Finding the similarity of biology sequences using BLAST on MapReduce Framework

Conference Paper · June 2013

DOI: 10.13140/2.1.2645.2800

4 authors, including:

Lang Van Tran
Vietnam Academy of Science and Technology
95 PUBLICATIONS **169** CITATIONS
SEE PROFILE

Nguyen Gia Khoa
Ho Chi Minh City Technical and Economic College, Ho Chi Minh City, Vietnam
4 PUBLICATIONS **0** CITATIONS
SEE PROFILE

TÌM MỨC ĐỘ TƯƠNG ĐỒNG CỦA CÁC TRÌNH TỰ SINH HỌC BẰNG THUẬT TOÁN BLAST TRÊN MÔ HÌNH MAPREDUCE

Đoàn Danh Đạt[1], Trần Văn Lăng[2], Trần Nguyễn Minh Hiếu[3], Nguyễn Gia Khoa[4]

[1]Học viện Công nghệ Bưu chính Viễn thông
[2]Viện Cơ học và Tin học ứng dụng, Viện Hàn lâm Khoa học và Công nghệ Việt Nam
[3]Trường Đại học Sài Gòn
[4]Cao đẳng kinh tế - kỹ thuật Phú Lâm
doandanhdat852004@gmail.com, langtv@vast.ac.vn, minhhieu2687@gmail.com, nguyengiakhoa@yahoo.com

TÓM TẮT—*Bài báo trình bày một số vấn đề về mô hình MapReduce, đây là mô hình lập trình cho phép xử lý và tạo ra lượng dữ liệu rất lớn để triển khai trên hệ thống phân tán. Mô hình này lấy ý tưởng từ hàm Map và Reduce trong các ngôn ngữ lập trình hướng hàm, qua đó định nghĩa một hàm ánh xạ để tạo ra các cặp khóa/giá trị từ dữ liệu vào, rồi định nghĩa một hàm quy hồi để kết hợp các giá trị có cùng khóa với nhau tạo thành dữ liệu xuất. Mô hình tính toán này được sử dụng trong các hệ thống xử lý và phân tích trên khối lượng rất lớn dữ liệu không cấu trúc, được lưu trữ phân tán trên nhiều cụm máy chủ. Bên cạnh đó, bài báo cũng trình bày cách thức xây dựng ứng dụng BLAST dùng mô hình MapReduce sử dụng Hadoop Framework, từ đó đánh giá tốc độ thực thi của ứng dụng Blast trên mô hình MapReduce.*

Từ khóa— *Tính toán song song, dữ liệu phân tán, thuật toán blast.*

I. GIỚI THIỆU

Sự phát triển nhanh chóng của công nghệ thông tin, mạng internet, mạng xã hội tạo ra sự bùng nổ về dữ liệu. Từ đó đòi hỏi phải có một mô hình mới để xử lý; mô hình MapReduce ra đời đã đưa ra một lời giải tốt (cho đến thời điểm hiện tại) cho bài toán trên. MapReduce là một "mô hình lập trình" (programming model), lần đầu báo cáo trong bài báo của Jefferey Dean và Sanjay Ghemawat ở Hội nghị OSDI 2004. Lấy ý tưởng từ lập trình hàm; MapReduce có hai tác vụ cơ bản là Map và Reduce. Đầu tiên, theo mô hình MapReduce, tác vụ "map" nhằm ánh xạ các dữ liệu vào thành một tập hợp các cặp "khóa/giá trị" rồi sau đó sử dụng tác vụ "reduce" để tập hợp các giá trị có khóa giống nhau lại.

