Môn học "Nhập môn Phân tích Dữ liệu Lớn – 71ITDS40303" Câu hỏi Thi Kết hợp (Tự luận + Vấn đáp) – HK242

Phần I Tổng quan

Câu 1: Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới khái niệm Dữ liệu Lớn, Tại sao cần áp dụng các công cụ lưu trữ và phân tích dữ liệu đặc thù (Hadoop HDFS, MapReduce) khi thực hiện phân tích Dữ liệu Lớn so với Dữ liệu truyền thống ?

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu khổng lồ, phức tạp, tăng nhanh, vượt khả năng xử lý của công cụ truyền thống, bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc.

Thuộc tính (5V):

1. **Volume**: Khối lượng dữ liệu lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý dữ liệu nhanh, gần thời gian thực.
3. **Variety**: Đa dạng nguồn và định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng nhất.
5. **Value**: Giá trị từ việc khai thác thông tin.

Tại sao cần Hadoop HDFS, MapReduce cho Dữ liệu Lớn?

Hạn chế của công cụ truyền thống:

- Không xử lý được khối lượng lớn, tốc độ cao, dữ liệu đa dạng.
- Chi phí mở rộng cao, kém linh hoạt.

Lý do dùng Hadoop HDFS, MapReduce:

1. **HDFS**:
    - Lưu trữ phân tán, chia nhỏ dữ liệu, chịu lỗi cao.
    - Phù hợp dữ liệu lớn, chi phí thấp.
2. **MapReduce**:
    - Xử lý song song, mở rộng dễ dàng.
    - Hỗ trợ dữ liệu đa dạng, hiệu quả cao.
3. **Tiết kiệm chi phí**: Dùng phần cứng giá rẻ, mã nguồn mở.
4. **Tích hợp tốt**: Nền tảng cho Hive, Spark, v.v.

Kết luận: HDFS và MapReduce giải quyết tốt các thách thức của Dữ liệu Lớn về lưu trữ, xử lý phân tán, mở rộng linh hoạt, chi phí thấp, phù hợp với khối lượng, tốc độ và đa dạng dữ liệu.

Câu 2: Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Liệt kê một số công nghệ và giải pháp phổ biến trong phân tích Dữ liệu Lớn. Cho một thí dụ thực tế áp dụng công nghệ và giải pháp phân tích Dữ liệu Lớn mà sinh viên đã tìm hiểu. Đâu là khó khăn chính khi thực hiện phân tích Dữ liệu Lớn ?

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu có khối lượng cực lớn, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của các công cụ truyền thống. Nó bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ nhiều nguồn như mạng xã hội, IoT, giao dịch trực tuyến.

Thuộc tính (5V):

1. **Volume**: Khối lượng dữ liệu khổng lồ (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý dữ liệu cao, gần thời gian thực.
3. **Variety**: Đa dạng định dạng và nguồn (văn bản, hình ảnh, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều, cần làm sạch dữ liệu.
5. **Value**: Giá trị khai thác từ dữ liệu để hỗ trợ quyết định.

Các công nghệ và giải pháp phổ biến trong phân tích Dữ liệu Lớn

1. **Hadoop**: Hệ thống lưu trữ (HDFS) và xử lý phân tán (MapReduce) cho dữ liệu lớn.
2. **Apache Spark**: Nền tảng xử lý nhanh, hỗ trợ phân tích thời gian thực và học máy.
3. **Apache Kafka**: Hệ thống xử lý luồng dữ liệu thời gian thực.
4. **NoSQL Databases**: MongoDB, Cassandra, HBase cho dữ liệu phi cấu trúc, mở rộng dễ dàng.
5. **Data Warehouse**: Snowflake, Google BigQuery để lưu trữ và phân tích dữ liệu lớn.
6. **Công cụ BI**: Tableau, Power BI để trực quan hóa và phân tích dữ liệu.
7. **Học máy và AI**: TensorFlow, PyTorch để dự đoán và phân tích nâng cao.

Thí dụ thực tế áp dụng phân tích Dữ liệu Lớn

Ứng dụng: Netflix sử dụng Big Data để gợi ý phim cá nhân hóa.

- **Công nghệ**: Apache Spark, AWS S3, và thuật toán học máy.

- **Cách hoạt động**: Netflix thu thập dữ liệu về lịch sử xem, đánh giá, tìm kiếm của người dùng. Spark xử lý khối lượng dữ liệu lớn này để phân tích hành vi, sau đó thuật toán học máy dự đoán sở thích và gợi ý nội dung phù hợp.
- **Kết quả**: Tăng mức độ hài lòng của người dùng, giữ chân khách hàng.

Khó khăn chính khi thực hiện phân tích Dữ liệu Lớn

1. **Quản lý khối lượng dữ liệu**: Lưu trữ và xử lý dữ liệu lớn đòi hỏi cơ sở hạ tầng mạnh, chi phí cao.
2. **Tốc độ xử lý**: Đáp ứng yêu cầu phân tích thời gian thực với dữ liệu tăng nhanh là thách thức.
3. **Chất lượng dữ liệu**: Dữ liệu không đồng nhất, thiếu chính xác, cần làm sạch và chuẩn hóa.
4. **Bảo mật và quyền riêng tư**: Dữ liệu lớn thường nhạy cảm, dễ bị tấn công hoặc vi phạm quy định (như GDPR).
5. **Thiếu nhân lực**: Yêu cầu chuyên gia có kỹ năng về Big Data, học máy, nhưng nguồn nhân lực còn hạn chế.
6. **Tích hợp hệ thống**: Kết nối dữ liệu từ nhiều nguồn khác nhau (cơ sở dữ liệu, API, IoT) phức tạp và tốn thời gian.

Kết luận

Dữ liệu Lớn mang lại giá trị lớn nhưng đòi hỏi công nghệ phù hợp như Hadoop, Spark, và giải pháp NoSQL để xử lý. Tuy nhiên, các khó khăn về cơ sở hạ tầng, chất lượng dữ liệu, bảo mật và nhân lực là những thách thức cần vượt qua để khai thác hiệu quả.

Câu 3: Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Mô tả hoạt động của công cụ phân tích Dữ liệu Lớn phổ biến trong môi trường điện toán đám mây Google Cloud Platform.

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu có khối lượng khổng lồ, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của các công cụ truyền thống. Nó bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ nhiều nguồn như mạng xã hội, IoT, giao dịch trực tuyến.

Thuộc tính (5V):

1. **Volume**: Khối lượng dữ liệu lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý dữ liệu cao, gần thời gian thực.
3. **Variety**: Đa dạng về nguồn và định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều, cần làm sạch.
5. **Value**: Giá trị khai thác từ dữ liệu để hỗ trợ quyết định.

