**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

---

**BÁO CÁO PROJECT 3**

ĐỀ TÀI

# Nghiên cứu về việc lưu trữ và xử lý dữ liệu lớn trên cụm Apache Spark Cluster

Giảng viên hướng dẫn: **TS. Nguyễn Tuấn Dũng**

Sinh viên thực hiện: Mai Xuân Thắng

MSSV: 201743369

***Hà Nội, tháng 1 năm 2021***

# Mục lục

# LỜI MỞ ĐẦU

Ngày nay, sự phát triển của công nghệ thông tin đã xâm nhập vào hầu hết các mặt của đời sống xã hội, với những ứng dụng rộng rãi hỗ trợ cho con người trên nhiều lĩnh vực. Cùng với đó là lượng dữ liệu lớn được sinh ra từng giây từng phút trong mọi lĩnh vực. Có thể nói dữ liệu là "dầu mỏ" của thế giới hiện đại ngày nay. Việc lưu trữ, quản lý và sử dụng nguồn tài nguyên này một cách hợp lý là bài toán bức thiết và quan trọng trong xã hội hiện nay.

Trong các công cụ quản lý dữ liệu lớn thì Apache Spark ngày càng đóng vai trò lớn hơn vì nó cho phép xây dựng các mô hình dự đoán nhanh chóng với việc tính toán được thực hiện trên một nhóm các máy tính, có có thể tính toán cùng lúc trên toàn bộ tập dữ liệu mà không cần phải trích xuất mẫu tính toán thử nghiệm. Vì vậy em quyết định chọn đề tài nghiên cứu về việc lưu trữ và xử lý dữ liệu trên framework này để xem hiệu năng cũng như sự tiện lợi của nó đối với cụm cluster.

Em xin gửi lời cảm ơn chân thành tới thầy Nguyễn Tuấn Dũng đã tin tưởng cho em chọn đề tài này, đồng thời đã nhiệt tình hướng dẫn trong suốt học kì vừa qua, giúp em học được nhiều kiến thức mới và thử nghiệm về framework sẽ gắn với chặng đường nghiên cứu và làm việc trong tương lai của em.

Do nhiều nguyên nhân cả chủ quan và khách quan, chủ yếu là do kiến thức còn hạn hẹp và lượng tài nguyên còn hạn hẹp, nghiên cứu của em thực hiện còn nhiều hạn chế, cụ thể là lượng dữ liệu chưa thật sự quá lớn, cụm cluster chưa thực hiện được trên thật nhiều máy. Rất mong nhận được sự chỉ dẫn của thầy để hoàn thiện hơn trong tương lai.

# 1. Lý thuyết

## 1.1. Spark

### 1.1.1. Spark là gì?

Apache Spark là một open source cluster computing framework được phát triển sơ khởi vào năm 2009 bởi AMPLab tại đại học California. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay.

Spark cho phép xây dựng các mô hình dự đoán nhanh chóng với việc tính toán được thực hiện trên một nhóm các máy tính, có có thể tính toán cùng lúc trên toàn bộ tập dữ liệu mà không cần phải trích xuất mẫu tính toán thử nghiệm. Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

### 1.1.2. Tại sao chọn Spark?

Với nhu cầu xử lý dữ liệu lớn cao như hiện nay, Apache Spark là một trong những framework nổi trội nhất về nhiều mặt. Thứ nhất, về mặt chi phí, Spark là một framework mã nguồn mở (open source framework), nghĩa là nó miễn phí.

Thứ hai, về tốc độ xử lý thì Spark nhanh hơn nhiều những framework khác. Ví dụ, khi so sánh với Hadoop thì Spark được cho là nhanh hơn Hadoop gấp 100 lần khi chạy trên RAM, và gấp 10 lần khi chạy trên ổ cứng. Hơn nữa, người ta cho rằng Spark sắp xếp (sort) 100TB dữ liệu nhanh gấp 3 lần Hadoop trong khi sử dụng ít hơn 10 lần số lượng hệ thống máy tính.