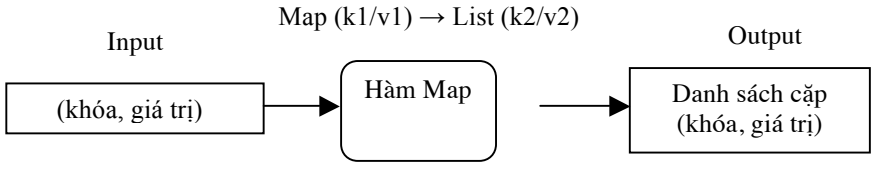
Trên thế giới, mô hình MapReduce đã và đang được quan tâm, ứng dụng hết sức rộng rãi như: Google, Yahoo, Facebook, MySpace đang sử dụng để xử lý dữ liệu; Amazon dùng Hadoop để cung cấp MapReduce như dịch vụ web; tháng 4 năm 2009 Yahoo đã dùng Hadoop để sắp xếp 100 terabyte dữ liệu trong vòng 173 phút trên hệ thống với 3400 node. Tại Việt Nam, MapReduce vẫn là công nghệ mới tuy nhiên đã có những nghiên cứu bước đầu như nhóm Phan Đắc Anh Huy, Lê Hoài Nam, Trần Văn Lăng sử dụng Hadoop/MapReduce trong việc thu thập nội dung mailling list và thống kê hồ sơ sinh viên (năm 2009); xây dựng ứng dụng tìm kiếm và sắp xếp trên một hệ thống cơ sở dữ liệu phân tán (hệ thống cơ sở dữ liệu khách hàng của công ty Cổ phần Viễn thông Tin học Bưu điện. Vinagame cũng đã triển khai Hadoop để xử lí dữ liệu dưới dạng thử nghiệm. Trong sinh tin học, thuật toán BLAST đang được dùng để so sánh các trình tự sinh học, tìm ra những trình tự sinh học tương đồng có trong các CSDL lớn như NCBI, EMBL, DDBJ. Khoa Toán – Tin của Trường Đại học Khoa học tự nhiên TP. HCM đã dùng BLAST để xây dựng hệ thống iGridPortal. Phân viện Công nghệ thông tin tại TPHCM (nay là Viện Cơ học và Tin học ứng dụng) đã có những dự án liên quan đến việc áp dụng BLAST trong việc xây dựng phần mềm HiBio để hỗ trợ việc phân tích gene và protein.

Với thực trạng triển khai mô hình MapReduce trên thế giới và những ưu điểm không thể phủ nhận của nó, yêu cầu cần phải thực hiện những nghiên cứu sâu hơn, xây dựng ứng dụng cụ thể hơn về mô hình MapReduce là cần thiết. Bài viết với tên gọi "Tìm mức độ tương đồng của các trình tự sinh học bằng thuật toán BLAST trên mô hình MapReduce" được trình bày trong 4 phần. Nội dung phần thứ nhất như trình bày, phần thứ II đưa ra phương pháp tiếp cận; phần thứ III là những thử nghiệm và đánh giá. Một số kết luận được trình bày trong phần cuối cùng.

II. PHƯƠNG PHÁP

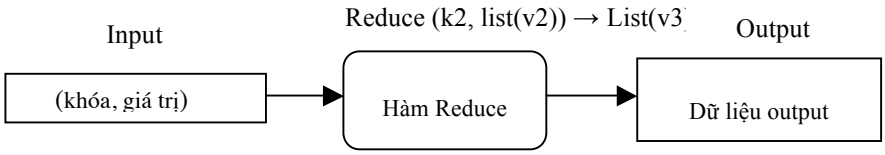
A. Mô hình MapReduce

Lấy ý tưởng từ lập trình hàm, mô hình MapReduce dựa trên hai hàm Map và Reduce, đó là lý do tại sao mô hình lập trình này được gọi là MapReduce, đây là hai hàm phổ dụng trong các ngôn ngữ lập trình hàm như Lisp. Để xử lý khối dữ liệu bao gồm rất nhiều cặp (khóa/giá trị), lập trình viên viết hai hàm Map và Reduce. Hàm Map có đầu vào là một cặp (k1/v1) và đầu ra là một danh sách các cặp (k2/v2), như vậy hàm Map có thể được viết một cách hình thức như sau [1, 3]:

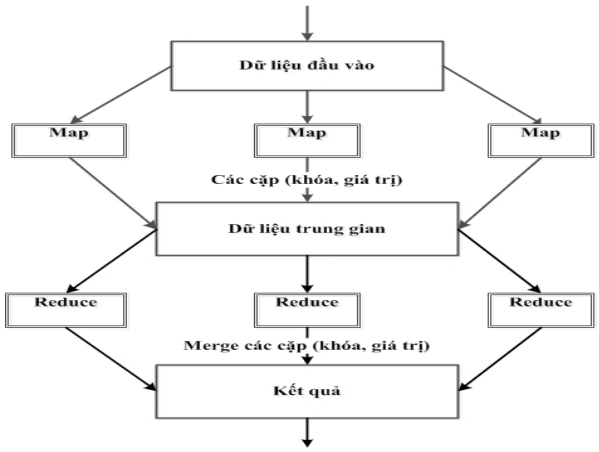


Mô hình MapReduce áp dụng hàm Map (do người dùng viết) vào từng cặp (khóa/giá trị) trong khối dữ liệu vào, chạy rất nhiều phiên bản của hàm Map song song với nhau trên các node của cluster. Sau khi giai đoạn này kết thúc, kết quả thu được là một tập hợp rất nhiều cặp (khóa/giá trị) gọi là các cặp (khóa/giá trị) trung gian. Các cặp này lại được nhóm một lần nữa theo khóa, như vậy các cặp (khóa/giá trị) trung gian có cùng khóa sẽ nằm cùng một nhóm trung gian.

Sau đó, hàm Reduce (cũng do người dùng viết) sẽ được áp dụng vào từng nhóm trung gian để tạo thành kết quả cuối cùng tùy theo yêu cầu đặt ra. Thư viện MapReduce tiếp nhận các kết quả này và xuất ra kết quả cuối cùng dưới các dạng dữ liệu khác nhau. Một cách hình thức, hàm này có thể được mô tả như sau [1, 3]:



Trong đó k2 là khóa chung của nhóm trung gian, list(v2) là tập các giá trị trong nhóm và list(v3) là một danh sách các giá trị trả về của hàm Reduce thuộc kiểu dữ liệu v3. Do hàm Reduce được áp dụng vào nhiều nhóm trung gian độc lập, chúng lại một lần nữa có thể được chạy song song với nhau.



Hình 1. Hình ảnh đơn giản hóa của quá trình xử lý MapReduce

B. HADOOP Framework

Apache Hadoop là một framework dùng để chạy những ứng dụng trên một cluster lớn được xây dựng trên những phần cứng thông thường. Hadoop hiện thực mô hình MapReduce, đây là mô hình mà ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn khác nhau, và các phần này sẽ được chạy song song trên nhiều node khác nhau. Thêm vào đó, Hadoop cung cấp 1 hệ thống file phân tán (HDFS) cho phép lưu trữ dữ liệu lên trên nhiều node. Cả MapReduce và HDFS đều được thiết kế sao cho framework sẽ tự động quản lý được các lỗi, các hư hỏng về phần cứng của các node.

1. Hệ thống tập tin phân tán của Hadoop Framework – HDFS

Dựa trên ý tưởng của mô hình MapReduce, HDFS được thiết kế để lưu trữ một khối lượng dữ liệu rất lớn (nhiều terabyte hay petabyte) trên một hệ thống gồm nhiều máy nên HDFS có những ưu điểm sau:

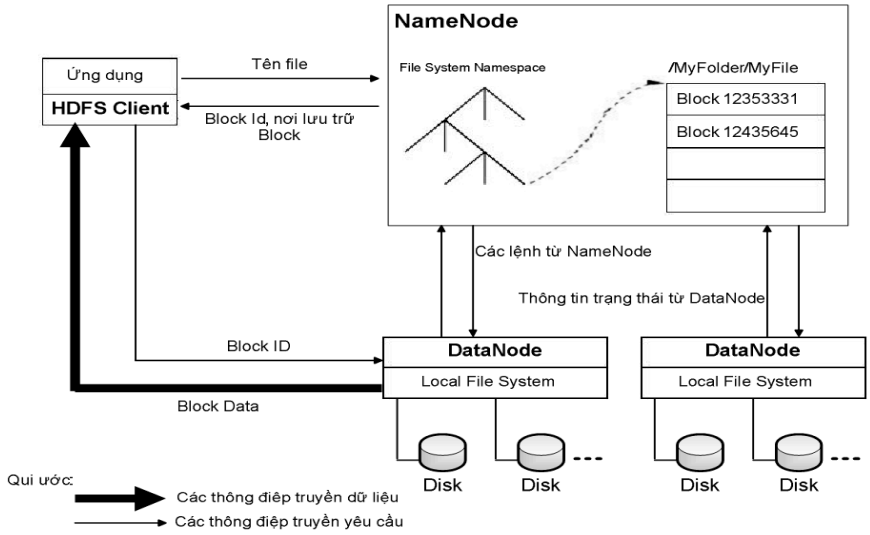
- Việc truyền dữ liệu trên mạng được tối ưu hóa.
- Hỗ trợ những tập tin có kích thước lớn hơn nhiều so với các hệ thống lưu trữ thông thường.
- Lưu trữ dữ liệu có sao lưu do đó HDFS cung cấp khả năng phục hồi truy cập rất nhanh.
- Cung cấp khả năng tích hợp cao nhất với mô hình MapReduce cho phép dữ liệu được xử lý nội bộ trong quá trình tính toán với khả năng cao nhất.