Mô tả hoạt động của công cụ phân tích Dữ liệu Lớn trên Google Cloud Platform (GCP)

Google Cloud Platform cung cấp nhiều công cụ mạnh mẽ để phân tích Dữ liệu Lớn, tích hợp tốt trong môi trường điện toán đám mây. Một số công cụ phổ biến và cách chúng hoạt động bao gồm:

1. **Google BigQuery**:
    - **Chức năng**: Kho dữ liệu (data warehouse) serverless, tối ưu cho phân tích dữ liệu lớn.
    - **Hoạt động**:
        - Lưu trữ dữ liệu trong định dạng cột (columnar format) để truy vấn nhanh.
        - Sử dụng SQL tiêu chuẩn, hỗ trợ xử lý petabyte dữ liệu mà không cần quản lý cơ sở hạ tầng.
        - Tự động mở rộng tài nguyên theo nhu cầu, giảm thời gian xử lý.
        - Ví dụ: Phân tích nhật ký web để phát hiện xu hướng người dùng trong vài giây.
    - **Ưu điểm**: Nhanh, dễ dùng, tích hợp với các công cụ BI như Looker, Tableau.
2. **Cloud Dataflow**:
    - **Chức năng**: Xử lý luồng (stream) và hàng loạt (batch) dữ liệu thời gian thực.
    - **Hoạt động**:
        - Dựa trên mô hình Apache Beam, lập trình pipeline để xử lý dữ liệu từ các nguồn như Pub/Sub, Kafka.
        - Tự động phân phối tác vụ trên nhiều máy chủ, tối ưu tài nguyên.
        - Ví dụ: Xử lý dữ liệu cảm biến IoT để giám sát thiết bị theo thời gian thực.
    - **Ưu điểm**: Linh hoạt, hỗ trợ cả dữ liệu thời gian thực và lịch sử.
3. **Cloud Dataproc**:
    - **Chức năng**: Quản lý và chạy các cụm Hadoop, Spark trên đám mây.
    - **Hoạt động**:
        - Triển khai nhanh các cụm Spark/Hadoop, tích hợp với BigQuery, Cloud Storage.
        - Xử lý dữ liệu lớn bằng Spark cho học máy hoặc MapReduce cho phân tích hàng loạt.
        - Ví dụ: Phân tích dữ liệu bán hàng để dự đoán doanh thu.
    - **Ưu điểm**: Dễ mở rộng, giảm chi phí quản lý cụm.
4. **Cloud Pub/Sub**:
    - **Chức năng**: Hệ thống nhắn tin thời gian thực để thu thập dữ liệu.
    - **Hoạt động**:

        - Nhận dữ liệu từ các nguồn như ứng dụng, thiết bị IoT, chuyển đến Dataflow hoặc BigQuery.
        - Đảm bảo truyền dữ liệu đáng tin cậy với độ trễ thấp.
        - Ví dụ: Thu thập dữ liệu từ ứng dụng di động để phân tích hành vi người dùng.
    - **Ưu điểm**: Tích hợp tốt với các dịch vụ GCP khác.
5. **AI Platform và AutoML**:
    - **Chức năng**: Áp dụng học máy để phân tích dữ liệu phức tạp.
    - **Hoạt động**:
        - Sử dụng dữ liệu từ BigQuery để huấn luyện mô hình dự đoán (như TensorFlow).
        - AutoML tự động tạo mô hình cho người dùng không chuyên.
        - Ví dụ: Dự đoán tỷ lệ rời bỏ khách hàng dựa trên lịch sử giao dịch.
    - **Ưu điểm**: Tăng tốc triển khai AI, giảm độ phức tạp.

Cách các công cụ phối hợp trên GCP

- **Quy trình điển hình**:
    1. **Thu thập**: Cloud Pub/Sub nhận dữ liệu từ nguồn (IoT, ứng dụng).
    2. **Lưu trữ**: Cloud Storage hoặc BigQuery lưu dữ liệu thô hoặc đã xử lý.
    3. **Xử lý**: Dataflow hoặc Dataproc làm sạch, chuyển đổi dữ liệu.
    4. **Phân tích**: BigQuery chạy truy vấn hoặc AI Platform xây dựng mô hình dự đoán.
    5. **Trực quan hóa**: Kết nối với Looker, Data Studio để hiển thị kết quả.
- **Lợi ích**:
    - Serverless, tự động mở rộng, giảm chi phí vận hành.
    - Tích hợp chặt chẽ, hỗ trợ toàn bộ chuỗi xử lý dữ liệu.
    - Bảo mật cao với IAM, mã hóa dữ liệu.

Kết luận

Dữ liệu Lớn với các đặc tính 5V đòi hỏi công cụ mạnh mẽ để xử lý. Trên GCP, BigQuery, Dataflow, Dataproc, Pub/Sub và AI Platform tạo thành hệ sinh thái linh hoạt, giúp lưu trữ, xử lý và phân tích dữ liệu lớn hiệu quả, tận dụng lợi thế đám mây để tối ưu chi phí và hiệu suất.

Câu 4: Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Mô tả hoạt động của công cụ phân tích Dữ liệu Lớn phổ biến trong môi trường điện toán đám mây Microsoft Azure

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu khổng lồ, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của công cụ truyền thống, bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ các nguồn như mạng xã hội, IoT, giao dịch.

Thuộc tính (5V):

1. **Volume**: Khối lượng lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý cao.
3. **Variety**: Đa dạng nguồn, định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều.
5. **Value**: Giá trị từ khai thác dữ liệu.

Hoạt động của công cụ phân tích Dữ liệu Lớn trên Microsoft Azure

Microsoft Azure cung cấp hệ sinh thái mạnh mẽ cho phân tích Dữ liệu Lớn. Các công cụ phổ biến bao gồm:

1. **Azure Synapse Analytics**:
    - **Chức năng**: Kho dữ liệu và phân tích Big Data serverless.
    - **Hoạt động**: Kết hợp SQL và Apache Spark để truy vấn dữ liệu lớn (có cấu trúc, phi cấu trúc). Tự động mở rộng tài nguyên, tích hợp Power BI để trực quan hóa.
    - **Ví dụ**: Phân tích dữ liệu bán hàng để dự báo doanh thu.
2. **Azure Data Lake Storage**:
    - **Chức năng**: Lưu trữ dữ liệu lớn, mở rộng linh hoạt.
    - **Hoạt động**: Lưu dữ liệu thô trong hệ thống tệp phân tán, tích hợp với Databricks, Synapse. Hỗ trợ bảo mật cấp cao.
    - **Ví dụ**: Lưu dữ liệu IoT để phân tích cảm biến.
3. **Azure Databricks**:
    - **Chức năng**: Phân tích dựa trên Spark, hỗ trợ học máy.
    - **Hoạt động**: Xử lý song song dữ liệu lớn bằng Python, Scala, SQL. Tích hợp với Data Lake, Synapse để xây dựng mô hình AI.
    - **Ví dụ**: Dự đoán nhu cầu khách hàng từ lịch sử mua sắm.
4. **Azure Stream Analytics**:
    - **Chức năng**: Xử lý luồng dữ liệu thời gian thực.
    - **Hoạt động**: Nhận dữ liệu từ Event Hubs, phân tích bằng truy vấn SQL-like, xuất kết quả đến Synapse hoặc Power BI.
    - **Ví dụ**: Giám sát giao thông qua dữ liệu GPS.
5. **Azure Machine Learning**:
    - **Chức năng**: Xây dựng mô hình học máy.
    - **Hoạt động**: Huấn luyện mô hình từ dữ liệu Synapse/Data Lake, triển khai API dự đoán. Hỗ trợ AutoML cho người không chuyên.
    - **Ví dụ**: Dự đoán hủy đơn hàng từ dữ liệu giao dịch.

Quy trình phối hợp

- **Thu thập**: Event Hubs nhận dữ liệu từ nguồn.
- **Lưu trữ**: Data Lake lưu dữ liệu thô.
- **Xử lý**: Databricks, Stream Analytics làm sạch dữ liệu.
- **Phân tích**: Synapse Analytics truy vấn, Azure ML dự đoán.
- **Trực quan**: Power BI hiển thị kết quả.

