

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GTVT**

**KHOA CÔNG NGHỆ THÔNG TIN**



***Học phần:* Big Data**

***Đề tài:* Đề 9**

**Giảng viên hướng dẫn: Th.S Khuất Thị Ngọc Ánh**

**Lớp: 71DCTT21**

**Thành viên nhóm gồm: Trương Bá Cát Trường**

**Nguyễn Thị Thùy Dung**

**Cấn Minh Quang**

**Trịnh Tiến Quang**

**Hà Nội, 2023**

**ĐỀ BÀI**

**Đề tài số 9.** Thực hiện các nội dung yêu cầu sau

**Câu 1.** Trình bày các thành phần của Hadoop (Khái niệm, đặc điểm)?

**Câu 2.** Trình bày về DStream Window?

**Câu 3.** Cho tập dữ liệu về danh sách 100 Tiktoker hàng đầu tại Pakistan có nội dung như bên dưới (File đính kèm), sử dụng Spark RDD hoặc Spark SQL để trả lời một số câu hỏi sau:

3.1. Tiktoker có thời gian tham gia sớm nhất?

3.2. Tiktoker có số lượng Fans nhiều nhất?

3.3. Liệt kê các năm gia nhập của các Tiktoker?

3.4. Liệt kê danh sách 3 Tiktoker hàng đầu của Pakistan theo số lượng Fan?

3.5. Liệt kê danh sách 3 Tiktoker hàng đầu của Pakistan theo Tổng số lượng Fan và số lượng View? Trong đó: số lượng Fan nhân hệ số 10

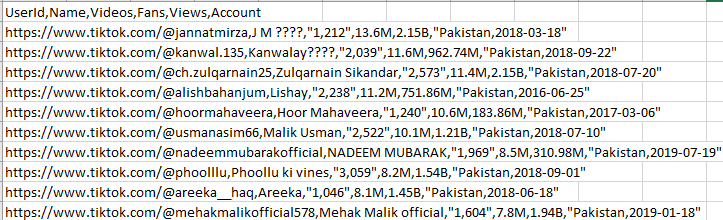
3.6. Liệt kê tỷ lệ giữa số lượng Fan và số lượng View của 3 Tiktoker có số lượng Fan thấp nhất?

3.7. Liệt kê tỷ lệ giữa số lượng Video và số lượng View của 3 Tiktoker có số lượng View cao nhất?

3.8. Tính số lượng View trung bình của tất cả các Tiktoker?

3.9. Tạo một cột mới nếu số lượng Video < 1000 điền là “Type1”, 1000 <= Số Video < 2000 điền là “Type2”, còn lại điền là “Type3”

3.10. Tạo một cột mới điền thông tin tỷ lệ giữa số lượng View và Fan, Video theo công thức Tỷ lệ = Số lượng View/(Số lượng Fan \* Số lượng Video)

****

**Câu 4.** Sử dụng số liệu thống kê các chuyến bay trong tháng 1,2,3 năm 2015 (file đính kèm) để thực hành streaming kết quả 3 hãng hàng không bay nhiều nhất. Trong đó cột Airline Identifier là mã định danh của các hãng hàng không.

**BÀI LÀM**

**Câu 1.** Trình bày các thành phần của Hadoop (Khái niệm, đặc điểm)?

1. **Khái niệm**
   1. **Hadoop**

Hadoop là một framework được viết bằng java cho phép phát triển các ứng dụng trên hệ thống phân tán.

Được thiết kế để mở rộng quy mô từ một máy chủ đơn sang nhiều máy tính khác có tính toán và lưu trữ cục bộ (local computation and storage)

Phát triển dựa trên mô hình Map-Reduce và hệ thống file phân tán GFS (Google File System - là hệ thống lưu trữ phân tán được phát triển bởi Google để lưu trữ và quản lý dữ liệu trên các máy chủ khác nhau)

Cơ chế hoạt động của GFS:

Phân chia tệp tin

Đánh dấu thời gian

Sao lưu dữ liệu

Cơ chế cache

Cơ chế đọc ghi

Nhờ cơ chế streaming hadoop có thể phát triển trên các ứng dụng phân tán bằng cả java và các ngôn ngữ lập trình khác như C++, Python, ….