Hơn nữa, một trong những ưu điểm lớn nhất của Spark là tính dễ sử dụng. Spark có giao diện người dùng thân thiện. Spark cung cấp các API thân thiện cho Scala Java, Python, và Spark SQL (hay còn gọi là Shark). Việc Spark được xây dựng từ các khối đơn giản nó giúp ta tạo các hàm do người dùng xác định một cách dễ dàng.

## 1.2. HDFS

### 1.2.1. HDFS là gì?

HDFS (tên viết tắt của từ Hadoop Distributed File System" là một hệ thống lưu dữ dữ liệu được sử dụng bởi Hadoop. Chức năng của hệ thống này là cung cấp khả năng truy cập với hiệu suất cao đến với các dữ liệu nằm trên các cụm máy tính.

HDFS tạo ra các mảnh nhỏ hơn của dữ liệu lớn rồi phân tán chúng lên các nodes khác nhau. Từ đó, sao chép mỗi miếng dữ liệu nhỏ hơn trên nhiều nodes khác. Do vậy, khi node bất kỳ có dữ liệu bị lỗi thì hệ thống sẽ tự động sử dụng dữ liệu từ 1 node khác rồi tiếp tục xử lý. Đây là một trong những tính năng đặc biệt quan trọng của HDFS.

### 1.2.2. Tại sao chọn HDFS?

HDFS cho phép dữ liệu có thể phân tán: Điều này có thể hiểu như sau: Nếu như có một cụm mà trong đó bao gồm 20 máy tính thì bạn chỉ cần đưa một file dữ liệu vào HDFS. Khi đó, thì file sẽ tự động được chia nhỏ thành nhiều phần rồi được lưu trữ ở 20 máy tính đó.

HDFS cho phép tính toán và phân tán song song: Thay vì chỉ sử dụng một máy để xử lý công việc, thì với HDFS thì bạn có thể để các máy hoạt động song song để xử lý chung một công việc để tiết kiệm thời gian.

HDFS cho phép nhân bản các file: Đặc điểm này sẽ giúp bạn đề phòng được các trường hợp một máy tính trong cụm phát sinh sự cố thì dữ liệu sẽ được backup lại mà không bị mất.

Và quan trọng nhất, mục đích của Apache Spark là xử lý dữ liệu. Tuy nhiên, để xử lý dữ liệu, hệ thống cần dữ liệu đầu vào từ thiết bị lưu trữ. Và với mục đích này, Spark sử dụng HDFS. Đây không phải là lựa chọn duy nhất, nhưng là lựa chọn phổ biến nhất vì Apache là bộ não đằng sau cả hai.

# 2. Bộ dữ liệu

Trước hết, để xử lý dữ liệu lớn thì điều kiện tiên quyết là cần phải có dữ liệu. Trong phạm vi đề tài thì em sử dụng là dữ liệu dạng text. Bộ dữ liệu nặng hơn 1,5GB do em crawl từ các trang báo online ở Việt Nam. Cụ thể:

- Thư viện sử dụng: newspaper3k, bs4
- Độ lớn: 1,62GB (chưa nén, trên hệ điều hành Ubuntu 20.04.1 LTS)
- Định dạng: Bộ dữ liệu sau khi crawl được chia thành 47 file txt đánh số từ 1 đến 47
- Crawl thành công: 467.949 / 545.662 đường link
- Nội dung: Nội dung các bài báo online bằng tiếng Việt trên 15 page báo mạng lớn
- Nguồn crawl:

| | |
|---|---|
| + https://nhandan.com.vn/ | 15.084 / 28.811 |
| + https://tuoitre.vn/ | 46.601 / 46.782 |
| + https://laodong.vn/ | 45.690 / 53.643 |
| + https://thanhnien.vn/ | 77.121 / 77.941 |
| + http://kinhtedothi.vn/ | 69.688 / 70.846 |
| + https://www.doisongphapluat.com/ | 11.345 / 11.406 |
| + https://vov.vn/ | 1.271 / 1.557 |
| + https://infonet.vietnamnet.vn/ | 27.972 / 34.894 |
| + https://vnexpress.net/ | 51.777 / 53.281 |
| + https://suckhoedoisong.vn/ | 42.521 / 53.72 |
| + https://dantri.com.vn/ | 42.313 / 73.048 |
| + https://vietnamnet.vn/ | 26.704 / 31.496 |
| + https://congly.vn/ | 24.874 / 24.887 |
| + http://baochinhphu.vn/ | 31.331 / 31.367 |
| + http://baovanhoa.vn/ | 257 / 331 |

