Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.00080

NHẬN DẠNG DẤU VÂN TAY DỮ LIỆU LỚN SỬ DỤNG GABOR WAVELET VÀ MAPREDUCE

Phan Anh Cang [1], Trần Hồ Đạt [2], Phan Thượng Cang [3]

[1] Khoa Công nghệ Thông tin, Trường Đại học Sư phạm Kỹ thuật Vĩnh Long
[2] Khoa Công nghệ Thông tin, Trường CĐ Kinh tế - Tài chính Vĩnh Long
[3] Khoa Công nghệ Thông tin, Trường Đại học Cần Thơ

cangpa@vlute.edu.vn, thdat88@gmail.com, ptcang@cit.ctu.edu.vn

***TÓM TẮT**: Nhận dạng dấu vân tay là một trong những phương pháp nhận dạng sinh trắc học phổ biến hiện nay. Nó được ứng dụng trong nhiều lĩnh vực trong đó có các hệ thống chấm công, truy tìm tội phạm, xác thực và bảo mật hệ thống,... Tuy nhiên, một trong những thách thức đối với các phương pháp truyền thống hiện nay là phụ thuộc vào thời gian trích xuất đặc trưng và nhận dạng. Điều này dẫn đến hạn chế của các hệ thống này là không thể nhận dạng hiệu quả trong điều kiện áp dụng cho môi trường dữ liệu lớn. Do đó, một phương pháp nhận dạng dấu vân tay dữ liệu lớn được đề xuất trong bài báo này dựa trên mô hình MapReduce trong môi trường Spark nhằm cải thiện thời gian thực hiện nhận dạng đáp ứng các hệ thống dữ liệu lớn. MapReduce được sử dụng trong khai phá và phân tích dữ liệu lớn mà không thể được xử lý trên các hệ thống thông thường do bị giới hạn bởi một số ràng buộc về tài nguyên như khả năng xử lý, bộ nhớ,... Kết quả thực nghiệm cho thấy phương pháp đề xuất đạt được kết quả nhận dạng dấu vân tay một cách tự động và hiệu quả.*

***Từ khóa:** Nhận dạng dấu vân tay, Gabor wavelet, rút trích đặc trưng vân tay, MapReduce, Spark*

I. GIỚI THIỆU

Vân tay của con người là bất biến và có thể phục hồi như cũ theo thời gian. Chính vì thế, nhận dạng vân tay là một trong những đặc điểm nhận dạng sinh trắc học phổ biến đang được sử dụng rộng rãi trong nhiều hệ thống. Với tốc độ phát triển dân số nhanh chóng qua các năm thì dữ liệu vân tay cũng ngày càng lớn đòi hỏi việc tính toán, xử lý cần rất nhiều thời gian và cần đến những hệ thống tính toán phức tạp. Các công việc xử lý trên dữ liệu đầu vào như: chuẩn hóa, làm rõ đường vân tay, trích xuất đặc trưng, nhận dạng đòi hỏi tốn nhiều thời gian và luôn là một thách thức đối với các hệ thống nhận dạng. Mặt khác, các hệ thống nhận dạng phải đáp ứng được độ tin cậy cho phép. MapReduce là kỹ thuật tính toán trên dữ liệu lớn mà điều này bị giới hạn khi thực hiện trên các máy tính cá nhân. Nội dung trình bày trong bài báo gồm các công việc liên quan giới thiệu về các thuật toán được sử dụng: phép biến đổi Gabor wavelet và kỹ thuật MapReduce; mô hình đề xuất nhận dạng vân tay sử dụng kỹ thuật MapReduce trong môi trường Spark; một số kết quả thực nghiệm đạt được.

II. CÔNG VIỆC LIÊN QUAN

Hiện nay, dấu vân tay được sử dụng rộng rãi trong các hệ thống xử lý nhận dạng hoặc xác thực vì dễ thu thập và lưu trữ. Các hệ thống này cho phép ứng dụng đa dạng dấu vân tay vào kiểm soát truy cập bảo mật cho điện thoại di động, các nhà sản xuất máy ATM và các nhà sản xuất ô tô để giảm đáng kể trộm cắp xe,… Bên cạnh đó, kỹ thuật MapReduce được ứng dụng nhiều trong các lĩnh vực để khai phá và phân tích dữ liệu lớn mà không thể được xử lý trên các hệ thống thông thường do bị giới hạn bởi một số ràng buộc về tài nguyên như khả năng xử lý, bộ nhớ,.... Nhiều nghiên cứu về nhận dạng vân tay được đề xuất. Đa số các thuật toán đề xuất đối sánh các điểm "minutiae" hoặc sự tương đồng giữa hai ảnh vân tay. Ackerman [1] đề xuất một phương pháp nhận dạng dấu vân tay dựa trên các điểm "minutiae". Phương pháp này phân tích các vùng đã được tiền xử lý để loại bỏ chi tiết không quan tâm và nhiễu trên ảnh để giảm thiểu thời gian xử lý. Bhuyan [2] và các đồng tác giả trình bày việc phân loại dấu vân tay bằng phương pháp khai phá dữ liệu. Ý tưởng chính của phương pháp là xem xét cả thông tin về hướng và các điểm đặc biệt xung quanh điểm trung tâm trong khu vực quan tâm để phân loại dấu vân tay. Phương pháp đề xuất có độ chính xác cao hơn và giảm các lỗi phân loại sai. Yenumula B Reddy [3] trình bày thuật toán phân loại dấu vân tay dựa trên việc rút trích các điểm "minutiae" và sử dụng các điểm này làm khóa chính để tìm kiếm các vân tay trong cơ sở dữ liệu. Mô hình MapReduce được thực hiện để nhận dạng và truy vấn dấu vân tay trong cơ sở dữ liệu. Philippe Parra [4] sử dụng phép biến đổi Gabor trong tiền xử lý vân tay đầu vào và thuật toán Ransac dùng để nhận dạng dấu vân tay dựa trên các điểm minutiae. Tác giả thực hiện hơn 9120 phép so sánh với thời gian hơn 20 giờ, phương pháp đề xuất đạt độ tin cậy cao. Tuy nhiên thời gian thực hiện của hệ thống là tương đối dài do chưa sử dụng kỹ thuật phân lớp vân tay. Maillo [5] và các cộng sự sử dụng tiếp cận MapReduce và thuật toán KNN trong phân lớp dữ liệu lớn, nhóm tác giả thực hiện phân lớp trên tập dữ liệu Poker-Hand với hơn 1 triệu mẫu khác nhau. Kỹ thuật phân lớp dựa trên MapReduce cho thấy được độ chính xác cao với thời gian thực hiện được cải thiện. Trong bài báo này, chúng tôi đề xuất một phương pháp nhận dạng dấu vân tay dựa trên việc rút trích các điểm "minutiae" với phép biến đổi Gabor wavelet và thuật toán Crossing Number. Sau đó, phương pháp phân loại KNN và thuật toán đối sánh dựa trên các điểm minutiae được áp dụng để nhận dạng dấu vân tay. Mô hình MapReduce trong môi trường Spark được áp dụng nhằm giảm thời gian thực hiện trong nhận dạng đáp ứng môi trường làm việc với dữ liệu lớn.