Hadoop sẽ được cài đặt trên các máy trong hệ phân tán, với các máy cài Hadoop được gọi là một Hadoop cluster

* 1. **Hadoop cluster**

Gồm 2 thành phần chính: HDFS và Map-Reduce

Có kiến trúc master - slave (chủ - tớ)

Trong Hadoop cluster chỉ có 1 NameNode và một node Jobtracker. Các node này có thể nằm trên cùng một máy

1. **Đặc điểm**
   1. **Các thành phần của Hadoop**

|  |  |
| --- | --- |
| Thành phần | Giải thích |
| Core | Cung cấp công cụ và giao diện cho hệ thống |
| MapReduce | Mô hình xử lý dữ liệu phân tán |
| HDFS | Hệ thống tệp phân tán |
| Hive | Kho dữ liệu phân tán, cung cấp SQL dựa trên ngôn ngữ truy vấn |
| Hbase | Cơ sở dữ liệu dựa trên cột phân tán |
| Pig | Ngôn ngữ dòng dữ liệu và môi trường thực thi |

* 1. **Kiến trúc Hadoop**

Hadoop có một cấu trúc liên kết master-slave. Trong cấu trúc này, chúng ta có một node master và nhiều node slave. Chức năng của node master là gán một tác vụ cho các node slave khác nhau và quản lý tài nguyên. Các node slave là máy tính thực tế có thể không mạnh lắm. Các node slave lưu trữ dữ liệu thực trong khi trên master chúng ta có metadata.

Kiến trúc của Hadoop gồm có ba lớp chính đó là

* HDFS (Hadoop Distributed File System)
* Map-Reduce
* Yarn
  + 1. **HDFS**

HDFS (Hadoop Distributed File System): Là hệ thống file phân tán, cung cấp khả năng lưu trữ dữ liệu khổng lồ và tính năng tối ưu hoá việc sử dụng băng thông giữa các node. HDFS có thể được sử dụng để chạy trên một cluster lớn với hàng chục ngàn node.

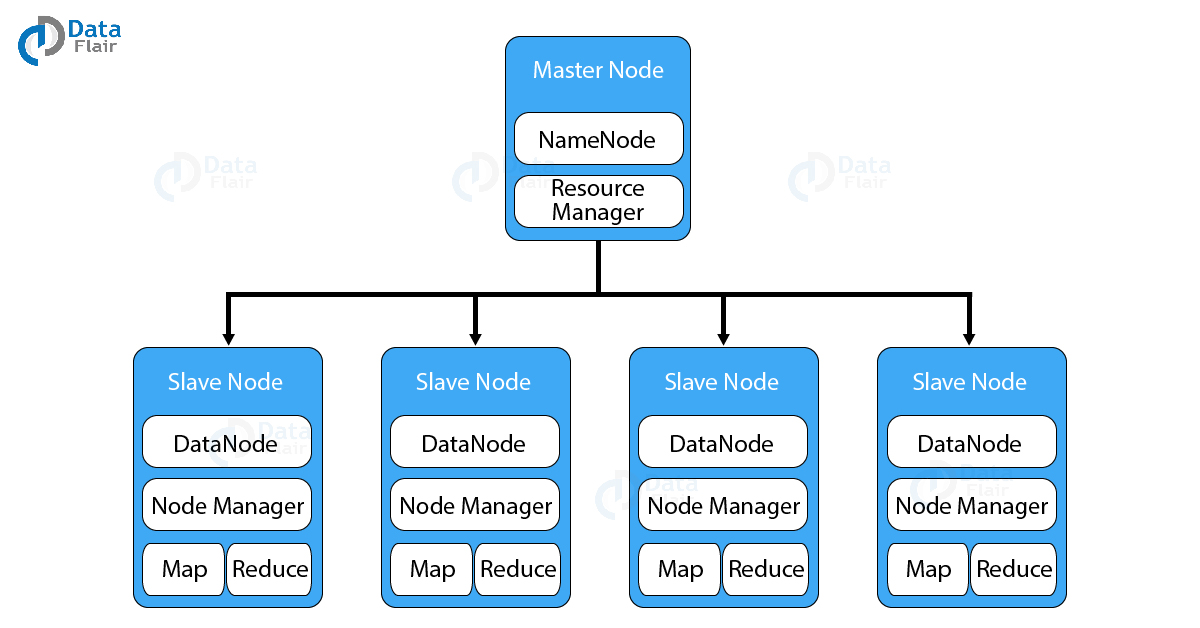
Cho phép truy xuất nhiều ổ đĩa như là 1 ổ đĩa. Nói cách khác, chúng ta có thể sử dụng một ổ đĩa mà gần như không bị giới hạn về dung lượng. Muốn tăng dung lượng chỉ cần thêm node (máy tính) vào hệ thống.

