**TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜&🙞🙜-----



**BÀI TẬP LỚN**

**NHẬP MÔN DỮ LIỆU LỚN**

**Cài đặt Hadoop, MongoDB và thuật toán TF-IDF sử dụng 3 nodes**

|  |  |  |
| --- | --- | --- |
| **Lớp 64CS3 – Nhóm 02** | | |
| **Thành viên nhóm** | Dương Gia Khánh  Vương Trung Thành  Ngô Đức Thịnh  Nguyễn Trần Lê Tuấn | 1526864  186864  189464  1553564 |
| **Giảng viên hướng dẫn** | ThS. Nguyễn Đình Quý | |

**HÀ NỘI, 07/2022**

Giáo viên hướng dẫn

Ký và ghi rõ họ tên

**Lời cảm ơn**

Nhóm 02 xin gửi lời cảm ơn đến thầy giáo, **ThS. Nguyễn Đình Quý** – giảng viên bộ môn Khoa học máy tính - khoa Công nghệ thông tin đã hướng dẫn chúng em trong suốt quá trình học hỏi và hướng dẫn làm bài tập lớn. Thầy đã dành nhiều thời gian của mình để tận tình chỉ bảo, hướng dẫn, định hướng cho nhóm thực hiện các bài thực hành, giúp cho chúng em hiểu biết thêm về môn học cũng như học hỏi thêm kinh nghiệm và kỹ năng để hoàn thành bài tập.

Mặc dù đã cố gắng hoàn thành tốt công việc được phân công trong phạm vi và khả năng cho phép song trong quá trình làm bài tập lớn do kiến thức của chúng em còn hạn chế nên không thể tránh khỏi một vài sai sót nhất định khi trình bày và đánh giá vấn đề. Rất mong nhận được sự góp ý, đánh giá của thầy để bài tập lớn của chúng em hoàn thiện hơn.

***Chúng em xin chân thành cảm ơn!***

**ĐẶT VẤN ĐỀ**

Hiện nay, có rất nhiều lớp các bài toán cần phải thực hiện khối lượng tính toán lớn trên rất nhiều dữ liệu đầu vào. Các bài toán như vậy có thể gặp trong nhiều lĩnh vực, từ tìm kiếm, truy hồi thông tin tới các bài toán sinh học. Một ví dụ điển hình là các máy tìm kiếm (search engine) lớn như Google, trong quá trình phân tích, đánh chỉ mục phải xử lý tới hàng tỷ trang Web và các tài liệu khác, tương đương lượng lớn dữ liệu ở mức pentabyte, exabyte. Hay trong lĩnh vực sinh học phân tử và tin sinh học.

Những vấn đề về xử lý ngôn ngữ tự nhiên, nhận dạng, xử lý ảnh ba chiều, dự bảo thời tiết... đều đòi hỏi phải xử lý dữ liệu với tốc độc rất cao, với khối lượng dữ liệu rất lớn. Để đáp ứng yêu cầu tỉnh toán và xử lý lớn như vậy, việc tính toàn tuần tự trên các máy tính riêng lẻ là không đáp ứng yêu cầu. Do đó cần phải có những hệ thống máy tính mạnh mẽ hơn để đáp ứng được những yêu cầu của thực tế.

Mặc dù tốc độ xử lý của các Bộ xử lý tăng nhanh trong những năm qua, nhưng do giới hạn về vật lý nên khả năng tính toán của chúng không thể tăng mãi được. Điều này dẫn tới là muốn tăng được khả năng tính toán của các hệ thống máy tính thì phải khai thác được khả năng xử lý song song của chúng. Ngày càng xuất hiện nhiều bài toán mà những hệ thống đơn một bộ xử lý không đáp ứng được yêu cầu xử lý về thời gian, do đó đòi hỏi phải sử dụng những hệ thống đa bộ xử lý và đòi hỏi phải xử lý song song.

Việc song song hóa để dãy nhanh tốc độ thực thi của một tiến trình có thể thực hiện ở nhiều mức, đầu tiên là mức vật lý, với việc xây dựng hệ thống máy tính liên kết với nhau. Mức thứ hai là mức ứng dụng, ngay chính bản thân các ứng dụng cũng phải hỗ trợ khả năng tính toán phân tán, tối ưu các thuật toán khi làm việc trên môi trường phân tán, mà tại đó rất hay gặp phải các vấn đề về lỗi, vấn đề bảo mật, tức là hệ thống không đồng đều và tính tin cậy không cao.

Các phương pháp song song hóa đã được đề cập từ lâu, khi mà J.Von Neumann giới thiệu mô hình tính toán song song có tên Otomat tế bào. Từ đó đến nay, lý thuyết về xử lý song song trở thành lĩnh vực nghiên cứu quan trọng và ngày càng đem lại những dấu hiệu khả quan trong việc xây dựng một mô hình lập trình mới có những tính năng vượt trội so với mô hình lập trình tuần tự truyền thống.

Năm 2004, Google giới thiệu mô hình toán (ở mức ứng dụng) MapReduce để trợ tính toán phân tán trên một tập dữ liệu lớn và trên các cụm nhiều máy tính. Cho đến nay đã có nhiều giải pháp triển khai như Apache Hadoop, Qizmt, Skynet và Greenplum, làm cho MapReduce trở lên phổ biến và theo Infoworld, MapReduce là công nghệ doanh nghiệp có ảnh hưởng lớn nhất (năm 2009).

MapReduce - nền tảng lập trình này đã đƣợc Google, Yahoo, Facebook, MySpace và nhiều hãng khác áp dụng để xử lý bộ dữ liệu khổng lồ hàng giờ. Ở dạng đơn giản nhất, MapReduce chia việc xử lý thành nhiều khối công việc nhỏ, phân tán gồm các nút tính toán (tiêu biểu là các server thông thường) và rồi thu thập các kết quả. Hỗ trợ xử lý song song có khả năng mở rộng cao, MapReduce là giải pháp nhanh, rẻ và an toàn.

Mục tiêu của bài tập lớn là tìm hiểu về cài đặt Hadoop cho 3 nodes, cài đặt MongoDB cho 3 nodes, cài đặt MapReduce cho thuật toán TF-IDF.

**MỤC LỤC**

[CHƯƠNG 1. CÀI ĐẶT HADOOP 3 NODES 1](#_Toc107772944)

[1.1 Apache Hadoop 1](#_Toc107772945)

[1.1.1 Giới thiệu tổng quan về Apache Hadoop 1](#_Toc107772946)

[1.1.2 Mục tiêu của Hadoop 1](#_Toc107772947)

[1.1.3 Ưu điểm của Apache Hadoop 1](#_Toc107772948)

[1.1.4 Các thành phần trong hệ sinh thái của Apache Hadoop 2](#_Toc107772949)

[1.2 Cách cài đặt Hadoop 3 nodes 4](#_Toc107772950)

[1.2.1 Chuẩn bị 4](#_Toc107772951)

[1.2.2 Kiến trúc Hadoop Cluster 4](#_Toc107772952)

[1.2.3 Cấu hình hệ thống 4](#_Toc107772953)

[1.2.4 Cấu hình Master Node 5](#_Toc107772954)

[1.2.5 Nhân bản cấu hình tới các Slave Node 9](#_Toc107772955)

[1.2.6 Chạy và monitor HDFS 10](#_Toc107772956)

[CHƯƠNG 2. CÀI ĐẶT MONGODB 12](#_Toc107772957)

[2.1 MongoDB 12](#_Toc107772958)

[2.1.1 Giới thiệu tổng quan về MongoDB 12](#_Toc107772959)

[2.1.2 Một số thuật ngữ hay sử dụng trong MongoDB 12](#_Toc107772960)

[2.1.3 Cách thức Mongo hoạt động 14](#_Toc107772961)