2.1. Rút trích đặc trưng

2.1.1. Phép biến đổi Gabor wavelet

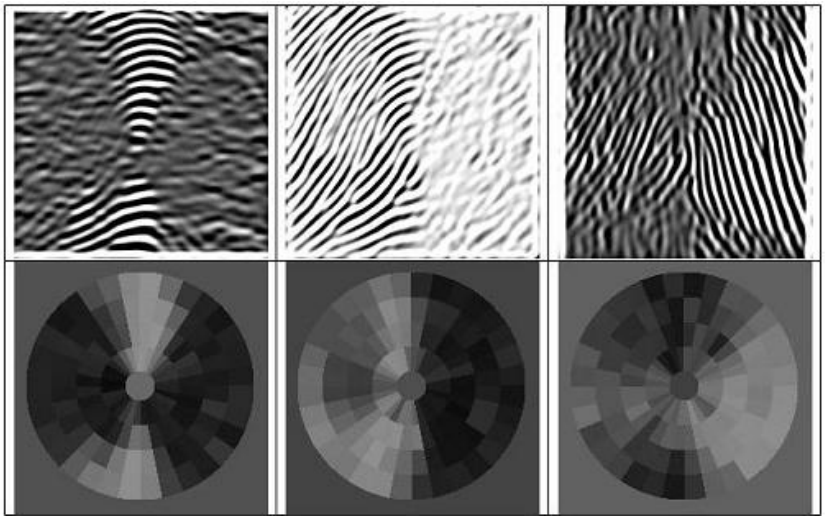
Trong thực tế, khi thu thập dữ liệu dấu vân tay không phải lúc nào cũng được kết quả tốt nhất. Hình ảnh dấu vân tay sau khi thu thập có thể bị nhiễu chẳng hạn như bị mờ, mất một phần dữ liệu,… Chính điều này làm ảnh hưởng đến độ tin cậy trong quá trình nhận dạng. Bộ lọc Gabor có hiệu quả trong việc tăng cường chất lượng ảnh vân tay đầu vào bằng cách chọn lọc trong miền không gian lẫn tần số. Hàm lọc Gabor có dạng như sau [6]:

$$G(x,y;f,\theta) = exp\left\{-\frac{1}{2}\left[\frac{x'^2}{\sigma_{x'}^2}+\frac{y'^2}{\sigma_{y'}^2}\right]\right\} cos(2\pi f x') \quad (1)$$

$$x' = x\,sin\theta + y\,cos\theta \quad (2)$$

$$y' = x\,cos\theta - y\,sin\theta \quad (3)$$

Trong đó: θ là hướng của bộ lọc, f là tần số của hàm sin, $\sigma_x$và σ_y là các độ lệch chuẩn dọc theo trục x và y thường được chọn từ thực nghiệm có giá trị [0,5]. Hàm lọc được thực hiện trong miền không gian với kích thước mặt nạ w x w.



Hình 1. Hình ảnh được lọc tương ứng cho các hướng 0^0, 22.5^0, 45^0 [6]

Thuật toán tăng cường chất lượng ảnh bằng bộ lọc Gabor thực hiện các giai đoạn như sau:

- Chuẩn hóa ảnh.
- Xác định trường định hướng của vân tay.
- Sử dụng hàm lọc Gabor cho ảnh đã chuẩn hóa trong miền tần số.
- Chia ảnh lọc thành từng khối nhỏ kích thước w x w.
- Xác định hướng của khối (dựa vào trường định hướng).
- Sử dụng phép biến đổi Fourier cho từng khối ảnh và hàm Gabor.

(a) (b)

Hình 2. Từ trái sang phải: (a) ảnh đầu vào, (b) ảnh được tăng cường với hàm lọc Gabor

Ảnh sau khi được chuẩn hoá bằng bộ lọc Gabor cho kết quả rất tốt (Hình 2a) và 2b)). Các đường vân được thể hiện rõ nét, là điều kiện quan trọng trong trích xuất đặc trưng. Sau khi áp dụng bộ lọc Gabor cải thiện chất lượng ảnh, bước tiếp theo chúng tôi sử dụng bộ lọc Gabor để trích xuất các đặc trưng trên ảnh vân tay [6], sau giai đoạn rút trích đặc trưng bằng bộ lọc Gabor số chiều của vector đặc trưng là rất lớn vì vậy chúng tôi tiếp tục sử dụng phương pháp PCA (trình bày trong mục 2.2) để giảm số chiều các đặc trưng trước khi bước vào giai đoạn kiểm tra.

2.1.2 Phát hiện các điểm đặc trưng bằng thuật toán Crossing Number

Hầu hết các hệ thống nhận dạng vân tay hiện nay thực hiện so sánh dựa trên các điểm minutiae vì cho kết quả nhận dạng rất cao. Do vậy, việc trích chọn các minutiae đáng tin cậy là một nhiệm vụ rất quan trọng. Vân tay sau khi được cải thiện bằng hàm lọc Gabor được nhị phân và làm mảnh để bắt đầu trích xuất các minutiae. Thuật toán phát hiện các điểm minutiae được sử dụng trong luận văn là Crossing Number (CN) [7] [8]. Thuật toán dùng một cửa sổ kích thước 3x3pixel, lấy tất cả các điểm ảnh trong cửa sổ, sau đó khảo sát giá trị logic của các điểm ảnh xung quanh điểm ảnh *[i,j]* ở chính giữa cửa sổ đó. Tùy vào kết quả tính toán của biểu thức CN thì kết luận: điểm *[i,j]* đang xét là một điểm phân nhánh, điểm cụt hay đang nằm trên một đường vân. Công thức tính như sau:

$$cn(p) = \frac{1}{2}\sum_{i=1}^{8}|p_i - p_{i+1}| \qquad p_9 = p_1 \tag{4}$$

Ảnh sử dụng phát hiện các minutiae phải là ảnh nhị phân, do đó val(p)∈{0,1}, các biến p1, p2, …, p8 thứ tự tạo thành các điểm lân cận điểm giữa cửa sổ đang khảo sát xét theo một chiều thuận hoặc ngược kim đồng hồ. Khi đó định nghĩa điểm *[i,j]* đang xét là:

- Điểm nằm trên đường vân nếu cn(p) =2.
- Điểm kết thúc (termination minutiae) nếu cn(p) =1.
- Điểm rẽ nhánh (bifurcation minutiae) nếu cn(p) =3.

0	0	1
0	1	0
0	0	0

a) cn(p)=1

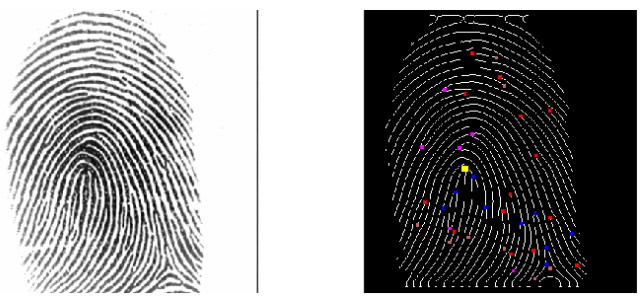
1	0	0
0	1	0
0	0	1

b) cn(p)=2

1	0	1
0	1	0
0	1	0

c) cn(p)=3

Hình 3. a) điểm kết thúc; b) điểm trên đường vân; d) điểm rẽ nhánh

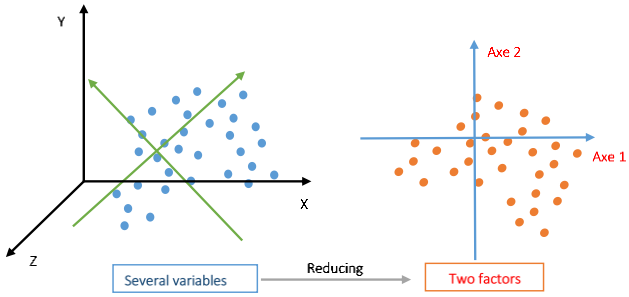


Hình 4. Ảnh ban đầu và ảnh sau khi trích xuất đặc trưng

Hình 4 minh họa kết quả rút trích đặc trưng với thuật toán CN. Các đặc trưng quan trọng được trích xuất bao gồm: core/delta, đường vân rẽ nhánh, đường vân kết thúc. Thuật toán trên không chỉ giúp tìm ra các đặc trưng quan trọng của ảnh vân tay mà còn giúp loại bỏ các đặc trưng sai trên các ảnh vân tay đầu vào.