**Đặc trưng**

* Phân tán
* Tính toàn vẹn
* Hỗ trợ lưu trữ lớn
* Tốc độ truy xuất cao
* Tính linh hoạt
* Tính chính xác

**Thiết kế HDFS**

* Có kiến trúc Master-Slave
* NameNode chạy trên máy chủ Master, có tác vụ quản lý Namespace và điều chỉnh truy cập tệp của client
* DataNode chạy trên các nút Slave. có tác vụ lưu trữ business data thực tế
* Một tập tin với định dạng HDFS được chia thành nhiều block, mỗi block có một ID duy nhất và những block này được lưu trữ trong một tập các DataNodes
* Kích thước 1 block thông thường là 64MB, kích thước này có thể thay đổi được bằng việc cấu hình



*Mô hình HDFS*

**Cơ chế đọc ghi dữ liệu của HDFS**

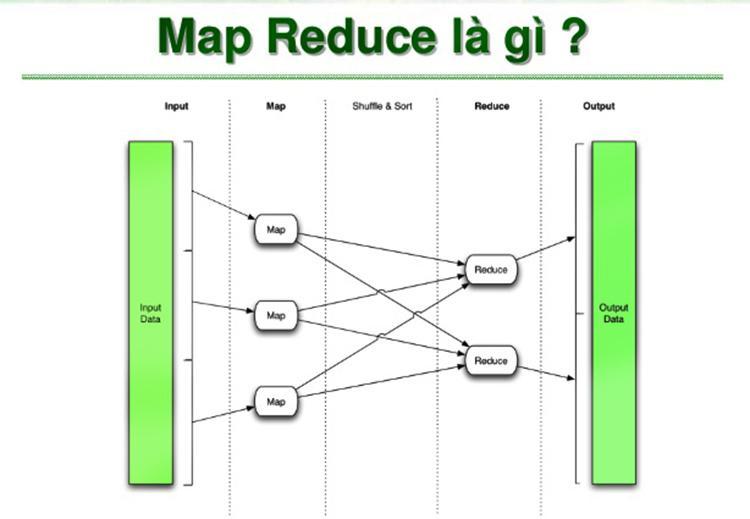
* Đọc dữ liệu từ HDFS:
  + Khi một ứng dụng hoặc người dùng muốn đọc dữ liệu từ HDFS, họ cần biết đường dẫn tới tệp dữ liệu trên HDFS.
  + HDFS sử dụng giao thức truy cập từ xa (Remote Procedure Call – RPC) để tạo kết nối giữa Client (Ứng dụng hoặc người dùng) và máy chủ NameNode (Một phần quản lý tệp và thư mục trên HDFS).
  + Client yêu cầu đọc tệp dữ liệu tới NameNode. NameNode trả về danh sách các khối dữ liệu và vị trí của chúng trên DataNodes (máy chủ lưu trữ dữ liệu thực sự).
  + Client sau đó liên hệ trực tiếp với các DataNodes để đọc từng khối dữ liệu. các khối này được đọc song song nếu có nhiều DataNodes chứa chúng
* Ghi dữ liệu vào HDFS:
  + Khi một ứng dụng hoặc người dùng muốn ghi dữ liệu vào HDFS, họ cũng cần biết đường dẫn tới vị trí lưu trữ dữ liệu trên HDFS
  + Vì HDFS chia dữ liệu thành các khối có kích thước cố định, nên khi một tệp dữ liệu mới được tạo, client gửi yêu cầu tới các NameNode để tạo một tệp mới và tạo danh sách các khối dữ liệu.
  + Client sau đó gửi các khối dữ liệu đến các DataNodes. Dữ liệu được chia thành các phần nhỏ và ghi song song và các DataNodes. Sau khi tất cả các khối được ghi thành công, NameNode cập nhật thông tin về tệp dữ liệu trong metadata của HDFS
* Lưu ý
  + HDFS được thiết kế để hỗ trợ đọc nhiều lần (Read-many) và ghi một lần (write-once). Điều này làm nó phù hợp với các ứng dụng xử lý dữ liệu lớn như xử lý lô (batch processing)
  + Dữ liệu trên HDFS thường được sao lưu để đảm bảo khả năng chịu lỗi. mỗi khối dữ liệu được sao lưu ít nhất một lần trên các DataNodes khác nhau trong cụm
  + NameNode duy trì thông tin về cấu trúc thư mục và tệp trên HDFS, trong khi DataNodes chịu trách nhiệm lưu trữ thực tế của dữ liệu
    1. **Map-Reduce**