Các ứng dụng cụ thể sử dụng Hadoop hầu hết đều sử dụng HDFS cho tác vụ đọc dữ liệu tuần tự, do đó HDFS được tối ưu hóa hiệu năng đọc theo luồng dữ liệu, hơn nữa phần lớn dữ liệu được ghi vào HDFS một lần và đọc nhiều lần nên HDFS không hỗ trợ nhiều việc cập nhật tập tin (các phiên bản trước Hadoop 0.19 đều không hỗ trợ) [5].

Thiết kế dựa trên hệ thống tập tin phân tán GFS của Google, HDFS cũng chia tập tin thành các khối có khích thước cố định (mặc định là 64 MB) [4]. Các khối này được lưu trữ trong một hệ thống gồm một (ít phổ biến) hay nhiều máy gọi là cluster, các máy trong cluster gọi là DataNode. Một tập tin được chia thành nhiều khối và các khối này không nhất thiết phải cùng nằm trên một máy, thông thường để đảm bảo tránh mất mát dữ liệu các khối được sao lưu và đặt trên nhiều máy một cách ngẫu nhiên, do đó để truy cập vào một tập tin có thể phải cần sự phối hợp của nhiều máy.

Hệ thống tập tin phân tán HDFS có cấu trúc kiểu master/slave, một HDFS cluster bao gồm một NameNode (master server) duy nhất làm nhiệm vụ quản lí hệ thống tập tin Namespace và quyền truy cập các tập tin, và các DataNode (slave

server) làm nhiệm vụ lưu trữ dữ liệu. Một tập tin được chia thành nhiều khối và được lưu trữ tại các DataNode, để thực hiện các thao tác trên tập tin như đóng, mở, đổi tên tập tin, … NameNode sẽ dựa vào các thông tin trên Namespace để xác định vị trí các khối và gửi yêu cầu đọc/ghi dữ liệu tương ứng đến các DataNode. Các Datanode cũng chịu trách nhiệm thực hiện các thao tác như tạo khối, xóa hay nhân bản các khối theo yêu cầu của NameNode [2, 5].



Hình 2. NameNode và DataNode trong HDFS

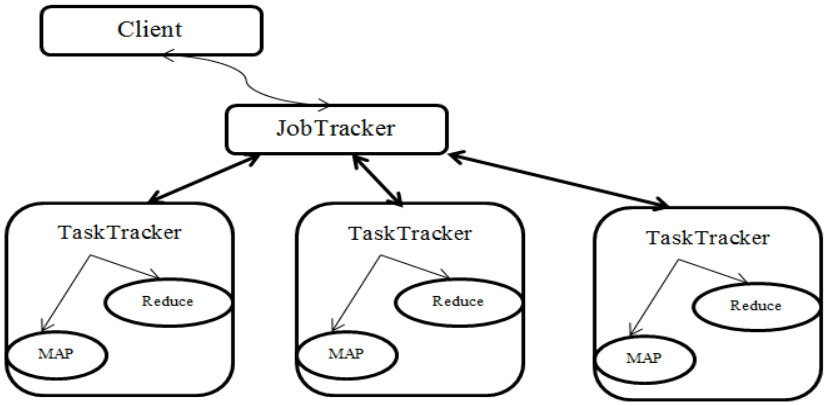
NameNode là thành phần quan trọng nhất của HDFS vì nó nắm giữ toàn bộ thông tin định danh để truy cập dữ liệu, việc xảy ra sự cố tại NameNode sẽ làm tê liệt hoàn toàn hệ thống tập tin. Để đề phòng sự cố, Hadoop đưa ra cơ chế dự phòng là SecondaryNameNode nhằm sao lưu định kỳ trạng thái của NameNode và thay thế vai trò của NameNode khi gặp sự cố.

