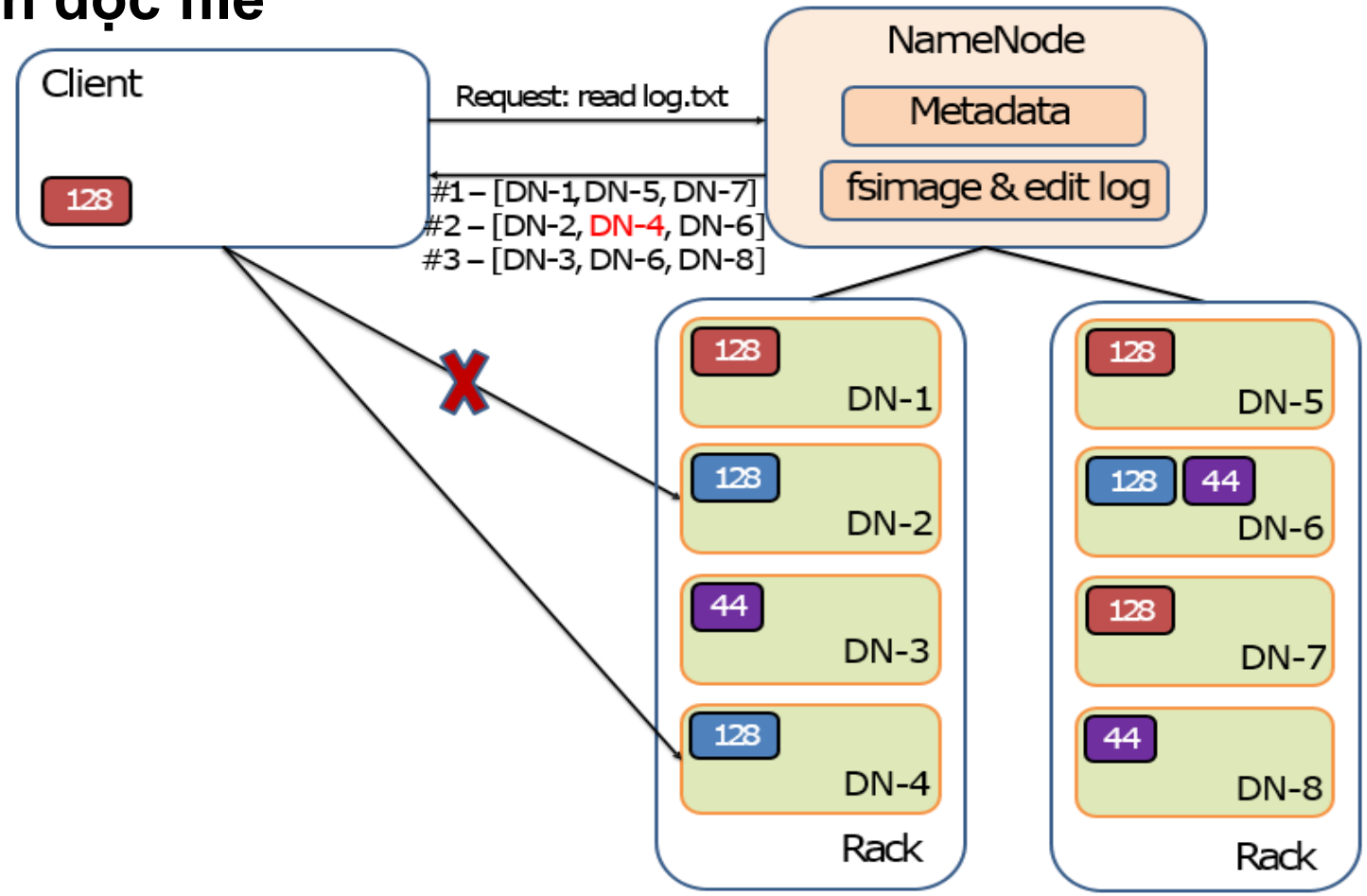


- Trong trường hợp không nhận được tín hiệu từ DataNode, sẽ tạo 1 đường truyền đến DataNode khác.