# 3. Thực nghiệm

Ở đây, để so sánh hiệu năng và sự thể hiện của Spark thì em chọn bài toán khá đơn giản là WordCount. Tuy vậy nhưng với lượng dữ liệu dạng text tương đối lớn thì chúng ta cũng có thể có cái nhìn về việc xử lý dữ liệu lớn thông thường so với bằng Spark, cũng như việc lưu trữ trên máy cục bộ (local memory) với lưu trữ trên HDFS trong Spark.

## 3.1. Cách thực hiện

### 3.1.1. Chuẩn bị

Trước lúc thực hiện các thứ nghiệm thì cần chuẩn bị một số điều như sau:

- Hai máy tính với hệ điều hành Ubuntu để kết nối cụm cluster.

- Apache Spark được cài ở cả 2 máy. Và được config thành một cụm cluster. Ở đây em dùng bản 2.4.5.

- HDFS được cài ở cả 2 máy.

- Ngoài ra máy cần cài đặt sẵn một số thứ khác như: Python 3, Java 8.

### 3.1.2. Nguyên tắc thực hiện

Để đảm bảo công bằng nhằm mục đích kết quả mang tính khách quan để so sánh, các thử nghiệm ở dưới được em ngầm định thực hiện theo một số nguyên tắc sau:

- Trước khi thực hiện, khởi động lại máy.

- Chỉ chạy chương trình thực hiện và không chạy thêm bất kỳ chương trình hay phần mềm nào thứ 3. Không tác động vào máy trong quá trình chạy code.

- Mỗi thử nghiệm chạy với 1 file data1.txt (50MB) và chạy trên toàn bộ tập dữ liệu (1,6GB), mỗi trường hợp 5 lần và lấy kết quả trung bình.

- Kết quả được quy đổi sang giây và được làm tròn đến chữ số thứ 4 sau phần thập phân.

Kết quả so sánh sẽ thông qua thời gian chạy chương trình và được tính bằng cách sử dụng thư viện datetime của Python.

### 3.1.3. Quá trình thực hiện

Trong phạm vi đề tài này, em thử nghiệm ở 3 trường hợp: chạy WordCount trên máy local với chương trình Python thông thường, chạy với Apache Spark nhưng dữ liệu load từ bộ nhớ cục bộ máy chủ và chạy chạy với Apache Spark với dữ liệu load từ bộ nhớ phân tán HDFS của cụm cluster.

Cách cài đặt và thực hiện chạy cụ thể được ghi rõ trong file Huongdan.pdf đi kèm.

## 3.2. Tiến hành

### 3.2.1. WordCount trên máy cục bộ bằng Python thuần

Ta tiến hành chạy file wordcount_python.py trên Python 3, thử nghiệm với 50mb và 1gb dữ liệu thu được kết quả:

| | 1 | 2 | 3 | 4 | 5 | Trung bình |
|---|---|---|---|---|---|---|
| 50MB | 3.7329s | 3.5917s | 3.6071s | 3.9143s | 3.9742s | 3.7640s |
| 1,6GB | 123.8567s | 124.1428s | 136.5576s | 121.3078s | 119.0908s | 124.9911s |

### 3.2.2. WordCount trên máy cục bộ thông qua Spark

Trong thử nghiệm này ta sử dung Spark, cụ thể là PySpark (giao diện sử dụng Spark thông qua Python để gọi các SparkAPI) để chạy file wordcount_spark.py viết bằng Python. Ở đây ta sử dụng các hàm có sẵn để xử lý dữ liệu dưới dạng RDD (Resilient Distributed Datasets - một cấu trúc dữ liệu cơ bản của Spark). Kết quả thu được như bảng dưới:

| | 1 | 2 | 3 | 4 | 5 | Trung bình |
|---|---|---|---|---|---|---|
| 50MB | 7.9356s | 8.1095s | 7.5843s | 7.5830s | 7.1349s | 7.6695s |
| 1,6GB | 149.5242s | 141.6332s | 144.4526s | 150.6672s | 145.2352s | 146.3025s |