2.2. Phương pháp phân tích thành phần chính

Phương pháp phân tích thành phần chính (PCA: Principal Component Analysis [9]) là một trong các phương pháp phân tích dữ liệu, ưu điểm chính của phương pháp này là giảm số chiều của dữ liệu với lượng thông tin bị mất là ít nhất. Ý tưởng của thuật toán được minh họa trong Hình 5.



Hình 5. Giảm số chiều của các điểm dữ liệu

Các bước chính thực hiện thuật toán PCA bao gồm [10], [11]:

- Bước 1: Tiền xử lý dữ liệu.
- Bước 2: Tính toán ma trận hiệp phương sai.
- Bước 3: Tính toán các trị riêng (eigenvalues) và vector riêng (eigenvectors).
- Bước 4: Chuyển dữ liệu từ không gian ban đầu vào không gian mới.

Các vector với số chiều đầu vào lớn gây khó khăn và mất nhiều thời gian tính toán. Phương pháp PCA rất hiệu quả góp phần cải thiện thời gian tính toán với lượng thông tin được đảm bảo. Các thông tin quan trọng được giữ lại phục vụ quá trình kiểm tra ảnh vân tay.

2.3. Thuật toán K-Láng giềng gần nhất (KNN: K-Nearest Neighbors)

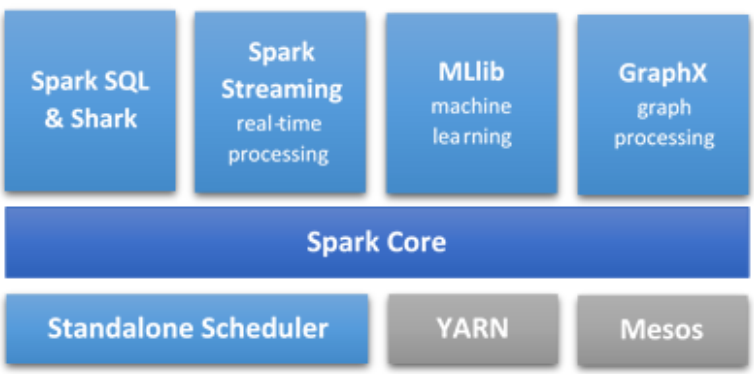
KNN [12] là thuật toán phân lớp các đối tượng dựa vào khoảng cách gần nhất giữa đối tượng cần phân lớp và tất cả các đối tượng trong tập huấn luyện. Đây là thuật toán đơn giản nhất trong các hệ thống máy học. Giai đoạn huấn luyện của thuật toán chỉ bao gồm việc lưu trữ các vectơ đặc trưng và nhãn của các ảnh huấn luyện [13]. Các bước thực hiện thuật toán KNN được mô tả như sau:

- Bước 1: Xác định giá trị tham số K (số láng giềng gần nhất).
- Bước 2: Tính khoảng cách giữa đối tượng cần phân lớp với tất cả các đối tượng trong tập huấn luyện (dựa vào khoảng cách Euclidean).
- Bước 3: Sắp xếp khoảng cách theo thứ tự tăng dần và xác định K láng giềng gần nhất với đối tượng cần phân lớp.
- Bước 4: Dựa vào phần phân lớp của láng giềng gần nhất để xác định lớp cho đối tượng cần phân lớp.

Ưu điểm của thuật toán KNN là dễ dàng cài đặt và sử dụng. Bên cạnh đó, hạn chế dễ nhận thấy của thuật toán này là cần phải chọn giá trị K phù hợp cho mỗi mô hình dự đoán, mất khá nhiều thời gian tính toán trong việc phân lớp cho một mẫu mới do phải tính toán và so sánh khoảng cách đến tất cả các mẫu trong tập huấn luyện.

2.4. Spark và hệ thống tập tin phân tán HDFS

Apache Spark được phát triển vào năm 2009 bởi AMPLab tại Đại học California. Spark cung cấp mô hình thực thi cho phép tính toán trên cụm nhằm làm tăng khả năng tính toán. Bên cạnh đó, Spark hỗ trợ tính toán tại bộ nhớ trong giúp truy xuất dữ liệu nhanh hơn bộ nhớ ngoài. Trong một số kết quả thực nghiệm cho thấy Spark có thể chạy nhanh gấp 10 đến 100 lần so với Hadoop [14] [15] [16]. Spark Core trong cấu trúc các tầng của Spark (Hình 6) là một thành phần của Spark: cung cấp những chức năng cơ bản nhất của Spark như lập lịch cho các tác vụ, quản lý bộ nhớ, phục hồi lỗi, tương tác với các hệ thống lưu trữ… Đặc biệt, Spark Core cung cấp API để định nghĩa RDD (Resilient Distributed DataSet) là tập hợp của các thành phần được phân tán trên các nút (node) của cụm (cluster) và có thể được xử lý song song.



Hình 6. Cấu trúc các tầng của Spark *[17]*

• Spark SQL cho phép truy vấn dữ liệu cấu trúc qua các câu lệnh SQL. Spark SQL có thể thao tác với nhiều nguồn dữ liệu như Hive tables, Parquet, và JSON.

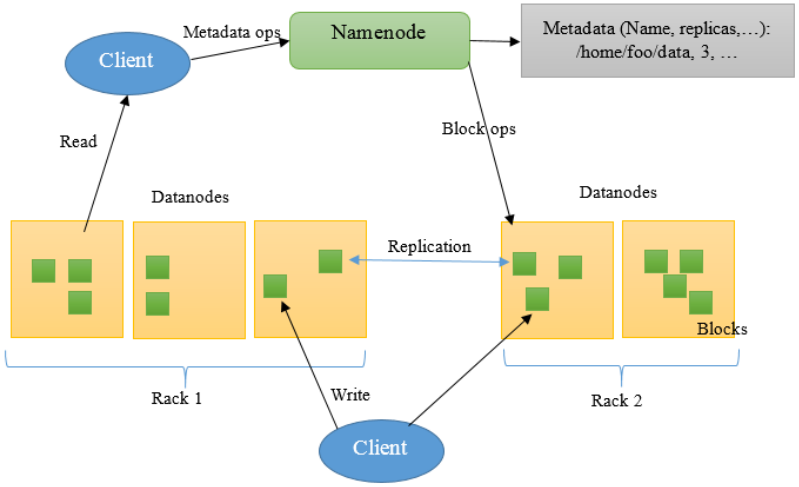
• Spark Streaming cung cấp API để dễ dàng xử lý dữ liệu stream,

• MLlib Cung cấp rất nhiều thuật toán của học máy như: classification, regression, clustering, collaborative filtering…

• GraphX là thư viện để xử lý đồ thị.