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

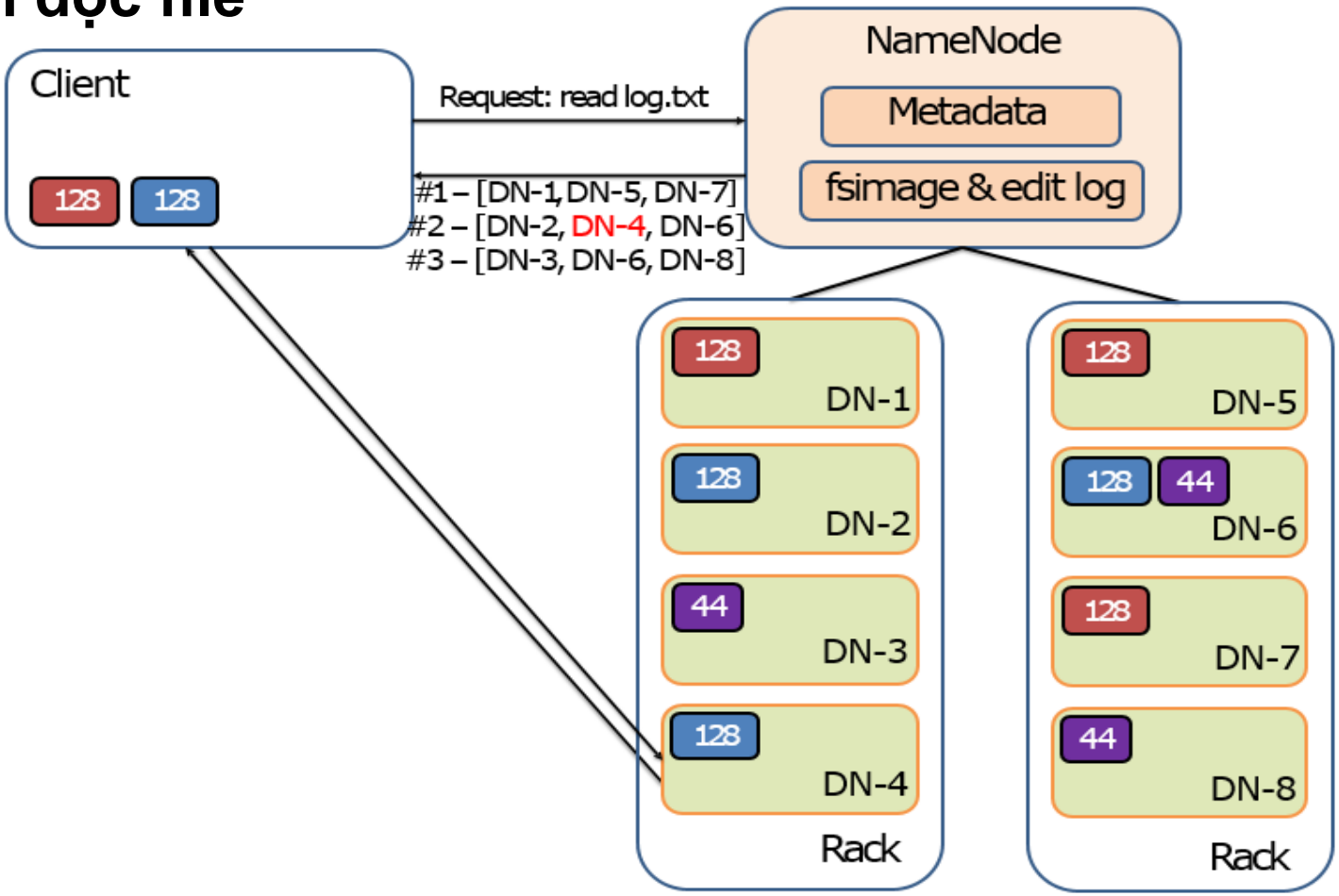
Quá trình đọc file



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

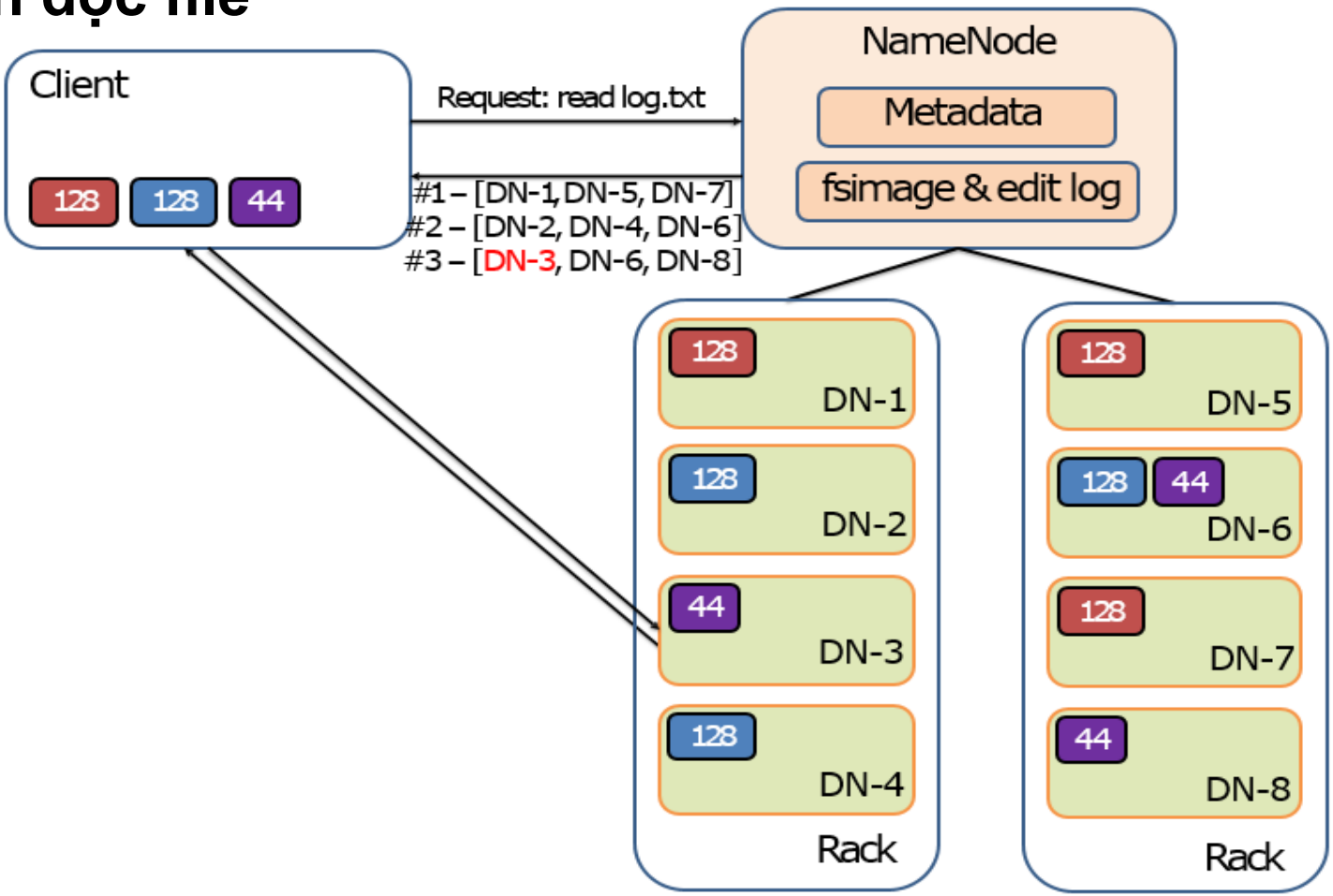
Quá trình đọc file



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

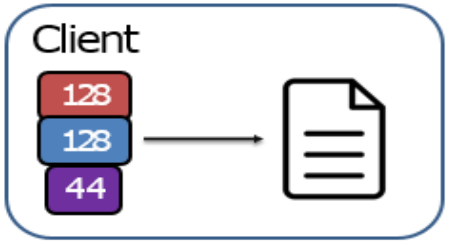
Quá trình đọc file

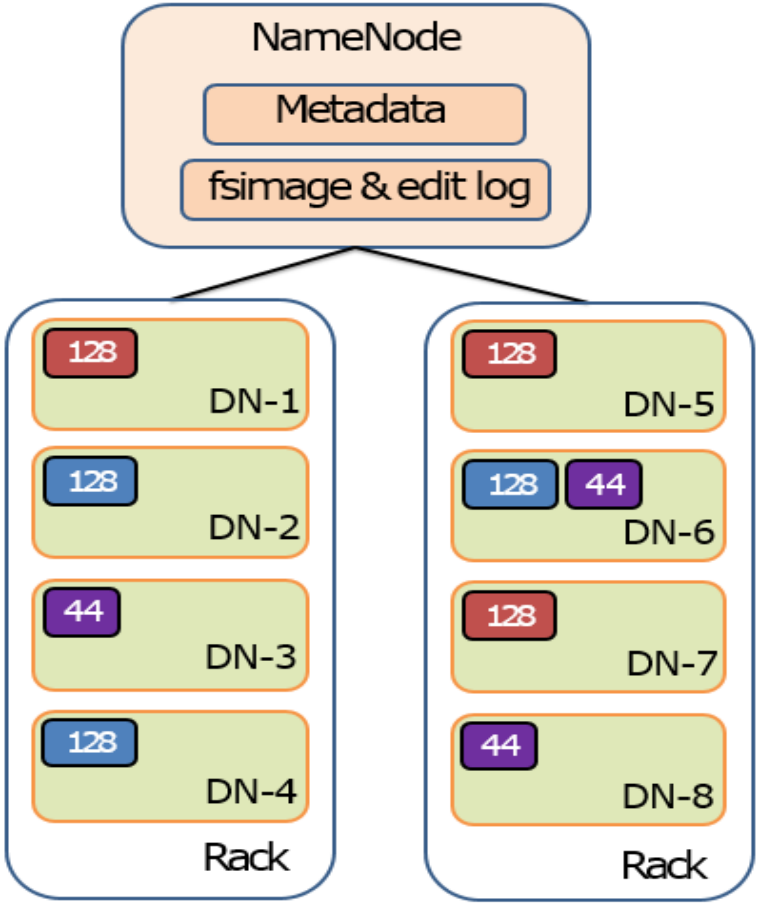


Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Quá trình đọc file





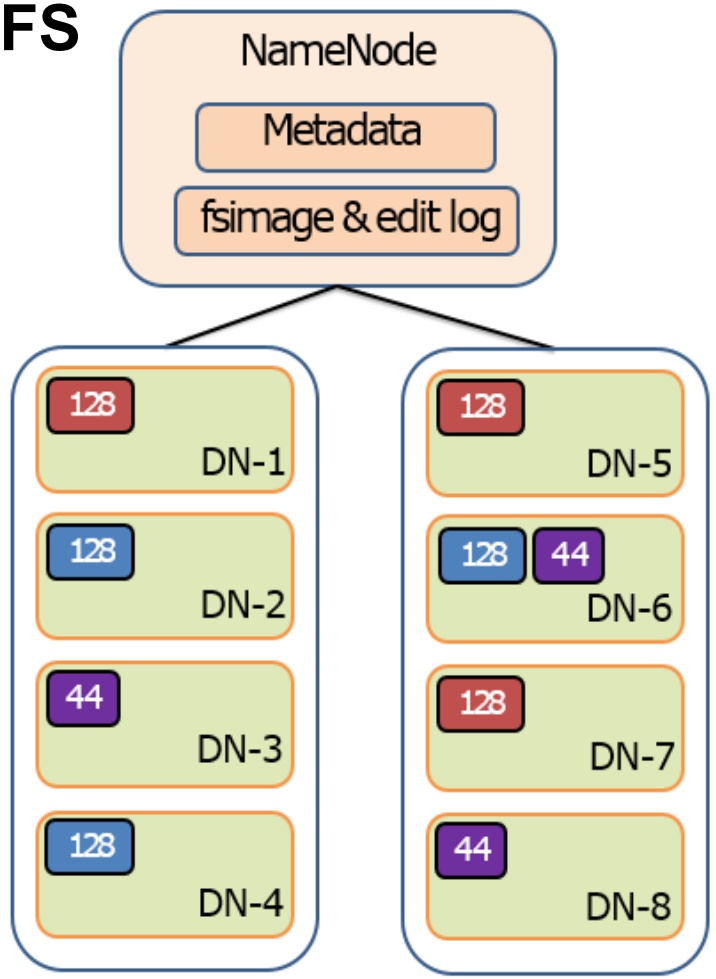
- Sau khi tải xong các block, các block sẽ được tổng hợp lại để tạo thành file hoàn chỉnh.

Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Khả năng chống chịu lỗi của HDFS

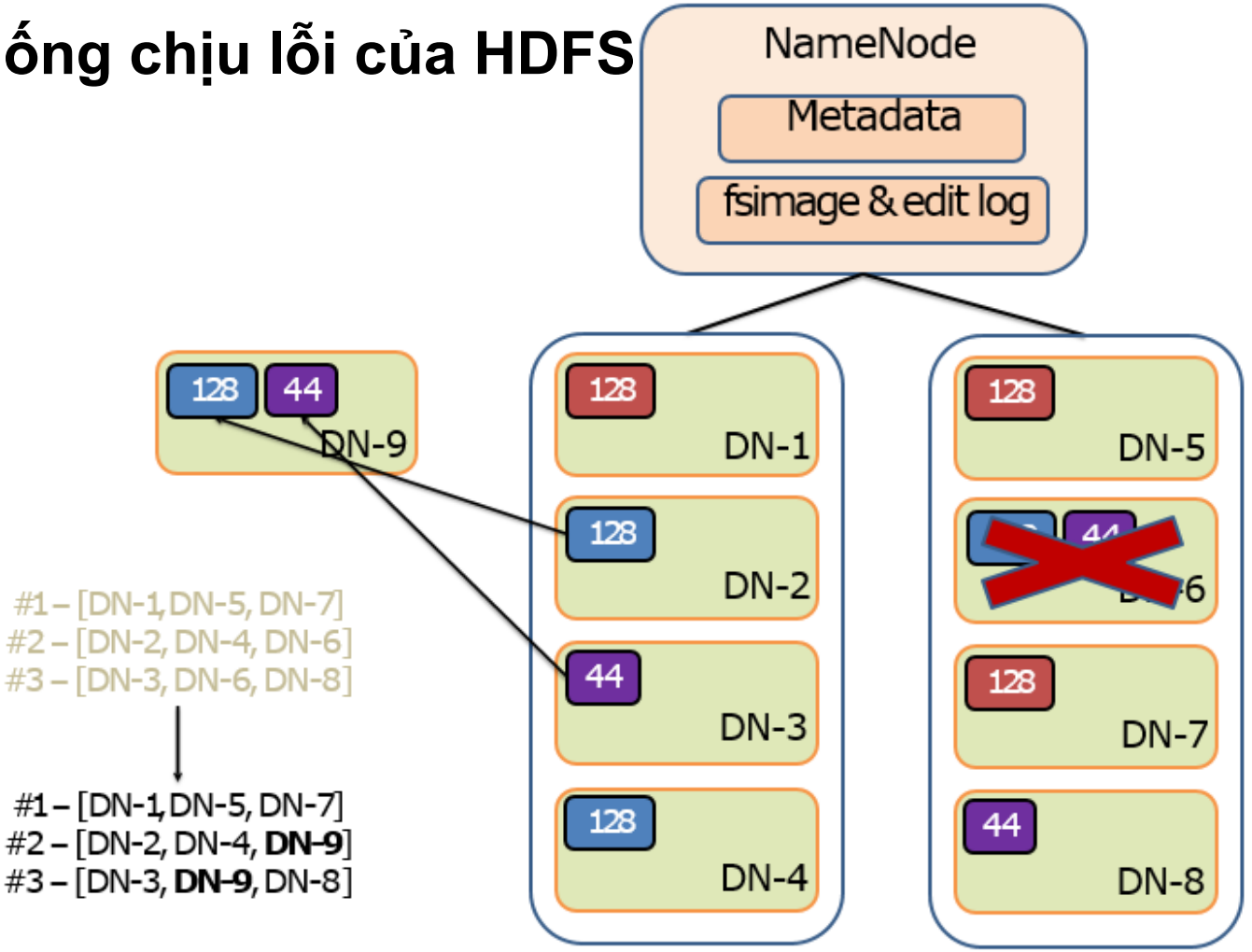
- DataNode gửi tín hiệu (Heartbeat) đến NameNode mỗi 3 giây để thông báo tình trạng hoạt động.
- Nếu sau 10 phút, NameNode không nhận được tín hiệu thì coi như DataNode đó không hoạt động và tiến hành tạo DataNode mới



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Khả năng chống chịu lỗi của HDFS

