TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VIỆN TRÍ TUỆ NHÂN TẠO

***

BÀI TẬP LỚN MÔN HỌC
KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN

CHỦ ĐỀ

Parallel BLAST on Hadoop for cellines analysis

Nhóm thực hiện:

1. Trần Xuân Bảo - 23020332
2. Hà Xuân Huy - 23020375
3. Phan Hoàng Dũng - 23020346

HÀ NỘI, 05/2025

Mục lục

1. Đặt vấn đề **4**
1.1. Giới thiệu về bài toán so khớp chuỗi sinh học 4
1.2. Vấn đề hiệu suất với BLAST truyền thống 5
1.3. Mục tiêu dự án 6
2. Cách tiếp cận **7**
2.1. Tư duy giải quyết bài toán 7
2.2. Lựa chọn công nghệ và công cụ 8
2.3. Đảm bảo tính tổng quát và mở rộng 8
3. Phân tích và thiết kế hệ thống **9**
3.1. Tổng quan 9
3.2. Thành phần chức năng chính 9
3.2.1. Hadoop HDFS 9
3.2.2. Distributed Cache 10
3.2.3. BlastMapper.java – Logic xử lý cốt lõi (Map Task) 10
3.2.4. BlastReducer.java – Gộp kết quả 10
3.2.5. BlastDriver.java – Điều phối toàn bộ hệ thống 11
3.2.6. FastaInputFormat.java – Định nghĩa định dạng đầu vào tùy chỉnh 11
3.2.7. FastaRecordReader.java – Trình đọc bản ghi FASTA 11
3.3. Luồng xử lý logic 12
3.4. Tính chất hệ thống 12
4. Triển khai hệ thống **14**
4.1. Chuẩn bị môi trường triển khai 14
4.2. Tải dữ liệu protein từ NCBI 14
4.3. Thực hiện BLAST với dữ liệu thô 15
4.3.1. BLAST truyền thống 15
4.3.2. Hadoop BLAST 17
4.4. Clustering và tái tổ chức dữ liệu 19
4.5. Chạy BLAST với tập dữ liệu đã thực hiện clustering 20
5. Thử nghiệm và đánh giá **21**
5.1. Mục tiêu đánh giá 21
5.2. Thiết lập thí nghiệm 21
5.3. Kết quả thí nghiệm 22
5.4. Phân tích kết quả 22
5.5. Đánh giá tổng quan 25

6. Cải tiến và hướng mở rộng **26**
6.1. Mở rộng quy mô xử lý 26
6.2. Nâng cấp công cụ xử lý 27
7. Kết luận **28**
8. Tài liệu tham khảo **30**

Phân công công việc

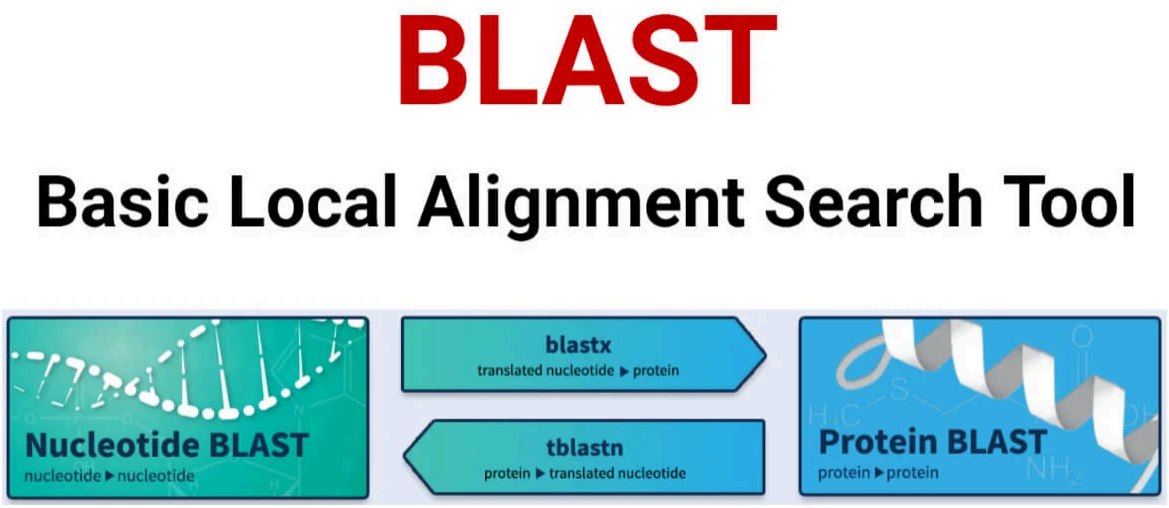
Thành viên	**Công việc**
Trần Xuân Bảo	● Tham khảo, lựa chọn dataset và chủ đề ● Quản lý bài tập lớn ● Thu thập một phần dữ liệu ● Đưa ra các công nghệ có thể sử dụng để tham khảo (Indexing, Clustering, ...) ● Tạo kiến trúc sơ bộ bài báo cáo, thuyết trình
Hà Xuân Huy	● Tìm nguồn dữ liệu, code thu thập dữ liệu. ● Thu thập 1 phần dữ liệu. ● Tạo code để xử lý MapReduce các file FASTA (Hadoop BLAST). ● Thử nghiệm, so sánh Hadoop thường và Hadoop BLAST. ● Trích xuất thông tin chi tiết để tạo biểu đồ trực quan hóa.
Phan Hoàng Dũng	● Thu thập một phần dữ liệu ● Đưa ra và vẽ biểu đồ pipeline cơ bản của hệ thống ● Tìm hiểu và thực hiện data clustering và tái tổ chức dữ liệu ● Viết báo cáo ● Làm slide

1. Đặt vấn đề

1.1. Giới thiệu về bài toán so khớp chuỗi sinh học

Trong lĩnh vực sinh học phân tử hiện đại, một trong những nhiệm vụ then chốt là phân tích và so khớp các chuỗi DNA, RNA hoặc protein. Các chuỗi sinh học này chứa đựng thông tin di truyền, quyết định chức năng và cấu trúc của các sinh vật sống. Việc so khớp các chuỗi giúp các nhà khoa học xác định mức độ tương đồng giữa các sinh vật, phát hiện gen đột biến, hoặc tìm kiếm các protein có chức năng tương tự trong các loài khác nhau.

Một trong những công cụ nổi tiếng và được sử dụng rộng rãi nhất cho bài toán này là BLAST (Basic Local Alignment Search Tool). BLAST giúp so khớp một chuỗi truy vấn với cơ sở dữ liệu khổng lồ các chuỗi đã biết, tìm ra các đoạn có mức độ tương đồng cao nhất. Chính vì vậy, BLAST là công cụ không thể thiếu trong các dự án giải mã bộ gen, phát triển thuốc, hay phân tích các dòng tế bào (cell lines) trong nghiên cứu y sinh học.



Tuy nhiên, trong bối cảnh bùng nổ dữ liệu sinh học hiện nay, với hàng triệu chuỗi gen và protein được giải mã mỗi năm, hiệu suất trở thành một vấn đề nghiêm trọng. BLAST truyền thống, dù chính xác, nhưng bị hạn chế bởi cơ chế xử lý tuần tự trên một máy đơn lẻ, dẫn đến thời gian tính toán kéo dài, đặc biệt khi làm việc với các cơ sở dữ liệu lớn.

1.2. Vấn đề hiệu suất với BLAST truyền thống

Cơ sở dữ liệu sinh học hiện nay đã phát triển với tốc độ chóng mặt. Ví dụ, cơ sở dữ liệu protein *nr* (Non-redundant) có dung lượng tới hàng chục GB, chứa hàng triệu chuỗi. Việc so khớp một tập các file FASTA (định dạng chứa chuỗi sinh học) với cơ sở dữ liệu này bằng BLAST đơn luồng đòi hỏi hàng giờ hoặc thậm chí hàng ngày để hoàn thành.

Điều này gây ra nhiều thách thức:

- Làm chậm quá trình nghiên cứu và khám phá.
- Hạn chế khả năng thực hiện phân tích quy mô lớn trên các dòng tế bào khác nhau.
- Đòi hỏi tài nguyên tính toán mạnh nếu muốn rút ngắn thời gian, dẫn đến chi phí tăng cao.

Trong bối cảnh đó, tăng tốc BLAST thông qua các giải pháp xử lý song song là xu hướng tất yếu. Một số giải pháp thương mại đã xuất hiện, như mpiBLAST (dựa trên MPI), hoặc các nền tảng điện toán đám mây. Tuy nhiên, những giải pháp này thường đòi hỏi cấu hình phức tạp hoặc chi phí đắt đỏ.