Lợi ích: Serverless, tích hợp tốt, bảo mật cao, tiết kiệm chi phí.

Kết luận

Dữ liệu Lớn với 5V cần công cụ mạnh mẽ. Azure cung cấp Synapse Analytics, Data Lake, Databricks, Stream Analytics, và Machine Learning để xử lý, phân tích dữ liệu lớn hiệu quả, tận dụng đám mây để tối ưu hiệu suất và chi phí.

Câu 5: Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Mô tả hoạt động của công cụ phân tích Dữ liệu Lớn trong môi trường điện toán đám mây Amazon Web Servers.

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu khổng lồ, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của các công cụ truyền thống. Nó bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ các nguồn như mạng xã hội, IoT, giao dịch trực tuyến.

Thuộc tính (5V):

1. **Volume**: Khối lượng lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý cao, gần thời gian thực.
3. **Variety**: Đa dạng nguồn và định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều, cần làm sạch.
5. **Value**: Giá trị khai thác để hỗ trợ quyết định.

Hoạt động của công cụ phân tích Dữ liệu Lớn trên Amazon Web Services (AWS)

Amazon Web Services (AWS) cung cấp một hệ sinh thái mạnh mẽ để phân tích Dữ liệu Lớn, tận dụng điện toán đám mây để xử lý dữ liệu hiệu quả. Các công cụ phổ biến bao gồm:

1. **Amazon S3 (Simple Storage Service)**:
    - **Chức năng**: Lưu trữ dữ liệu lớn, bền bỉ và linh hoạt.

    - **Hoạt động**: Lưu trữ dữ liệu thô (có cấu trúc, phi cấu trúc) trong các "bucket" với dung lượng không giới hạn. Tích hợp với các dịch vụ phân tích như Redshift, Athena. Hỗ trợ phân loại dữ liệu để tối ưu chi phí.
    - **Ví dụ**: Lưu trữ nhật ký web để phân tích hành vi người dùng.
    - **Ưu điểm**: Độ bền cao (99.999999999%), chi phí thấp, dễ tích hợp.
2. **Amazon Redshift**:
    - **Chức năng**: Kho dữ liệu (data warehouse) để phân tích dữ liệu lớn.
    - **Hoạt động**: Sử dụng kiến trúc cột (columnar) và xử lý song song để chạy truy vấn SQL trên petabyte dữ liệu. Tự động mở rộng cụm, tích hợp với S3, Glue.
    - **Ví dụ**: Phân tích dữ liệu bán hàng để tối ưu chiến lược kinh doanh.
    - **Ưu điểm**: Nhanh, hiệu quả cho dữ liệu có cấu trúc, tích hợp BI tools.
3. **Amazon Athena**:
    - **Chức năng**: Truy vấn dữ liệu serverless trên S3.
    - **Hoạt động**: Sử dụng SQL chuẩn để truy vấn dữ liệu trực tiếp trên S3 mà không cần tải lên kho dữ liệu. Dựa trên Presto, tự động tối ưu hiệu suất.
    - **Ví dụ**: Phân tích log ứng dụng để phát hiện lỗi hệ thống.
    - **Ưu điểm**: Không cần quản lý hạ tầng, trả phí theo truy vấn, dễ dùng.
4. **AWS Glue**:
    - **Chức năng**: Dịch vụ ETL (Extract, Transform, Load) cho Big Data.
    - **Hoạt động**: Tự động khám phá dữ liệu trên S3, tạo danh mục dữ liệu (Data Catalog). Chuyển đổi dữ liệu bằng script Python/Scala, tích hợp với Redshift, Athena.
    - **Ví dụ**: Chuẩn hóa dữ liệu khách hàng từ nhiều nguồn trước khi phân tích.
    - **Ưu điểm**: Tự động hóa ETL, tiết kiệm thời gian, hỗ trợ dữ liệu đa dạng.
5. **Amazon Kinesis**:
    - **Chức năng**: Xử lý dữ liệu luồng thời gian thực.
    - **Hoạt động**: Thu thập và xử lý dữ liệu từ nguồn như IoT, ứng dụng qua Kinesis Data Streams/Data Firehose. Phân tích luồng bằng Kinesis Analytics, lưu kết quả vào S3 hoặc Redshift.
    - **Ví dụ**: Giám sát giao dịch tài chính để phát hiện gian lận tức thời.
    - **Ưu điểm**: Độ trễ thấp, xử lý thời gian thực, tích hợp tốt.
6. **Amazon SageMaker**:
    - **Chức năng**: Xây dựng và triển khai mô hình học máy.
    - **Hoạt động**: Sử dụng dữ liệu từ S3, Redshift để huấn luyện mô hình (TensorFlow, PyTorch). Hỗ trợ AutoML và triển khai API dự đoán thời gian thực.
    - **Ví dụ**: Dự đoán nhu cầu sản phẩm dựa trên lịch sử bán hàng.
    - **Ưu điểm**: Dễ dùng, hỗ trợ AI, tích hợp toàn diện.

Quy trình phối hợp trên AWS

- **Thu thập**: Kinesis nhận dữ liệu từ nguồn (IoT, ứng dụng).

- **Lưu trữ**: S3 lưu dữ liệu thô, Redshift lưu dữ liệu đã xử lý.
- **Xử lý**: Glue làm sạch, chuyển đổi dữ liệu; Athena truy vấn nhanh.
- **Phân tích**: Redshift chạy phân tích sâu, SageMaker xây dựng mô hình AI.
- **Trực quan hóa**: Kết nối với Amazon QuickSight hoặc Tableau để hiển thị kết quả.
- **Lợi ích**:
    - Serverless, tự động mở rộng, giảm chi phí vận hành.
    - Tích hợp chặt chẽ, hỗ trợ toàn bộ quy trình Big Data.
    - Bảo mật cao với IAM, mã hóa dữ liệu.

Kết luận

Dữ liệu Lớn với đặc tính 5V đòi hỏi công cụ mạnh để xử lý. AWS cung cấp S3, Redshift, Athena, Glue, Kinesis và SageMaker, tạo hệ sinh thái toàn diện cho lưu trữ, xử lý và phân tích dữ liệu lớn, tận dụng đám mây để tối ưu hiệu suất, chi phí và khả năng mở rộng.

Câu 6: Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Mô tả hoạt động của công cụ phân tích Dữ liệu Lớn trong môi trường Hadoop DevOps trong mội trường điện toán đám mây riêng, kết nối MS Visual Studio Code bằng SSH với máy chủ của Khoa CNTT.

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu khổng lồ, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của các công cụ truyền thống, bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ các nguồn như mạng xã hội, IoT, giao dịch trực tuyến.

Thuộc tính (5V):

1. **Volume**: Khối lượng lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý cao, gần thời gian thực.
3. **Variety**: Đa dạng nguồn và định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều, cần làm sạch.
5. **Value**: Giá trị khai thác để hỗ trợ quyết định.