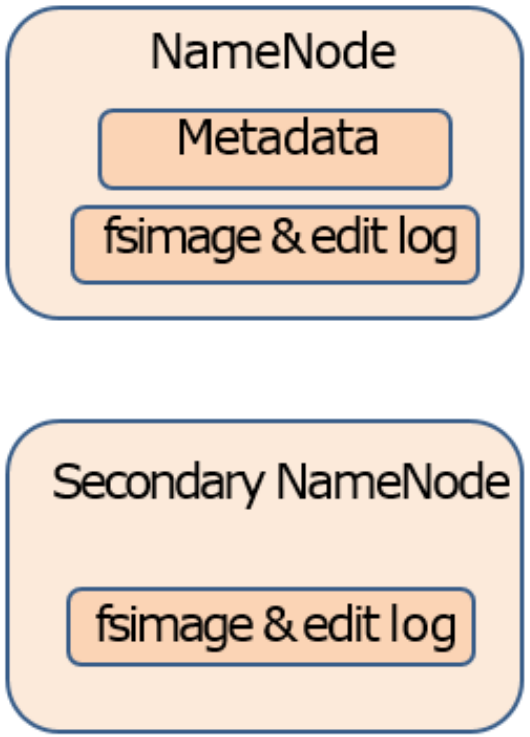


Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Secondary NameNode

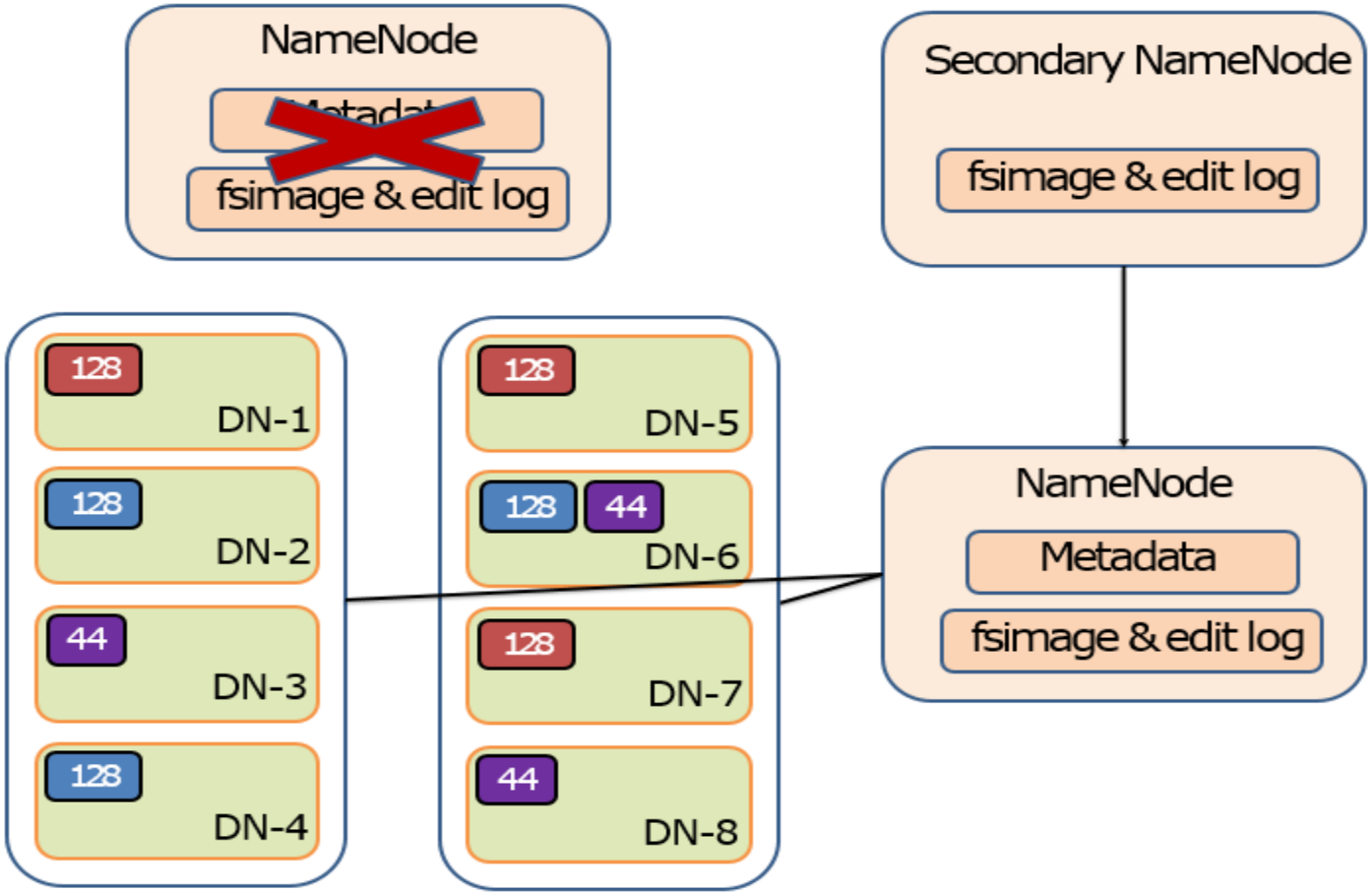
- Secondary NameNode lưu bản sao của fsimage & edit log của NameNode, phục vụ mục đích khôi phục lại NameNode mới trong trường hợp NameNode hiện tại hỏng.



Kiến trúc HDFS và Google File System

❖ Kiến trúc HDFS

Secondary NameNode



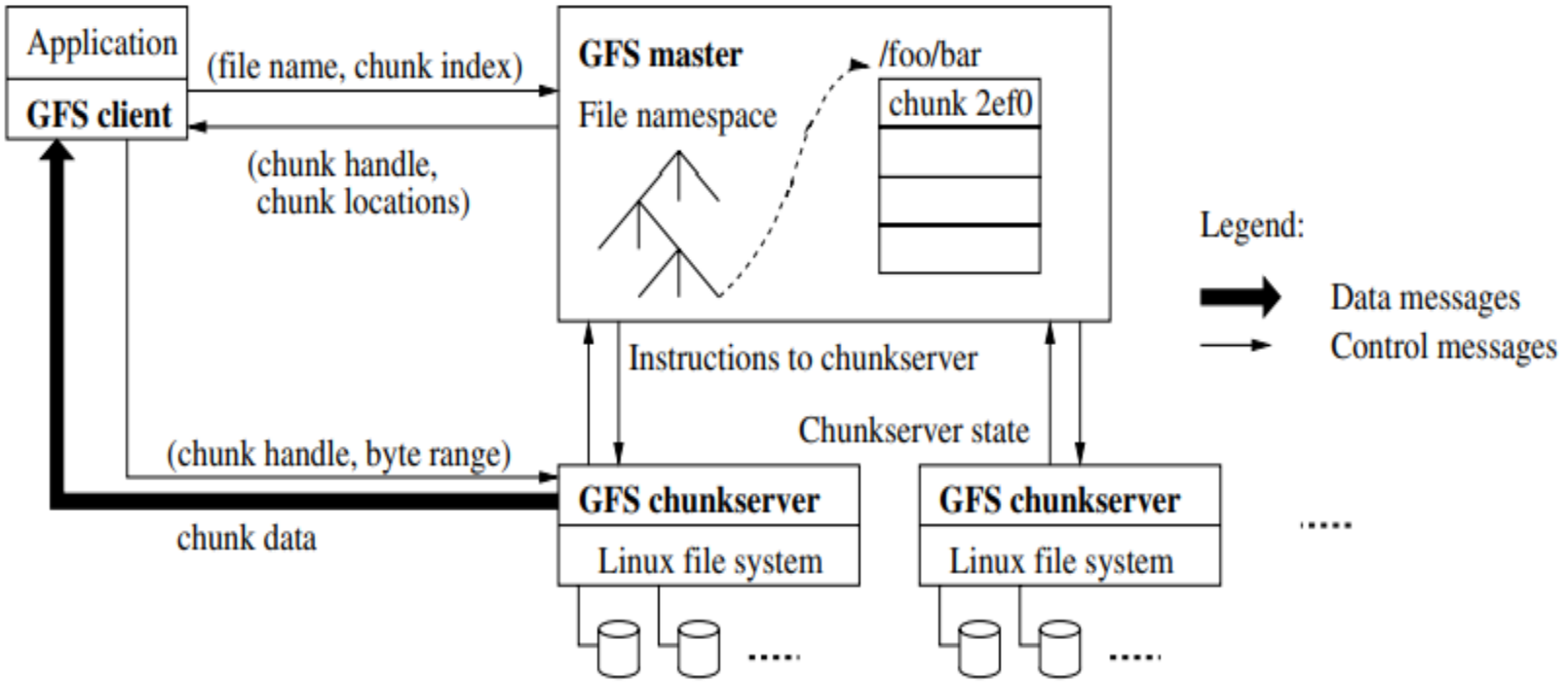
Kiến trúc HDFS và Google File System

❖ Google File System

- Google File System **(GFS):** hệ thống tệp phân tán có khả năng mở rộng đáp ứng nhu cầu xử lý dữ liệu ngày càng tăng của Google.
- **GFS** cung cấp khả năng chịu lỗi, độ tin cậy, khả năng mở rộng, tính khả dụng và hiệu suất cho các mạng lớn và các nút được kết nối. Quản lý hai loại dữ liệu là *File metadata* và *File Data*.