Một lựa chọn khả thi và tiết kiệm là tận dụng Hadoop, nền tảng mã nguồn mở cho xử lý dữ liệu phân tán, nổi tiếng với mô hình MapReduce. Với khả năng chia nhỏ công việc và phân phối lên hàng chục, hàng trăm nút tính toán, Hadoop cho phép xử lý song song hiệu quả các bài toán dữ liệu lớn, bao gồm cả bài toán so khớp chuỗi sinh học.

1.3. Mục tiêu dự án

Dự án này hướng tới việc triển khai một phiên bản song song của công cụ BLAST trên nền tảng Hadoop sử dụng mô hình MapReduce. Mục tiêu quan trọng nhất là tận dụng khả năng phân tán và xử lý song song của Hadoop để tăng tốc quá trình so khớp chuỗi sinh học, phục vụ cho các nghiên cứu về cell lines.

Cụ thể, dự án nhằm đạt được các mục tiêu sau:

- Chuyển đổi cơ chế BLAST tuần tự sang mô hình song song, trong đó mỗi tác vụ mapper sẽ xử lý một file FASTA riêng biệt.
- Phân phối dữ liệu BLAST (bao gồm chương trình thực thi và cơ sở dữ liệu) tới các nút tính toán thông qua cơ chế Distributed Cache của Hadoop.
- Tối ưu hóa việc xử lý file đầu vào để đảm bảo các file FASTA lớn có thể được xử lý trọn vẹn bởi từng mapper, tránh tình trạng phân mảnh hoặc mất dữ liệu.
- Rút ngắn đáng kể thời gian xử lý so với cách truyền thống, từ hàng giờ xuống còn vài phút hoặc thậm chí vài giây, tùy vào số lượng node trong cluster Hadoop.
- Giữ nguyên độ chính xác của BLAST, đảm bảo kết quả so khớp không bị ảnh hưởng bởi cơ chế song song hóa.
- Xây dựng một giải pháp dễ mở rộng, có thể triển khai trên các cluster với quy mô từ nhỏ (vài node) đến lớn (hàng trăm node), phục vụ cho các dự án sinh học quy mô quốc tế.

Dự án lựa chọn mô hình Map-Only trong MapReduce bởi tính chất công việc phù hợp: mỗi file FASTA đầu vào có thể được xử lý hoàn toàn độc lập. Cách tiếp cận này giúp đơn giản hoá kiến trúc, đồng thời tận dụng tối đa khả năng song song hóa tự nhiên của Hadoop.

Như vậy, có thể nói, mục tiêu của dự án không chỉ dừng lại ở việc tăng tốc BLAST, mà còn hướng tới việc xây dựng một nền tảng mở, sẵn sàng phục vụ cho các dự án phân tích chuỗi sinh học quy mô lớn trong tương lai.

2. Cách tiếp cận

Dự án được tiếp cận dưới góc nhìn tổng hợp giữa bài toán sinh học phân tử, cụ thể là xử lý dữ liệu chuỗi protein từ các dòng tế bào, và các kỹ thuật xử lý dữ liệu lớn nhằm đảm bảo hiệu năng khi làm việc với khối lượng dữ liệu tăng trưởng nhanh theo thời gian. Thay vì giải quyết bài toán theo hướng cài đặt đơn lẻ hoặc chạy tuần tự, nhóm tiếp cận bài toán với hai nguyên tắc:

- Tối ưu hiệu suất thông qua tổ chức dữ liệu logic và thông minh, bao gồm: chuẩn hóa, phân cụm, và tạo chỉ mục.
- Tận dụng sức mạnh xử lý song song của hệ sinh thái Hadoop, đặc biệt là khả năng phân phối dữ liệu và tính toán trên nhiều node.

2.1. Tư duy giải quyết bài toán

Bài toán đặt ra là: làm sao để so sánh nhanh chóng một chuỗi protein mới với một tập dữ liệu lớn các chuỗi protein đã có, nhằm phát hiện những chuỗi tương đồng? Trong khi BLAST vốn được thiết kế để xử lý tuần tự, hoặc xử lý song song theo CPU nhưng trên một đơn máy. Khi số lượng chuỗi trong cơ sở dữ liệu tăng lên hàng trăm ngàn hoặc hàng triệu, việc tìm kiếm tương đồng cho mỗi chuỗi mới trở nên cực kỳ tốn thời gian và tài nguyên.

Trước những thách thức trên, nhóm lựa chọn tiếp cận bài toán không theo cách xử lý tuyến tính hay đơn lẻ, mà theo tư duy xây dựng hệ thống xử lý theo luồng kết hợp với hạ tầng xử lý phân tán. Bài toán được chia thành nhiều giai đoạn rõ ràng, mỗi giai đoạn giải quyết một phần cụ thể của vấn đề:

- **Giảm kích thước dữ liệu mà vẫn giữ tính đại diện** → áp dụng phân cụm.
- **Tạo cấu trúc tìm kiếm nhanh thay vì so sánh tuyến tính** → dùng chỉ mục BLAST.
- **Phân tán xử lý trên nhiều nút để rút ngắn thời gian truy vấn** → tích hợp với Hadoop.
- **Tối ưu xử lý dữ liệu mới để tránh tính toán lại toàn bộ** → xây dựng mô hình tái sử dụng chỉ mục và phân cụm.

Một trong những điểm khó khăn thực tế là: dữ liệu protein không chỉ lớn mà còn liên tục phát sinh mới. Việc xử lý lại toàn bộ tập dữ liệu mỗi khi có thêm

vài chuỗi mới là không hiệu quả. Nhóm giải quyết vấn đề này bằng cách tái sử dụng kết quả phân cụm trước đó, chỉ so sánh dữ liệu mới với đại diện cụm để xác định cụm phù hợp. Ngoài ra, nhóm chỉ lập chỉ mục cho phần dữ liệu mới, sau đó ghép nối vào chỉ mục cũ nếu cần.

2.2. Lựa chọn công nghệ và công cụ

Việc lựa chọn công cụ không đơn thuần dựa trên sự phổ biến, mà dựa trên tiêu chí:

- **Phù hợp bản chất dữ liệu**: Dữ liệu dạng FASTA (chuỗi protein), cần công cụ hiểu ngữ nghĩa sinh học → chọn BLAST.
- **Phù hợp mô hình xử lý song song**: Dữ liệu lớn, nhiều chuỗi → chọn Hadoop để chia nhỏ và xử lý phân tán.
- **Phù hợp bài toán tăng trưởng theo thời gian**: Dữ liệu cập nhật liên tục → xây dựng pipeline linh hoạt, tái sử dụng kết quả cũ.

Từ đó, nhóm hình thành chiến lược xử lý theo tầng, giúp tách biệt rõ ràng các giai đoạn (tiền xử lý, phân cụm, lập chỉ mục, tìm kiếm, lưu trữ), hỗ trợ dễ bảo trì và mở rộng sau này.

2.3. Đảm bảo tính tổng quát và mở rộng

Ngay từ đầu, nhóm định hướng xây dựng một hệ thống không chỉ chạy được trên dữ liệu hiện tại mà còn mở rộng tốt với:

- Dữ liệu mới liên tục cập nhật.
- Khả năng chia sẻ và tái sử dụng cao (bằng cách gửi chỉ mục BLAST thay vì toàn bộ dữ liệu).
- Tái sử dụng kết quả phân cụm để lọc vùng tìm kiếm (truy vấn trực tiếp đại diện của cụm).

Như vậy, cách tiếp cận mà nhóm lựa chọn không chỉ mang tính giải pháp tức thời, mà còn đặt nền tảng cho một hệ thống sinh học tin học có thể tái sử dụng lâu dài, tích hợp tốt vào các cơ sở dữ liệu quy mô lớn hơn trong tương lai, như cơ sở dữ liệu gen người, động vật, vi khuẩn, hoặc các hệ thống hỗ trợ phân tích đa dạng sinh học và y sinh học phân tử.