Hoạt động của công cụ phân tích Dữ liệu Lớn trong môi trường Hadoop DevOps trên đám mây riêng

Trong môi trường **Hadoop DevOps** trên **đám mây riêng** (private cloud), các công cụ phân tích Dữ liệu Lớn được triển khai để xử lý dữ liệu hiệu quả, với quy trình tự động hóa và tích hợp liên tục (CI/CD). Dưới đây là mô tả hoạt động của các công cụ phổ biến, kết hợp với việc kết nối **MS Visual Studio Code (VS Code)** qua **SSH** đến máy chủ của Khoa CNTT:

1. Các công cụ phân tích Dữ liệu Lớn trong Hadoop

- **Hadoop Distributed File System (HDFS)**:
  - **Chức năng**: Lưu trữ dữ liệu lớn trên nhiều máy chủ (nodes) trong đám mây riêng.
  - **Hoạt động**: Dữ liệu được chia thành các khối (blocks), phân phối và sao chép trên các DataNodes để đảm bảo chịu lỗi và truy cập nhanh. NameNode quản lý siêu dữ liệu (metadata).

  - **Trong DevOps**: Tích hợp với công cụ như Ansible hoặc Terraform để tự động hóa triển khai và quản lý cụm HDFS.
  - **Ví dụ**: Lưu trữ dữ liệu giao dịch ngân hàng để phân tích rủi ro.
- **MapReduce**:
  - **Chức năng**: Xử lý song song dữ liệu lớn.
  - **Hoạt động**: Chia tác vụ thành các giai đoạn Map (phân tích dữ liệu) và Reduce (tổng hợp kết quả), chạy trên các nodes gần dữ liệu để tối ưu hiệu suất (data locality).

  - **Trong DevOps**: Sử dụng Jenkins để tự động hóa pipeline chạy job MapReduce, kiểm tra và triển khai mã.
  - **Ví dụ**: Phân tích log website để tối ưu trải nghiệm người dùng.
- **Apache Spark**:
  - **Chức năng**: Xử lý nhanh dữ liệu lớn, hỗ trợ thời gian thực và học máy.
  - **Hoạt động**: Sử dụng bộ nhớ trong (in-memory) để xử lý dữ liệu nhanh hơn MapReduce. Tích hợp với HDFS, hỗ trợ Spark SQL, MLlib.

    - **Trong DevOps**: Triển khai Spark trên cụm Hadoop qua Kubernetes, tự động mở rộng tài nguyên bằng Helm charts.
    - **Ví dụ**: Dự đoán xu hướng tiêu dùng từ dữ liệu bán hàng.
- **Apache Hive**:
    - **Chức năng**: Kho dữ liệu để truy vấn dữ liệu lớn.
    - **Hoạt động**: Sử dụng HiveQL (giống SQL) để truy vấn dữ liệu trên HDFS, phù hợp với dữ liệu có cấu trúc.

    - **Trong DevOps**: Tự động hóa quy trình ETL (Extract, Transform, Load) bằng Airflow, tích hợp với Hive.
    - **Ví dụ**: Phân tích dữ liệu sinh viên để đánh giá hiệu quả học tập.
- **Apache Sqoop và Flume**:
    - **Chức năng**: Thu thập và nhập dữ liệu.
    - **Hoạt động**: Sqoop nhập dữ liệu từ cơ sở dữ liệu quan hệ vào HDFS; Flume thu thập dữ liệu luồng (log, sensor).

    - **Trong DevOps**: Tự động hóa pipeline nhập dữ liệu bằng script CI/CD.
    - **Ví dụ**: Nhập dữ liệu từ MySQL của Khoa CNTT vào HDFS.

2. Triển khai trong đám mây riêng

- **Môi trường đám mây riêng**: Đám mây riêng (private cloud) được triển khai trên cơ sở hạ tầng nội bộ của Khoa CNTT, sử dụng các công cụ như OpenStack hoặc VMware để tạo cụm Hadoop. Điều này đảm bảo kiểm soát bảo mật và quyền riêng tư dữ liệu.
- **DevOps trong Hadoop**:
    - **Tự động hóa**: Sử dụng Ansible, Terraform để triển khai và cấu hình cụm Hadoop, đảm bảo tính nhất quán.
    - **CI/CD**: Jenkins hoặc GitLab CI/CD tự động hóa việc xây dựng, kiểm tra và triển khai mã MapReduce, Spark.
    - **Giám sát**: Prometheus và Grafana giám sát hiệu suất cụm, cảnh báo khi tài nguyên vượt ngưỡng.
    - **Containerization**: Docker và Kubernetes quản lý các container chạy Spark, Hive, đảm bảo mở rộng linh hoạt.

3. Kết nối MS Visual Studio Code qua SSH đến máy chủ Khoa CNTT

- **Mục đích**: Sử dụng VS Code để viết, gỡ lỗi và quản lý mã phân tích Dữ liệu Lớn trên máy chủ Hadoop từ xa.
- **Quy trình kết nối**:
    1. **Cấu hình SSH trên máy chủ Khoa CNTT**:
        - Máy chủ phải cài OpenSSH Server (sudo apt install openssh-server).
        - Đảm bảo người dùng có quyền truy cập SSH (thêm vào file /etc/ssh/sshd_config nếu cần).
        - Tạo cặp khóa SSH trên máy local: ssh-keygen -t rsa.
        - Sao chép khóa công khai vào máy chủ: ssh-copy-id user@server_ip.
    2. **Cài đặt VS Code và tiện ích mở rộng**:
        - Cài VS Code trên máy local.
        - Cài tiện ích **Remote - SSH** từ Marketplace.
    3. **Kết nối VS Code qua SSH**:
        - Mở VS Code, nhấn Ctrl+Shift+P, chọn **Remote-SSH: Connect to Host**.
        - Nhập user@server_ip (ví dụ: student@192.168.1.100).
        - Chọn tệp cấu hình SSH (thường là ~/.ssh/config) hoặc nhập thông tin thủ công.
        - Sau khi kết nối, VS Code mở giao diện làm việc trên máy chủ từ xa.
    4. **Làm việc với Hadoop**:
        - Cài các extension như **Python**, **Java**, **Hadoop Snippets** trên VS Code để viết mã MapReduce, Spark hoặc HiveQL.
        - Truy cập HDFS, chạy job Spark hoặc truy vấn Hive trực tiếp từ terminal trong VS Code.
        - Ví dụ: Chạy lệnh spark-submit my_script.py để xử lý dữ liệu trên cụm Hadoop.
    5. **Quản lý mã**:
        - Sử dụng Git trong VS Code để đẩy mã lên repository (GitLab của Khoa CNTT).
        - Tích hợp với Jenkins để tự động chạy job khi mã được cập nhật.
- **Lợi ích kết nối VS Code qua SSH**:
    - Làm việc từ xa mà không cần cài đặt Hadoop cục bộ.
    - Gỡ lỗi mã trực tiếp trên môi trường sản xuất.
    - Tích hợp DevOps pipeline (Git, CI/CD) trong một giao diện duy nhất.

4. Quy trình tổng thể trong Hadoop DevOps

- **Thu thập dữ liệu**: Sqoop/Flume nhập dữ liệu từ cơ sở dữ liệu hoặc nguồn luồng vào HDFS.
- **Lưu trữ**: HDFS lưu dữ liệu phân tán trên cụm đám mây riêng.
- **Xử lý**: Spark/MapReduce làm sạch, chuyển đổi dữ liệu.

- **Phân tích**: Hive hoặc Spark SQL truy vấn dữ liệu, SageMaker (nếu tích hợp AWS) xây dựng mô hình AI.
- **Triển khai DevOps**: Tự động hóa bằng Jenkins, Ansible; giám sát bằng Prometheus.
- **Kết nối từ xa**: VS Code qua SSH cho phép lập trình viên truy cập cụm, viết mã, chạy job.