Kiến trúc HDFS và Google File System

❖ Google File System

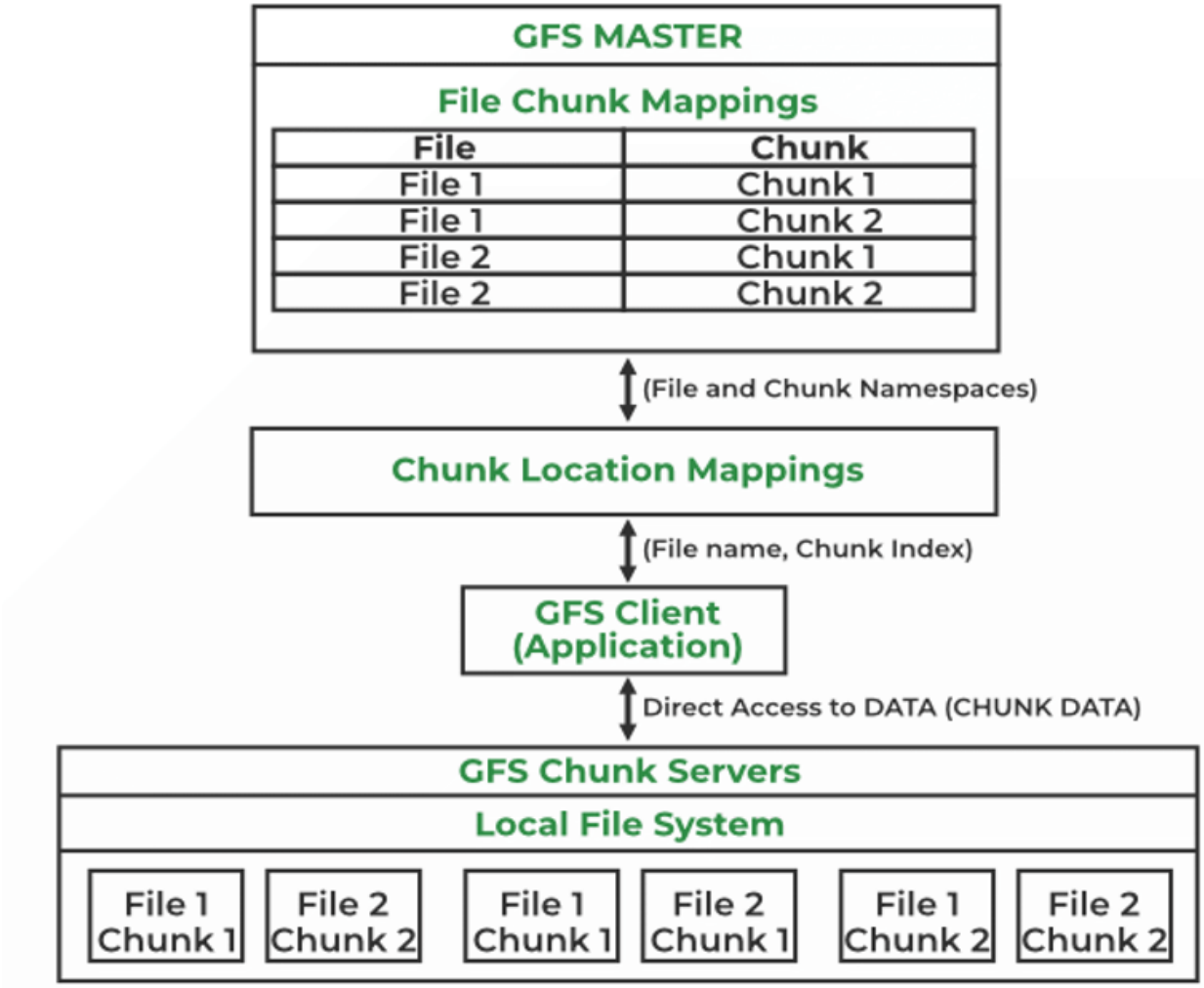


Kiến trúc Google File System

http://pdos.csail.mit.edu/6.824/papers/gfs.pdf

Kiến trúc HDFS và Google File System

❖ Google File System



Kiến trúc HDFS và Google File System

❖ Google File System

Các thành phần chính

- **GFS Client:** là các chương trình máy tính hoặc ứng dụng để yêu cầu truy cập và sửa đổi các tập tin đã tồn tại hoặc thêm các tập tin mới vào hệ thống.
- **GFS Master Server:** đóng vai trò là điều phối viên của cụm, lưu giữ hồ sơ về các hành động của cụm trong nhật ký hoạt động, theo dõi dữ liệu mô tả các khối, siêu dữ liệu.
- **Máy chủ Chunk GFS:** lưu trữ các chunk.GFS tạo nhiều bản sao của mỗi chunk và lưu trữ chúng trên nhiều máy chủ chunk khác nhau để đảm bảo tính ổn định;

Kiến trúc HDFS và Google File System

❖ Google File System

Các tính năng của GFS

- Quản lý và khóa không gian tên.
- Khả năng chịu lỗi.
- Giảm tương tác giữa máy khách và máy chủ chính vì kích thước máy chủ lớn.
- Tính khả dụng cao.
- Sao chép dữ liệu quan trọng.
- Phục hồi dữ liệu tự động và hiệu quả.
- Thông lượng tổng hợp cao.

Kiến trúc HDFS và Google File System

❖ Google File System

Các tính năng của GFS

- Quản lý và khóa không gian tên.
- Khả năng chịu lỗi.
- Giảm tương tác giữa máy khách và máy chủ chính vì kích thước máy chủ lớn.
- Tính khả dụng cao.
- Sao chép dữ liệu quan trọng.
- Phục hồi dữ liệu tự động và hiệu quả.
- Thông lượng tổng hợp cao.

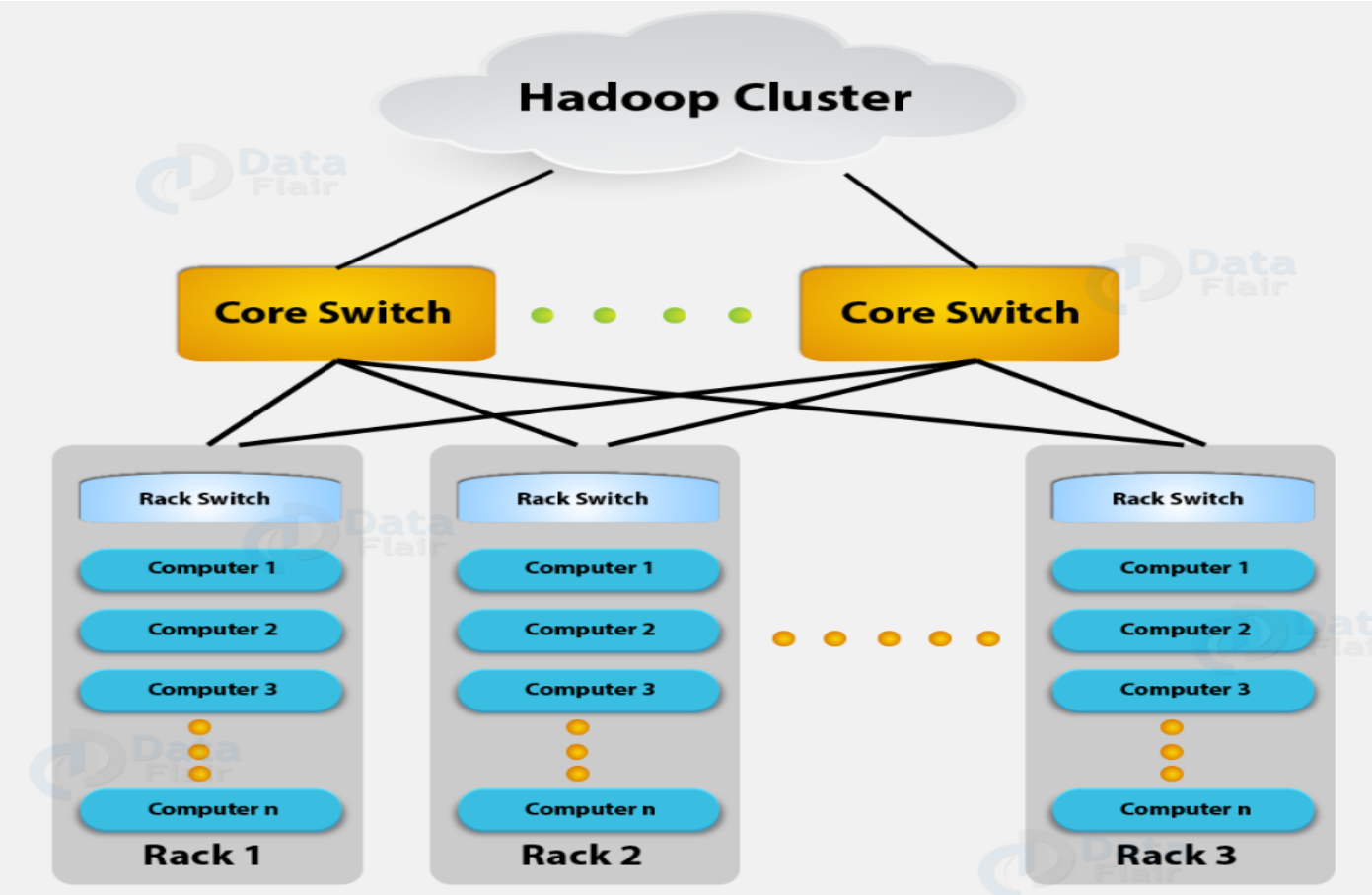
Cụm Hadoop

❖ Hadoop Cluster

- **Cụm Hadoop:** mục đích lưu trữ cũng như phân tích lượng lớn dữ liệu phi cấu trúc trong môi trường tính toán phân tán.
- Hai thành phần chính gồm cụm và nút:
  - ***Cụm:*** một tập hợp các nút là những gì chúng ta gọi là cụm.
  - ***Nút:*** là điểm giao nhau/kết nối trong một mạng (tức là một máy chủ)

Cụm Hadoop

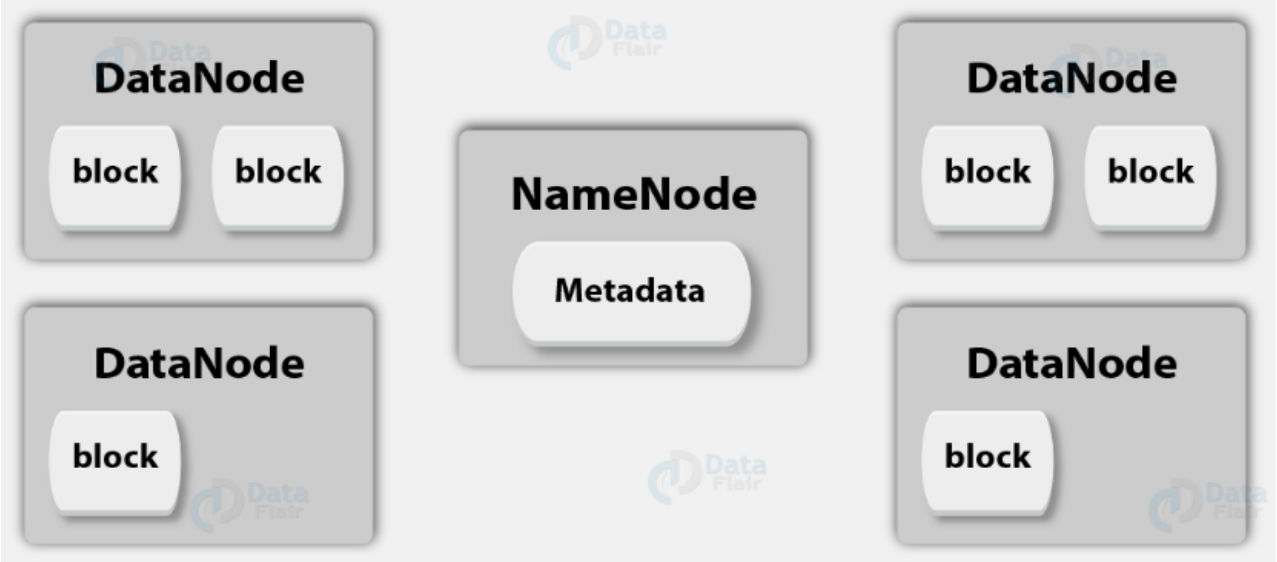
❖ Hadoop Cluster



Cụm Hadoop

❖ Hadoop Cluster

- **NameNode** là HDFS master, quản lý không gian tên hệ thống tệp và điều chỉnh quyền truy cập vào tệp của máy khách và cũng tham khảo **DataNode** (HDFS slave) trong khi sao chép dữ liệu hoặc chạy các hoạt động MapReduce.
- **DataNode** quản lý lưu trữ được gắn vào các nút mà chúng chạy trên đó, một DataNode cho mỗi slave trong cụm.



Cấu trúc liên kết mạng và Hadoop

Giao thức giao tiếp

- **Hadoop** sử dụng TCP IP: tất cả các giao thức giao tiếp HDFS đều được phân lớp. Trên máy NameNode: máy khách thiết lập kết nối với cổng TCP có thể định cấu hình.
- NameNode sẽ trao đổi ClientProtocol.
- Sử dụng Giao thức DataNode, DataNode sẽ trao đổi với NameNode.
- Remote Procedure Call (RPC) chứa cả Giao thức DataNode cũng như Giao thức Client. NameNode chỉ phản hồi các yêu cầu RPC do DataNode hoặc máy khách đưa ra.