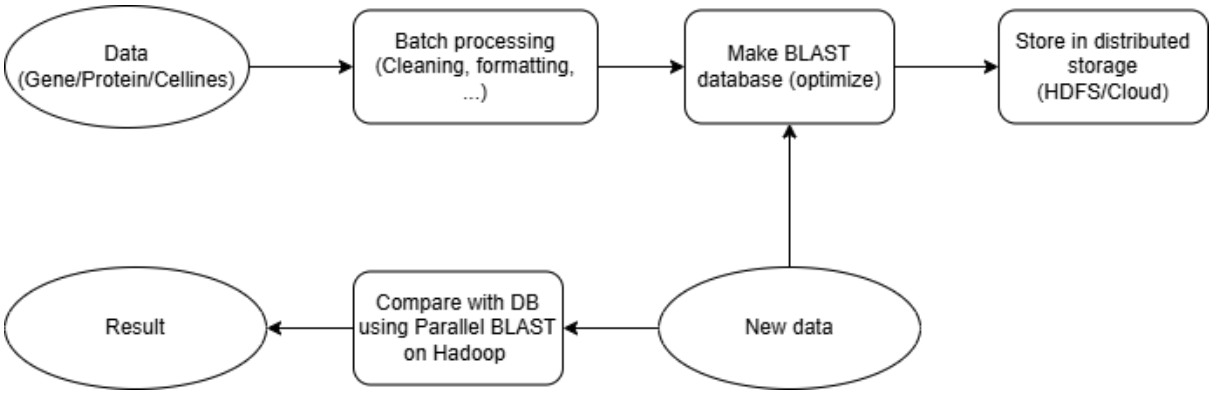
3. Phân tích và thiết kế hệ thống

3.1. Tổng quan

Hệ thống được thiết kế theo mô hình pipeline gồm các thành phần chính sau:

- **Tầng thu thập và chuẩn bị dữ liệu:** nơi tiếp nhận các chuỗi protein (dạng FASTA), xử lý định dạng và chuẩn hóa.
- **Cơ sở dữ liệu BLAST:** được tạo từ các chuỗi protein tham chiếu, tối ưu hoá cho việc tra cứu bằng BLAST.
- **Hệ thống lưu trữ phân tán:** sử dụng HDFS làm nơi lưu trữ các file đầu vào và đầu ra, đảm bảo tính sẵn sàng và khả năng chia sẻ dữ liệu giữa các node.
- **Hệ thống thực thi BLAST song song:** triển khai trên Hadoop MapReduce, mỗi mapper chịu trách nhiệm xử lý một hoặc một số file FASTA đầu vào.
- **Tầng đầu ra và tổng hợp kết quả:** lưu kết quả so khớp chuỗi của từng mapper về HDFS, phục vụ cho các bước phân tích sau.

Luồng dữ liệu trong hệ thống được mô tả trực quan qua sơ đồ pipeline:



3.2. Thành phần chức năng chính

3.2.1. Hadoop HDFS

Hệ thống lưu trữ chính của toàn bộ dữ liệu đầu vào, đầu ra và tệp tin thực thi. HDFS chia nhỏ các tệp lớn thành các khối (blocks) và phân phối chúng qua các node trong cluster, hỗ trợ khả năng lưu trữ song song và truy xuất hiệu quả. Việc đặt các file đầu vào trên HDFS giúp Hadoop tự động phân phối công việc đến các node gần dữ liệu, tối ưu hiệu năng.

3.2.2. Distributed Cache

Là cơ chế trong Hadoop giúp phân phối các file dùng chung như chương trình thực thi BLAST và cơ sở dữ liệu BLAST đến tất cả các node trong cụm. Điều này đảm bảo mọi mapper đều có thể truy cập được các tài nguyên cần thiết mà không cần tải về thủ công từ HDFS, giảm độ trễ và tăng độ ổn định.

3.2.3. BlastMapper.java – Logic xử lý cốt lõi (Map Task)

Blast Mapper là thành phần trung tâm trong xử lý BLAST song song. Mỗi mapper được khởi tạo từ BlastMapper.java chịu trách nhiệm xử lý một đơn vị dữ liệu (một file FASTA) như sau:

- **Nhận cặp đầu vào**: mapper nhận một cặp key-value, trong đó key là tên file FASTA, value là nội dung hoặc đường dẫn file.
- **Chuẩn bị môi trường**: giải nén dữ liệu từ Distributed Cache, tạo thư mục làm việc local.
- **Thực thi tiến trình ngoài**: dùng ProcessBuilder để gọi lệnh *blastp* trên file FASTA đầu vào, so khớp với database protein tham chiếu.
- **Ghi kết quả**: kết quả đầu ra (.txt hoặc .out) được ghi lên HDFS.

Mỗi mapper chạy độc lập nên hệ thống dễ mở rộng theo chiều ngang: càng nhiều mapper (tương ứng với số file FASTA), hiệu năng càng cao.

3.2.4. BlastReducer.java – Gộp kết quả

Trong trường hợp người dùng mong muốn gộp toàn bộ kết quả từ các mapper thành một file duy nhất để dễ phân tích, hệ thống có thể kích hoạt lớp **Blast Reducer**.

Reducer sẽ:

- Thu nhận tất cả kết quả .txt từ các mapper.
- Gộp vào một file duy nhất theo đúng thứ tự hoặc tiêu chí mong muốn.
- Có thể áp dụng thêm xử lý hậu kỳ như nén .gz, ghi thêm header, thống kê sơ bộ,...

Trong một số trường hợp, bước này có thể được bỏ qua nếu mỗi mapper tạo ra file riêng biệt phục vụ truy vấn độc lập.

3.2.5. BlastDriver.java – Điều phối toàn bộ hệ thống

Lớp **BlastDriver** có vai trò cấu hình và khởi tạo toàn bộ job MapReduce:

- Chỉ định lớp Mapper (Blast Mapper), lớp Reducer, định nghĩa input/output key-value types.
- Thiết lập Input Format tùy chỉnh.
- Thêm file thực thi và database vào Distributed Cache.
- Chạy job Hadoop với thông số cần thiết (thư mục input, output, đối số BLAST).

Blast Driver chính là điểm khởi đầu và điều phối toàn bộ pipeline xử lý phân tán.

3.2.6. FastaInputFormat.java – Định nghĩa định dạng đầu vào tùy chỉnh

Vì dữ liệu FASTA là một cấu trúc phức tạp (nhiều dòng cho một bản ghi), không thể xử lý tốt bằng các TextInputFormat mặc định của Hadoop, hệ thống đã xây dựng lớp FastaInputFormat kế thừa từ FileInputFormat.

Lớp này được thiết kế để:

- Đảm bảo mỗi mapper nhận nguyên vẹn một file FASTA.
- Tránh chia file thành các split nhỏ làm hỏng cấu trúc dữ liệu.

Từ đó giúp mỗi tác vụ Map có thể xử lý độc lập từng chuỗi protein theo đúng chuẩn định dạng, không bị phân mảnh logic.

3.2.7. FastaRecordReader.java – Trình đọc bản ghi FASTA

FastaRecordReader là lớp đi kèm với FastaInputFormat, định nghĩa cách đọc từng bản ghi FASTA từ HDFS. Thay vì chỉ đọc từng dòng, nó sẽ:

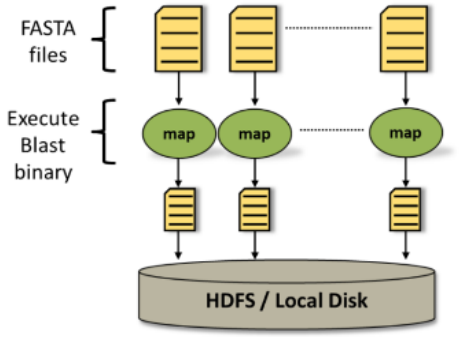
- Xác định rõ ranh giới giữa các bản ghi dựa trên ký tự ‘>’.
- Gom các dòng lại thành một chuỗi FASTA đầy đủ.
- Trả về key là tên file FASTA và value là nội dung của file.

Điều này giúp mapper xử lý đúng toàn bộ chuỗi FASTA như một đơn vị dữ liệu duy nhất, tránh việc bị cắt ngang trong khi đọc.

3.3. Luồng xử lý logic

Toàn bộ hệ thống hoạt động theo luồng dữ liệu phân tán như sau:

- Người dùng hoặc chương trình nhập dữ liệu (protein) từ các nguồn như NCBI, sau đó chuẩn hóa định dạng.
- Dữ liệu đầu vào được đưa lên HDFS với database BLAST.
- Một job MapReduce được kích hoạt:
    - Các file FASTA được cấp phát cho các mapper.
    - Các file dùng chung được tải từ Distributed Cache.
    - Tại mỗi node, tiến trình BLAST được chạy cục bộ.
- Kết quả của mỗi mapper được ghi về HDFS.