Kết luận

Dữ liệu Lớn với đặc tính 5V đòi hỏi công cụ mạnh như HDFS, MapReduce, Spark, Hive trong môi trường Hadoop DevOps trên đám mây riêng để xử lý hiệu quả. Kết nối VS Code qua SSH đến máy chủ Khoa CNTT cho phép lập trình viên làm việc từ xa, tích hợp CI/CD và quản lý mã linh hoạt, tối ưu hóa quy trình phân tích Dữ liệu Lớn trong môi trường an toàn và kiểm soát.

Câu 7: Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Vai trò của ngôn ngữ lập trình Python, các thư viện Python hỗ trợ phân tích dữ liệu lớn là gì ? Cho thí dụ thực tế mô tả hoạt động phân tích dữ liệu lớn bằng lập trình Python.

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu khổng lồ, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của các công cụ truyền thống. Nó bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ các nguồn như mạng xã hội, IoT, giao dịch trực tuyến.

Thuộc tính (5V):

1. **Volume**: Khối lượng lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý cao, gần thời gian thực.
3. **Variety**: Đa dạng nguồn và định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều, cần làm sạch.
5. **Value**: Giá trị khai thác để hỗ trợ quyết định.

Vai trò của Python và các thư viện hỗ trợ phân tích Dữ liệu Lớn

Vai trò của Python:

- **Dễ sử dụng**: Cú pháp đơn giản, dễ học, phù hợp cho cả người mới và chuyên gia.
- **Hệ sinh thái mạnh mẽ**: Có nhiều thư viện chuyên biệt cho xử lý, phân tích và học máy trên Dữ liệu Lớn.
- **Tích hợp linh hoạt**: Kết nối với các nền tảng Big Data như Hadoop, Spark, AWS, Azure, GCP.

- **Hỗ trợ đa nền tảng**: Chạy trên local, cloud hoặc cụm phân tán.
- **Cộng đồng lớn**: Hỗ trợ tài liệu, mã nguồn mở phong phú.

Các thư viện Python phổ biến hỗ trợ phân tích Dữ liệu Lớn:

1. **Pandas**:
    - **Chức năng**: Xử lý và phân tích dữ liệu dạng bảng (DataFrame).
    - **Ứng dụng**: Làm sạch, chuyển đổi, tổng hợp dữ liệu lớn (phù hợp với dữ liệu vừa phải, hoặc kết hợp với Dask cho dữ liệu lớn hơn).
2. **NumPy**:
    - **Chức năng**: Tính toán số học trên mảng đa chiều, tối ưu hiệu suất.
    - **Ứng dụng**: Xử lý dữ liệu số, hỗ trợ các thư viện khác như Pandas, SciPy.
3. **Dask**:
    - **Chức năng**: Xử lý song song dữ liệu lớn vượt quá bộ nhớ RAM.
    - **Ứng dụng**: Mở rộng Pandas và NumPy để xử lý dữ liệu lớn trên nhiều CPU hoặc cụm.
4. **PySpark**:
    - **Chức năng**: Giao diện Python cho Apache Spark, xử lý dữ liệu phân tán.
    - **Ứng dụng**: Phân tích dữ liệu lớn trên cụm Hadoop/Spark, hỗ trợ thời gian thực và học máy.
5. **TensorFlow** và **PyTorch**:
    - **Chức năng**: Xây dựng mô hình học máy và học sâu.
    - **Ứng dụng**: Phân tích dự đoán, xử lý dữ liệu phi cấu trúc (hình ảnh, văn bản).
6. **Scikit-learn**:
    - **Chức năng**: Cung cấp thuật toán học máy cơ bản.
    - **Ứng dụng**: Phân loại, hồi quy, cụm hóa trên dữ liệu lớn (kết hợp với Dask hoặc Spark).
7. **Boto3**:
    - **Chức năng**: Tương tác với AWS (S3, Redshift, Athena).
    - **Ứng dụng**: Truy xuất và xử lý dữ liệu lớn trên đám mây AWS.

Thí dụ thực tế: Phân tích dữ liệu lớn bằng Python

Ứng dụng: Phân tích dữ liệu giao dịch bán hàng của một siêu thị để dự đoán doanh thu.

Mô tả hoạt động:

- **Dữ liệu**: Bộ dữ liệu giao dịch (10 triệu bản ghi) lưu trên Amazon S3, bao gồm thông tin về sản phẩm, thời gian, giá bán, khách hàng.
- **Công cụ**: Python với PySpark và Pandas, chạy trên AWS EMR (Elastic MapReduce).

  Kết nối và tải dữ liệu:

```
from pyspark.sql import SparkSession
import boto3

# Khởi tạo Spark
spark = SparkSession.builder.appName("SalesAnalysis").getOrCreate()

# Tải dữ liệu từ S3
s3_path = "s3://bucket-name/sales_data.csv"
df = spark.read.csv(s3_path, header=True, inferSchema=True)
```

Làm sạch dữ liệu:

```
# Loại bỏ giá trị null và trùng lặp
df_clean = df.dropna().dropDuplicates()

# Chuyển đổi định dạng ngày
from pyspark.sql.functions import to_date
df_clean = df_clean.withColumn("date", to_date("date", "yyyy-MM-dd"))
```

Phân tích dữ liệu:

```
# Tổng hợp doanh thu theo sản phẩm
revenue_by_product = df_clean.groupBy("product_id").sum("price").alias("total_revenue")

# Chuyển một phần dữ liệu sang Pandas để trực quan hóa
import pandas as pd
revenue_pd = revenue_by_product.limit(1000).toPandas()
```

Dự đoán doanh thu:

```
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler

# Chuẩn bị dữ liệu cho học máy
assembler = VectorAssembler(inputCols=["quantity", "month"], outputCol="features")
```

```
data = assembler.transform(df_clean.select("quantity", "month", "price"))

# Huấn luyện mô hình
lr = LinearRegression(featuresCol="features", labelCol="price")
model = lr.fit(data)

# Dự đoán
predictions = model.transform(data)
```

Trực quan hóa:

```
import matplotlib.pyplot as plt

# Vẽ biểu đồ doanh thu
revenue_pd.plot(kind="bar", x="product_id", y="sum(price)")
plt.title("Doanh thu theo sản phẩm")
plt.show()
```

- **Kết quả**:
  - Phát hiện các sản phẩm có doanh thu cao nhất.
  - Dự đoán doanh thu tương lai dựa trên số lượng bán và thời gian.
  - Hỗ trợ siêu thị tối ưu kho hàng và chiến lược tiếp thị.
- **Ưu điểm của Python**:
  - PySpark xử lý dữ liệu lớn hiệu quả trên cụm phân tán.
  - Pandas và Matplotlib hỗ trợ phân tích và trực quan hóa cục bộ.
  - Tích hợp dễ dàng với AWS qua Boto3.

Kết luận

Dữ liệu Lớn với đặc tính 5V đòi hỏi công cụ mạnh để xử lý. Python, với các thư viện như PySpark, Pandas, Dask, và TensorFlow, đóng vai trò quan trọng trong việc xử lý, phân tích và dự đoán trên dữ liệu lớn. Thí dụ thực tế về phân tích dữ liệu bán hàng cho thấy Python giúp đơn giản hóa quy trình, từ làm sạch đến học máy, mang lại giá trị thực tiễn cho doanh nghiệp.