Hadoop là nền tảng hỗ trợ cho phép xử lý phân tán các tập dữ liệu lớn trên các cụm máy tính. Hadoop cung cấp hệ thống file phân tán (HDFS) và hỗ trợ mô hình MapReduce cho phép các ứng dụng làm việc với nhiều nút với hàng petabyte dữ liệu. Một cụm máy tính cài đặt hệ thống HDFS có hai loại nút: Nút tên (NameNode), hay còn gọi là nút chủ (master) và nút dữ liệu (DataNodes), hay còn gọi là nút tớ (worker). NameNode quản lý không gian tên hệ thống tập tin. Nó duy trì cây hệ thống tập tin và siêu dữ liệu cho tất cả các tập tin và thư mục trong cây. NameNode nhận biết

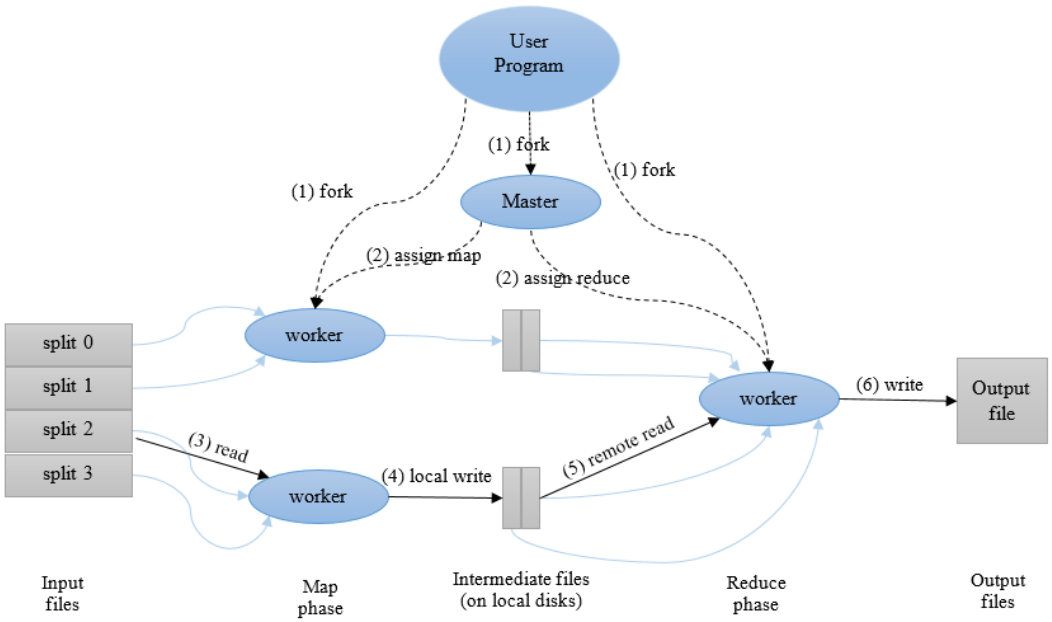
các DataNode mà trên đó tất cả các khối cho một tập tin được đặt phân tán. Các DataNode lưu trữ và lấy khối khi nó được gọi (bởi người dùng hoặc từ nút chủ) [18]. Hình 7 biểu diễn kiến trúc hệ thống tập tin phân tán HDFS [19].



Hình 7. Kiến trúc hệ thống tập tin phân tán HDFS *[20]*

2.5. Mô hình MapReduce

MapReduce được Google đưa ra vào năm 2004 với mục đích rút ngắn thời gian xử lý toàn bộ dữ liệu bằng cách sử dụng các máy tính hoạt động song song nhưng độc lập với nhau. Mô hình tính toán MapReduce khác biệt ở chỗ mã chương trình được sao chép tới các nút lưu trữ để thực thi. Đây là một trong những điểm mấu chốt tiên tiến của MapReduce vì quan điểm di chuyển mã chương trình thì tiết kiệm và hiệu quả hơn di chuyển dữ liệu mà có thể lên tới hàng TB [21]. MapReduce là mô hình lập trình được sử dụng để tính toán tập dữ liệu lớn. Một tiến trình xử lý MapReduce cơ bản có thể tính toán đến terabytes hoặc petabyte dữ liệu trên hệ thống được kết nối thành cụm các nodes. Dữ liệu được chia thành các mảnh nhỏ rồi đưa vào các nodes độc lập, vì vậy số lượng và kích thước của các mảnh phụ thuộc vào số nodes được kết nối trong mạng [22]. Các bước Map và Reduce được thiết kế tách biệt, riêng rẻ và hoàn toàn độc lập. Mỗi bước Map và Reduce được thực hiện song song trên các cặp dữ liệu (key, value). Do đó, chương trình được chia thành hai giai đoạn riêng biệt là Map và Reduce [23]. Mô hình thực hiện MapReduce tổng quát được biểu diễn trong Hình 8 dưới đây:



Hình 8. Mô hình thực hiện MapReduce tổng quát *[24]*

MapReduce là một thư viện ứng dụng xử lý dữ liệu phân tán và song song. Nó cho phép các lập trình viên thông thường lập trình các ứng dụng phân tán mà không cần biết chi tiết về kiến trúc xử lý phân tán. Kiến trúc của nó dựa trên chế độ master và slave. Master là một nút đặc biệt để phối hợp hoạt động giữa nhiều nút worker. Nó nhận dữ liệu đầu vào được xử lý. Dữ liệu đầu vào được chia thành các khối nhỏ hơn và tất cả các khối này được xử lý song song trên nhiều nút worker phân tán. Bước này được gọi là "Map". Nút worker trả về kết quả cho master và master bắt đầu một nhiệm vụ khác để kết hợp các kết quả này, được gọi là "Reduce". Công việc này cũng chạy trên nhiều nút worker phân tán và song song.

2.6. Phương pháp đối sánh vân tay dựa trên các điểm minutiae (matching)

Trong một ảnh vân tay chất lượng tốt có khoảng từ 70 đến 80 điểm minutiae [25] [26]. Ý tưởng chính của thuật toán đối sánh vân tay dựa trên các điểm minutiae là trích xuất các đặc trưng trên vân tay đầu vào và lưu trữ dưới dạng vectơ đặc trưng: [X,Y,CN,Theta,Flag,1], trong đó 3 thành phần quan trọng trong việc đối sánh bao gồm: X và Y chứa tọa độ của điểm minutiae, Theta là hướng của điểm minutiae.