Hadoop BLAST dataflow

3.4. Tính chất hệ thống

- **Song song hóa hoàn toàn:** Toàn bộ hệ thống được thiết kế theo mô hình Map-Only, trong đó mỗi mapper hoạt động độc lập và không phụ thuộc lẫn nhau, tương ứng với mỗi file FASTA đầu vào. Điều này cho phép hệ thống thực thi đồng thời nhiều tác vụ BLAST một cách tự nhiên, không cần chia nhỏ chuỗi hoặc tổng hợp kết quả phức tạp. Nếu hệ thống có đủ tài nguyên phần cứng (CPU, RAM), Hadoop có thể khởi tạo hàng chục hoặc hàng trăm mapper song song, mỗi mapper xử lý một file .fasta.
- **Tính mở rộng:** Hệ thống có khả năng mở rộng tốt theo cả hai hướng: chiều ngang (scale-out) và chiều dọc (scale-up). Mở rộng theo chiều ngang nghĩa là có thể bổ sung thêm nhiều node (máy tính) vào cụm Hadoop, giúp tăng số lượng mapper hoạt động song song. Mở rộng theo chiều dọc nghĩa là có thể nâng cấp tài nguyên cho từng node như CPU, RAM, ổ cứng để tăng hiệu suất xử lý cho mỗi mapper. Nhờ thiết kế linh

hoạt này, hệ thống có thể dễ dàng phát triển từ môi trường thử nghiệm nhỏ trên một máy đơn đến một cụm xử lý mạnh mẽ phục vụ các bài toán sinh học quy mô lớn.

- **Độ tin cậy:** Hệ thống kế thừa khả năng chịu lỗi mạnh mẽ của Hadoop. Cụ thể, dữ liệu được lưu trữ trên HDFS với nhiều bản sao (replication), và nếu một node bị lỗi hoặc không hoạt động, Hadoop sẽ tự động lấy bản sao từ các node khác để tiếp tục xử lý. Bên cạnh đó, nếu một mapper bị gián đoạn, Hadoop có thể khởi động lại mapper đó trên node khác. Cơ chế này giúp hệ thống hoạt động ổn định và liên tục, đảm bảo quá trình xử lý dữ liệu không bị gián đoạn, ngay cả khi một số thành phần gặp sự cố.
- **Tính lặp lại:** Do được thiết kế theo mô hình pipeline xử lý dữ liệu, hệ thống có thể dễ dàng thực hiện lặp lại quá trình xử lý theo chu kỳ. Điều này có nghĩa là mỗi khi có dữ liệu protein mới, người dùng chỉ cần chạy lại pipeline với tập tin đầu vào mới mà không cần thay đổi cấu trúc chương trình. Việc này giúp hệ thống phù hợp với các mô hình phân tích định kỳ trong nghiên cứu sinh học, nơi dữ liệu được cập nhật thường xuyên. Ngoài ra, người dùng có thể viết script để tự động hóa toàn bộ quá trình, từ tải dữ liệu, chạy Hadoop BLAST đến trích xuất kết quả.

4. Triển khai hệ thống

Sau khi phân tích và thiết kế hệ thống xử lý chuỗi protein song song bằng BLAST trên nền tảng Hadoop, nhóm tiến hành triển khai giải pháp qua nhiều giai đoạn, từ chuẩn bị môi trường, xử lý dữ liệu đầu vào, đến thực thi chương trình BLAST dưới dạng một job MapReduce.

4.1. Chuẩn bị môi trường triển khai

Để triển khai hệ thống BLAST song song trên Hadoop, nhóm đã tiến hành thiết lập môi trường làm việc trên máy ảo Linux Ubuntu. Toàn bộ các thành phần của Hadoop (HDFS, YARN, MapReduce) đã được cài đặt và cấu hình sẵn trong quá trình học.

Ngoài Hadoop, nhóm cài thêm các công cụ hỗ trợ:

- **NCBI BLAST+** để chạy BLAST truyền thống và tích hợp trong Hadoop.
- **CD-HIT** để thực hiện clustering chuỗi protein.

4.2. Tải dữ liệu protein từ NCBI

Nguồn dữ liệu chính được sử dụng là NCBI Protein Database - một kho dữ liệu sinh học uy tín do Viện Y tế Quốc gia Hoa Kỳ (NIH) duy trì. Các chuỗi protein được truy vấn và tải về ở định dạng FASTA thông qua dịch vụ Entrez Programming Utilities (E-utilities).

Để đảm bảo tính hiệu quả và chính xác trong quá trình thu thập dữ liệu, nhóm sử dụng ngôn ngữ Python kết hợp với thư viện *Biopython*, cụ thể là module *Bio.Entrez*. Mỗi gen được truy vấn qua từ khóa tìm kiếm theo cú pháp "GENE_NAME[Gene]", đảm bảo chỉ lấy các bản ghi có liên quan trực tiếp tới gene đó.

Sau khi chạy script, nhóm đã thu thập thành công gần 200 gene đầu vào. Các bản ghi được lưu theo định dạng chuẩn FASTA trong thư mục fasta_nr, mỗi file tương ứng với một gene. Tổng dung lượng dữ liệu FASTA thu được là khoảng 291 MB (~400000 chuỗi protein).

4.3. Thực hiện BLAST với dữ liệu thô

Ở giai đoạn đầu tiên, nhóm sử dụng toàn bộ dữ liệu protein thu được (chưa xử lý trùng lặp, chưa phân nhóm) để thử nghiệm hai mô hình: BLAST truyền thống và Hadoop BLAST.

4.3.1. BLAST truyền thống

- Sau khi tải về các file FASTA, mỗi file chứa một chuỗi protein từ NCBI, nhóm tiến hành hợp nhất tất cả các file này thành một file FASTA lớn để tiện cho việc chạy BLAST.

```
cat *.fasta > all_proteins.fasta
```

File all_proteins.fasta sau khi hợp nhất chứa đầy đủ các file protein cần phân tích.

- Để có thể thực hiện so khớp bằng *blastp*, cần chuyển đổi tập hợp các chuỗi protein trong all_proteins.fasta thành một cơ sở dữ liệu BLAST có cấu trúc tối ưu. BLAST+ cung cấp công cụ *makeblastdb* chuyên dùng cho mục đích này.

```
makeblastdb -in all_proteins.fasta -dbtype prot -out protein_db
```

Sau khi thực thi, *makeblastdb* tạo ra một bộ file nhị phân gồm các phần: *db_proteins.pin*: file chỉ mục cho thông tin; *db_proteins.psq*: chứa dữ liệu chuỗi đã nén; *db_proteins.phr*: chứa tiêu đề chuỗi (header) ...

Những file này là cần thiết để *blastp* hoạt động hiệu quả.

- Chuẩn bị file dữ liệu FASTA cần so sánh. Một chuỗi protein mới được tạo thành file *query.fasta*. File này có cấu trúc chuẩn định dạng FASTA, có thể đến từ cơ sở dữ liệu mới, hoặc là chuỗi được người dùng cung cấp cần so khớp với cơ sở dữ liệu đã có để tìm protein tương đồng.

- Thực hiện lệnh BLAST: Công cụ *blastp* được sử dụng để so sánh chuỗi truy vấn với cơ sở dữ liệu đã xây dựng trước đó (protein_db). Nhóm lựa chọn định dạng đầu ra là *outfmt 6* (tabular), định dạng phổ biến và dễ xử lý bằng code:

```
blastp -query query.fasta -db protein_db -out result.txt -outfmt 6
```

Sau khi lệnh thực hiện xong, kết quả được lưu tại file *result.txt*. Có thể xem kết quả trực tiếp bằng lệnh

```
cat result.txt
```

Kết quả gồm 12 cột:

Cột	Tên Trường	Ý nghĩa cụ thể
1	**Query ID**	Tên chuỗi cung cấp trong file query.fasta (>my_query)
2	**Subject ID**	Tên chuỗi trong database BLAST khớp với query (ID của chuỗi đã index)
3	**% Identity**	Tỉ lệ phần trăm giống nhau giữa 2 chuỗi (cao hơn → giống hơn)
4	**Alignment Length**	Số lượng ký tự được so sánh (có alignment)
5	**Mismatches**	Số ký tự khác nhau giữa query và subject
6	**Gap Opens**	Số lần có khoảng trống (gap) trong alignment

Cột	Tên Trường	Ý nghĩa cụ thể
7	**Query Start**	Vị trí bắt đầu của chuỗi query trong đoạn so khớp
8	**Query End**	Vị trí kết thúc của chuỗi query trong đoạn so khớp
9	**Subject Start**	Vị trí bắt đầu của chuỗi subject (trong DB)
10	**Subject End**	Vị trí kết thúc của chuỗi subject
11	**E-value**	Xác suất tìm được kết quả "tình cờ" như vậy. Càng nhỏ càng tốt.
12	**Bit Score**	Điểm số phản ánh chất lượng của kết quả so khớp. Càng cao càng tốt.