### 3.2.3. WordCount trên cụm cluster thông qua Spark và HDFS

Ta đẩy dữ liệu lên HDFS. Sau đó chạy wordcount trên cụm spark gồm 2 máy: một master và một datanode với dữ liệu đọc từ HDFS. Tương tự như chạy Spark trên máy cục bộ, chỉ cần đổi sang dùng hdfs_path thay vì local_path ta thu được bảng kết quả:

| | 1 | 2 | 3 | 4 | 5 | Trung bình |
|---|---|---|---|---|---|---|
| 50MB | 7.7042s | 7.9409s | 7.3133s | 8.2271s | 7.9214s | 7.8214s |
| 1,6GB | 141.1353s | 138.4363s | 143.3554s | 146.7432s | 136.1536s | 141.1648s |

# 4. Tổng kết

## 4.1. Nhận xét

Qua các bảng kết quả thu được, ta có một vài so sánh như sau:

- Đối với chạy trên một file dữ liệu nhỏ (50MB):

+ So với chạy Python thuần là 3.7640s thì thời gian chạy trên Spark với dữ liệu cục bộ tốn thời gian là 7.6695s gấp 2.03 lần, và chạy trên cụm Spark Cluster với HDFS tốn thời gian là 7.8214s gấp 2.08 lần.

=> Thời gian chạy trên dữ liệu nhỏ thì thời gian khởi tạo môi trường để chạy Spark tốn nhiều hơn so với thời gian xử lý khiến việc chạy trên Spark tốn gấp đôi thời gian so với chạy Python thuần.

+ Thời gian chạy Spark cluster chậm hơn 2% so với chạy Spark trên máy local (7.8214s so với 7.6695s) là khá tương đồng nhau.

=> Có sự chênh lệch rất nhỏ, do thời gian kết nối giữa hai máy.

- Đối với toàn bộ tập dữ liệu (1.6GB):

+ Thời gian chạy Spark với một máy cục bộ là 146.3025s và thời gian chạy trên cụm Spark Cluster là 141.1648s, lần lượt gấp 1.17 lần và 1.13 lần so với thời gian chạy Python thuần là 124.9911s.

=> Vẫn chậm hơn nhưng có thể thấy so với tỉ lệ gấp hơn 2 lần khi chạy dữ liệu ít thì thời gian chạy đã chênh lệch ít hơn. Có thể thấy khi tăng dung lượng dữ liệu cần xử lý thì tốc độ xử lý bằng Spark đã nhanh hơn so với thời gian xử lý bằng Python thông thường.

+ Thời gian chạy trên cụm Spark Cluster đã nhanh hơn so với thời gian chạy trên Spark với một máy local. Cụ thể là nhanh hơn 4.5%.

=> Khi chạy bằng nhiều máy hơn, hiệu suất xử lý dữ liệu trên Spark đã được cải thiện. Nếu ta tăng số máy lên nhiều hơn nữa, lên đến vài trăm hay vài nghìn máy vật lý thì hiệu suất xử lý có thẻ sẽ ấn tượng hơn nữa.

## 4.2. Kết luận

Dù bài toán còn nhiều hạn chế nhưng có thể phần nào thấy được lợi ích của việc lưu trữ và xử lý dữ liệu lớn trên các hệ thống phân tán dữ liệu, cụ thể ở đây là Spark và HDFS. Dự đoán với một hệ thống đủ lớn và xử lý dữ liệu lên đến nhiều TB thì Spark sẽ phát huy hiệu quả tốt hơn nhiều.

Qua quá trình tìm hiểu và thử nghiệm, em đã học và tiếp cận được nhiều công cụ và kiến thức về xử lý dữ liệu lớn nói chung cũng như về công nghệ nói riêng. Thời gian tiếp em sẽ thử nghiệm với dữ liệu lớn hơn và thử nghiệm với nhiều máy hơn. Bên cạnh đó sẽ phát triển thành một bài toán có ứng dụng rõ ràng hơn và áp dụng cả học máy và học sâu vào xử lý dữ liệu.