Câu 8 : Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Hãy trình bày mô hình và giải thích cơ chế hoạt động của Hadoop Map/ Reduce.

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu khổng lồ, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của các công cụ truyền thống. Nó bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ các nguồn như mạng xã hội, IoT, giao dịch trực tuyến.

Thuộc tính (5V):

1. **Volume**: Khối lượng lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý cao, gần thời gian thực.
3. **Variety**: Đa dạng nguồn và định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều, cần làm sạch.
5. **Value**: Giá trị khai thác để hỗ trợ quyết định.

Mô hình và cơ chế hoạt động của Hadoop MapReduce

Mô hình MapReduce: MapReduce là một mô hình lập trình và xử lý dữ liệu phân tán trong Hadoop, được thiết kế để xử lý khối lượng dữ liệu lớn trên nhiều máy chủ (nodes). Nó chia nhỏ công việc thành các tác vụ nhỏ, chạy song song để tối ưu hiệu suất.

Cơ chế hoạt động: MapReduce hoạt động theo hai giai đoạn chính: **Map** và **Reduce**, với các bước chi tiết như sau:

1. **Input Data**:
    - Dữ liệu đầu vào được lưu trữ trên **HDFS** (Hadoop Distributed File System), chia thành các khối (blocks, thường 128MB).
    - Mỗi khối được xử lý độc lập trên các DataNode.
2. **Map Phase**:
    - **Chức năng**: Phân tích và chuyển đổi dữ liệu đầu vào thành các cặp khóa-giá trị (key-value pairs).
    - **Hoạt động**:
        - Một hàm **Mapper** được viết bởi người dùng (thường bằng Java, Python, hoặc ngôn ngữ khác) xử lý từng bản ghi trong khối dữ liệu.
        - Mỗi Mapper tạo ra các cặp key-value trung gian.
        - Ví dụ: Trong phân tích log web, Mapper có thể đếm số lần truy cập của mỗi URL, tạo cặp <URL, 1>.
    - **Song song**: Nhiều Mapper chạy đồng thời trên các DataNode, tận dụng tính phân tán.
3. **Shuffle and Sort**:
    - **Chức năng**: Sắp xếp và phân phối dữ liệu từ Map đến Reduce.
    - **Hoạt động**:

        - Hadoop tự động thu thập tất cả cặp key-value từ các Mapper.
        - Sắp xếp theo khóa (key) và nhóm các giá trị có cùng khóa.
        - Chuyển các nhóm key-value đến các Reducer.
        - Ví dụ: Các cặp <URL, 1> được nhóm thành <URL, [1, 1, 1,...]>.
    - **Tối ưu**: Dữ liệu được truyền qua mạng tối thiểu nhờ **data locality** (xử lý gần nơi lưu trữ).
4. **Reduce Phase**:
    - **Chức năng**: Tổng hợp dữ liệu từ giai đoạn Map để tạo kết quả cuối cùng.
    - **Hoạt động**:
        - Một hàm **Reducer** được viết bởi người dùng xử lý các nhóm key-value.
        - Reducer tổng hợp giá trị theo khóa để tạo đầu ra cuối cùng (cũng là cặp key-value).
        - Ví dụ: Reducer tính tổng số lần truy cập cho mỗi URL, tạo cặp <URL, total_count>.
    - **Song song**: Nhiều Reducer chạy đồng thời trên các nodes.
5. **Output Data**:
    - Kết quả từ Reducer được ghi lại vào HDFS dưới dạng tệp.
    - Có thể được sử dụng cho phân tích tiếp theo hoặc xuất ra các hệ thống khác.
    - Ví dụ: Tệp chứa danh sách <URL, total_count> được lưu trên HDFS.

Các thành phần quản lý trong MapReduce:

- **JobTracker** (trong Hadoop 1.x) hoặc **ResourceManager** (Hadoop YARN):
    - Quản lý và phân phối công việc đến các nodes.
    - Theo dõi tiến độ job, xử lý lỗi nếu node thất bại.
- **TaskTracker** (Hadoop 1.x) hoặc **NodeManager** (YARN):
    - Thực thi các tác vụ Map và Reduce trên từng node.
    - Báo cáo trạng thái về JobTracker/ResourceManager.

Ví dụ minh họa (Đếm từ - Word Count):

- **Input**: Tệp văn bản lớn chứa nhiều dòng, lưu trên HDFS.
- **Map**:
    - Mỗi Mapper đọc một dòng, tách thành các từ, tạo cặp <word, 1>.
    - Ví dụ: Dòng “hello world” tạo ra <hello, 1>, <world, 1>.
- **Shuffle and Sort**: Nhóm các cặp theo từ, ví dụ: <hello, [1, 1, 1]>, <world, [1, 1]>.
- **Reduce**: Tính tổng giá trị cho mỗi từ, tạo kết quả <hello, 3>, <world, 2>.
- **Output**: Tệp trên HDFS chứa số lần xuất hiện của mỗi từ.

Ưu điểm của MapReduce:

- **Khả năng mở rộng**: Xử lý dữ liệu lớn trên hàng nghìn nodes.

- **Chịu lỗi**: Tự động chạy lại tác vụ nếu node thất bại.
- **Tối ưu hiệu suất**: Xử lý gần dữ liệu (data locality), giảm truyền dữ liệu qua mạng.

Hạn chế:

- **Độ trễ cao**: Phù hợp với xử lý hàng loạt (batch), không tối ưu cho thời gian thực.
- **Phức tạp**: Yêu cầu viết mã Map và Reduce thủ công.
- **Bị thay thế dần**: Các công cụ như Apache Spark nhanh hơn nhờ xử lý trong bộ nhớ.

Kết luận

Dữ liệu Lớn với đặc tính 5V đòi hỏi công cụ mạnh để xử lý khối lượng, tốc độ và đa dạng dữ liệu. Hadoop MapReduce cung cấp mô hình phân tán hiệu quả, chia tác vụ thành Map và Reduce để xử lý song song trên HDFS. Dù có hạn chế về tốc độ và độ phức tạp, MapReduce vẫn là nền tảng quan trọng cho phân tích Dữ liệu Lớn, đặc biệt trong các hệ thống cần độ bền và khả năng mở rộng cao.

Câu 9 : Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Mô tả kiến trúc của hệ thống tập tin phân tán HDFS và vai trò của HDFS trong phân tích Dữ liệu Lớn. Để quản trị HDFS, chúng ta có thể sử dụng công cụ nào ? Hãy liệt kê một số lệnh cơ bản để quản trị HDFS.

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu khổng lồ, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của các công cụ truyền thống. Nó bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ các nguồn như mạng xã hội, IoT, giao dịch trực tuyến.

Thuộc tính (5V):

1. **Volume**: Khối lượng lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý cao, gần thời gian thực.
3. **Variety**: Đa dạng nguồn và định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều, cần làm sạch.
5. **Value**: Giá trị khai thác để hỗ trợ quyết định.

Kiến trúc của hệ thống tập tin phân tán HDFS

HDFS (Hadoop Distributed File System) là hệ thống tập tin phân tán được thiết kế để lưu trữ và quản lý dữ liệu lớn trên nhiều máy chủ, tối ưu cho xử lý khối lượng dữ liệu khổng lồ với độ tin cậy cao.