4.3.2. Hadoop BLAST

Sau khi đã chuẩn bị đầy đủ dữ liệu đầu vào (file FASTA cần so khớp), cơ sở dữ liệu protein và chương trình thực thi BLAST, hệ thống được triển khai và thực thi bằng các lệnh Hadoop sau đây:

- Tạo thư mục lưu dữ liệu đầu vào trên HDFS

```
hdfs dfs -mkdir -p /input
```

Lệnh này tạo thư mục /input trên hệ thống file phân tán HDFS, nơi sẽ chứa các file FASTA cần so khớp.

- Tải dữ liệu FASTA đầu vào lên HDFS

```
hdfs dfs -put query.fasta /input/
```

File *query.fasta* là tệp chứa chuỗi protein cần được so khớp. Sau khi upload, Hadoop sẽ có thể chia tệp này và phân phối cho các mapper xử lý song song.

- Xóa thư mục đầu ra cũ (nếu có)

```
hdfs dfs -rm -r /output
```

Để tránh lỗi khi Hadoop ghi đè lên một thư mục đã tồn tại, thư mục /output được xóa trước nếu tồn tại từ lần chạy trước đó.

- Thực thi job Hadoop BLAST

```
hadoop jar blastjob.jar /input /output
C:\Projects\HadoopBLAST\protein_db\protein_db
```

Trong đó:

  - blastjob.jar: file JAR đã được build, chứa mã MapReduce.
  - /input: đường dẫn thư mục chứa file FASTA trên HDFS.
  - /output: nơi lưu kết quả đầu ra của Hadoop.
  - C:\Projects\HadoopBLAST\protein_db\protein_db: đường dẫn tuyệt đối trên máy cục bộ đến thư mục chứa cơ sở dữ liệu BLAST và binary (đã được nạp thông qua Distributed Cache).
- Tải kết quả từ HDFS về máy cục bộ

```
hdfs dfs -get /output/part-r-00000
```

Sau khi job hoàn tất, kết quả BLAST được lưu trong file *part-r-00000*. Đây là file đầu ra mặc định của Hadoop (trong trường hợp job chỉ có 1 reducer hoặc là Map-only). Lệnh trên sẽ tải file về thư mục hiện hành trong máy để phục vụ cho việc phân tích tiếp theo.

4.4. Clustering và tái tổ chức dữ liệu

Trong các bài toán phân tích chuỗi sinh học nói chung, và so khớp protein bằng BLAST nói riêng, việc xử lý trực tiếp toàn bộ tập dữ liệu đầu vào thường dẫn đến hiện tượng dư thừa do có nhiều chuỗi gần giống nhau hoặc trùng lặp về mặt sinh học. Điều này không chỉ làm tăng thời gian xử lý mà còn có thể tạo ra kết quả không cần thiết hoặc khó phân tích sau này.

Để khắc phục vấn đề đó, nhóm đã thực hiện bước clustering dữ liệu chuỗi protein, nhằm giảm thiểu sự dư thừa, tối ưu hiệu suất xử lý, và đồng thời giữ lại những chuỗi đại diện cho các nhóm tương đồng về sinh học.

Mục tiêu của clustering:

- Giảm số lượng chuỗi cần xử lý mà không làm mất thông tin sinh học quan trọng.
- Loại bỏ trùng lặp và các chuỗi có độ tương đồng cao.
- Rút ngắn thời gian thực thi BLAST (cả truyền thống và song song).
- Tăng độ rõ ràng cho kết quả đầu ra khi mỗi chuỗi đại diện phản ánh tốt hơn cho từng cụm chức năng.

Để giảm thiểu trùng lặp trong tập dữ liệu chuỗi protein và tối ưu hiệu quả xử lý trong các bước BLAST, nhóm đã sử dụng công cụ **CD-HIT** để thực hiện bước phân cụm (clustering) các chuỗi tương đồng. CD-HIT (Cluster Database at High Identity with Tolerance) là một công cụ mạnh mẽ thường được sử dụng trong tin sinh học để phân nhóm các chuỗi protein hoặc nucleotide có mức độ tương đồng cao, giúp loại bỏ dữ liệu dư thừa mà không làm mất thông tin sinh học quan trọng.

Các bước:

- Chạy CD-HIT với ngưỡng giống nhau 90% (0.9) trên tập all_proteins.fasta

```
cd-hit -i all_proteins.fasta -o clustered_proteins.fasta -c 0.9 -n
5
```

Kết quả sau khi thực hiện lệnh này là một file đầu ra *clustered_proteins.fasta* chứa tập hợp rút gọn các chuỗi protein sao cho không còn các chuỗi gần giống

nhau vượt quá ngưỡng 90% về mặt trình tự. Mỗi chuỗi trong tập tin đầu ra đại diện cho một cụm chuỗi tương đồng, giúp:

- **Giảm số lượng chuỗi đầu vào cần xử lý** trong BLAST, từ đó tiết kiệm thời gian và tài nguyên tính toán.
- **Duy trì tính đa dạng của dữ liệu**, vì các chuỗi đại diện vẫn giữ nguyên đặc điểm sinh học đặc trưng của từng nhóm.
- **Tạo thuận lợi cho việc đánh giá**, khi kết quả BLAST được thực hiện trên một tập dữ liệu rút gọn nhưng vẫn đủ đại diện.

Ngoài ra, CD-HIT cũng sinh ra một file *clustered_proteins.fasta.clstr* chứa thông tin về cách các chuỗi được phân cụm, có thể dùng để truy ngược chuỗi gốc nào thuộc về cụm nào trong quá trình phân tích kết quả.

4.5. Chạy BLAST với tập dữ liệu đã thực hiện clustering

Sau khi áp dụng bước clustering chuỗi protein bằng công cụ **CD-HIT**, nhóm đã thu được một tập dữ liệu mới có kích thước nhỏ hơn, loại bỏ được các chuỗi trùng lặp hoặc có mức độ tương đồng cao (> 90%). Việc chạy lại BLAST với tập dữ liệu đã được phân cụm này giúp đánh giá ảnh hưởng của bước clustering đến hiệu năng xử lý và kết quả sinh học.

Toàn bộ các bước triển khai hệ thống, bao gồm: tạo database, upload lên HDFS, thực thi job Hadoop, và lấy kết quả đầu ra... vẫn được giữ nguyên như khi xử lý dữ liệu thô. Điểm duy nhất thay đổi là tập dữ liệu đầu vào: thay vì sử dụng file all_proteins.fasta chứa chuỗi gốc, nhóm sử dụng file clustered_proteins.fasta đã được thực hiện cluster.

Tổng kết lại, quá trình triển khai hệ thống đã cho phép nhóm hiện thực hóa toàn bộ pipeline xử lý dữ liệu protein từ khâu thu thập, tiền xử lý, phân cụm, cho đến thực thi BLAST bằng cả mô hình truyền thống và mô hình phân tán trên Hadoop. Việc thực hiện thử nghiệm trên cả hai bộ dữ liệu trước và sau khi clustering giúp nhóm không chỉ kiểm chứng hiệu quả của song song hóa mà còn đánh giá được tác động rõ rệt của việc giảm trùng lặp dữ liệu. Kết quả triển khai thành công này là cơ sở vững chắc để nhóm tiến hành các phân tích hiệu năng sâu hơn trong chương tiếp theo, cũng như mở rộng hệ thống trong các ứng dụng thực tế quy mô lớn.

5. Thử nghiệm và đánh giá

5.1. Mục tiêu đánh giá

Sau khi triển khai thành công hệ thống thực thi BLAST song song trên nền tảng Hadoop, nhóm tiến hành đánh giá hiệu quả của hệ thống thông qua các thí nghiệm thực tế. Mục tiêu của quá trình đánh giá là:

- So sánh hiệu năng giữa BLAST truyền thống và Hadoop BLAST.
- Đo lường thời gian thực thi tổng thể trong các tình huống sử dụng khác nhau.
- Phân tích tác động của song song hóa khi thay đổi kích thước tập truy vấn đầu vào.
- Đánh giá hiệu quả của clustering (phân cụm dữ liệu) đối với tốc độ xử lý và khả năng mở rộng hệ thống.