Gọi T và I là các biểu diễn của ảnh vân tay mẫu và vân tay đầu vào: T = $\{m_1, m_{2,\dots,} m_i\}$; $m_u=\{x_u,y_u,\theta_u\}$, u = 1,…i, I = $\{m_1, m_{2,\dots,} m_j\}$; $m_t=\{x_t,y_t,\theta_t\}$, t = 1,…,j. Quá trình đối sánh lần lượt từng điểm đặc trưng ảnh vân tay đầu vào với tập điểm đặc trưng huấn luyện. Với i và j là số các điểm đặc trưng trong T và I. Một điểm đặc trưng m_t trong I và một điểm đặc trưng m_u trong T được xem là đối sánh nếu khoảng cách không gian (sd) giữa chúng nhỏ hơn mức dung sai cho trước $r_0$và sự khác nhau về hướng (dd) giữa chúng là nhỏ hơn góc dung sai θ_0:

$$sd(m_i, m_j) = 1 \Leftrightarrow \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \le r_0 \quad (5)$$

$$dd(m_i, m_j) = 1 \Leftrightarrow min(|\theta_i - \theta_j|, 360 - |\theta_i - \theta_j|) < \theta_0 \quad (6)$$

Hai vân tay sẽ được so sánh dựa trên các điểm minutiae trích ra từ các vân tay. Công thức tính độ tương tự hai vân tay được cho theo công thức sau:

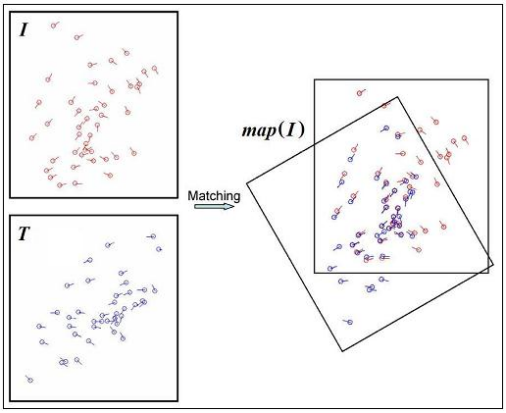
$$S(T, I) = max\left(\forall_m \sum_{i=1}^{n} md(m_i, map_m(m_i'))\right) \quad (7)$$

$$md(m_i, m_j) = sd(m_i, m_j).dd(m_i, m_j) \quad (8)$$

Trong đó:

n: là số lượng điểm minutiae trong tập đầu vào I.

m: là số lần biến đổi tương đương với số lượng minutiae trong tập mẫu T.



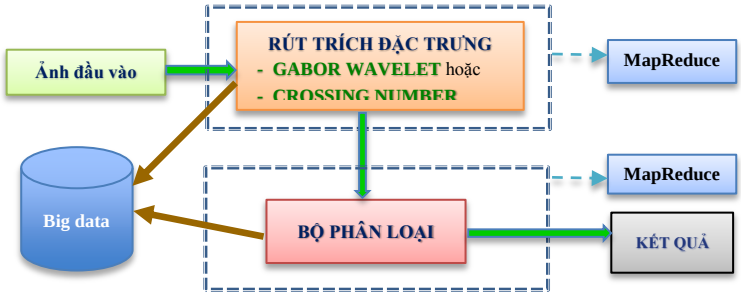
Hình 9. Đối sánh vân tay dựa trên minutiae *[26]*

Hàm *map* trong công thức (6) và hình 9 thực hiện *m* lần biến đổi tương đương sao cho các điểm m_i trong T chính là điểm m_j trong I. Hàm *md* dùng để xác định hai điểm minutiae trên hai vân tay có trùng khớp với nhau hay không $md(m_i, m_j)$=1 nếu $sd(m_i, m_j) \le r_0$ và $dd(m_i, m_j)<\theta_0$, ngược lại bằng 0.

III. MÔ HÌNH ĐỀ XUẤT

3.1. Phương pháp tổng quát

Đối với các hệ thống nhận dạng truyền thống là mặc dù thuật toán này nhận dạng chính xác các vân tay đầu vào nhưng nó cũng đối mặt với thách thức cho các hệ thống nhận dạng trong môi trường dữ liệu lớn, đặc biệt là thời gian tính toán trong giai đoạn nhận dạng cùng lúc nhiều ảnh vân tay đầu vào. Hầu hết các thuật toán không được thiết kế để xử lý tập dữ liệu lớn trong một khoảng thời gian hợp lý hoặc với khả năng bộ nhớ, bộ xử lý hợp lý.



Hình 10. Mô hình nhận dạng vân tay sử dụng MapReduce

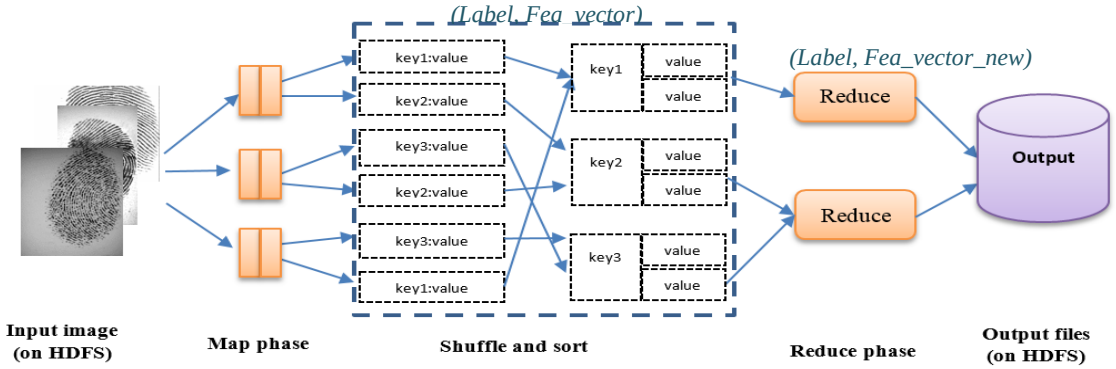
Để giải quyết thách thức này, mô hình MapReduce song song hoá hai giai đoạn rút trích đặc trưng và phân loại mô tả trong Hình 10 được ứng dụng vào thuật toán nhận dạng vân tay. Sau đây là cách hoạt động của mô hình này:

- Bước 1: Thực thi pha Map: Mỗi ảnh vân tay đầu vào được rút trích thành các dữ liệu vectơ đặc trưng để xử lý và chúng được chia nhỏ thành nhiều khối nhỏ hơn. Sau đó, các khối dữ liệu này đi qua một pha Map để ánh xạ và được xử lý song song trên nhiều nút worker phân tán. Kết quả bước này là tập dữ liệu trung gian dưới dạng (key, value). Đồng thời trong bước này, tập dữ liệu trung gian này cũng được trộn và sắp xếp theo thứ tự trước khi gửi đến pha Reduce.
- Bước 2: Thực thi pha Reduce: Nút worker trả về kết quả cho master và master bắt đầu một nhiệm vụ khác để kết hợp các kết quả này, được gọi là pha "Reduce". Các kết quả đầu ra của pha Map là tập dữ liệu trung gian dưới dạng (key, value) được sắp xếp theo thứ tự và xử lý gửi cho mỗi bộ Reducer. Các tập dữ liệu trung gian đi qua pha Reduce để chúng được kết hợp nhau và tạo ra kết quả cuối cùng đó là các dữ liệu output dưới dạng (key, value_new).
- Bước 3: Dựa trên các dữ liệu output, việc nhận dạng được quyết định dựa trên kết quả của các dữ liệu output này và giá trị tham số được chọn.

Như vậy, mô hình MapReduce có sự kết hợp chặt chẽ giữa pha Map và Reduce để xử lý dữ liệu. Mô hình này có lợi thế vì có thể mở rộng số lượng các nút trong hệ thống, nhờ đó khả năng tính toán của hệ thống cũng sẽ tăng lên. Chính điều này giúp hệ thống có thể hoạt động tốt khi kho dữ liệu là lớn. Vì vậy, MapReduce rất linh hoạt và mạnh mẽ để giải quyết các công việc xử lý dữ liệu lớn. Nó không phù hợp để thực hiện các phép tính trên tập dữ liệu bình thường có thể được nạp trực tiếp vào bộ nhớ máy tính và phân tích các kỹ thuật truyền thống.