[2.1.4 Ưu điểm của MongoDB 14](#_Toc107772962)

[2.1.5 Nhược điểm của MongoDB 16](#_Toc107772963)

[2.1.6 Các trường hợp sử dụng MongoDB 16](#_Toc107772964)

[2.2 Cách cài đặt MongoDB 3 nodes replica set cluster 17](#_Toc107772965)

[2.2.1 Giới thiệu Replica Set trong MongoDB 17](#_Toc107772966)

[2.2.2 Cấu hình trong MongoDB 19](#_Toc107772967)

[CHƯƠNG 3. THUẬT TOÁN TF-IDF 23](#_Toc107772968)

[3.1 TF-IDF 23](#_Toc107772969)

[3.2 TF 23](#_Toc107772970)

[3.3 IDF 23](#_Toc107772971)

[CHƯƠNG 4. THỰC NGHIỆM 25](#_Toc107772972)

[4.1 Thực nghiệm Hadoop 25](#_Toc107772973)

[4.2 Thực nghiệm MongoDB 28](#_Toc107772974)

[4.3 Thuật toán TF-IDF trong MapReduce 32](#_Toc107772975)

[4.3.1 Tính Term Frequency (TF) 32](#_Toc107772976)

[4.3.2 Tính Inverse Document Frequency (IDF) 33](#_Toc107772977)

[4.3.3 Tính TF-IDF 34](#_Toc107772978)

[TÀI LIỆU THAM KHẢO 36](#_Toc107772979)

# CÀI ĐẶT HADOOP 3 NODES

## Apache Hadoop

### Giới thiệu tổng quan về Apache Hadoop

Apache đã phát triển hệ thống mã mở Hadoop cùng với nhân là khung phần mềm MapReduce. Hadoop chạy trên môi trường Unix và được thiết kế để hỗ trợ các ứng dụng sử dụng được số lượng lớn dữ liệu cấu trúc và phi cấu trúc. Hệ thống phần mềm mã mở này được tối ưu cho tính tin cậy, khả năng mở rộng và tính toán phân tán.

Không giống như các hệ quản trị cơ sở dữ liệu truyền thống, Hadoop được thiết kế để làm việc với nhiều loại dữ liệu và dữ liệu nguồn. Công nghệ HDFS của Hadoop cho phép khối lượng lớn công việc được chia thành các khối dữ liệu nhỏ hơn được nhân rộng và phân phối trên các phần cứng của một cluster để xử lý nhanh hơn. Công nghệ này đã được sử dụng rộng rãi bởi một số trang web lớn nhất thế giới, chẳng hạn như Facebook, eBay, Amazon, Baidu, và Yahoo. Các nhà quan sát nhấn mạnh rằng Yahoo là một trong những nhà đóng góp lớn nhất đối với Hadoop.

### Mục tiêu của Hadoop

Mục tiêu chính của Hadoop là:

* Lưu trữ dữ liệu khả mở, tiết kiệm chi phí:

+ Xử lý dữ liệu phân tán với mô hình lập trình đơn giản, thân thiện hơn như MapReduce

+ Hadoop thiết kế để mở rộng thông qua kỹ thuật scale-out, tăng số lượng máy chủ

+ Thiết kế để vận hành trên phần cứng phổ thông, có khả năng chống chịu lỗi phần cứng

* Xử lý dữ liệu mạnh mẽ

+ Thách thức: các thiết bị lưu trữ tốc độ chậm, máy tính thiếu tin cậy, lập trình song song phân tán không dễ dàng.

### Ưu điểm của Apache Hadoop

Thư viện phần mềm Apache Hadoop là một framework cho phép các xử lý phân tán các tập dữ liệu lớn trên các cụm nhiều máy tính sử dụng sử dụng mô hình lập trình đơn giản. Nó được thiết kế để mở rộng từ một vài máy chủ đơn đến hàng nghìn máy tính, mà mỗi máy đều cung cấp một đơn vị lưu trữ và tính toán. Thay vì dựa trên các máy tính có độ tin cậy cao, thì cái thư viện được thiết kế để tự nó phát hiện và xử lý lỗi ở tầng ứng dụng, do đó nó dung cấp một dịch vụ có độ tin cậy cao trên cụm nhiều máy tính mà mỗi máy đều có thể bị lỗi.

### Các thành phần trong hệ sinh thái của Apache Hadoop

Ảnh có chứa văn bản

Mô tả được tạo tự động

*Lõi (core) Hadoop*: bao gồm một tập hợp của các thành phần và các giao diện cung cấp chức năng truy cập vào các hệ thống tập tin phân tán và vào ra tổng quát. Các thành phần cốt lõi cũng cung cấp sự tối ưu hóa dựa vào phân nhóm địa lý của máy chủ nhằm giảm thiểu lưu lượng mạng giữa các máy chủ trong các cụm tính toán.

*Hệ thống tập tin phân tán* (HDFS - Hadoop Distribute File System): là hệ thống lưu trữ chính được sử dụng bởi các ứng dụng Hadoop. HDFS, như tên gọi của nó, một hệ thống tập tin phân tán cung cấp truy cập thông lượng cao vào dữ liệu của ứng dụng, tạo ra nhiều bản sao của khối dữ liệu và phân phối chúng trên các nút tính toán trong một cụm để cho phép tính toán song song, đáng tin cậy và nhanh chóng.

*MapReduce Framework*: là một mô hình lập trình và khung phát triển phần mềm cho phép viết các ứng dụng nhanh chóng xử lý một lượng lớn dữ liệu song song dựa vào một cụm lớn các máy tính gọi là các nút tính toán (node) dựa trên kiến trúc của nền tảng Hadoop. MapReduce sử dụng HDFS để truy cập vào các khối (phân đoạn tập tin) và lưu trữ kết quả rút gọn.

*Apache Pig*: là ngôn ngữ xử lí dòng dữ liệu. Apache Pig là một nền tảng cho việc phân tích dữ liệu lớn bao gồm một ngôn ngữ cấp cao để diễn tả các chương trình phân tích dữ liệu. Đặc điểm chính của chương trình Pig là cấu trúc của chúng có thể được song song hóa cho phép nó xử lý các tập hợp dữ liệu rất lớn, cú pháp đơn giản. Các tính năng xây dựng sẵn (built-in functionality) cung cấp một mức độ trừu trượng để cho phát triển các công việc Hadoop nhanh hơn và dễ dàng hơn để viết hơn so với MapReduce truyền thống.

*Apache Hive*: là một kho dữ liệu cơ sở hạ tầng được xây dựng trên Hadoop. Hive cung cấp các công cụ để cho phép tóm tắt dữ liệu, truy vấn không chuẩn (ad-hoc) và phân tích các bộ dữ liệu lớn được lưu trữ trong các tập tin Hadoop. Nó cung cấp một cơ chế để định cấu trúc cho loại dữ liệu này và cung cấp một ngôn ngữ truy vấn đơn giản gọi là Hive QL, dựa SQL, cho phép người sử dụng quen thuộc với SQL để truy vấn dữ liệu này.

*Apache Hbase*: là một cơ sở dữ liệu phân tán theo cột. HBase sử dụng HDFS cho việc lưu trữ cơ bản của nó. Nó ánh xạ dữ liệu HDFS vào một cơ sở dữ liệu có cấu trúc giống và cung cấp các giao diện lập trình được cho Java (Java API) truy cập vào CSDL này. Nó hỗ trợ hàng loạt kiểu tính toán sử dụng các truy vấn MapReduce và đọc ngẫu nhiên. HBase thường được sử dụng trong Hadoop khi có truy cập đọc/ ghi ngẫu nhiên, thời gian thực. Mục tiêu của nó là lưu trữ các bảng rất lớn đang chạy trên cụm thiết bị phần cứng.