5.2. Thiết lập thí nghiệm

Để có cái nhìn toàn diện về hiệu năng hệ thống, nhóm thực hiện các thử nghiệm trên hai bộ dữ liệu đầu vào với kích thước khác nhau:

- Tập truy vấn nhỏ: khoảng 120 chuỗi gen, dung lượng 87 KB.
- Tập truy vấn lớn: khoảng 480 chuỗi gen, dung lượng 351 KB.

Cả hai tập được chạy trong hai điều kiện:

- Sử dụng cơ sở dữ liệu gốc chưa clustering.
- Sử dụng cơ sở dữ liệu đã thực hiện clustering.

Việc sử dụng hai quy mô dữ liệu truy vấn khác nhau giúp nhóm đánh giá khả năng mở rộng theo chiều ngang (scale-out) của hệ thống Hadoop BLAST.

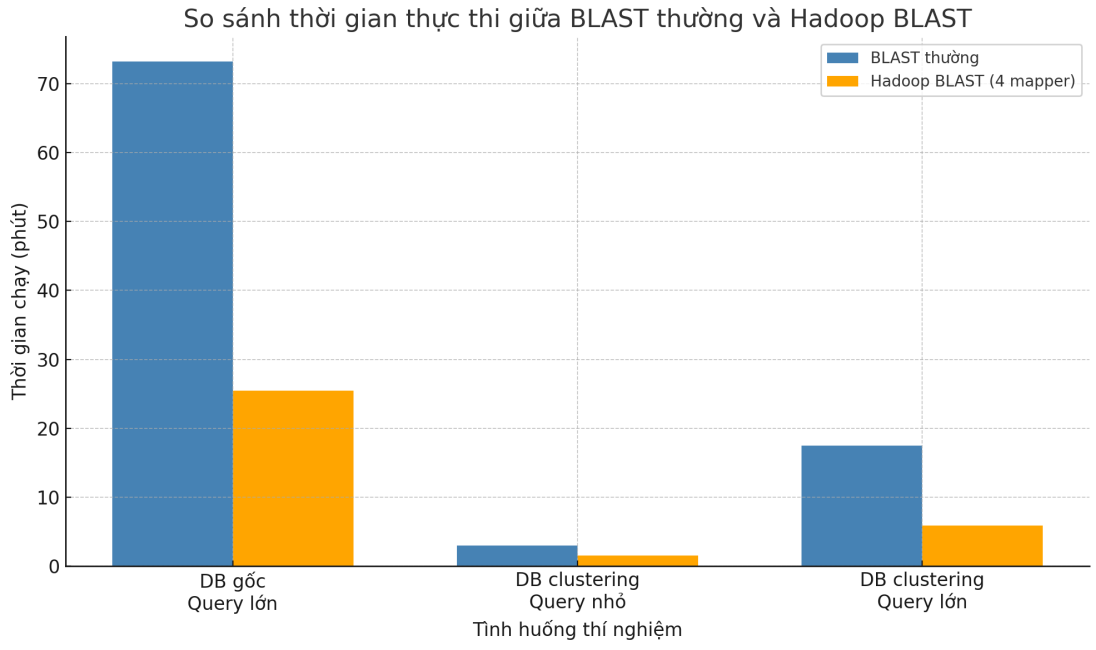
Cả hai mô hình xử lý đều được thực thi trên cùng một máy, đảm bảo môi trường thử nghiệm công bằng. Số mapper mặc định của Hadoop BLAST là 4, tương đương với khả năng chạy 4 tiến trình BLAST đồng thời.

5.3. Kết quả thí nghiệm

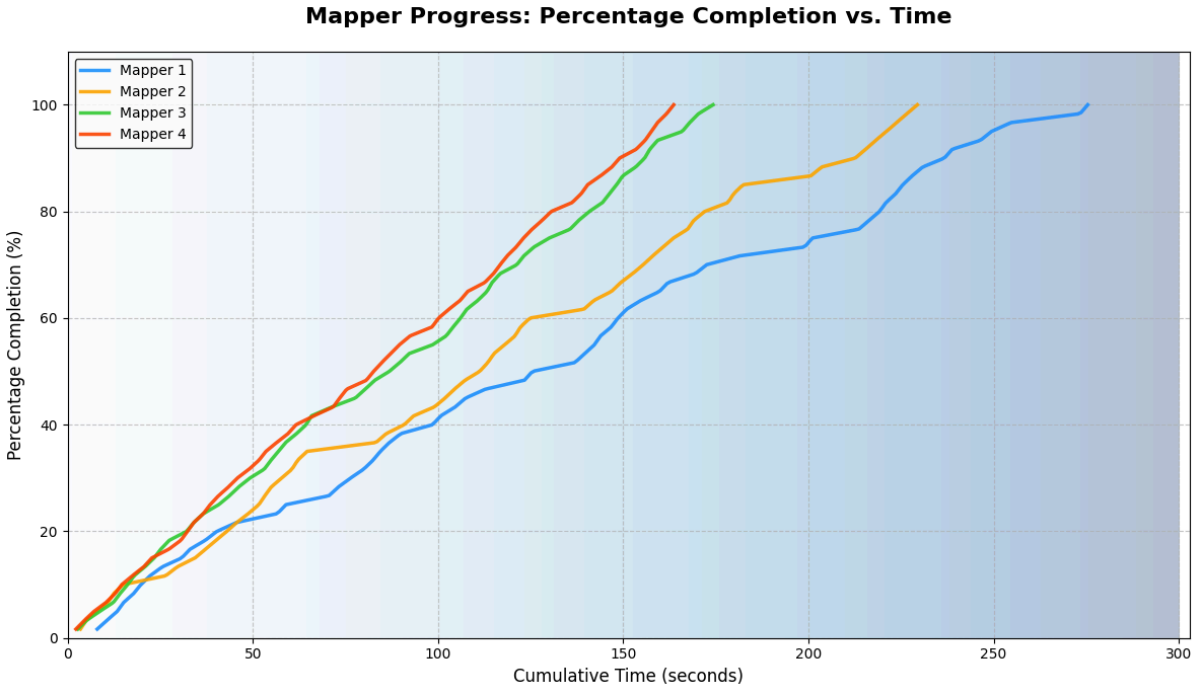
STT	Cơ sở dữ liệu	Tập truy vấn	Số chuỗi	Dung lượng	BLAST thường	Hadoop BLAST
1	Gốc (chưa clustering)	Lớn	480	351 KB	73.21 phút	25.47 phút
2	Đã clustering	Nhỏ	120	87 KB	3.00 phút	1.55 phút
3	Đã clustering	Lớn	480	351 KB	17.48 phút	5.88 phút

5.4. Phân tích kết quả

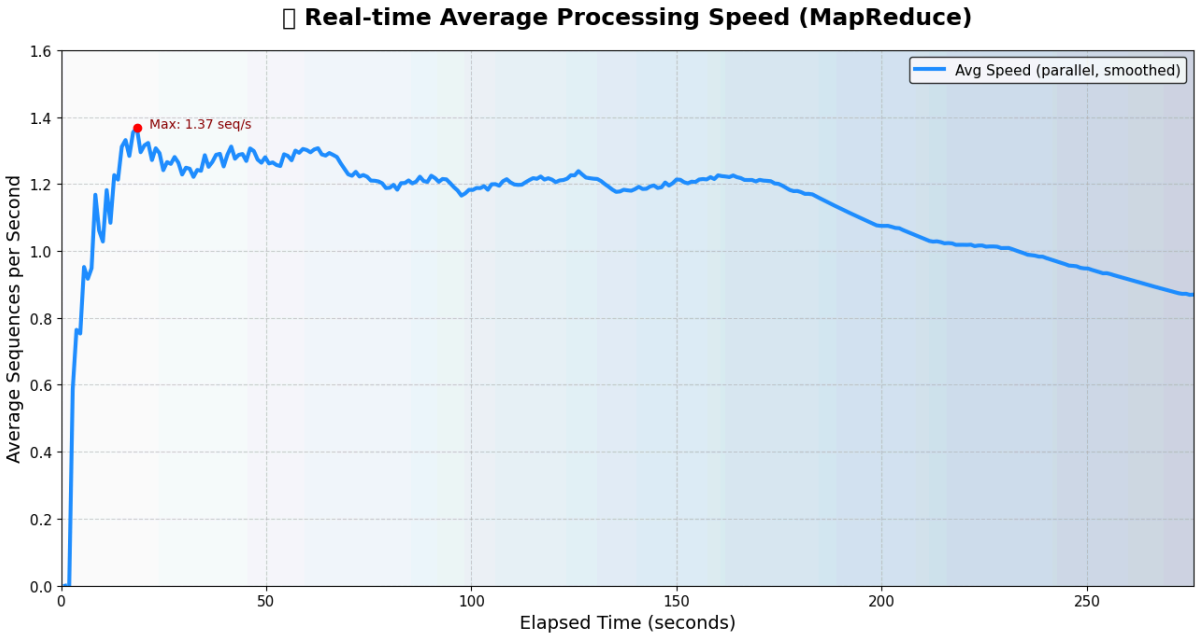
Biểu diễn trực quan kết quả thí nghiệm



Tiến trình thực hiện của các Mapper, thể hiện mức độ hoàn thành theo thời gian thực. Có thể thấy các Mapper có sự chênh lệch về tốc độ thực hiện, điều này có thể là do loại gen khác nhau được phân chia cho các mapper.