3.2. Huấn luyện ảnh vân tay với mô hình MapReduce trong môi trường Spark

Giai đoạn huấn luyện hàng loạt ảnh vân tay được mô tả trong Hình 11. Gọi (*Label, Fea_vector*) tương ứng với nhãn và vector đặc trưng của tập vân tay huấn luyện và (*Label, Fea_vector_new*) tương ứng với nhãn và vector đặc trưng đầu ra của tập vân tay huấn luyện được lưu trữ trên HDFS.



Hình 11. Huấn luyện ảnh vân tay với mô hình MapReduce trong môi trường Spark

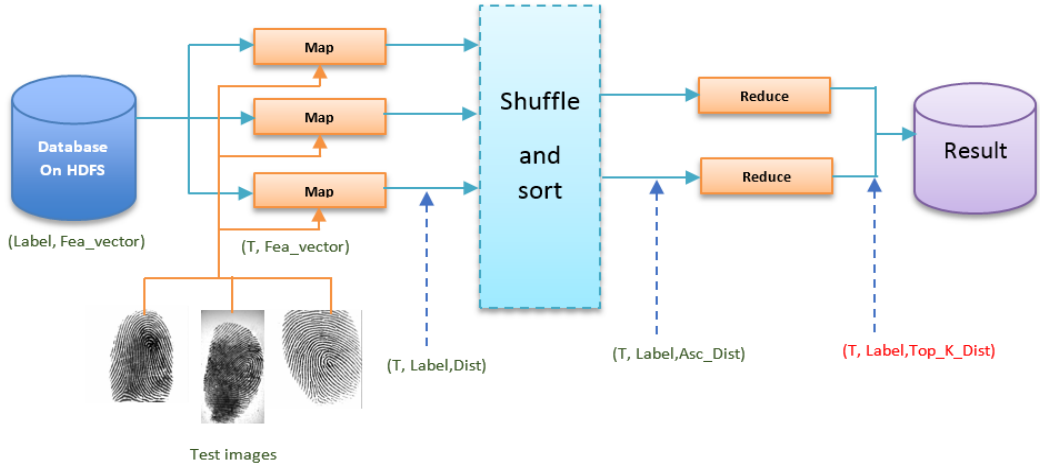
Quá trình thực hiện chi tiết được mô tả như sau:

- Giai đoạn Map:
  - Đầu vào hàm Map: tạo danh sách chứa nhãn (*Label*) và giá trị đặc trưng (*Fea_vector*).
  - Thủ tục Map: lấy tất cả các ảnh từ hệ thống tập tin phân tán HDFS → sử dụng bộ lọc Gabor tăng cường chất lượng ảnh → nhị phân hoá → làm mỏng ảnh → trích xuất đặc trưng → trộn và sắp xếp nhãn tương ứng với các giá trị đặc trưng.
  - Đầu ra hàm Map: vector đặc trưng của tập ảnh vân tay trên HDFS.

- Giai đoạn Reduce:
  - Đầu vào hàm Reduce: danh sách chứa nhãn và đặc trưng của tập ảnh vân tay.
  - Thủ tục Reduce: duyệt qua từng vector đặc trưng → sử dụng phương pháp PCA để giảm số chiều các vector đặc trưng → cập nhật lại danh sách nhãn và giá trị đặc trưng tương ứng.
  - Đầu ra hàm Reduce: danh sách đầu ra với nhãn (*Label*) và giá trị đặc trưng (*Fea_vector_new*).

3.3. Kiểm tra ảnh vân tay với mô hình MapReduce trong môi trường Spark

Sau quá trình huấn luyện ảnh vân tay, bước tiếp theo chúng tôi thực hiện kiểm tra ảnh vân tay để đánh giá độ chính xác của hệ thống. Mô hình tổng quát được thể hiện trong Hình 12. Gọi (*Label, Fea_vector*) tương ứng với nhãn và vector đặc trưng của tập vân tay huấn luyện được lưu trữ trên hệ thống tập tin phân tán HDFS và (*T, Fea_vector*) tương tứng với nhãn và vector đặc trưng của tập vân tay kiểm tra. Quá trình kiểm tra trên hai thuật toán KNN và matching bao gồm các giai đoạn:



Hình 12. Kiểm tra ảnh vân tay trên thuật toán KNN với mô hình MapReduce trong môi trường Spark

3.3.1. Kiểm tra với thuật toán KNN:

- Giai đoạn Map: Đầu vào hàm Map: tạo danh sách chứa các giá trị của tập dữ liệu kiểm tra. Thủ tục Map: truy xuất cơ sở dữ liệu có chứa tập dữ liệu kiểm tra (T, Fea_vector) → mở tập tin chứa tập dữ liệu huấn luyện (Label, Fea_vector) → tính toán cùng lúc giá trị khoảng cách từ tập dữ liệu kiểm tra với tập dữ liệu huấn luyện (T, Label, Dist) → ghi nhận giá trị khoảng cách tương ứng với nhãn và sắp xếp giá trị theo thứ tự tăng dần (T, Label, Asc_Dist). Đầu ra hàm Map: giá trị khoảng cách tương ứng với nhãn của tập dữ liệu kiểm tra.

- Giai đoạn Reduce: Đầu vào hàm Reduce: giá trị K láng giềng, dữ liệu kiểm tra. Thủ tục Reduce: tạo bộ đếm cho tất cả các nhãn → tìm K khoảng cách đầu tiên (Top_K_Dist) tương ứng của tập kiểm tra và tăng dần bộ đếm cho tất cả các nhãn. Đầu ra hàm Reduce: nhãn vân tay có bộ đếm lớn nhất trong tập kiểm tra.

3.3.2. Kiểm tra với thuật toán Matching:

- Giai đoạn Map: Đầu vào hàm Map: tạo danh sách chứa các giá trị của tập dữ liệu kiểm tra. Thủ tục Map: truy xuất cơ sở dữ liệu có chứa tập dữ liệu kiểm tra (T, Fea_vector) → mở tập tin chứa tập dữ liệu huấn luyện (Label, Fea_vector) → tính toán cùng lúc giá trị độ tương tự score từ tập dữ liệu kiểm tra với tập dữ liệu huấn luyện (T, Label, score) (dựa trên công thức (7)) → ghi nhận giá trị score tương ứng với nhãn và sắp xếp giá trị theo thứ tự giảm dần (T, Label, Desc_score). Đầu ra hàm Map: giá trị score tương ứng với nhãn của tập dữ liệu kiểm tra.
- Giai đoạn Reduce: Đầu vào hàm Reduce: max(score), dữ liệu kiểm tra. Thủ tục Reduce: tìm giá trị tương tự lớn nhất và gán vào max(score) tương ứng với nhãn của tập dữ liệu kiểm tra. Đầu ra hàm Reduce: nhãn vân tay với giá trị max(score).