*Apache Sqoop:* là một công cụ được thiết kế để hỗ trợ xuất và nhập dữ liệu hàng loạt vào HDFS từ các kho dữ liệu có cấu trúc như cơ sở dữ liệu quan hệ, kho dữ liệu doanh nghiệp và hệ thống NoSQL. Nó là một công cụ di chuyển dữ liệu dựa trên kiến ​​trúc trình kết nối hỗ trợ các plugin để cung cấp kết nối với các hệ thống bên ngoài mới.Hỗ trợ import tất cả các bảng, một bảng hay 1 phần của bảng vào HDFS. Thông qua Map only hoặc MapReduce job, kết quả là 1 thư mục trong HDFS chứ các tập tin văn bản phân tách các trường theo ký tự phân tách (vd. , hoặc \t).Hỗ trợ export dữ liệu ngược trở lại từ Hadoop ra bên ngoài.

*Apache Kafka*: là hệ thống message pub/sub phân tán (distributed messaging system). Bên pulbic dữ liệu được gọi là producer, bên subscribe nhận dữ liệu theo topic được gọi là consumer. Kafka có khả năng truyền một lượng lớn message theo thời gian thực, trong trường hợp bên nhận chưa nhận message vẫn được lưu trữ sao lưu trên một hàng đợi và cả trên ổ đĩa bảo đảm an toàn. Đồng thời nó cũng được replicate trong cluster giúp phòng tránh mất dữ liệu.

*Apache Oozie*: Oozie là một hệ thống lập lịch luồng công việc để quản lý các công việc thực thi trên cụm Hadoop. Luồng workflow của Oozie là đồ thị vòng có hướng (Directed Acyclical Graphs (DAGs)) của các khối công việc. Oozie hỗ trợ đa dạng các loại công việc như: thực thi MapReduce jobs, thực thi Pig hay Hive scripts, thực thi các chương trình Java hoặc Shell, tương tác với dữ liệu trên HDFS, chạy chương trình từ xa qua SSH, gửi nhận email

*Apache Zookeeper*: là một công cụ cấu hình cụm (cluster) và quản lý sự dãy hóa (serialization) rất hữu ích để xây dựng các cụm lớn các nút của Hadoop, dịch vụ hiệu năng cao cho các ứng dụng phân tán. Nó tập trung vào các dịch vụ như quản

lí thông tin cấu hình, đặt tên, đồng bộ hóa phân tán cũng và các dịch vụ nhóm.

*YARN – Yet Another Resource Negotiator*: là một framework hỗ trợ phát triển ứng dụng phân tán YARN cung cấp daemons và APIs cần thiết cho việc phát triển ứng dụng phân tán, đồng thời xử lý và lập lịch sử dụng tài nguyên tính toán (CPU hay memory) cũng như giám sát quá trình thực thi các ứng dụng đó. YARN đóng vai trò cấp phát lượng tài nguyên phù hợp cho các ứng dụng khi có yêu cầu. YARN được đưa ra từ Hadoop 2.0, cho phép MapReduce và non MapReduce cùng chạy trên 1 cụm Hadoop. Với MapReduce job, vai trò của job tracker được thực hiện bởi application tracker.

## Cách cài đặt Hadoop 3 nodes

### Chuẩn bị

Với phần này, ta phải đi chuẩn bị 3 máy tính, rồi thiết lập cấu hình, môi trường cho hệ thống trên máy và tự thiết lập địa chỉ IP.

Node-master: 10.0.20.10

Node-1: 10.0.20.20

Node-2: 10.0.20.30

### Kiến trúc Hadoop Cluster

Kiến trúc Hadoop Cluster gồm 2 node chính:

Master node: lưu giữ thông tin về hệ thống file phân tán, tương đương với bảng inode của ext3, ngoài ra còn có nhiệm vụ lên kế hoạch phân bố tài nguyên. Trong guide này, node master có 2 nhiệm vụ chính:

+ Name node: quản lý hệ thôngống file phân tán, nắm thông tin block dữ liệu nào nằm ở đâu trong cluster.

+ ResourceManager: quản lý các job của YARN và quản lý các job được xếp lịch chạy trên các node slave.

Slave nodes: lưu dữ liệu thực và cung cấp sức mạnh phần cứng để chạy các job, trong lab này là node1 và node2:

+ Data node: quản lý các block dữ liệu về mặt vật lý.

+ NodeManager: quản lý thực hiện các task trên node.

### Cấu hình hệ thống

#### Tạo file host trên mỗi node

Để mỗi node giao tiếp với nhau bằng tên, ta đi sửa file /etc/hosts để thêm địa chỉ IP riêng của ba máy chủ.

|  |
| --- |
| 10.0.20.10 node-master  10.0.20.20 node1  10.0.20.30 node2 |

#### Tạo Key-pairs xác thực cho Hadoop User

Master node sẽ sử dụng ssh để kết nối tới các node khác và quản lý cluster. Thực hiện:

1. Đăng nhập vào node-master với hadoop user, tạo 1 ssh-key gán cho hadoop user.

|  |
| --- |
| ssh hadoop@node-master  ssh-keygen –b 4096 |

2. Sau đó thử ssh bằng hadoop user vào từng node, nếu không hỏi password tức là thành công.

|  |
| --- |
| ssh-copy-id -i $HOME/.ssh/id\_rsa.pub hadoop@node-master  ssh-copy-id -i $HOME/.ssh/id\_rsa.pub hadoop@node1  ssh-copy-id -i $HOME/.ssh/id\_rsa.pub hadoop@node2 |

#### Tải và giải nén bộ cài Hadoop

Đăng nhập vào node-master với hadoop user, tải bộ cài Hadoop từ trang chủ. Nhớ kiểm tra phiên bản đã cài vì hadoop có cập nhật các phiên bản.

|  |
| --- |
| cd  wget http://apache.cs.utah.edu/hadoop/common/current/hadoop-3.1.2.tar.gz  tar -xzf hadoop-3.1.2.tar.gz  mv hadoop-3.1.2 hadoop |

#### Thiết lập biến môi trường

1. Thêm folder hadoop vừa giải nén vào biến môi trường PATH. Sửa /home/hadoop/.profile và thêm dòng:

|  |
| --- |
| PATH=/home/hadoop/hadoop/bin:/home/hadoop/hadoop/sbin:$PATH |

Thêm Hadoop vào PATH cho shell. Sửa .bashrc và thêm các dòng:

|  |
| --- |
| export HADOOP\_HOME=/home/hadoop/hadoop  export PATH=${PATH}:${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin |

### Cấu hình Master Node

#### Thiết lập JAVA\_HOME

1. Tìm đường dẫn cài đặt Java của bạn. Đây được gọi là JAVA\_HOME. Nếu bạn đã cài đặt open-jdk từ trình quản lý gói của mình, bạn có thể tìm thấy đường dẫn bằng lệnh:

|  |
| --- |
| update-alternatives --display java |

2. Vào ~/hadoop/etc/hadoop/hadoop-env.sh và sửa dòng:

|  |
| --- |
| export JAVA\_HOME=/usr/java/jdk1.8.0\_171-amd64 |

Đường dẫn JAVA\_HOME thay đổi tùy từng lúc.

#### Đặt đường dẫn NameNode

Cập nhật file ~/hadoop/etc/hadoop/core-site.xml, để đặt vị trí NameNode thành node-master trên cổng 9000:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  <configuration>  <property>  <name>fs.default.name</name>  <value>hdfs://node-master:9000</value>  </property>  </configuration> |

Chỉnh sửa file hdfs-site.conf

|  |
| --- |
| <configuration>  <property>  <name>dfs.namenode.name.dir</name>  <value>/home/hadoop/data/nameNode</value>  </property>    <property>  <name>dfs.datanode.data.dir</name>  <value>/home/hadoop/data/dataNode</value>  </property>    <property>  <name>dfs.replication</name>  <value>1</value>  </property>  </configuration> |

Lưu ý thuộc tính dfs.replication, thuộc tính này là số lần dữ liệu được nhân bản (nhằm mục đích dự phòng, node này chết thì còn bản sao ở node khác). ở đây mình có 2 node  có thể set mục này tối đa bằng 2 để dữ liệu được nhân bản trên cả 2 node (tốn tài nguyên). Không được đặt con số này nhiều hơn số datanode mà chúng ta có.