Tốc độ xử lý trung bình khi song song hóa 4 mapper theo thời gian thực.



Có thể thấy tốc độ cao nhất là 1.37 chuỗi trên 1 giây và có xu hướng giảm dần khi kết thúc. Lý do là vì có các Mapper hoàn thành công việc trước nên sẽ giảm tốc độ tổng thể khi sắp kết thúc.

Hiệu quả của song song hóa kết hợp Clustering

Kết quả cho thấy, việc thực thi BLAST song song bằng Hadoop mang lại hiệu quả cải thiện hiệu suất rõ rệt, đặc biệt khi kích thước dữ liệu truy vấn tăng lên. Trong trường hợp tập query lớn, thời gian xử lý giảm gần 3 lần, phản ánh đúng bản chất của bài toán: mỗi chuỗi có thể được xử lý độc lập, do đó phù hợp hoàn hảo với mô hình MapReduce.

Ở tập dữ liệu nhỏ hơn (120 chuỗi), tốc độ cải thiện chỉ khoảng 2 lần. Điều này dễ hiểu vì:

- Tổng thời gian xử lý vốn đã ngắn, nên chi phí khởi tạo mapper và overhead của Hadoop chiếm tỷ lệ lớn hơn.
- Với dữ liệu nhỏ, hiệu quả song song hóa chưa được tận dụng tối đa.

Tuy nhiên, khi mở rộng tập truy vấn lên 480 chuỗi, Hadoop BLAST càng phát huy ưu thế: số lượng file tăng → số lượng mapper tăng tương ứng → thời gian xử lý mỗi mapper vẫn giữ ổn định → thời gian tổng thể giảm rõ rệt.

Tác động của clustering

Clustering giúp giảm trùng lặp và độ tương đồng giữa các chuỗi, nhờ đó:

- Rút gọn cơ sở dữ liệu BLAST: Chỉ giữ lại các chuỗi đại diện trong mỗi cụm, làm giảm đáng kể kích thước của database cần tra cứu.
- Giảm số phép so khớp cần thực hiện: Với cơ sở dữ liệu nhỏ hơn, mỗi truy vấn yêu cầu ít so khớp hơn, từ đó thời gian xử lý cho mỗi chuỗi cũng giảm.
- Rút ngắn thời gian chạy ở cả hai mô hình BLAST.

Sự hiệu quả của clustering thể hiện rất rõ trong trường hợp xử lý cùng một tập truy vấn lớn (480 chuỗi):

- Khi sử dụng cơ sở dữ liệu gốc: Hadoop BLAST mất khoảng 25.47 phút để hoàn thành.
- Khi sử dụng cơ sở dữ liệu đã clustering: chỉ mất khoảng 5.88 phút.

Tốc độ xử lý tăng gần 4.3 lần so với khi dùng database gốc. Điều này cho thấy rằng, clustering không chỉ là bước tiền xử lý bổ trợ, mà là một phần cốt lõi trong chiến lược tối ưu hiệu suất cho cả hệ thống BLAST tuần tự lẫn hệ thống song song hóa.

Hiệu quả tổng hợp khi kết hợp song song hóa và clustering

Sự kết hợp giữa song song hóa với Hadoop và clustering dữ liệu đầu vào chính là yếu tố quyết định giúp hệ thống đạt hiệu suất vượt trội. Trong trường hợp tốt nhất, tổng thời gian thực thi giảm từ hơn 73 phút (BLAST thường với dữ liệu gốc) xuống chỉ còn 5.88 phút (Hadoop BLAST với dữ liệu đã clustering), tức là nhanh hơn hơn 12 lần.

Tóm lại, đây không chỉ là sự cải thiện tốc độ đơn thuần, mà còn là chiến lược xử lý dữ liệu hiệu quả và có khả năng mở rộng cao, đặc biệt hữu ích trong bối cảnh các bài toán sinh học hiện đại đang phải đối mặt với khối lượng dữ liệu ngày càng khổng lồ.

5.5. Đánh giá tổng quan

- **Tốc độ cải thiện tỷ lệ thuận với quy mô đầu vào:** Một trong những quan sát rõ nét nhất từ quá trình thử nghiệm là hiệu quả song song hóa của Hadoop BLAST tỏ ra tỷ lệ thuận với kích thước của tập dữ liệu đầu vào. Khi số lượng chuỗi gen cần xử lý tăng lên, tổng thời gian thực thi của hệ thống BLAST truyền thống tăng lên một cách tuyến tính, do toàn bộ khối lượng công việc được xử lý theo cách tuần tự. Ngược lại, Hadoop BLAST tận dụng được kiến trúc MapReduce để phân tán công việc đến nhiều mapper, mỗi mapper xử lý một phần dữ liệu độc lập, nhờ đó giảm tải xử lý tại mỗi node và tiết kiệm tổng thời gian thực thi.
- **Clustering có vai trò quan trọng không kém việc song song hóa:** Nó giúp giảm số lượng chuỗi cần so khớp trong database, giảm số phép tính trùng lặp trong quá trình BLAST, giảm số lần truy xuất I/O trong hệ thống, rút gọn không gian tìm kiếm, đồng thời giữ lại đại diện sinh học chính xác cho từng nhóm protein tương đồng.
- **Sự kết hợp giữa clustering + Hadoop BLAST** là hướng tối ưu nhất: Qua phân tích, có thể thấy rằng mỗi phương pháp – clustering và song song hóa – đều mang lại hiệu quả nhất định. Tuy nhiên, chính sự kết hợp hai kỹ thuật này mới tạo nên giải pháp xử lý dữ liệu protein thực sự hiệu quả và có tính mở rộng, vừa đảm bảo được tốc độ, độ chính xác, vừa mở ra khả năng mở rộng lên hàng nghìn hoặc hàng triệu chuỗi sinh học trong các dự án thực tế.

6. Cải tiến và hướng mở rộng

Sau khi hoàn thiện hệ thống BLAST song song trên nền tảng Hadoop và đánh giá hiệu năng thông qua các thực nghiệm thực tế, nhóm nhận thấy vẫn còn nhiều không gian để cải tiến cả về hiệu suất, khả năng mở rộng, tính linh hoạt và khả năng tích hợp của hệ thống. Những cải tiến và hướng mở rộng sau đây được đề xuất để tiếp tục phát triển hệ thống trở nên mạnh mẽ hơn, phù hợp hơn với các yêu cầu xử lý dữ liệu sinh học quy mô lớn trong tương lai.

6.1. Mở rộng quy mô xử lý

- **Chạy trên nhiều máy thay vì một máy**

Trong quá trình triển khai, hệ thống Hadoop của nhóm được cấu hình ở chế độ mô phỏng (pseudo-distributed), tức là mọi thành phần đều chạy trên cùng một máy ảo. Mô hình này đơn giản, dễ thử nghiệm, nhưng không tận dụng được hết khả năng mở rộng thực sự của Hadoop. Khi số lượng chuỗi gen cần xử lý tăng lên hàng ngàn hoặc hàng chục ngàn chuỗi, hệ thống hiện tại sẽ dần chậm lại vì toàn bộ công việc vẫn xử lý trên một máy duy nhất.

Giải pháp mở rộng: Trong tương lai, nhóm có thể xây dựng một cụm Hadoop thực sự gồm nhiều máy (multi-node cluster). Trong đó, mỗi máy trong mạng LAN có thể đóng vai trò là một node để thực thi mapper song song. Ngoài ra, nếu không có điều kiện xây dựng cụm vật lý, nhóm cũng có thể sử dụng các dịch vụ điện toán đám mây để triển khai Hadoop trên nền tảng cloud. Các dịch vụ này cho phép cấu hình cụm Hadoop nhanh chóng, dễ dàng tăng số lượng node xử lý chỉ bằng vài cú click chuột.