IV. KẾT QUẢ NGHIÊN CỨU

Chúng tôi thực nghiệm hệ thống, trong đó một master được cài đặt ubuntu server để làm nút master quản lý và vận hành mô hình, hai worker được cài đặt ubuntu desktop có cấu hình Core 2 Duo-E6700-3.2Ghz, Ram 2Gb, HDD 320GB. Ngoài ra, chúng tôi sử dụng Hadoop 2.6.0, Spark 1.6.0. Ngôn ngữ lập trình được sử dụng là Matlab R2016b. Chúng tôi sử dụng cơ sở dữ liệu dấu vân tay FVC2000, FVC2002, FVC2004. Mỗi cơ sở dữ liệu bao gồm 320 ảnh vân tay của 40 người. Tổng cộng: 960 ảnh vân tay (tổng dung lượng xấp xỉ 130MB). Để kiểm tra độ chính xác của thuật toán, trên mỗi cơ sở dữ liệu ảnh vân tay chúng tôi tiến hành chia tập dữ liệu theo tỉ lệ 3:1. Nghĩa là sử dụng 240 mẫu vân tay để huấn luyện và 80 mẫu vân tay để kiểm tra.

Quá trình thực nghiệm được mô tả như sau: đầu tiên chúng tôi kiểm tra dữ liệu trên 1 máy tính với cấu hình máy tính Laptop Core i5-2410 (2.30Ghz), Ram 4Gb, HDD SSD 120Gb. Sau đó, tiến hành kiểm tra dữ liệu trên hệ thống tập tin phân tán HDFS trên 3 nút với 1 master và 2 worker. Hadoop được sử dụng để phát triển ứng dụng xử lý dữ liệu trong môi trường tính toán phân tán. Nó cung cấp hệ thống file phân tán (HDFS) và hỗ trợ mô hình MapReduce cho phép các ứng dụng làm việc với nhiều nút với hàng petabyte dữ liệu. Chúng tôi cũng thực nghiệm trên nhiều giá trị tham số K và chọn giá trị tốt nhất với K = 4. Việc kiểm tra độ chính xác nhận dạng vân tay trong 4 trường hợp:

- Trường hợp 1: Gabor + PCA + KNN.
- Trường hợp 2: Gabor + PCA + KNN (sử dụng mô hình MapReduce trong Spark).
- Trường hợp 3: Gabor + CN + Matching.
- Trường hợp 4: Gabor + CN + Matching (sử dụng mô hình MapReduce trong Spark).

Kết quả độ chính xác của thuật toán đề xuất với 4 trường hợp thực nghiệm được trình bày trong Bảng 1 như sau:

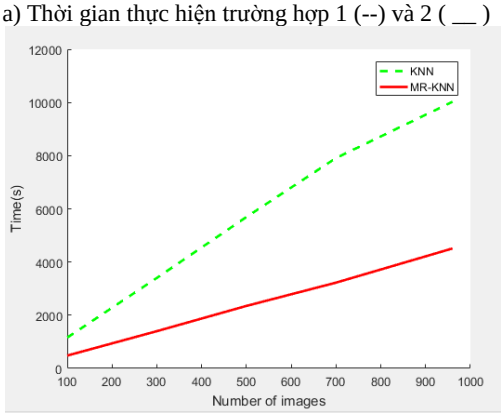
Từ Bảng 1 chúng ta có thể nhận xét: kết quả nhận dạng vân tay trên các cơ sở dữ liệu vân tay FVC đều cho độ chính xác là như nhau trong các trường hợp 1 và 2 hoặc 3 và 4. Hơn nữa, trường hợp 3 và 4 cho kết quả độ chính xác

cao hơn so với trường hợp 1 và 2 vì nó xem xét cả thông tin của các điểm minutiae xung quanh điểm trung tâm trong khu vực quan tâm để phân loại dấu vân tay. Tuy nhiên, thời gian thực hiện trường hợp 2 giảm đi rõ rệt so với trường hợp 1 do có ứng dụng mô hình MapReduce trong môi trường Spark như minh họa trong Hình 13a) và Bảng 2. Tương tự, thời gian thực hiện trường hợp 4 giảm đi rõ rệt so với trường hợp 3 do có ứng dụng mô hình MapReduce như minh họa trong Hình 13b) và Bảng 2.

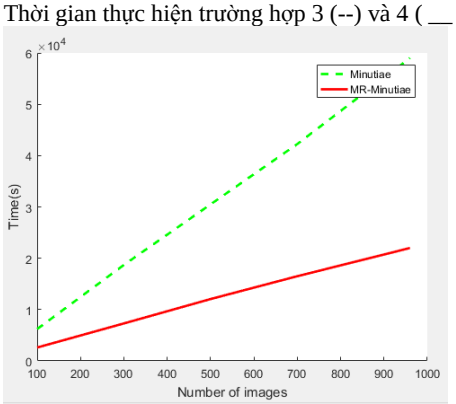
Bảng 1. Độ chính xác của thuật toán nhận dạng vân tay trong 4 trường hợp

Tập dữ liệu	**Độ chính xác nhận dạng vân tay** (%)			
	Trường hợp 1	Trường hợp 2	Trường hợp 3	Trường hợp 4
FVC2000	87.5	87.5	95	95
FVC2002	85	85	95	95
FVC2004	88.75	88.75	**96.25**	**96.25**

a) Thời gian thực hiện trường hợp 1 (--) và 2 (__)



b) Thời gian thực hiện trường hợp 3 (--) và 4 (__)



Hình 13. Biểu đồ so sánh thời gian thực hiện của các trường hợp 1-2-3-4

Bảng 2. Bảng thời gian thực hiện của các trường hợp 1-2-3-4

Số mẫu	**Thời gian thực hiện (giây)**			
	Trường hợp 1	Trường hợp 2	Trường hợp 3	Trường hợp 4
100	1158	482	6200	2604
300	3402	1400	18700	7300
500	5693	2350	30500	12050
700	7911	3225	42250	16500
960	10024	4510	59000	22000

V. KẾT LUẬN

Một giải pháp phù hợp giúp loại bỏ việc di chuyển dữ liệu vào và ra khỏi hệ thống lưu trữ trong khi phát huy khả năng tính toán song song và xử lý dữ liệu trực tiếp trên bộ nhớ trong là rất cần thiết. Đây là một vấn đề đang trở nên quan trọng do số lượng dữ liệu vân tay trên các hệ thống ứng dụng ngày càng tăng. Với các hệ thống nhận dạng truyền thống, một khi dữ liệu lớn hoàn toàn khó có khả năng đáp ứng về thời gian nhận dạng trong khi phải đối mặt với những giới hạn về tài nguyên bộ nhớ và khả năng xử lý của máy tính. Do đó, xử lý hiệu quả dữ liệu lớn là một bước quan trọng để phân tích và nhận dạng vân tay. Điều này chứng tỏ được tầm quan trọng của việc sử dụng mô hình MapReduce trên nền tảng xử lý phân tán để xử lý khối lượng lớn các dữ liệu vân tay.

Trong bài báo này, chúng tôi đã đề xuất một phương pháp hiệu quả về mặt thời gian thực hiện nhận dạng vân tay với việc ứng dụng mô hình MapReduce trong môi trường Spark để xử lý lượng lớn dữ liệu. Hệ thống hỗ trợ tính toán trực tiếp trên bộ nhớ trong làm cho quá trình truy vấn dữ liệu nhanh hơn so với các hệ thống dựa trên đĩa cứng như Hadoop. Hệ thống nhận dạng vân tay dựa trên mô hình MapReduce đáp ứng được các yêu cầu cơ bản về hiệu suất xử lý. Kết quả thực nghiệm cũng cho thấy về độ chính xác là như nhau trong khi thời gian thực hiện giảm rõ rệt so với các phương pháp truyền thống do sử dụng kỹ thuật tính toán song song và xử lý dữ liệu trực tiếp trên bộ nhớ trong. Trong nội dung bài báo, chúng tôi chỉ tiến hành thực nghiệm trên tập dữ liệu nhỏ với số nút hạn chế nhằm xây dựng mô hình mẫu đánh giá tính khả thi của hệ thống nhận dạng vân tay khi hoạt động dựa trên mô hình MapReduce trong môi trường Spark. Các công việc tiếp theo, chúng tôi sẽ thực hiện nhận dạng trên tập dữ liệu NIST [27] với số nút lớn hơn và so sánh độ chính xác cũng như thời gian nhận dạng trên thuật toán MCC (Minutia Cylinder-Code) [28].