2. Hadoop Core MapReduce

Hadoop cung cấp hai tác vụ giúp quản lý ứng dụng MapReduce [1]:

JobTracker: là daemon làm nhiệm vụ kết nối giữa ứng dụng MapReduce và hệ thống Hadoop cluster. Khi mã thực thi chương trình được gửi đến hệ thống, JobTracker có nhiệm vụ lên kế hoạch thực thi bằng cách xác định các tập tin cần xử lý, giao nhiệm vụ cho các máy con và giám sát tiến trình xử lý trên các máy con, nếu một tiến trình thất bại JobTracker sẽ cố gắng tự khởi động lại tiến trình trên một máy khác. Một hệ thống chỉ có duy nhất một JobTracker và thường được chạy trên cùng máy chủ với NameNode.

TaskTracker: Cũng giống như các daemon quản lý dữ liệu (DataNode và NameNode), các daemon quản lý tính toán (JobTracker và TaskTracker) cũng làm việc theo mô hình master/slave, JobTracker giám sát tổng thể việc thực hiện một ứng dụng MapReduce và TaskTracker quản lý các tiến trình nhỏ trên các máy con. Mỗi máy con chỉ có một TaskTracker và mỗi TaskTracker chịu trách nhiệm thực hiện một số tiến trình do JobTracker giao cho, tuy nhiên một TaskTracker lại có khả năng sinh ra và quản lý nhiều máy ảo Java (JVM) để thực hiện song song nhiều tiến trình map/reduce.



Hình 3. Tương tác giữa JobTracker và TaskTracker

Khác với DataNode, TaskTracker liên lạc định kì với JobTracker và nếu sau một khoảng thời gian JobTracker không nhận được bản tin thì nó sẽ đánh dấu TaskTracker bị sự cố và giao lại nhiệm vụ tương ứng cho máy khác.

C. Thuật toán BLAST

Hai thuật toán tìm kiếm trong cơ sở dữ liệu phổ biến trên thế giới hiện nay là BLAST và FastA. Thuật toán BLAST được đánh giá nhanh hơn và hiện đang được sử dụng rộng rãi, hơn nữa BLAST là thuật toán tìm kiếm heuristic. BLAST cần đầu vào là 2 chuỗi: một là trình tự truy vấn (hay còn gọi là trình tự đích) và một cơ sở dữ liệu các trình tự. BLAST tìm kiếm các trình tự con trong câu truy vấn giống với các trình tự trong cơ sở dữ liệu sinh học. Thông thường, khi sử dụng, trình tự truy vấn là nhỏ hơn rất nhiều so với cơ sở dữ liệu. Chẳng hạn, trình tự truy vấn có thể chỉ gồm 1.000 nucleotide trong khi cơ sở dữ liệu trình tự có hàng tỉ nucleotide.

BLAST tìm kiếm những bắt cặp trình tự có điểm số cao giữa trình tự truy vấn và các trình tự trong cơ sở dữ liệu bằng cách sử dụng phương pháp dựa trên kinh nghiệm (heuristic) để có thể có tìm được kết quả gần tốt bằng với giải thuật Smith-Waterman. Thuật toán bắt cặp trình tự tối ưu của Smith-Waterman là quá chậm khi tìm kiếm trong một cơ sở dữ liệu gen quá lớn như GenBank. Bởi vậy, giải thuật BLAST dùng một hướng tiếp cận heuristic, dù ít chính xác hơn Smith-Waterman nhưng lại cho tốc độ nhanh hơn gấp 50 lần. Thuật toán của BLAST có 2 phần, một phần tìm kiếm và một phần đánh giá thống kê dựa trên kết quả tìm được [6].