**Định nghĩa**

* Map-Reduce là mô hình xử lý tính toán song song và phân tán trên hệ thống phân tán
* Phân rã nghiệp vụ chính thành các công việc con để thực hiện xử lý song song, sau đó thu lại kết quả

**Chức năng:** Map-Reduce thực hiện 2 chức năng chính là **Map** và **Reduce**

* **Map**: Sẽ thực hiện đầu tiên, có chức năng tải, phân tích dữ liệu đầu vào và được chuyển đổi thành tập dữ liệu theo cặp key/value
* **Reduce**: Sẽ nhận kết quả đầu ra từ tác vụ Map, kết hợp dữ liệu lại với nhau thành tập dữ liệu nhỏ hơn
* Giữa hàm Map và Reduce có một giai đoạn xử lý trung gian gọi là hàm Shuffle. Hàm này có nhiệm vụ sắp xếp các từ và tổng hợp dữ liệu đầu vào cho Reduce từ các kết quả đầu ra của hàm Map.



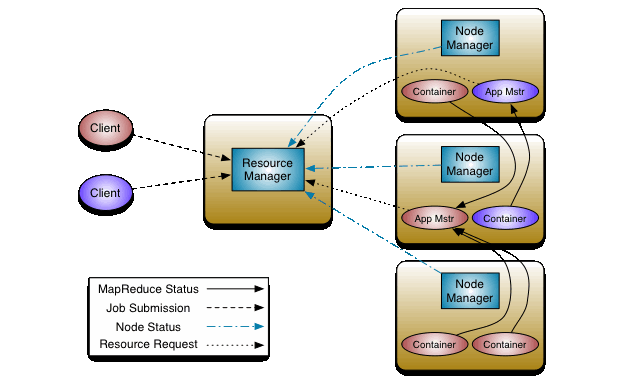
*Mô hình Map-Reduce*

* + 1. **Yarn**

YARN (Yet-Another-Resource-Negotiator) là một framework hỗ trợ phát triển ứng dụng phân tán, YARN cung cấp daemons và APIs cần thiết cho việc phát triển ứng dụng phân tán, đồng thời xử lý và lập lịch sử dụng tài nguyên tính toán (CPU hay memory) cũng như giám sát quá trình thực thi các ứng dụng đó.