- **Tạo giao diện dễ dùng**

Hiện tại, hệ thống chủ yếu được vận hành bằng dòng lệnh. Điều này tuy phù hợp với người có kinh nghiệm lập trình, nhưng lại là rào cản đối với người dùng phổ thông hoặc các nhà nghiên cứu sinh học không chuyên về công nghệ thông tin. Nếu muốn hệ thống thực sự được ứng dụng trong thực tế, đặc biệt là trong các phòng thí nghiệm sinh học, cần phải có một giao diện đơn giản, trực quan hơn.

Giải pháp đề xuất: Nhóm có thể phát triển thêm một giao diện web đơn giản giúp người dùng tải file FASTA lên và nhận kết quả BLAST nhanh chóng, thay

vì phải dùng dòng lệnh. Việc bổ sung giao diện không chỉ nâng cao trải nghiệm người dùng mà còn mở ra khả năng tích hợp hệ thống với các công cụ phân tích dữ liệu khác trong tương lai.

6.2. Nâng cấp công cụ xử lý

- Sử dụng Spark thay cho Hadoop MapReduce

Hadoop MapReduce là nền tảng xử lý phân tán nổi tiếng, nhưng nó cũng có một số điểm hạn chế, đặc biệt là tốc độ xử lý bị ảnh hưởng do phải ghi và đọc dữ liệu từ ổ đĩa giữa các bước. Điều này gây ra độ trễ cao khi xử lý các khối lượng dữ liệu lớn hoặc khi cần phản hồi nhanh.

Giải pháp đề xuất: Trong các phiên bản tương lai, hệ thống có thể được chuyển sang xử lý BLAST song song bằng Apache Spark, hoặc xây dựng pipeline xử lý sinh học sử dụng Spark làm nền tảng.

- Kết nối với các công cụ sinh học khác

Hiện tại hệ thống chỉ tập trung vào việc chạy BLAST để tìm chuỗi tương đồng, trong khi thực tế phân tích sinh học cần nhiều bước xử lý tiếp theo. Do đó, một hướng mở rộng hợp lý là kết nối hệ thống với các công cụ phân tích sinh học khác, hoặc sử dụng mô hình học máy để phân loại protein, phát hiện các đột biến bất thường, hoặc phân tích mối liên hệ giữa các chuỗi gen. Việc tích hợp này sẽ giúp mở rộng hệ thống từ một công cụ BLAST đơn lẻ thành một pipeline phân tích sinh học đầy đủ, hỗ trợ các nghiên cứu sâu hơn trong các lĩnh vực như y học chính xác, di truyền học và công nghệ sinh học.

7. Kết luận

Trong bối cảnh khoa học sự sống ngày càng tiến bộ và dữ liệu sinh học không ngừng tăng trưởng, nhu cầu về các giải pháp tính toán hiệu suất cao để xử lý các bài toán phân tích chuỗi sinh học trở nên cấp thiết hơn bao giờ hết. Dự án "Parallel BLAST on Hadoop for Cell Lines Analysis" đã được thực hiện nhằm đáp ứng nhu cầu đó, với mục tiêu triển khai một hệ thống giúp tăng tốc quá trình so khớp chuỗi sinh học thông qua việc tận dụng sức mạnh xử lý phân tán của nền tảng Hadoop.

Sau quá trình phân tích, thiết kế và triển khai, nhóm đã xây dựng thành công một phiên bản song song của công cụ BLAST, trong đó các file chuỗi sinh học đầu vào được chia nhỏ và phân phối tới các tác vụ mapper chạy song song trên các node của cụm Hadoop. Nhờ cơ chế này, thời gian thực thi các tác vụ BLAST đã được rút ngắn đáng kể so với phương pháp truyền thống vốn chỉ chạy trên một máy đơn. Hệ thống đã vận hành ổn định, các kết quả đầu ra đảm bảo tính chính xác, phản ánh đúng nội dung so khớp chuỗi của công cụ BLAST nguyên gốc.

Điểm nổi bật trong giải pháp của nhóm là việc áp dụng cơ chế Distributed Cache của Hadoop để phân phối các thành phần cần thiết như chương trình BLAST và cơ sở dữ liệu tới các node trong cụm, giúp hệ thống hoạt động hiệu quả mà không phát sinh chi phí truyền dữ liệu dư thừa. Cách tiếp cận sử dụng mô hình Map-Only trong MapReduce cũng đã chứng minh được tính phù hợp với bài toán: mỗi file FASTA đầu vào được xử lý một cách độc lập, không yêu cầu giai đoạn Reduce, nhờ đó hệ thống tận dụng được tối đa khả năng song song hóa tự nhiên của Hadoop.

Thông qua quá trình triển khai dự án, nhóm không chỉ hoàn thành mục tiêu kỹ thuật mà còn thu nhận được nhiều bài học kinh nghiệm quý báu. Trước hết, nhóm đã có cơ hội tiếp cận sâu hơn với các kiến trúc xử lý dữ liệu lớn như Hadoop, hiểu rõ cách thức vận hành các thành phần HDFS, MapReduce và Distributed Cache. Các thành viên cũng có cơ hội nâng cao kỹ năng lập trình xử lý dữ liệu sinh học trong môi trường phân tán, từ việc tùy chỉnh Input Format cho tới việc gọi các tiến trình BLAST bên ngoài trong từng mapper. Quan trọng hơn cả, dự án đã giúp cả nhóm ý thức rõ hơn về thách thức và cơ hội khi đưa các thuật toán truyền thống như BLAST vào các môi trường điện toán song song hiện đại.

Dự án không chỉ có ý nghĩa về mặt học thuật mà còn mang lại những giá trị thực tiễn rõ rệt. Trong các bài toán nghiên cứu về dòng tế bào, việc so khớp chuỗi sinh học là một bước quan trọng giúp các nhà khoa học phát hiện các biến thể di truyền, xác định các gen đích, hoặc đánh giá mức độ tương đồng giữa các mẫu sinh học. Việc tăng tốc quá trình này không chỉ giúp rút ngắn thời gian nghiên cứu mà còn góp phần nâng cao hiệu suất toàn bộ quy trình phân tích sinh học, từ đó thúc đẩy nhanh hơn các bước tiến trong các lĩnh vực như y học chính xác, công nghệ sinh học và nghiên cứu gen.

Nhìn chung, kết quả của dự án đã chứng minh rằng các công nghệ xử lý dữ liệu lớn như Hadoop hoàn toàn có thể được ứng dụng hiệu quả trong lĩnh vực sinh học, mở ra hướng đi mới cho việc giải quyết các bài toán sinh học quy mô lớn. Giải pháp mà nhóm đề xuất vừa đảm bảo khả năng mở rộng linh hoạt theo số lượng node trong cụm, vừa duy trì tính chính xác, đồng thời có thể tiếp tục được nâng cấp trong tương lai để đáp ứng các yêu cầu phức tạp hơn.

8. Tài liệu tham khảo

N Khare, A Khare, F Khan: *HCudaBLAST: an implementation of BLAST on Hadoop and Cuda (2017)*

M Meng, J Gao, J Chen: *Blast-Parallel: The parallelizing implementation of sequence alignment algorithms based on Hadoop platform (2013)*

Aisling O'Driscoll, Vladislav Belogrudov, John Carroll, Kai Kropp, Paul Walsh, Peter Ghazal, Roy D. Sleator: *HBLAST: Parallelised sequence similarity – A Hadoop MapReducable basic local alignment search tool (2015)*

M Gaikwad, S Ahirrao: *BLAST using big data technologies: a survey (2018)*

T Khawla, M Fatiha, Z Azeddine, N Said: *A Blast implementation in Hadoop MapReduce using low cost commodity hardware (2018)*

JJ Alnasir, HP Shanahan: *The application of hadoop in structural bioinformatics (2020)*

Ramanti Dharayani, Wahyu Catur Wibowo, Yova Ruldeviyani, Arfive Gandhi: *Genomic Anomaly Searching with BLAST Algorithm using MapReduce Framework in Big Data Platform (2019)*

X Yang, Y Liu, C Yuan, Y Huang: *Parallelization of BLAST with MapReduce for long sequence alignment (2011)*

C Romeu Farré: *Design and implementation BLAST tool big data (2019)*

AR Kumar, HP Singh, GA Kumar: *A Review on Design and Development of Performance Evaluation Model for Bio-Informatics Data Using Hadoop (2021)*

L Zhang, B Tang: *Parka: a parallel implementation of BLAST with MapReduce (2018)*