Kiến trúc HDFS:

1. **NameNode**:
    - Vai trò: "Bộ não" của HDFS, quản lý siêu dữ liệu (metadata) như cấu trúc thư mục, vị trí các khối dữ liệu (blocks), quyền truy cập.
    - Lưu trữ: Giữ tệp **fsimage** (ảnh hệ thống tệp) và **edit log** (nhật ký thay đổi).
    - Đặc điểm: Chỉ có một NameNode chính (single point of failure), nhưng có thể cấu hình **Secondary NameNode** hoặc **High Availability (HA)** để dự phòng.
    - Nhiệm vụ: Phân bổ khối dữ liệu, xử lý yêu cầu đọc/ghi từ client.
2. **DataNode**:
    - Vai trò: Lưu trữ dữ liệu thực tế dưới dạng các khối (blocks, thường 128MB hoặc 256MB).
    - Số lượng: Nhiều DataNode chạy trên các máy chủ khác nhau, đảm bảo phân tán dữ liệu.
    - Nhiệm vụ: Đọc/ghi dữ liệu theo lệnh từ NameNode, báo cáo trạng thái và danh sách khối định kỳ.
3. **Block**:
    - Dữ liệu được chia thành các khối cố định, lưu trữ trên DataNode.
    - Mỗi khối được **sao chép (replication)** (mặc định 3 bản) trên các DataNode khác nhau để đảm bảo chịu lỗi.
4. **Client**:
    - Giao tiếp với NameNode để lấy siêu dữ liệu và với DataNode để đọc/ghi dữ liệu.
    - Không lưu trữ dữ liệu mà chỉ điều phối yêu cầu.

Cơ chế hoạt động:

- **Ghi dữ liệu**:
    1. Client gửi yêu cầu ghi đến NameNode.
    2. NameNode xác định vị trí lưu trữ, trả về danh sách DataNode.
    3. Client gửi dữ liệu đến DataNode đầu tiên, dữ liệu được sao chép qua các DataNode khác (pipeline replication).
    4. DataNode xác nhận hoàn tất với NameNode.
- **Đọc dữ liệu**:
    1. Client yêu cầu NameNode để lấy vị trí khối dữ liệu.
    2. NameNode trả về danh sách DataNode chứa khối.
    3. Client đọc trực tiếp từ DataNode gần nhất (data locality).

Đặc điểm nổi bật:

- **Chịu lỗi**: Sao chép khối dữ liệu đảm bảo dữ liệu vẫn an toàn nếu một DataNode lỗi.
- **Khả năng mở rộng**: Thêm DataNode để tăng dung lượng và hiệu suất.

- **Tối ưu đọc tuần tự**: Phù hợp với xử lý dữ liệu lớn, ít sửa đổi.

Vai trò của HDFS trong phân tích Dữ liệu Lớn

1. **Lưu trữ dữ liệu lớn**:
    - HDFS lưu trữ khối lượng dữ liệu khổng lồ (Volume) trên nhiều máy chủ giá rẻ, hỗ trợ mở rộng dễ dàng.
    - Ví dụ: Lưu trữ log giao dịch ngân hàng hàng triệu bản ghi.
2. **Hỗ trợ dữ liệu đa dạng**:
    - Xử lý dữ liệu có cấu trúc (CSV), bán cấu trúc (JSON), phi cấu trúc (video, ảnh), đáp ứng thuộc tính Variety.
    - Ví dụ: Lưu trữ dữ liệu từ IoT, mạng xã hội.
3. **Xử lý phân tán**:
    - HDFS cho phép các công cụ như MapReduce, Spark, Hive truy cập dữ liệu gần nơi lưu trữ (data locality), giảm độ trễ, tăng tốc xử lý (Velocity).
    - Ví dụ: Phân tích thời gian thực dữ liệu bán hàng.
4. **Độ tin cậy cao**:
    - Sao chép dữ liệu đảm bảo tính Veracity và bảo vệ dữ liệu khi phần cứng lỗi.
    - Ví dụ: Phân tích dữ liệu y tế cần độ chính xác cao.
5. **Nền tảng cho phân tích giá trị**:
    - HDFS cung cấp dữ liệu thô cho các công cụ phân tích (Hive, Spark), giúp khai thác thông tin hữu ích (Value).
    - Ví dụ: Dự đoán xu hướng thị trường từ dữ liệu khách hàng.

Công cụ quản trị HDFS

Để quản trị HDFS, có thể sử dụng các công cụ sau:

1. **HDFS Command Line Interface (CLI)**:
    - Công cụ chính, sử dụng lệnh hdfs dfs để quản lý tệp, thư mục.
2. **Hadoop Web UI**:
    - Giao diện web (thường chạy trên cổng 50070 của NameNode) để giám sát trạng thái HDFS, dung lượng, DataNode.
3. **Hue (Hadoop User Experience)**:
    - Giao diện đồ họa để quản lý tệp, chạy truy vấn, giám sát HDFS.
4. **Ambari**:
    - Công cụ quản lý cụm Hadoop, hỗ trợ cấu hình, giám sát HDFS.
5. **Cloudera Manager**:
    - Tương tự Ambari, dùng cho các cụm Hadoop thương mại.

Một số lệnh cơ bản để quản trị HDFS

Dùng lệnh hdfs dfs trong terminal để quản lý HDFS. Dưới đây là các lệnh phổ biến:

1. **Xem nội dung thư mục**: hdfs dfs -ls /path/to/directory
2. **Tạo thư mục**: hdfs dfs -mkdir /path/to/new_directory
3. **Tải tệp lên HDFS**: hdfs dfs -put /local/path/file.txt /hdfs/path/
4. **Tải tệp từ HDFS về máy cục bộ**: hdfs dfs -get /hdfs/path/file.txt /local/path/
5. **Xem nội dung tệp**: hdfs dfs -cat /hdfs/path/file.txt
6. **Xóa tệp hoặc thư mục**: hdfs dfs -rm /hdfs/path/file.txt ,hdfs dfs -rm -r /hdfs/path/directory
7. **Kiểm tra dung lượng**: hdfs dfs -du -h /hdfs/path/
8. **Sao chép tệp trong HDFS:** hdfs dfs -cp /hdfs/source/file.txt /hdfs/destination/
9. **Kiểm tra trạng thái hệ thống:** hdfs dfsadmin -report

Kết luận

Dữ liệu Lớn với đặc tính 5V đòi hỏi hệ thống lưu trữ mạnh mẽ như HDFS, với kiến trúc NameNode và DataNode đảm bảo phân tán, chịu lỗi và mở rộng. HDFS đóng vai trò nền tảng trong phân tích Dữ liệu Lớn, cung cấp dữ liệu cho các công cụ như MapReduce, Spark. Quản trị HDFS có thể dùng CLI, Hue, Ambari, với các lệnh cơ bản như ls, put, get, rm giúp quản lý tệp hiệu quả.

Câu 10: Hãy trình bày định nghĩa và các thuộc tính cơ bản liên quan tới Dữ liệu Lớn. Các thành phần chính trong kiến trúc phân tích dữ liệu lớn Hadoop là gì ? Giải thích các khái niệm HDFS, Namenode, Datanode, Resource Manager

Định nghĩa: Dữ liệu Lớn là tập hợp dữ liệu khổng lồ, phức tạp, tăng trưởng nhanh, vượt khả năng xử lý của các công cụ truyền thống. Nó bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc từ các nguồn như mạng xã hội, IoT, giao dịch trực tuyến.