Trong phần đánh giá thống kê, BLAST dựa trên cơ sở đánh giá của một cặp trình tự để tính ra một giá trị gọi là (Bit-Score). Giá trị càng cao chứng tỏ khả năng tương tự của các bắt cặp càng cao.

Ngoài ra BLAST tính toán một giá trị trông đợi E-Score (Expect-Score) phụ thuộc vào Bit-Score.

Giải thuật này được hiện thực qua các chương trình: blastp, blastn, blastx, tblastn và tblastx của NCBI với các chức năng như sau:

- blastp: so sánh trình tự amino acid với cơ sở dữ liệu là các trình tự protein.
- blastn: so sánh trình tự nucleotid với cơ sở dữ liệu là các trình tự nucleotid.
- blastx: so sánh biến đổi six-frame của trình tự nucleotid với cơ sở dữ liệu protein.
- blastn: so sánh trình tự protein với cơ sở dữ liệu nucleotid.
- tblastx: so sánh biến đổi six-frame của một trình tự nucleotid với biến đổi six-frame của các trình tự trong cơ sở dữ liệu nucleotid.

III. THỬ NGHIỆM VÀ ĐÁNH GIÁ HIỆU SUẤT

A. Môi trường ứng dụng

Hệ thống được cài đặt trên:

- Hệ điều hành Ubuntu 10.04 (32 bit).
- Hadoop framework 0.20.2.
- JDK 1.6.0_43.
- File đầu vào (có định dạnh FASTA) là danh sách các trình tự protein cần tìm kiếm (Bài báo dùng 1 file đầu vào có 1000 trình tự cần tìm kiếm).
- Sử dụng BLASTX(so sánh biến đổi six-frame của trình tự nucleotid với cơ sở dữ liệu protein) của NCBI.
- Cơ sở dữ liệu protein sử dụng là NR (Non-Redudant) của NCBI.

B. Thiết lập

Thực thi Blast trên Hadoop bao gồm 3 phần:

- Xử lý các file đầu vào cho chương trình Blast.
- Ứng với n tác vụ MAP thì chương trình tự động chia file đầu vào thành n file để cho mỗi MAP thực thi trên 1 file.
- Hàm Map.
- Hàm chính (Main function).

BLAST không đòi hỏi hàm Reduce để xử lý các đầu ra của hàm Map, vì vậy chương trình không cần dùng hàm Reduce. Đầu vào chuẩn của chương trình Blast file FASTA, do đó định dạng file đầu vào cho ứng dụng BLAST trên Hadoop cũng là một tập các file FASTA.

Hình 4. Mô hình Blast trên Hadoop Framework

Hàm Map tạo một tiến trình (process) Java gọi chương trình Blast. Hàm Map có khóa là tên file và giá trị là đường dẫn file trong HDFS. Mỗi tác vụ (task) Map sẽ download file đầu vào trong HDFS, và truyền vào chương trình BLAST.

Sử dụng hàm copyToLocalFile API để copy các file input từ HDFS tới thư mục hiện hành. Lớp OutputHandler để xử lý từng dòng của các file đầu vào.

C. Đánh giá hiệu suất

Chương trình Blast – Hadoop được chạy trên Ubuntu Ubuntu 10.04 (32 bit).