TÀI LIỆU THAM KHẢO

[1] A. a. R. O. Ackerman, "Fingerprint recognition," *UCLA Computer Science Department*, 2012.

[2] M. H. S. S. a. D. K. B. Bhuyan, "An effective method for fingerprint classification," *arXiv preprint arXiv:1211.4658*, 2012.

[3] Y. B. Reddy, "Latent Fingerprint Matching in Large Databases Using High Performance Computing," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 13, 2015.

[4] P. Parra, "Fingerprint minutiae extraction and matching for identification procedure," *University of California, San Diego La Jolla, CA*, pp. 92093--0443, 2004.

[5] J. a. T. I. a. H. F. Maillo, "A mapreduce-based k-nearest neighbor approach for big data classification," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, IEEE, 2015, pp. 167--172.

[6] M. Y. J. M. Umer Munir, "Fingerprint Matching using Gabor Filters," *National Conference on Emerging Technologies*, 2004.

[7] B. M. Mehtre, "Fingerprint image analysis for automatic identification," *Machine Vision and Applications*, vol. 6, pp. 124-139, 1993.

[8] S. a. D. M. a. B. B. Kasaei, "Fingerprint feature extraction using block-direction on reconstructed images," *TENCON'97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE*, vol. 1, pp. 303--306, 1997.

[9] I. T. Jolliffe, "Principal Component Analysis and Factor Analysis," in *Principal component analysis*, Springer, 1986, pp. 115--128.

[10] L. I. a. o. Smith, "A tutorial on principal components analysis," *Cornell University, USA*, vol. 51, p. 65, 2002.

[11] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[12] T. M. C. a. P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, pp. 21-27, 1967.

[13] J. a. K. B.-S. a. S. S. Kim, "Comparing Image Classification Methods: K-nearest-neighbor and Support-vector-machines," *World Scientific and Engineering Academy and Society (WSEAS)*, pp. 133-138, 2012.

[14] M. C. M. J. F. S. S. a. I. S. Matei Zaharia, "Spark: cluster computing with working sets," *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, p. 10, 2010.

[15] A. P. a. M. S. Ranjani, "Spark-An Efficient Framework for Large Scale Data Analytics," *International Journal of Scientific & Engineering Research*, vol. 7, 2016.

[16] [Online]. Available: https://spark.apache.org/.

[17] H. a. K. A. a. W. P. a. Z. M. Karau, "Learning spark: lightning-fast big data analysis," *O'Reilly Media, Inc.*, 2015.

[18] T. C. Đệ, "Điện toán đám mây và bài toán xử lý dữ liệu lớn theo mô hình ánh xạ - rút gọn," *Tạp chí Khoa học Trường Đại học Cần Thơ*, pp. 56-63, 2013.

[19] D. Borthakur, "HDFS architecture guide," *Hadoop Apache Project*, pp. 1-13, 2008.

[20] D. Borthakur, "HDFS Architecture Guide," 2013. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

[21] T. V. Trung, "Dữ liệu lớn và làm chủ công nghệ dữ liệu lớn tại Việt Nam," *Kỷ yếu Hội thảo khoa học "Thống kê Nhà nước với Dữ liệu lớn"*, pp. 71-75, 2015.

[22] W. Tom, "Hadoop_The definitive Guide - 3rd edition," *O'reilly*, 2012.

[23] J. D. a. S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," 2004.

[24] G. S. Dean. J, "MapReduce : Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 1-13, 2008.

[25] J. T. GOVINDARAJU V., "Minutiae-based partial fingerprint recognition," *ProQuest Dissertations and Theses*, vol. 38, pp. 169-169 p., 2005.

[26] Ł. WIE□CŁAW, "A minutiae-based matching algorithms in fingerprint recognition systems," *Journal of Medical Informatics & Technologies*, vol. 13, pp. 65-71, 2009.

[27] C. I. Watson, "Nist special database 29," 2001.

[28] R. M. F. a. D. M. Cappelli, "Minutia cylinder-code: A new representation and matching technique for fingerprint recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2128-2141, 2010.

[29] X. W. a. V. Kumar, "The Top Ten Algorithms in Data Mining," *Chapman & Hall/CRC Data Mining and Knowledge Discovery*, 2009.

[30] I. T. F. H. Jesús Maillo, "A MapReduce-Based k-Nearest Neighbor Approach for Big Data Classification," *Trustcom/BigDataSE/ISPA, 2015 IEEE*, 2015.

[31] R. A. N. H. S. M. A. A. A. A. A. Rashid, "Security system using biometric technology: Design and implementation of voice recognition system (VRS)," *Proceedings of the International Conference on Computer and Communication Engineering 2008, ICCCE08: Global Links for Human Development*, pp. 898-902, 2008.

[32] C. Yang, L. Chen and W. &. W. K. Chou, "Implementation of a medical image file accessing system on cloud," *IEEE 13th International Conference on Computational Science and Engineering (CSE)*, pp. 321-326, 2010.

[33] S. N. S. P. Pelle Jakovits M.Sc., "Large-scale Image Processing Using MapReduce," *Thesis of TARTU UNIVERSITY Faculty of Mathematics and Computer Science Institute of Computer Science Computer Science*, 2013.

[34] Y. T. L. J. D. L. White Brandyn, "Web-Scale Computer Vision using MapReduce for Multimedia Data Mining," *Computing*, pp. 1-10, 2010.

[35] L. Liu, "Performance comparison by running benchmarks on Hadoop, Spark, and HAMR," *PhD thesis, University of Delaware*, 2015.

[36] B. Mehtre, "Fingerprint Image Analysis for Automatic Identification," *Machine Vision and Application*, 1993.

[37] S. Kasaei, "Fingerprint feature extraction using block-direction on reconstructed images," 1998.

[38] J. C. Amengual, "Real-time minutiae extraction in fingerprint images," 1997.

BIG DATA FINGERPRINT RECOGNITION USING GABOR WAVELET AND MAPREDUCE PARALLEL MODEL

Phan Anh Cang, Tran Ho Dat, Phan Thuong Cang

ABSTRACT: *Fingerprint recognition is one of the most popular biometric identification methods in the present time. It is used in many fields including attendance-controlled systems, criminal tracking, authentication and system security, and so on. However, one of the challenges of traditional methods is time-dependent on extracting minutiae and recognition. This issue leads to the limitation of these systems which is not able to effectively identify for the big data environment. Therefore, a big data fingerprint identification method which is proposed in this paper is based on the MapReduce model in a Spark environment to improve the recognition time for large data systems. MapReduce is used for exploring and analyzing big data sets which cannot be processed by traditional systems due to the processing power, memory constraints,… Experimental results show that our proposed method achieves automatic and effective fingerprint recognition.*

Keywords: *fingerprint recognition, Gabor wavelet, minutiae fingerprint extraction, MapReduce, Spark.*