#### Thiết lập YARN làm job Scheduler

Đổi tên file mapred-site.xml.template thành mapred-site.xml. Nội dung:

|  |
| --- |
| <configuration>  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  </configuration> |

Lưu ý giá trị là yarn, tốt nhất tìm vị trí khoá như trên rồi sửa lại.  
Sửa file yarn-site.xml.

|  |
| --- |
| <configuration>  <property>  <name>yarn.acl.enable</name>  <value>0</value>  </property>    <property>  <name>yarn.resourcemanager.hostname</name>  <value>node-master</value>  </property>    <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  </configuration> |

#### Cấu hình Slave

Khai báo danh sách các datanode mà ta có với namenode, sửa file ~/hadoop/etc/hadoop/slaves  
Điền mỗi namenode 1 dòng

|  |
| --- |
| node1  node2 |

#### Cấu hình phân bổ RAM

Việc phân bổ ram có thể làm để những node có ram yếu có thể chạy được. các giá trị mặc định được thiết kế cho các máy 8GB ram trở lên. Dưới đây là những tuỳ chỉnh cho những node 2GB Ram.

#### Các thuộc tính phân bổ RAM

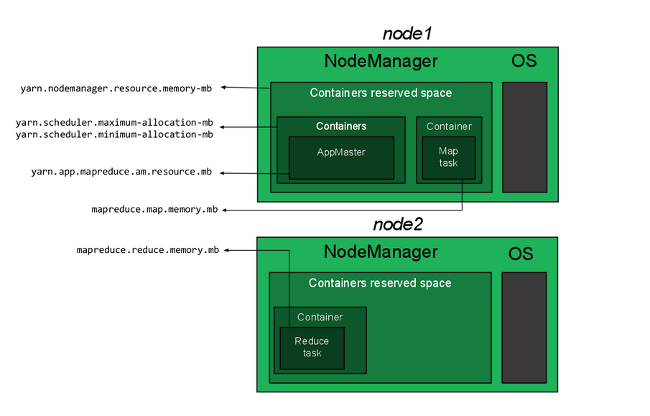
Một Yarn job được chạy với 2 loại:

* Một application manager: chịu trách nhiệm giám sát ứng dụng và phối hợp để phân phối excecutor trong cluster (phân phối xem node nào sẽ thực hiện job).
* Một số executor được tạo bởi application manager chạy các job. Đối với các mapredure job, chúng thực hiện các tiến trình map và redure song song.

Cả 2 app trên chạy trên slave node. Mỗi slave node chạy một NodeManager dạng deamon, chịu trách nhiệm đối với việc tạo container trên mỗi node. Toàn bộ cluster chịu sự quản lý của ResourceManager, lên kế hoạch phân bổ các container trên toàn bộ slavenode, dựa trên nhu cầu cần thiết cho mỗi action hiện tại.

Bốn hình thức cấp phát tài nguyên được cấu hình chuẩn để cluster có thể làm việc:

* Số bộ nhớ RAM có thể cung cấp cho Yarn container trên mỗi node. Giới hạn này nên để cao, nếu không việc cung cấp tài nguyên cho container sẽ bị reject, ứng dụng sẽ fail. Tuy nhiên, không được cấp toàn bộ số RAM trên node. Giá trị này được cấu hình trên file yarn-site.xml với key = yarn.nodemanager.resource.memory-mb
* Lượng bộ nhớ mà 1 node đơn có thể chiếm và số bộ nhớ cấp phát nhỏ nhất được cho phép. Được cấu hình tại yarn-site.xml với key = yarn.scheduler.minimum-allocation-mb
* Lượng RAM được cấp cho mỗi tiến trình map hoặc redure. Nên nhỏ hơn số bộ nhớ tối đa. Được cấu hình tại file mapred-site.xml với key = mapreduce.map.memory.mb và mapreduce.reduce.memory.mb
* Lượng RAM sẽ được cấp cho ApplicationMaster.  
  Hình minh hoạ cho các cấu hình này:

Ví dụ về cấu hình cho node 2GB RAM

|  |  |
| --- | --- |
| Property  yarn.nodemanager.resource.memory-mb  yarn.scheduler.maximum-allocation-mb  yarn.scheduler.minimum-allocation-mb  yarn.app.mapreduce.am.resource.mb  mapreduce.map.memory.mb  mapreduce.reduce.memory.mb | Value  1536  1536  128  512  256  256 |

Sửa file /home/hadoop/hadoop/etc/hadoop/yarn-site.xml và thêm các dòng sau:

|  |
| --- |
| <property>  <name>yarn.nodemanager.resource.memory-mb</name>  <value>1536</value>  </property>    <property>  <name>yarn.scheduler.maximum-allocation-mb</name>  <value>1536</value>  </property>    <property>  <name>yarn.scheduler.minimum-allocation-mb</name>  <value>128</value>  </property>    <property>  <name>yarn.nodemanager.vmem-check-enabled</name>  <value>false</value>  </property> |

Sửa file /home/hadoop/hadoop/etc/hadoop/mapred-site.xml và thêm các dòng sau:

|  |
| --- |
| <property>  <name>yarn.app.mapreduce.am.resource.mb</name>  <value>512</value>  </property>    <property>  <name>mapreduce.map.memory.mb</name>  <value>256</value>  </property>    <property>  <name>mapreduce.reduce.memory.mb</name>  <value>256</value>  </property> |

### Nhân bản cấu hình tới các Slave Node

1. Copy hadoop sang slave node:

|  |
| --- |
| cd /home/hadoop/  scp hadoop-\*.tar.gz node1:/home/hadoop  scp hadoop-\*.tar.gz node2:/home/hadoop |

Kết nối tới node1, node2 qua SSH, giải nén 2 gói vừa copy sang.

|  |
| --- |
| ssh node1 |
| tar -xzf hadoop-3.1.2.tar.gz  mv hadoop-3.1.2 hadoop  exit |

Thực hiện giống trên với node2.

Quay lại node-master, copy config sang.

|  |
| --- |
| for node in node1 node2; do  scp ~/hadoop/etc/hadoop/\* $node:/home/hadoop/hadoop/etc/hadoop/;  done |

Format HDFS

HDFS sau khi tạo cũng cần phải được format trước khi sử dụng, tương tự như đối với bất kỳ file system truyền thống nào khác.  
Trên node master chúng ta gõ lệnh:

|  |
| --- |
| hdfs namenode -format |

Đến đây thì việc cài đặt HDFS đã hoàn tất và sẵn sàng để chạy.