Bảng 1. Thời gian thực thi chương trình BLAST không cài đặt trên hadoop

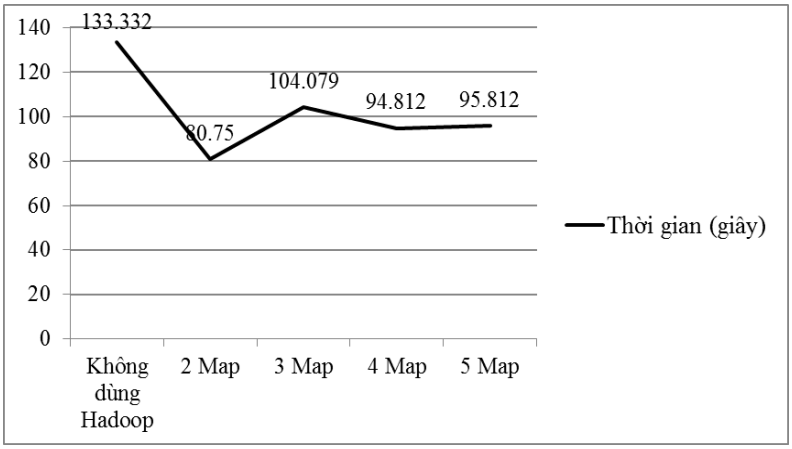
Thời gian thực thi (giây)
133.332

Bảng 2. Thời gian thực thi chương trình BLAST không cài đặt trên hadoop

Số Map	Thời gian thực thi (giây)
2	80.75
3	104.079
4	94.812
5	95.812

Với kết quả thử nghiệm, cho thấy tốc độ thực thi của BLAST được cải tiến đáng kể so với dùng chương trình BLAST thông thường khi số Map lần lượt 2, 3, 4, 5. Nếu được cài đặt trên môi trường có cấu hình tốt hơn, thì sẽ giảm đáng kể thời gian thực thi tìm kiếm sự tương đồng trên BLAST.

Phần thử nghiệm cũng cho thấy, khi cài đặt hệ thống Hadoop nên thiết lập cấu hình số Map sao cho phù hợp với từng hệ thống để tối ưu hóa tài nguyên CPU, RAM (Số Map được chọn phải bằng với số CPU trên hệ thống).



Hình 5. Đánh giá tốc độ thực thi khi chạy BLAST trên Hadoop

IV. KẾT LUẬN

Nhìn chung, mô hình MapReduce đã được sử dụng rộng rãi trên thế giới, đặc biệt là Google, cho nhiều mục đích khác nhau. Riêng Hadoop cũng đã được lựa chọn làm công nghệ chính cho rất nhiều hệ thống nổi tiếng trên thế giới, trong đó có Yahoo, Facebook, Amazon, … Thế mạnh của MapReduce là sự đơn giản và dễ hiểu đến mức một lập trình viên không cần biết nhiều về hệ thống phân tán vẫn có thể viết được một chương trình. Ngoài ra, MapReduce được thiết kế phục vụ cho một mục đích duy nhất là cung cấp khả năng xử lý lượng dữ liệu phân tán khổng lồ trong một thời gian chấp nhận được, vì vậy nếu hệ thống chỉ xử lý trên lượng dữ liệu không quá lớn và có đòi hỏi cao ở các tiêu chí khác, thì MapReduce chưa hẳn là sự lựa chọn tốt nhất. Việc triển khai ứng dụng BLAST dùng mô hình MapReduce trên hapdoop framework giúp thực thi song song Blast. Từ đó cải tiền được tốc độ thực thi của ứng dụng Blast.

V. TÀI LIỆU THAM KHẢO

[1] Tom White, Hadoop: The Definitive Guide, MapReduce for the Cloud. O'Reilly, 06/2009.
[2] Jason Venner, Pro Hadoop, Apress, 06/2009.
[3] Jeffrey Dean và Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, 12/2004.
[4] Ralf Lammel, Google's MapReduce Programming Model-Revisited, 2007.
[5] http://hadoop.apache.org.
[6] http://blast.ncbi.nlm.nih.gov/Blast.cgi

FIND REGIONS OF LOCAL SIMILARITY BETWEEN SEQUENCES BY BLAST ALGORITHM ON MAPREDUCE MODEL

ABSTRACT— *the paper presents some techniques with MapReduce model, this is the programing model allows programmers to create and process huge amounts of data to deploy the distributed system. This model is inspired by the Map and Reduce functions in the function-oriented programming language, which defines a mapping function to generate the key pair / value from the input data, and then define the recursive function combine the values that has same key together to create output data. This model was used in the process and analysis data system that data in the system is huge and unstructured data that was stored distributed over multiple server clusters. Besides, the paper also shows how to build Blast on MapReduce using Hadoop Framework, from that evaluate the performance of Blast on MapReduce model.*