Bên trong YARN, chúng ta có hai trình quản lý ResourceManager và NodeManage

* ResourceManager: Quản lý toàn bộ tài nguyên tính toán của cluster.
* NodeManger: Giám sát việc sử dụng tài nguyên của container và báo cáo với ResourceManger. Các tài nguyên ở đây là CPU, memory, disk, network, ...



Quá trình 1 ứng dụng chạy trên YARN được mô tả bằng sơ đồ trên qua các bước sau:

* Client giao 1 task cho Resource Manager
* Resource Manager tính toán tài nguyên cần thiết theo yêu cầu của ứng dụng và tạo 1 App Master (App Mstr). Application Master được chuyển đến chạy 1 một node tính toán. Application Master sẽ liên lạc với các NodeManager ở các node khác để ra yêu cầu công việc cho node này.
* Node Manager nhận yêu cầu và chạy các task trên container
* Các thông tin trạng thái thay vì được gửi đến JobTracker sẽ được gửi đến App Master.

ResourceManger có hai thành phần quan trọng đó là Scheduler và ApplicationManager

**Scheduler** có trách nhiệm phân bổ tài nguyên cho các ứng dụng khác nhau. Đây là Scheduler thuần túy vì nó không thực hiện theo dõi trạng thái cho ứng dụng. Nó cũng không sắp xếp lại các tác vụ bị lỗi do lỗi phần cứng hoặc phần mềm. Bộ lập lịch phân bổ các tài nguyên dựa trên các yêu cầu của ứng dụng

**ApplicationManager** có chức năng sau:

* Chấp nhận nộp công việc.
* Đàm phán container đầu tiên để thực thi ApplicationMaster. Một nơi chứa kết hợp các yếu tố như CPU, bộ nhớ, đĩa và mạng.
* Khởi động lại container ApplicationMaster khi không thành công.

**Câu 2.** Trình bày về DStream Window?

**Khái niệm:**

* Dstream Window là một khung thời gian trượt trên Dstream (Distributed Stream) trong Spark Streaming.
* Cho phép xử lý dữ liệu trong một khoảng thời gian xác định và tạo ra kết quả

**Một số Khái niệm trong Dstream Window:**

* ***Kích thước cửa sổ (Window Size):*** xác định khoảng thời gian muốn xử lý dữ liệu trong đó
  + *Ví dụ:* nếu thiết lập cửa sổ 5 phút, Spark Streaming sẽ tự động xử lý dữ liệu trong cửa sổ 5 phút liên tiếp
* ***Kích thước bước (Slide Interval):*** là khoảng thời gian giữa các cửa sổ liên tiếp. Nó xác định tần suất mà Dstream Window được cập nhật
  + *Ví dụ:* nếu thiết lập kích thước bước là 1 phút trong cửa sổ 5 phút. Thì cửa sổ mới sẽ được tạo mỗi 1 phút và xử lý dữ liệu trong 5 phút gần nhất
* ***Xử lý trong cửa sổ:*** Trong mỗi cửa sổ, có thể thực hiện các hoạt động xử lý dữ liệu như áp dụng hàm map, reduce, filter hoặc bất kì phép biến đổi nào khác. Dữ liệu trong cửa sổ sẽ được tự động chuyển đến các phép biến đổi này để tính toán kết quả

**Ứng dụng:**

* Dstream Window thường được sử dụng để thực hiện các tác vụ như tính toán tổng của dữ liệu trong khoảng thời gian, xác định giá trị trung bình, tối đa, tối thiểu hoặc xác định sự thay đổi của dữ liệu theo thời gian.
* Giúp xác định xu hướng và thông tin quan trọng trong dữ liệu thời gian thực một cách hiệu quả.

*Ví Dụ:* Nếu muốn theo dõi tổng số lượng truy cập vào 1 trang web trong cửa sổ 1 giờ, có thể sử dụng Dstream Window với kích thước cửa sổ là 1 giờ và kích thước bước là 10 phút. Spark streaming sẽ tự động tính tổng lượng truy cập trong mỗi cửa sổ 1 giờ và cập nhật kết quả sau mỗi 10 phút