### Chạy và monitor HDFS

1. Start và Stop HDFS

Mọi thao tác được thực hiện trên node-master:

|  |
| --- |
| start-dfs.sh  stop-dfs.sh |

Sau khi chạy lệnh start, master sẽ tự động SSH sang node1+node2 (qua key đã tạo ở trên) để tiến hành bật datanode. Kiểm tra các tiến trình đã được khởi động trên mỗi node bằng lệnh jps , kết quả như bên dưới, bao gồm PID và process name.

|  |
| --- |
| 21922 Jps  21603 NameNode  21787 SecondaryNameNode |
| 19728 DataNode  19819 Jps |

2. Monitor HDFS cluster

Lấy thông tin về cluster đang chạy bằng lệnh:

|  |
| --- |
| hdfs dfsadmin -report |

Lấy các thông tin về các lệnh được hỗ trợ bằng lệnh help.

|  |
| --- |
| hdfs dfsadmin -help |

3. Test đẩy dữ liệu lên HDFS

Sử dụng lệnh hdfs dfs để thao tác với dữ liệu trên hdfs. Đầu tiên, ta tạo một thư mục mặc định, tất cả các lệnh khác sẽ sử dụng đường dẫn quan hệ tới thư mục home mặc định này.

|  |
| --- |
| hdfs dfs -mkdir -p /user/hadoop |

Tạo thêm thư mục books bên trong hdfs.

|  |
| --- |
| hdfs dfs -mkdir books |

# CÀI ĐẶT MONGODB

## MongoDB

### Giới thiệu tổng quan về MongoDB

MongoDB là một cơ sở dữ liệu mã nguồn mở và là một cơ sở dữ liệu NoSQL hàng đầu, được hàng triệu người sử dụng. MongoDB được viết bằng C++. Ngoài ra, MongoDB là một cơ sở dữ liệu đa nền tảng, hoạt động trên các khái niệm Collection và Document, nó cung cấp hiệu suất cao, tính khả dụng cao và khả năng mở rộng dễ dàng. MongoDB sẽ tránh cấu trúc table-based của cơ sở dữ liệu quan hệ để thích ứng với các tài liệu như JSON có một schema rất linh hoạt gọi là BSON. MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích thước và các document khác nhau. Các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.

### Một số thuật ngữ hay sử dụng trong MongoDB

* **\_id**: Là trường bắt buộc có trong mỗi document. Trường **\_id** đại diện cho một giá trị duy nhất trong document MongoDB. Trường **\_id** cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì MongoDB sẽ tự động sinh ra một **\_id** đại diện cho document đó và là duy nhất trong cơ sở dữ liệu MongoDB.
* **Collection**: Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.
* **Cursor**: Đây là một con trỏ đến tập kết quả của một truy vấn. Máy khách có thể lặp qua một con trỏ để lấy kết quả.
* **Database**: Nơi chứa các Collection, giống với cơ sở dữ liệu RDMS chúng chứa các bảng. Mỗi Database có một tập tin riêng lưu trữ trên bộ nhớ vật lý. Một máy chủ MongoDB có thể chứa nhiều Database.
* **Document**: Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các trường tên và giá trị.
* **Field**: Là một cặp name – value trong một document. Một document có thể có không hoặc nhiều trường. Các trường giống các cột ở cơ sở dữ liệu quan hệ.
* **JSON**: Viết tắt của JavaScript Object Notation. Con người có thể đọc được ở định dạng văn bản đơn giản thể hiện cho các dữ liệu có cấu trúc. Hiện tại JSON đang hỗ trợ rất nhiều ngôn ngữ lập trình.
* **Index**: Là những cấu trúc dữ liệu đặc biệt, dùng để chứa một phần nhỏ của các tập dữ liệu một cách dễ dàng để quét. Chỉ số lưu trữ giá trị của một fields cụ thể hoặc thiết lập các fields, sắp xếp theo giá trị của các fields này. Index hỗ trợ độ phân tích một cách hiệu quả các truy vấn. Nếu không có chỉ mục, MongoDB sẽ phải quét tất cả các documents của collection để chọn ra những document phù hợp với câu truy vấn. Quá trình quét này là không hiệu quả và yêu cầu MongoDB để xử lý một khối lượng lớn dữ liệu.
* **Replication:** có ý nghĩa là “nhân bản”, là có một phiên bản giống hệt phiên bản đang tồn tại, đang sử dụng. Với cơ sở dữ liệu, nhu cầu lưu trữ lớn, đòi hỏi cơ sở dữ liệu toàn vẹn, không bị mất mát trước những sự cố ngoài dự đoán là rất cao. Vì vậy, người ta nghĩ ra khái niệm “nhân bản”, tạo một phiên bản cơ sở dữ liệu giống hệt cơ sở dữ liệu đang tồn tại, và lưu trữ ở một nơi khác, đề phòng có sự cố.
* **Aggregation:** Các Aggregation operation xử lý các bản ghi dữ liệu và trả về kết quả đã được tính toán. Các phép toán tập hợp nhóm các giá trị từ nhiều Document lại với nhau, và có thể thực hiện nhiều phép toán đa dạng trên dữ liệu đã được nhóm đó để trả về một kết quả duy nhất. Trong SQL, count(\*) và GROUP BY là tương đương với Aggregation trong MongoDB.
* **Lưu trữ file:** MongoDB được dùng như một hệ thống file tận dụng những function trên và hoạt động như một cách phân phối qua sharding

**Hãy lưu ý sự khác biệt của các trường và \_id trong một document. Một \_id được dùng để đại diện cho một document và chúng được sinh ra khi thêm một Document vào Collection.**

### Ưu điểm của MongoDB

* Ít schema hơn: Vì schema được sinh ra là để nhóm các đối tượng vào 1 cụm, dễ quản lý. Ví dụ như tạo 1 schema tên là Students chẳng hạn thì chỉ có những gì liên quan đến student thì mới được cho vào schema này. Trong khi đó trong mongodb thì chỉ 1 collection ta có thể chứa nhiều document khác nhau. Với mỗi document thì số trường, nội dung, kích thước lại có thể khác nhau.

Diagram

Description automatically generated

* Cấu trúc của một đối tượng rõ ràng
* Không có các join phức tạp
* Khả năng mở rộng cực lớn: việc mở rộng dữ liệu mà không phải lo đến các vấn đề như khoá ngoại, khoá chính, kiểm tra ràng buộc,… MongoDB cho phép thực hiện replication và sharding nên việc mở rộng cũng thuận lợi hơn.

Text

Description automatically generated with low confidence

* Sử dụng bộ nhớ trong để lưu giữ cửa sổ làm việc cho phép truy cập dữ liệu nhanh hơn. Việc cập nhật được thực hiện nhanh gọn nhờ update tại chỗ (in-place).

Diagram

Description automatically generated with low confidence

* Dữ liệu lưu trữ phi cấu trúc, không có tính ràng buộc, toàn vẹn nên tính sẵn sàng cao, hiệu suất lớn và dễ dàng mở rộng lưu trữ.
* Dữ liệu được caching (ghi đệm) lên RAM, hạn chế truy cập vào ổ cứng nên tốc độ đọc và ghi cao.

### Cách thức Mongo hoạt động

Diagram

Description automatically generated

MongoDB hoạt động dưới một tiến trình ngầm service, luôn mở một cổng (cổng mặc định là 27017) để lắng nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý.

Mỗi một bản ghi của MongoDB được tự động gắn thêm một field có tên “\_id” thuộc kiểu dữ liệu ObjectId mà nó quy định để xác định được tính duy nhất của bản ghi này so với bản ghi khác, cũng như phục vụ các thao tác tìm kiếm và truy vấn thông tin về sau. Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.

Mỗi khi có một truy vấn dữ liệu, bản ghi được cache (ghi đệm) lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.

Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng.

### Nhược điểm của MongoDB

* Dữ liệu được caching, lấy RAM làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ RAM lớn.
* Như đã giới thiệu ở trên, mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống ổ cứng ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao.
* Không ứng dụng được cho các mô hình giao dịch nào có yêu cầu độ chính xác cao do không có ràng buộc.
* Không có cơ chế transaction (giao dịch) để phục vụ các ứng dụng ngân hàng.