Cấu trúc liên kết mạng và Hadoop

Cấu hình các nút trong Hadoop

- Hai loại cấu hình:

- Cấu hình mặc định chỉ đọc: core-default.xml, hdfs-default.xml, yarn-default.xml, mapred-default.xml.
- Cấu hình dành riêng: etc/hadoop/core-site.xml, etc/hadoop/hdfs-site.xml, etc/hadoop/yarn-site.xml và etc/hadoop/mapred-site.xml.

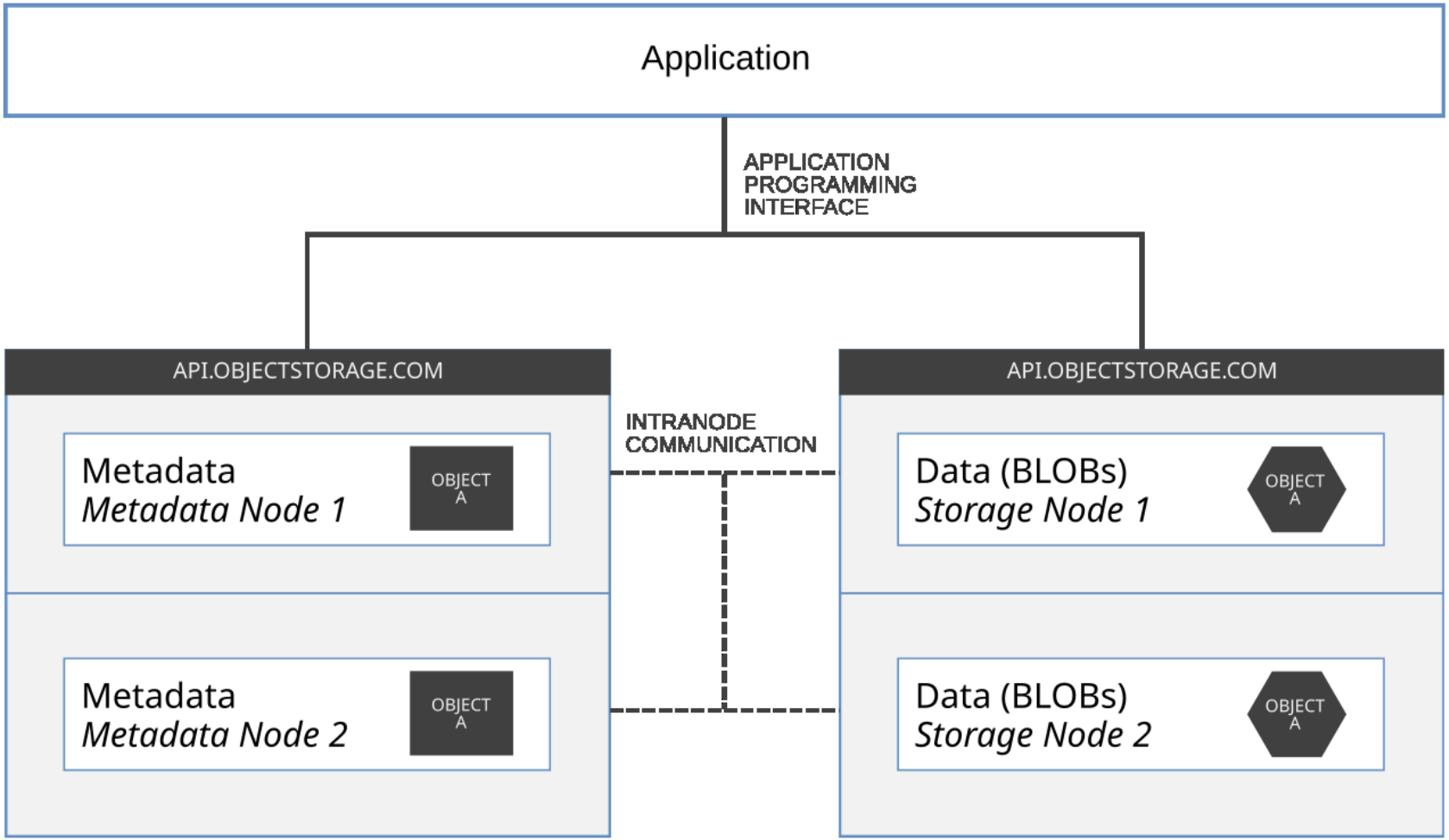
- Thiết lập các tệp và etc/hadoop/hadoop-env.sh và etc/hadoop/yarn-env.sh giúp kiểm soát các lệnh Hadoop
- Cấu hình môi trường thực thi cho cụm Hadoop:

- Các daemon HDFS: NameNode và DataNode.
- Daemon YARN: ResourceManager và NodeManager.

Object Storage

- Lưu trữ đối tượng (Object Storage): là một phương pháp lưu trữ dữ liệu máy tính, quản lý dữ liệu dưới dạng "blob" hoặc "đối tượng", khác với các kiến trúc lưu trữ khác như hệ thống tệp, quản lý dữ liệu dưới dạng phân cấp tệp và lưu trữ khối, quản lý dữ liệu dưới dạng khối trong các sector và track.
- Mỗi đối tượng được liên kết với một metadata và một mã định danh duy nhất.
- Triển khai ở nhiều cấp độ: cấp độ thiết bị (thiết bị lưu trữ đối tượng), cấp độ hệ thống và cấp độ giao diện.

Object Storage



Object Storage

- Cho phép lưu giữ lượng lớn dữ liệu phi cấu trúc trong đó dữ liệu được ghi một lần và đọc một lần (hoặc nhiều lần, sử dụng cho các mục đích như lưu trữ các đối tượngvideo và ảnh trên Facebook, bài hát trên Spotify hoặc tệp trong các dịch vụ cộng tác trực tuyến, chẳng hạn như Dropbox.
- Hạn chế của lưu trữ đối tượng:

- Không dành cho dữ liệu giao dịch, vì lưu trữ đối tượng không được thiết kế để thay thế việc truy cập và chia sẻ tệp NA.
- Không hỗ trợ các cơ chế khóa và chia sẻ cần thiết để duy trì một phiên bản tệp duy nhất được cập nhật chính xác.

Các công cụ phổ biến

- HBase và Big Table
- MapReduce, Hive, và Pig
- YARN
- Ceph
- MinIO

Các công cụ phổ biến

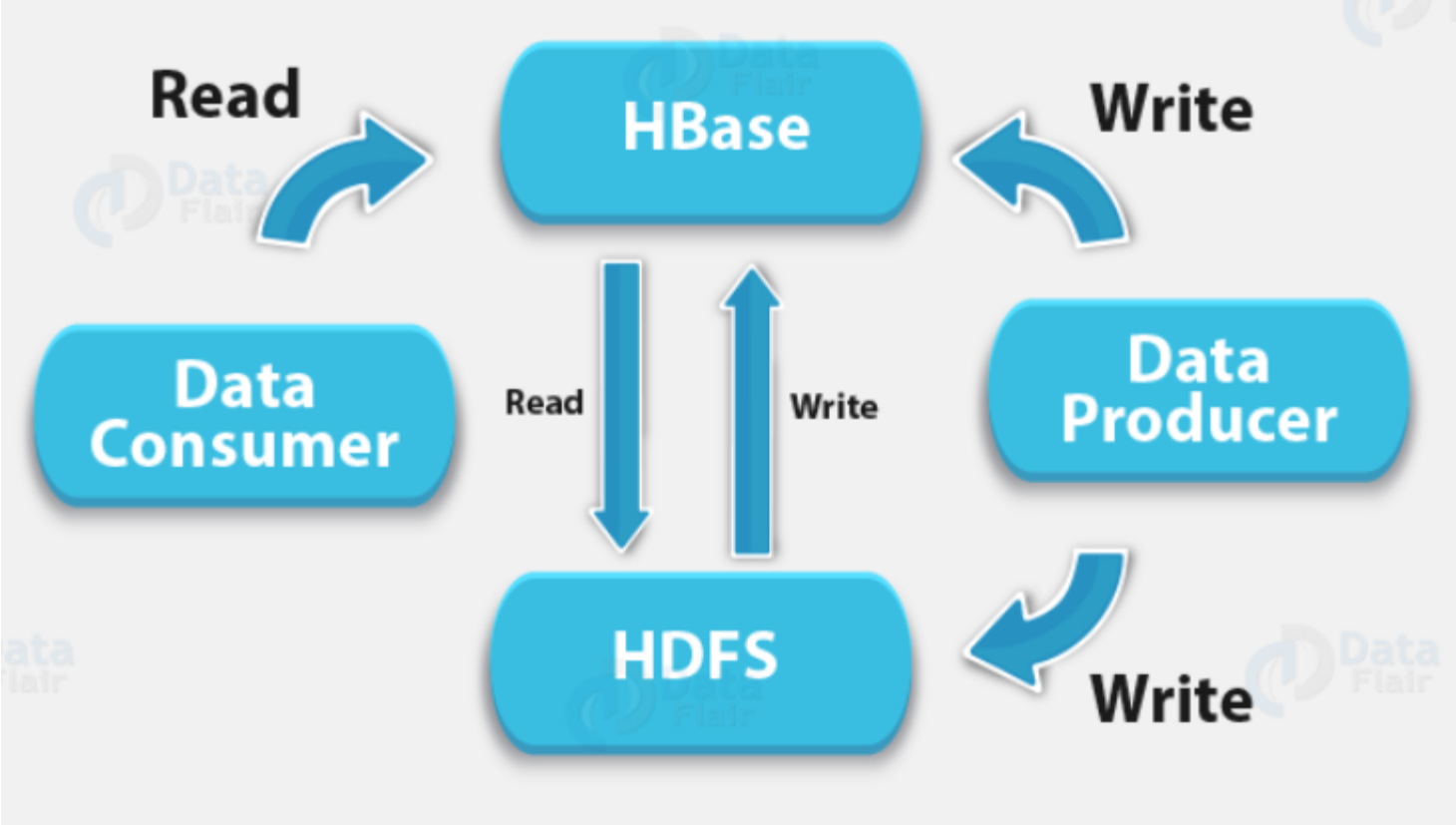
❖ HBase và Big Table

HBase là một dự án Hadoop là cơ sở dữ liệu Hadoop phân tán, mã nguồn mở có nguồn gốc từ Bigtable của Google.

- Sử dụng ngôn ngữ lập trình Java.
- Là một phần không thể thiếu của Apache Software Foundation và hệ sinh thái Hadoop.
- Là một cơ sở dữ liệu có tính khả dụng cao chỉ chạy trên HDFS.
- Đây là cơ sở dữ liệu theo cột được xây dựng trên HDFS.