Thuộc tính (5V):

1. **Volume**: Khối lượng lớn (terabyte, petabyte).
2. **Velocity**: Tốc độ tạo và xử lý cao, gần thời gian thực.
3. **Variety**: Đa dạng nguồn và định dạng (văn bản, video, JSON...).
4. **Veracity**: Độ chính xác không đồng đều, cần làm sạch.
5. **Value**: Giá trị khai thác để hỗ trợ quyết định.

Các thành phần chính trong kiến trúc phân tích Dữ liệu Lớn Hadoop

Kiến trúc Hadoop là một hệ sinh thái mã nguồn mở được thiết kế để xử lý và phân tích Dữ liệu Lớn. Các thành phần chính bao gồm:

1. **HDFS (Hadoop Distributed File System)**:
   - Hệ thống tập tin phân tán để lưu trữ dữ liệu lớn trên nhiều máy chủ, đảm bảo chịu lỗi và khả năng mở rộng.
2. **YARN (Yet Another Resource Negotiator)**:
   - Quản lý tài nguyên và lập lịch công việc, phân bổ CPU, bộ nhớ cho các ứng dụng như MapReduce, Spark.
3. **MapReduce**:
   - Mô hình lập trình xử lý dữ liệu phân tán, chia tác vụ thành Map (phân tích) và Reduce (tổng hợp) để xử lý song song.
4. **Hadoop Common**:
   - Các thư viện và tiện ích hỗ trợ các thành phần khác của Hadoop.
5. **Công cụ hệ sinh thái**:
   - **Hive**: Kho dữ liệu, truy vấn SQL-like.
   - **Pig**: Xử lý dữ liệu bằng ngôn ngữ Pig Latin.
   - **HBase**: Cơ sở dữ liệu NoSQL phân tán.
   - **Sqoop**: Chuyển dữ liệu giữa Hadoop và cơ sở dữ liệu quan hệ.
   - **Flume**: Thu thập dữ liệu luồng.
   - **Oozie**: Quản lý luồng công việc.
   - **Spark**: Xử lý nhanh, thay thế MapReduce trong nhiều trường hợp.

Giải thích các khái niệm: HDFS, NameNode, DataNode, Resource Manager

1. **HDFS (Hadoop Distributed File System)**:
   - **Định nghĩa**: Hệ thống tập tin phân tán, lưu trữ dữ liệu lớn trên nhiều máy chủ, tối ưu cho đọc/ghi tuần tự và xử lý khối lượng dữ liệu khổng lồ.
   - **Chức năng**:
     - Chia dữ liệu thành các khối (blocks, thường 128MB hoặc 256MB), phân phối trên các DataNode.
     - Sao chép khối (mặc định 3 bản) để chịu lỗi.
     - Tận dụng **data locality** (xử lý dữ liệu gần nơi lưu trữ) để tăng hiệu suất.
   - **Vai trò trong Big Data**: Lưu trữ dữ liệu lớn, đa dạng (Volume, Variety), cung cấp nền tảng cho phân tích bằng MapReduce, Spark, Hive.
   - **Ví dụ**: Lưu trữ log giao dịch ngân hàng để phân tích gian lận.
2. **NameNode**:
   - **Định nghĩa**: Thành phần quản lý siêu dữ liệu (metadata) của HDFS, đóng vai trò "bộ não" của hệ thống.
   - **Chức năng**:
     - Lưu trữ thông tin về cấu trúc thư mục, vị trí khối dữ liệu, quyền truy cập trong tệp **fsimage** và **edit log**.

        - Phân bổ khối dữ liệu đến DataNode, xử lý yêu cầu đọc/ghi từ client.
        - Theo dõi trạng thái DataNode qua báo cáo định kỳ.
    - **Đặc điểm**:
        - Chỉ có một NameNode chính (single point of failure), nhưng có thể cấu hình **Secondary NameNode** (sao lưu siêu dữ liệu) hoặc **High Availability (HA)** với NameNode dự phòng.
    - **Ví dụ**: Khi client yêu cầu đọc tệp, NameNode cung cấp danh sách DataNode chứa khối dữ liệu.
3. **DataNode**:
    - **Định nghĩa**: Thành phần lưu trữ dữ liệu thực tế trong HDFS, chạy trên các máy chủ riêng biệt.
    - **Chức năng**:
        - Lưu trữ các khối dữ liệu (blocks) và thực hiện lệnh đọc/ghi từ client.
        - Báo cáo danh sách khối và trạng thái (sống/chết) định kỳ cho NameNode.
        - Tham gia vào quá trình sao chép khối để đảm bảo chịu lỗi.
    - **Đặc điểm**:
        - Có thể có nhiều DataNode, dễ mở rộng bằng cách thêm máy chủ.
        - Không lưu siêu dữ liệu, chỉ chứa dữ liệu thô.
    - **Ví dụ**: Một DataNode lưu trữ khối dữ liệu chứa log giao dịch và gửi dữ liệu này khi client yêu cầu.
4. **Resource Manager**:
    - **Định nghĩa**: Thành phần chính trong YARN, quản lý tài nguyên và lập lịch công việc trên cụm Hadoop.
    - **Chức năng**:
        - **Resource Manager** gồm hai module:
            - **Scheduler**: Phân bổ tài nguyên (CPU, RAM) cho các ứng dụng (MapReduce, Spark) dựa trên chính sách lập lịch (FIFO, Fair Scheduler).
            - **Application Manager**: Quản lý vòng đời ứng dụng, khởi tạo và giám sát **ApplicationMaster** cho mỗi job.
        - Nhận yêu cầu từ client, phân phối tài nguyên đến **NodeManager** trên các máy chủ.
    - **Đặc điểm**:
        - Chạy trên một máy chủ chính, có thể cấu hình HA để tránh lỗi.
        - Tách biệt quản lý tài nguyên khỏi xử lý dữ liệu, cải thiện hiệu suất so với Hadoop 1.x (JobTracker).
    - **Ví dụ**: Khi chạy job MapReduce, Resource Manager phân bổ tài nguyên cho các Mapper và Reducer trên NodeManager.

Mối quan hệ giữa các thành phần

- **HDFS (NameNode, DataNode)**: Lưu trữ và quản lý dữ liệu lớn, cung cấp dữ liệu cho các ứng dụng phân tích.
- **YARN (Resource Manager)**: Quản lý tài nguyên để chạy các ứng dụng như MapReduce, Spark trên dữ liệu HDFS.
- **Tích hợp**: NameNode và DataNode phối hợp để lưu/truy xuất dữ liệu, trong khi Resource Manager đảm bảo các job phân tích sử dụng tài nguyên hiệu quả.

Kết luận

Dữ liệu Lớn với đặc tính 5V đòi hỏi kiến trúc mạnh mẽ như Hadoop, với các thành phần chính là HDFS, YARN, MapReduce và các công cụ hệ sinh thái. HDFS (NameNode, DataNode) cung cấp lưu trữ phân tán, chịu lỗi; Resource Manager trong YARN quản lý tài nguyên và lập lịch công việc. Các thành phần này phối hợp để xử lý và phân tích Dữ liệu Lớn hiệu quả, đáp ứng yêu cầu về khối lượng, tốc độ và đa dạng dữ liệu.