### Các trường hợp sử dụng MongoDB

* Sử dụng MongoDB trong trường hợp:
* Nếu website của bạn có tính chất INSERT cao bởi vì mặc định MongoDB có sẵn cơ chế ghi với tốc độ cao và an toàn. Website của bạn ở dạng thời gian thực nhiều, nghĩa là nhiều người thao tác với ứng dụng. Nếu trong quá trình load bị lỗi tại một điểm nào đó thì nó sẽ bỏ qua phần đó nên sẽ an toàn.
* Website bạn có nhiều dữ liệu quá. Giả sử web bạn có đến 10 triệu records thì đó là cơn ác mộng với MYSQL. Bởi vì MongoDB có khả năng tìm kiếm thông tin liên quan cũng khá nhanh nên trường hợp này nên dùng nó.
* Máy chủ không có hệ quản trị CSDL Trường hợp này thường bạn sẽ sử dụng SQLITE hoặc là MongoDB.
* Không nên sử dụng MongoDB trong trường hợp:
* Các ứng dụng cần sử dụng nhiều transaction (như ngân hàng) do Mongodb không có cơ chế transaction (giao dịch) để phục vụ cho các ứng dụng ngân hàng
* Các ứng dụng cần SQL (sử dụng joins).

## Cách cài đặt MongoDB 3 nodes replica set cluster

### Giới thiệu Replica Set trong MongoDB

Replica set trong MongoDB là một group các mongod processes để duy trì cùng một cơ sở dữ liệu. Replica set cung cấp khả năng dự phòng và tính sẵn sàng cao.

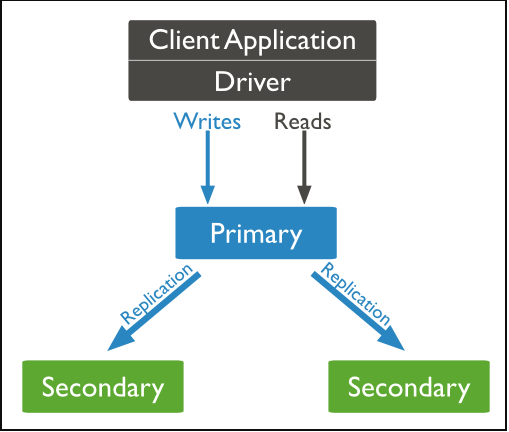
1. Tính dự phòng và sẵn sàng cao

Replication cung cấp tính dự phòng và nâng cao tính khả dụng của mongoDB. Với nhiều bản sao chép trên nhiều database server khác nhau, replication nâng cao mức độ chịu lỗi chống lại việc mất dữ liệu trên signle database.

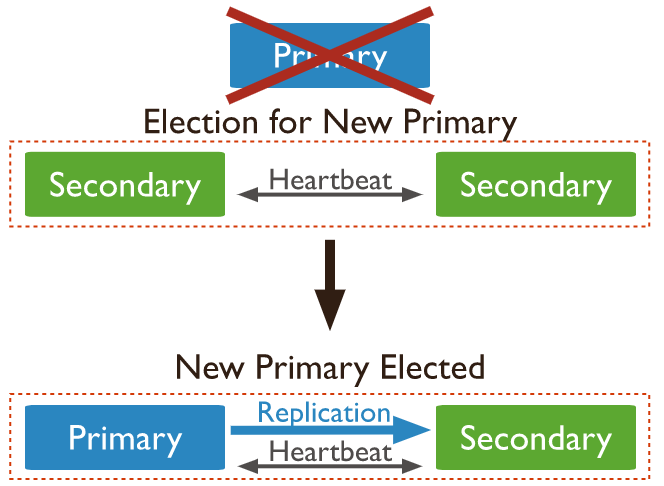
Trong một số trường hợp, replication có thể cải thiện khả năng đọc dữ liệu vì client có thể đọc dữ liệu trên nhiều database server khác nhau. Duy trì các bản sao dữ liệu làm tăng tính cục bộ và khả dụng của dữ liệu.

Các replica cũng có thể duy trì cho các mục đích như khôi phục dữ liệu, sao lưu.

1. Replication trong MongoDB



Replica set là một group các mongod instances để duy trì cùng một bộ cơ sở dữ liệu. Trong replica set bao gồm một số node mang dữ liệu, trong các nút này chỉ duy nhất một nút là nút chính (Primary) để ghi dữ liệu trong khi các nút còn lại là các nút phụ (Secondary) được sao chép dữ liệu từ nút chính. Nếu nút chính bị lỗi một trong những nút phụ sẽ được thay thế thành một nút chính mới.



Trong một số các trường hợp khác (giả sử bạn có một primary node và secondary node nhưng vì vấn đề ràng buộc nên không thể thêm được một secondary node nữa), lúc này bạn có thể chọn thêm một mongod instances vào replica set là **arbiter**. **Arbiter** tham gia bầu cử nhưng không chứa dữ liệu (không cung cấp dữ liệu thừa).

Diagram

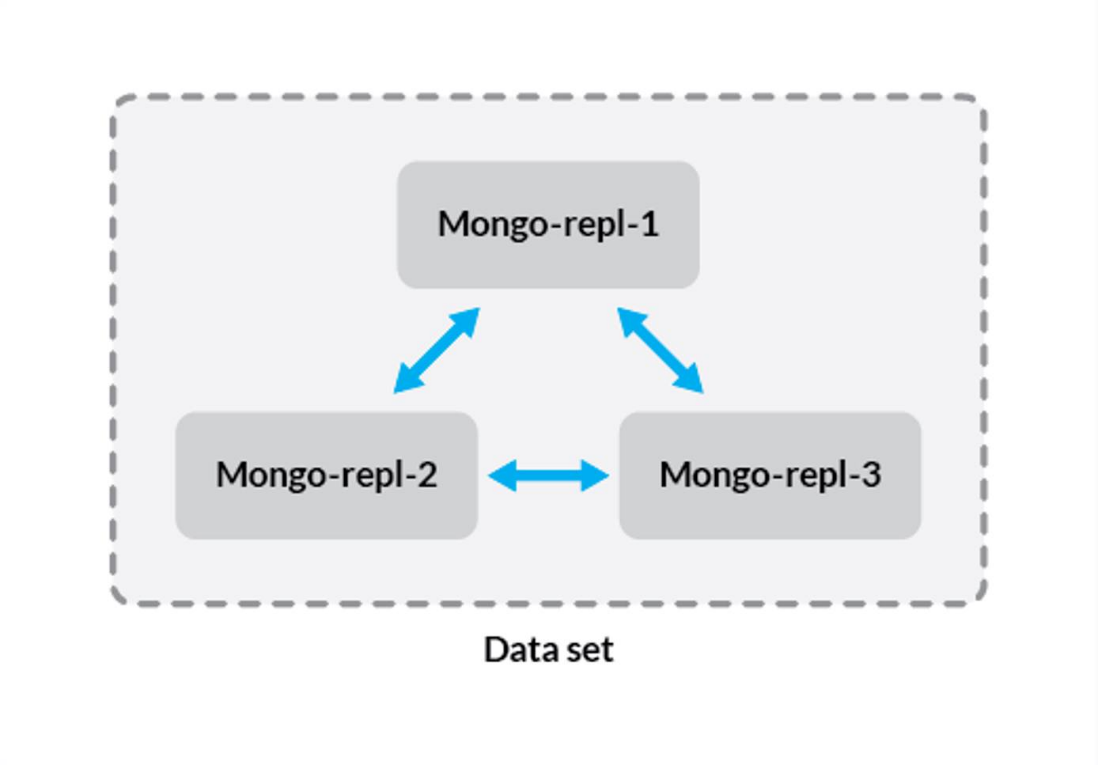
Description automatically generated

**Arbiter** sẽ giữ nguyên vai trò trong suốt quá trình bầu cử, trong khi **primary** có thể xuống trở thành **secondary** và một **secondary** có thể lên thành **primary**.