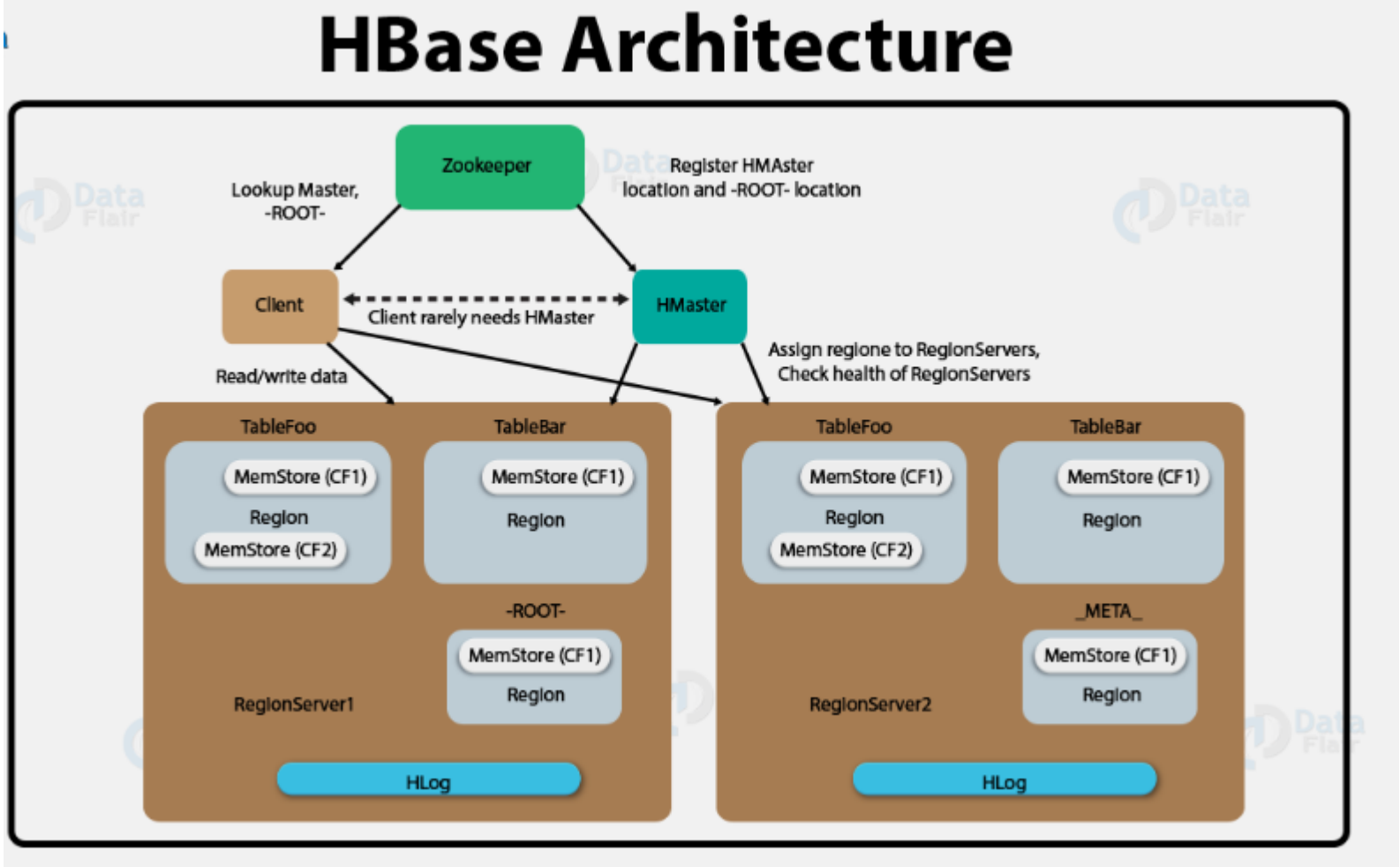
Các công cụ phổ biến

❖ HBase và Big Table



Các công cụ phổ biến

❖ HBase và Big Table



Các công cụ phổ biến

❖ MapReduce, Hive, và Pig

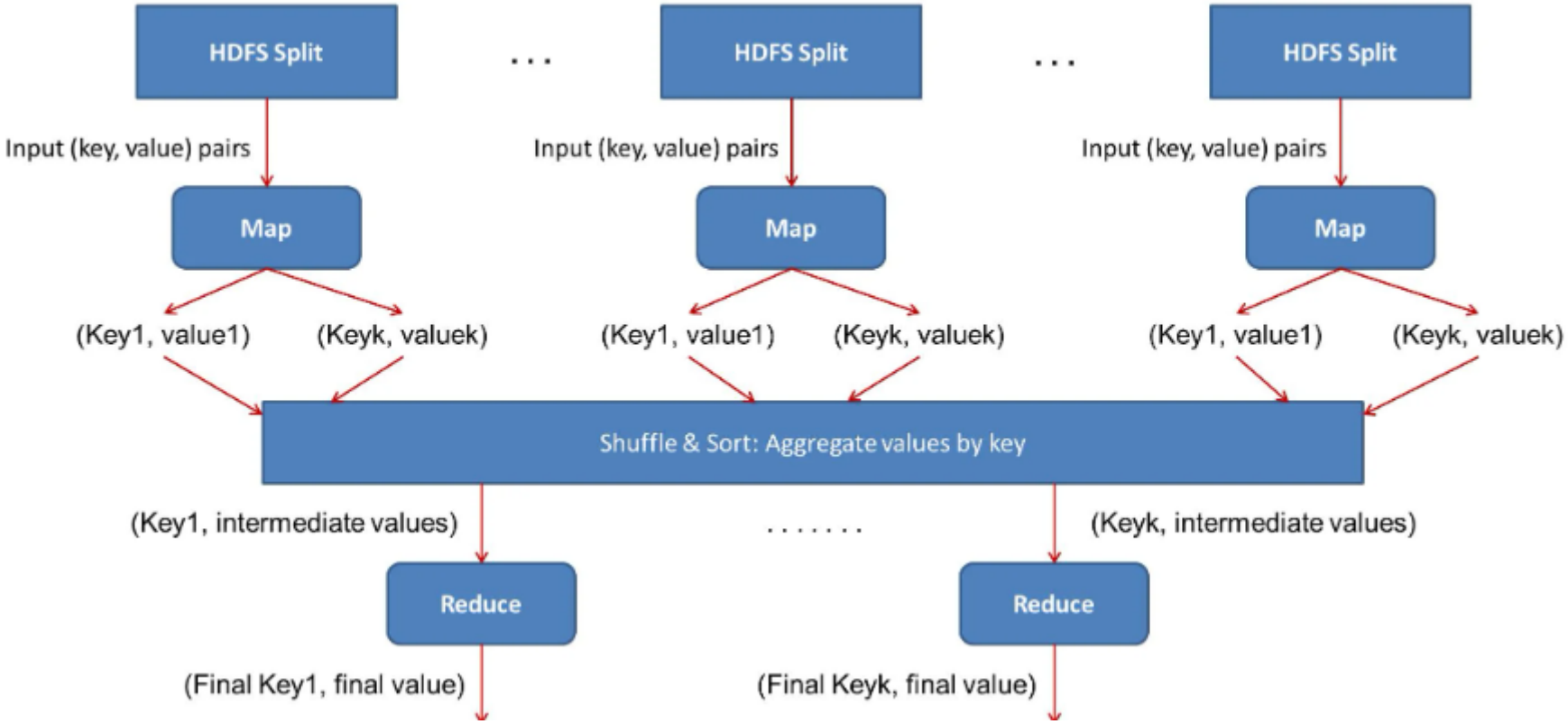
MapReduce: là lớp xử lý của Hadoop.

- Mô hình lập trình MapReduce được thiết kế để xử lý khối lượng dữ liệu lớn song song bằng cách chia công việc thành một tập hợp các tác vụ độc lập.
- Đưa nghiệp vụ vào MapReduce xử lý. Công việc (công việc hoàn thành) do người dùng gửi đến master được chia thành các công việc nhỏ (nhiệm vụ) và được giao cho các slave.

Các công cụ phổ biến

- MapReduce, Hive, và Pig

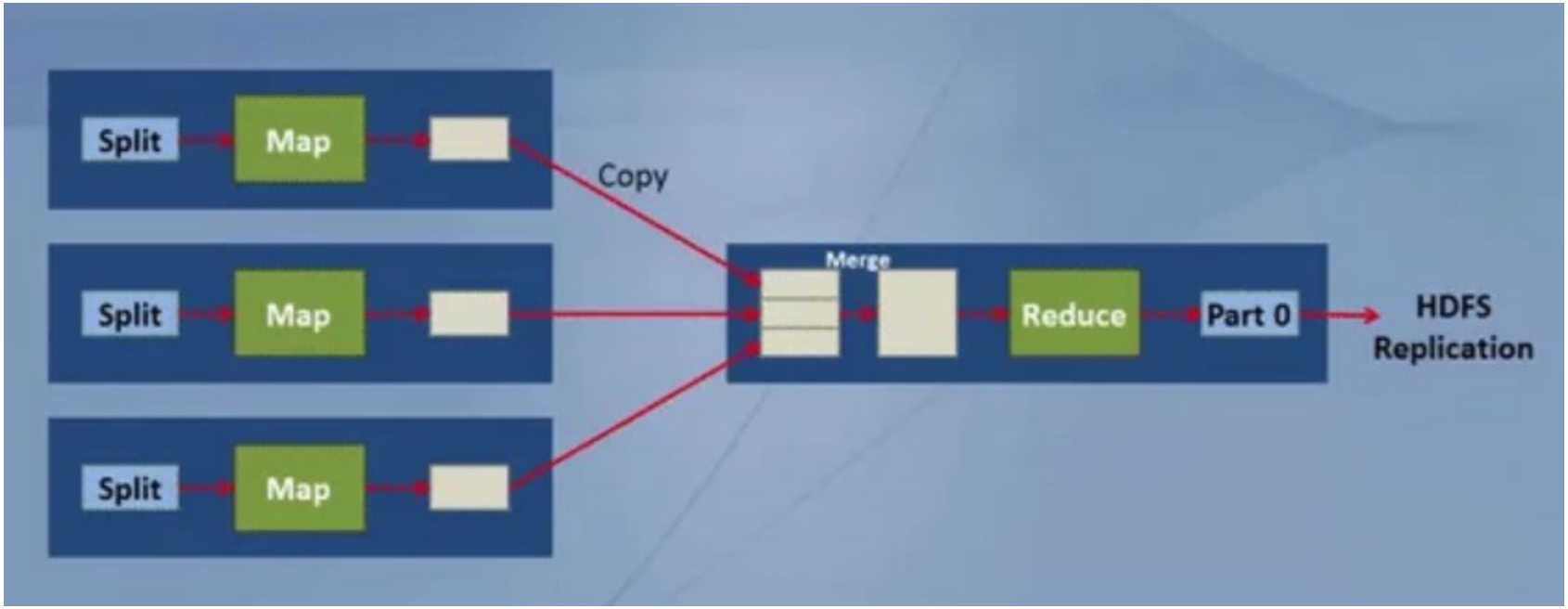
MapReduce:



Các công cụ phổ biến

- MapReduce, Hive, và Pig

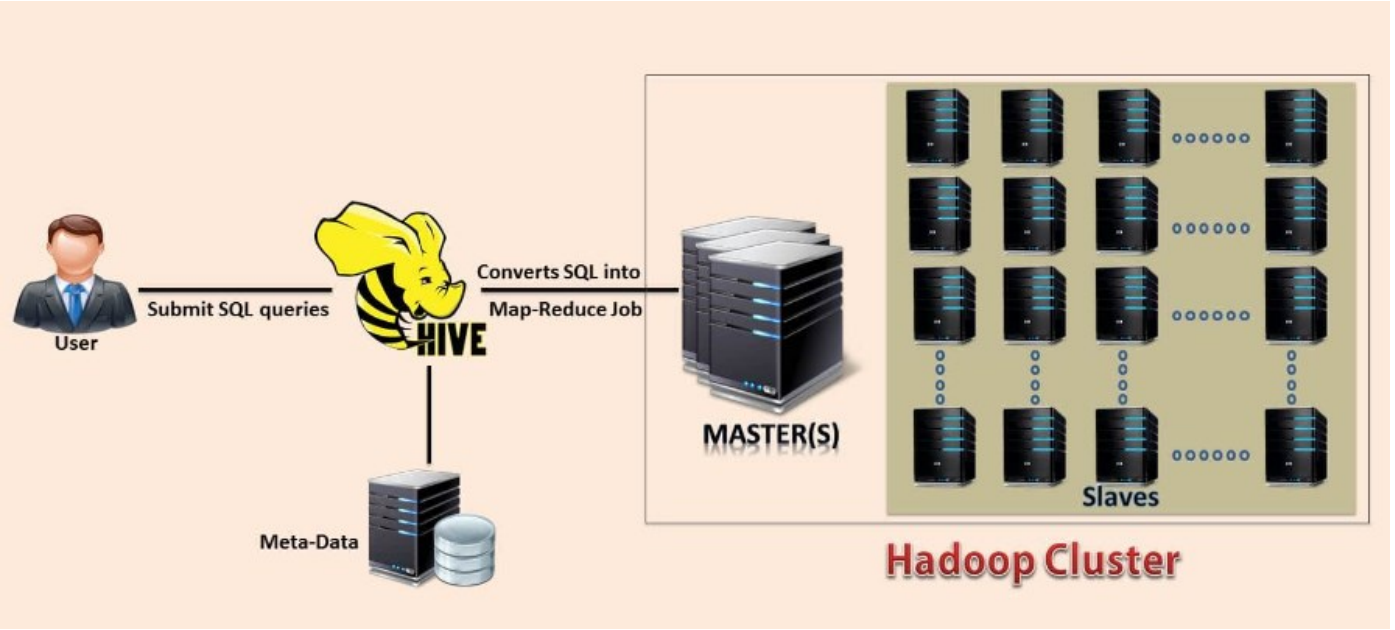
MapReduce:



Các công cụ phổ biến

❖ MapReduce, Hive, và Pig

Apache Hive: hệ thống nguồn mở được xây dựng trên Hadoop dùng để truy vấn và phân tích các tập dữ liệu lớn được lưu trữ trong các tệp Hadoop *(xử lý dữ liệu có cấu trúc và bán cấu trúc trong Hadoop)*.



Các công cụ phổ biến

❖ MapReduce, Hive, và Pig

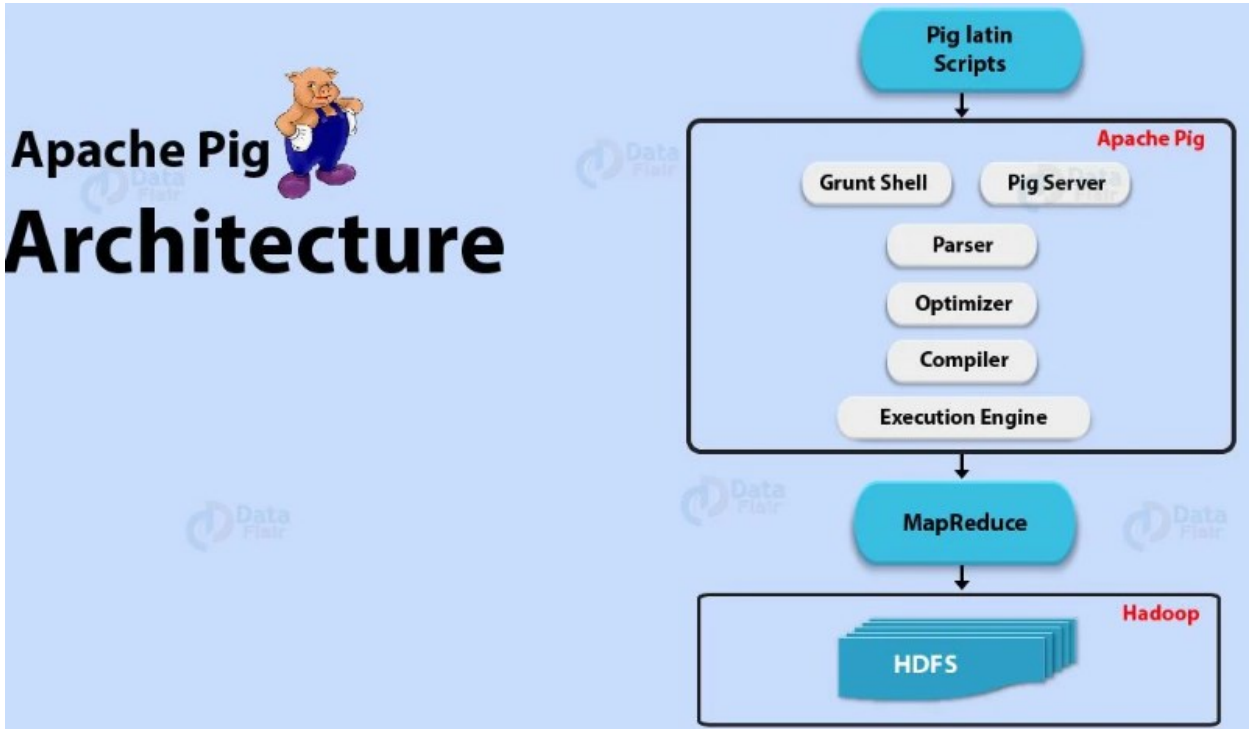
Apache Hive có ưu điểm:

- Hive sử dụng ngôn ngữ được gọi là HiveQL (HQL), tương tự như SQL.
- HiveQL tự động dịch các truy vấn giống SQL thành các tác vụ MapReduce.
- Hive thường chạy trên máy trạm của bạn và chuyển đổi truy vấn SQL của bạn thành một loạt các tác vụ để thực thi trên cụm Hadoop.
- Apache Hive sắp xếp dữ liệu thành các bảng, cung cấp phương tiện để gắn cấu trúc vào dữ liệu được lưu trữ trong HDFS.

Các công cụ phổ biến

❖ MapReduce, Hive, và Pig

Apache Pig được phát triển như một dự án nghiên cứu vào năm 2006 tại Yahoo, để tạo và thực thi các công việc MapReduce trên các tập dữ liệu được tạo ra.



Các công cụ phổ biến

❖ MapReduce, Hive, và Pig

Apache Pig có ưu điểm:

- Sử dụng Pig Latin giúp việc dễ dàng thực hiện các tác vụ MapReduce.
- Cung cấp nhiều toán tử tích hợp, để hỗ trợ các hoạt động dữ liệu như nối, bộ lọc, sắp xếp,... giống SQL.
- Cung cấp các kiểu dữ liệu lồng nhau không có trong MapReduce như tuples, bags và maps.

Các công cụ phổ biến

❖ YARN

- **YARN** (Yet-Another-Resource-Negotiator) là một framework hỗ trợ phát triển ứng dụng phân tán.
- Cho phép các công cụ xử lý dữ liệu khác nhau như xử lý đồ thị, xử lý tương tác, xử lý luồng cũng như xử lý hàng loạt chạy và xử lý dữ liệu được lưu trữ trong HDFS (Hệ thống tệp phân tán Hadoop) do đó làm cho hệ thống hiệu quả hơn nhiều, phân bổ động nhiều tài nguyên khác nhau và lên lịch xử lý ứng dụng.

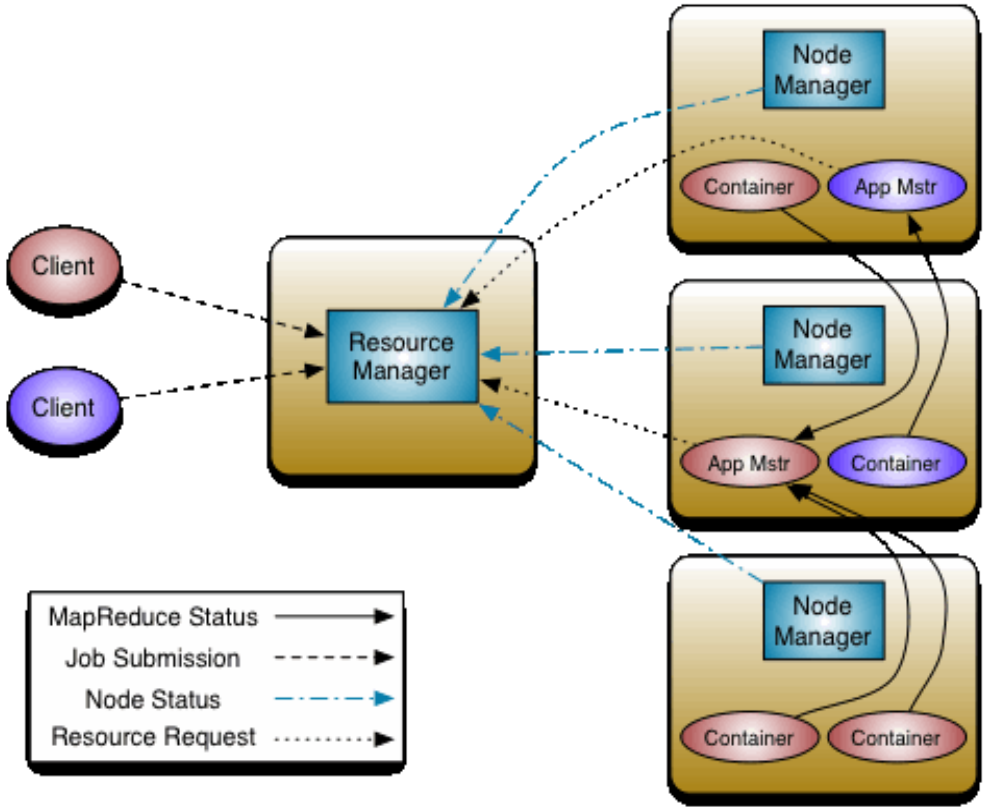
Các công cụ phổ biến

❖ YARN

Gồm hai trình quản lý:

- **ResourceManage**
Quản lý toàn bộ tà
nguyên tính toán củ
cluster.