### Cấu hình trong MongoDB

#### Bước chuẩn bị

Để cho phép nhân bản nhất quán, mỗi nút sẽ cần giao tiếp với tất cả các nút khác trong cụm. Ví dụ, trong một tập hợp ba nút, truyền dữ liệu sẽ giống như sau:



Có hai cách chính để cho phép các node trong tập hợp bản sao giao tiếp nhau.  
- Phương pháp đầu tiên là sử dụng địa chỉ IP riêng cho từng node của tập hợp bản sao. Điều này cho phép các Linodes trong tập hợp bản sao giao tiếp mà không để lộ dữ liệu của mình lên internet công cộng. Phương pháp này được khuyến khích, nhưng lưu ý rằng nó yêu cầu tất cả các node của tập hợp bản sao phải ở trong cùng một trung tâm dữ liệu.

- Phương pháp thứ hai là chỉ cần sử dụng địa chỉ IP công cộng được gán cho mỗi Linode. Ta sẽ cần sử dụng phương pháp này nếu Linodes của mình được đặt ở các trung tâm dữ liệu khác nhau, mặc dù điều này không được khuyến khích vì độ trễ của mạng sẽ có tác động tiêu cực đến việc nhân rộng. Nếu phải sử dụng địa chỉ IP công cộng, ta nên định cấu hình mã hóa SSL / TLS cho dữ liệu được gửi giữa các máy chủ của mình hoặc định cấu hình chúng để giao tiếp qua VPN.

Cho dù đang sử dụng địa chỉ IP công cộng hay riêng để gửi dữ liệu, ta vẫn cần bảo mật từng Linode bằng tường lửa trước khi triển khai bộ bản sao của mình vào sản xuất.

#### Cấu hình Hostsosts Files

Mỗi thành viên của tập hợp bản sao của bạn phải có một tên máy chủ để xác định nó là thành viên của tập hợp. Bằng cách này, bạn sẽ có thể giữ cho cơ sở hạ tầng của mình được tổ chức trên quy mô lớn (ví dụ: nếu bạn thêm nhiều bộ bản sao hơn). Để đơn giản hóa cấu hình của tập hợp bản sao, hãy thêm các dòng sau vào tệp /etc/hosts trên mỗi thành viên của tập hợp bản sao:

|  |
| --- |
| 192.0.2.1 mongo-repl-1  192.0.2.2 mongo-repl-2  192.0.2.3 mongo-repl-3 |

Nếu đang sử dụng nhiều hơn ba Linodes, hãy thêm tất cả các máy chủ ở giai đoạn này. Thay thế tên máy chủ bằng tên máy chủ thực tế của bản thân và cả địa chỉ IP bằng địa chỉ IP của Linodes của mình.

#### Thiết lập MongoDB Authentication

Trong phần này, ta sẽ tạo một tệp khóa sẽ được sử dụng để bảo mật xác thực giữa các thành viên trong tập hợp bản sao của chính mình. Dưới đây, ta sẽ đi sử dụng tệp khóa được tạo bằng openssl.

*a,Tạo tệp khóa*

Tạo tệp khóa bằng lệnh:

|  |
| --- |
| openssl rand -base64 756 > mongo-keyfile |

Khi đã tạo khóa, đi sao chép khóa đó cho từng thành viên trong tập hợp bản sao.

Sau đó ta sẽ thực hiện trên từng thành viên của tập hợp bản sao, để tất cả chúng đều có tệp khóa nằm trong cùng một thư mục, với các quyền giống hệt nhau. Tạo file /opt/mongo lưu tệp khóa.

|  |
| --- |
| sudo mkdir /opt/mongo |

Giả sử rằng tệp khóa của ta nằm trong thư mục chính cho người dùng, hãy di chuyển tệp đó đến / opt / mongo và gán cho nó các quyền chính xác:

|  |
| --- |
| sudo mv ~/mongo-keyfile /opt/mongo  sudo chmod 400 /opt/mongo/mongo-keyfile |

Cập nhật quyền sở hữu tệp khóa để tệp đó thuộc về người dùng MongoDB. Sử dụng lệnh thích hợp cho bản phân phối của bạn:

|  |
| --- |
| sudo chown mongodb:mongodb /opt/mongo/mongo-keyfile |

*b, Tạo Administrative user*

Trên Linode mà ta định sử dụng làm thành viên primary của tập hợp nhân bản của mình, hãy đăng nhập vào shell mongo:

|  |
| --- |
| mongo |

Kết nối với admin database:

|  |
| --- |
| use admin |

Tạo người dùng quản trị có đặc quyền root. Thay thế "mật khẩu" bằng một mật khẩu mạnh mà ta chọn:

|  |
| --- |
| db.createUser({user: "mongo-admin", pwd: "password", roles:[{role: "root", db: "admin"}]}) |

#### Cấu hình MongoDB

Trên mỗi Linodes, ta hãy thực hiện các thay đổi sau đối với tệp /etc/mongod.conf:

|  |
| --- |
| net:  port: 27017  bindIp: 127.0.0.1,192.0.2.1  security:  keyFile: /opt/mongo/mongo-keyfile  replication:  replSetName: rs0 |

Cổng mặc định là 27017. Chỉ thị bindIp chỉ định địa chỉ IP mà daemon MongoDB sẽ lắng nghe và vì ta đang kết nối một số máy chủ, đây phải là địa chỉ IP tương ứng với Linode mà bạn đang định cấu hình nó (cùng một địa chỉ đã được thêm vào tệp máy chủ trong phần trước). Để mặc định 127.0.0.1 cũng cho phép bạn kết nối cục bộ, điều này có thể hữu ích cho việc thử nghiệm sao chép.

Bỏ thông báo phần security và sử dụng tùy chọn keyFile để hướng MongoDB đến khóa ta đã tạo trước đó.

Phần replication cần được bỏ ghi chú để được kích hoạt. Các chỉ thị trong phần này là những gì ảnh hưởng trực tiếp đến cấu hình của bộ bản sao của bạn. Giá trị rs0 là tên đang sử dụng cho tập hợp bản sao. Có thể đặt tên khác nếu muốn.

Khi đã thực hiện những thay đổi này, hãy khởi động lại dịch vụ mongod:

|  |
| --- |
| sudo systemctl restart mongod |

#### Bắt đầu Replication và thêm thành viên (node)

*a, Kết nối qua SSH với Linode mà ta định sử dụng làm primary*

Khi bạn đã đăng nhập, hãy kết nối với shell MongoDB bằng người dùng quản trị mà bạn đã tạo trước đó:

|  |
| --- |
| mongo -u mongo-admin -p --authenticationDatabase admin |

*b, Từ mongo shell, tập hợp replica set:*

|  |
| --- |
| rs.initiate() |

*c, Trong khi vẫn được kết nối với mongo shell, hãy thêm các máy chủ khác vào tập hợp bản sao:*

|  |
| --- |
| rs.add("mongo-repl-2")  rs.add("mongo-repl-3") |

Nếu đã cấu hình các thành viên khác cho tập hợp bản sao của mình, hãy thêm chúng bằng cách sử dụng lệnh tương tự và tên máy chủ mà ta đã đặt trong tệp /etc/hosts của mình.

*d, Khi tất cả các thành viên đã được thêm vào, hãy kiểm tra cấu hình của bộ bản sao của mình:*

|  |
| --- |
| rs.conf() |

Điều này sẽ hiển thị một đối tượng cấu hình tập hợp bản sao với thông tin về từng thành viên cũng như một số siêu dữ liệu về tập hợp bản sao.