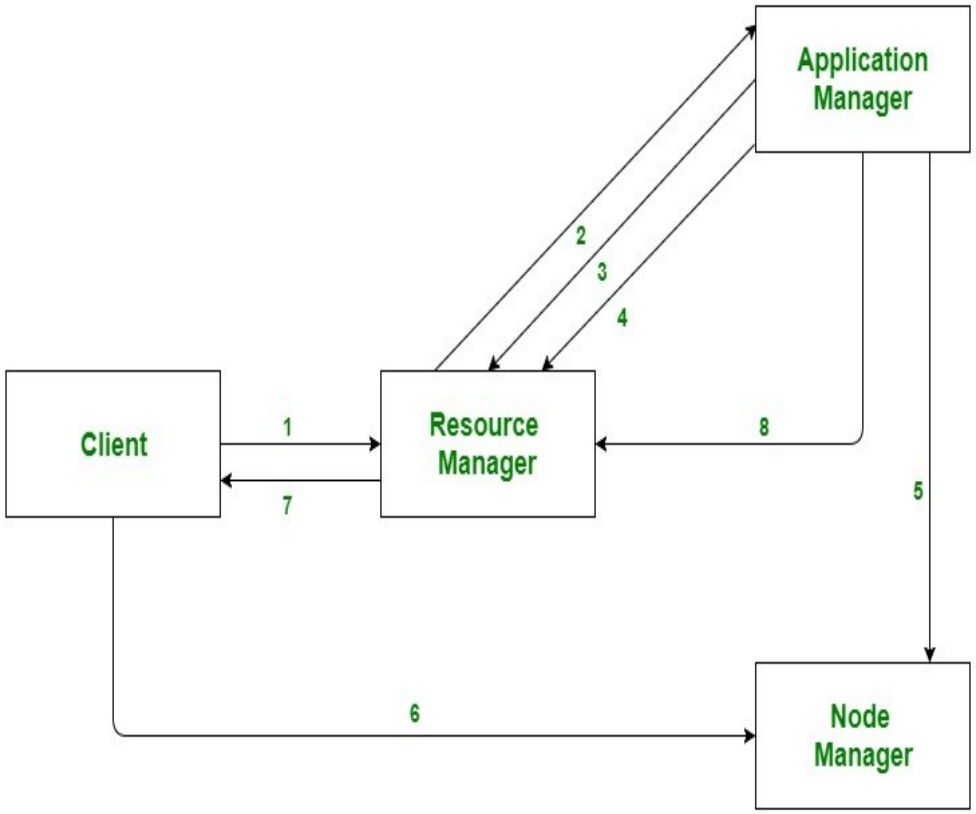
- Node**Manger:** Giá
sát việc sử dụng tà
nguyên của containe
và báo cáo vớ
ResourceManger nh
CPU, memory, disk
network,...



Các công cụ phổ biến

❖ YARN

1. Khách hàng nộp đơn
2. Trình quản lý tài nguyên phân bổ một vùng chứa để khởi động Trình quản lý ứng dụng
3. Trình quản lý ứng dụng tự đăng ký với Trình quản lý tài nguyên
4. Trình quản lý ứng dụng đàm phán các container từ Trình quản lý tài nguyên
5. Trình quản lý ứng dụng thông báo cho Trình quản lý nút để khởi chạy các vùng chứa
6. Mã ứng dụng được thực thi trong container
7. Khách hàng liên hệ với Resource Manager/Application Manager để theo dõi trạng thái của ứng dụng
8. Sau khi quá trình xử lý hoàn tất, Trình quản lý ứng dụng sẽ hủy đăng ký với Trình quản lý tài nguyên



Qui trình làm việc của YARN trong Hadoop

Các công cụ phổ biến

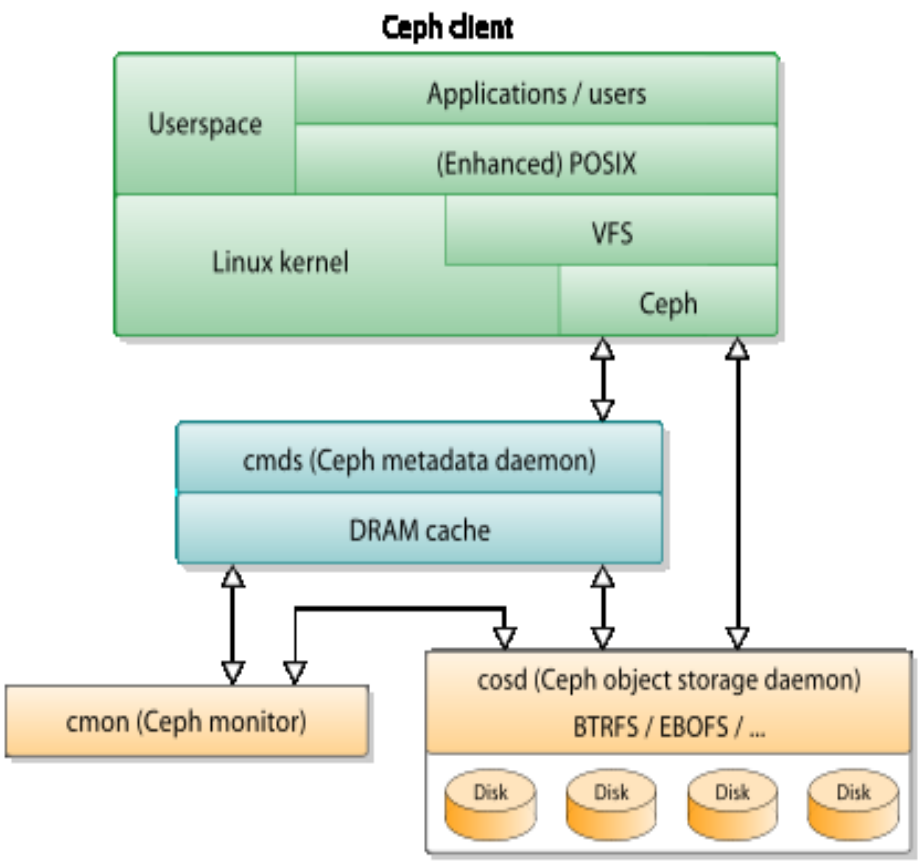
❖ Ceph

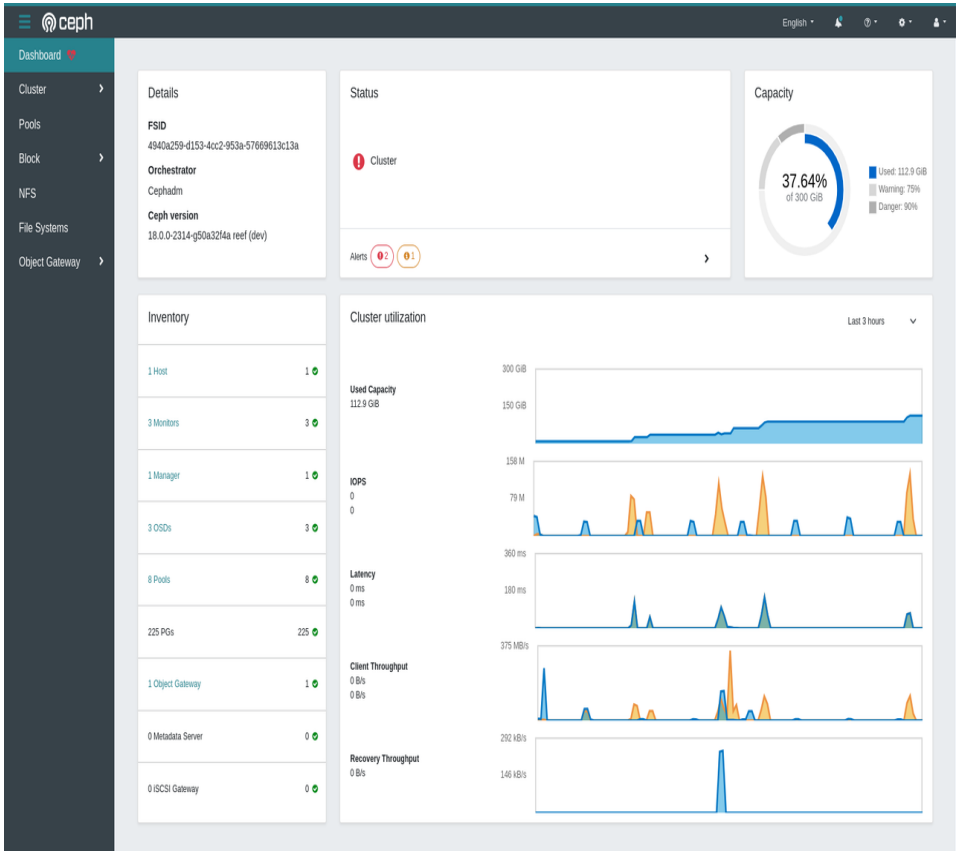
Ceph là một nền tảng lưu trữ được định nghĩa bằng phần mềm mã nguồn mở, cung cấp lưu trữ đối tượng, lưu trữ khối và lưu trữ tệp được xây dựng trên nền tảng cụm phân tán chung.

- Ceph cung cấp hoạt động phân tán mà không có điểm lỗi đơn và khả năng mở rộng đến cấp exabyte.
- Ceph sao chép dữ liệu với khả năng chịu lỗi, sử dụng phần cứng thông dụng và IP Ethernet.
- Ceph có tính khả dụng cao và đảm bảo độ bền dữ liệu mạnh mẽ thông qua các kỹ thuật bao gồm sao chép, mã hóa xóa, ảnh chụp nhanh và bản sao. Hệ thống vừa tự phục hồi vừa tự quản lý, giảm thiểu thời gian quản trị và các chi phí khác.

Các công cụ phổ biến

- Ceph





Các công cụ phổ biến

❖ MinIO

- **MinIO:** là hệ thống lưu trữ đối tượng hiệu suất cao, được thiết kế để thay thế cho hệ thống lưu trữ đám mây gốc. API tương thích với Amazon S3.
- Cung cấp nhiều tùy chọn triển khai: có thể chạy như một ứng dụng gốc trên hầu hết các kiến trúc phổ biến được triển khai như một ứng dụng chứa trong container bằng Docker hoặc Kubernetes.
- MinIO là phần mềm nguồn mở, công cụ để phát triển và thử nghiệm, cũng như các kịch bản DevOps.

Các công cụ phổ biến

- MinIO

http://github.com/minio/minio