*e, Ta có thể kiểm tra trạng thái của bộ bản sao của mình:*

|  |
| --- |
| rs.status() |

# THUẬT TOÁN TF-IDF

## TF-IDF

**TF-IDF** (Term Frequency – Inverse Document Frequency) là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện trong văn bản nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu. Một vài biến thể của TF-IDF thường được sử dụng trong các hệ thống tìm kiếm như một công cụ chính để đánh giá và sắp xếp văn bản dựa vào truy vấn của người dùng. TF-IDF cũng được sử dụng để lọc những từ stopwords trong các bài toán như tóm tắt văn bản và phân loại văn bản.

**TF-IDF = TF x IDF**

Trong đó:

TF-IDF: Trọng số của từ trong cụm văn bản

TF: Tần suất xuất hiện của từ

IDF: Nghịch đảo tần suất của văn bản

## TF

**TF**: Term Frequency (Tần suất xuất hiện của từ) là số lần từ xuất hiện trong văn bản. Vì các văn bản có thể có độ dài ngắn khác nhau nên một số từ có thể xuất hiện nhiều lần trong một văn bản dài hơn là một văn bản ngắn. Như vậy, term frequency thường được chia cho độ dài văn bản (tổng số từ trong một văn bản).

Trong đó:

tf(t, d): Tần suất xuất hiện từ t trong văn bản d

f(w, d): Số lần xuất hiện của từ t trong văn bản d

nd: Tổng số từ có trong văn bản d

## IDF

**IDF**: Inverse Document Frequency(Nghịch đảo tần suất của văn bản), giúp đánh giá tầm quan trọng của một từ . Khi tính toán TF , tất cả các từ được coi như có độ quan trọng bằng nhau. Nhưng một số từ như “is”, “of” và “that” thường xuất hiện rất nhiều lần nhưng độ quan trọng là không cao. Như thế chúng ta cần giảm độ quan trọng của những từ này xuống.

Trong đó:

idf(t, D) : giá trị nghịch đảo tần suất của từ t trong tập văn bản

|D| : Tống số văn bản trong tập D

: thể hiện số văn bản trong tập D có chứa từ t

Cơ số logarit trong công thức này không thay đổi giá trị idf của từ mà chỉ thu hẹp khoảng giá trị của từ đó. Vì thay đổi cơ số sẽ dẫn đến việc giá trị của các từ thay đổi bởi một số nhất định và tỷ lệ giữa các trọng lượng với nhau sẽ không thay đổi. (nói cách khác, thay đổi cơ số sẽ không ảnh hưởng đến tỷ lệ giữa các giá trị IDF). Việc sử dụng logarit nhằm giúp giá trị tf-idf của một từ nhỏ hơn, do chúng ta có công thức tính tf-idf của một từ trong 1 văn bản là tích của tf và idf của từ đó.

# THỰC NGHIỆM

## Thực nghiệm Hadoop

Bắt đầu chúng ta sẽ đi khởi động các node trong hadoop: namenode, datanode, secondary namenode, resourcemanager, nodemanager.

|  |
| --- |
| start-master.sh  start-all.sh |

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ta đi liệt kê các file có trong hadoop bằng lệnh:

|  |
| --- |
| hadoop fs -ls |

Ảnh có chứa văn bản, màn hình, ảnh chụp màn hình, thiết bị điện tử

Mô tả được tạo tự động

Thêm file vào hadoop, ta sẽ đi thực hiện bằng dòng:

|  |
| --- |
| hadoop fs -put /root/file1.txt |

Ảnh có chứa văn bản

Mô tả được tạo tự động

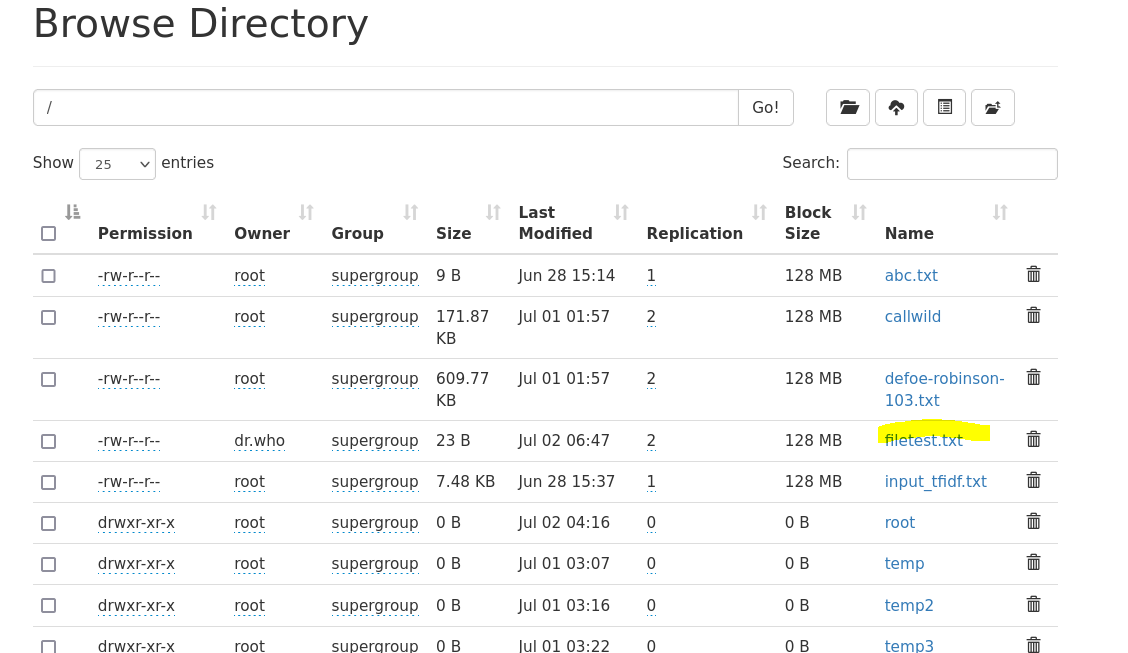
Ta sẽ đi đọc file trong hadoop xem file có chạy không:

|  |
| --- |
| hadoop fs -tail filetest.txt |

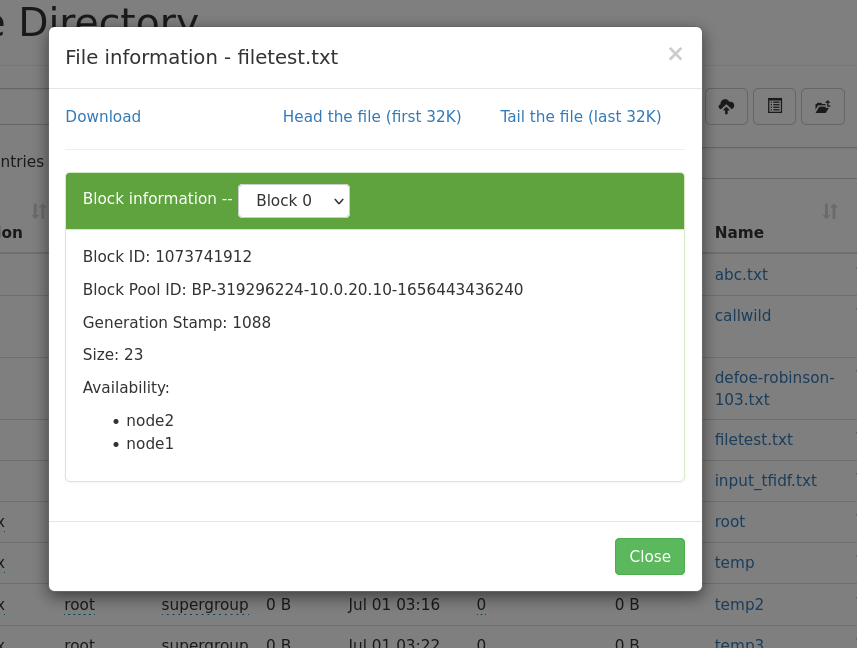
Ảnh có chứa văn bản

Mô tả được tạo tự động

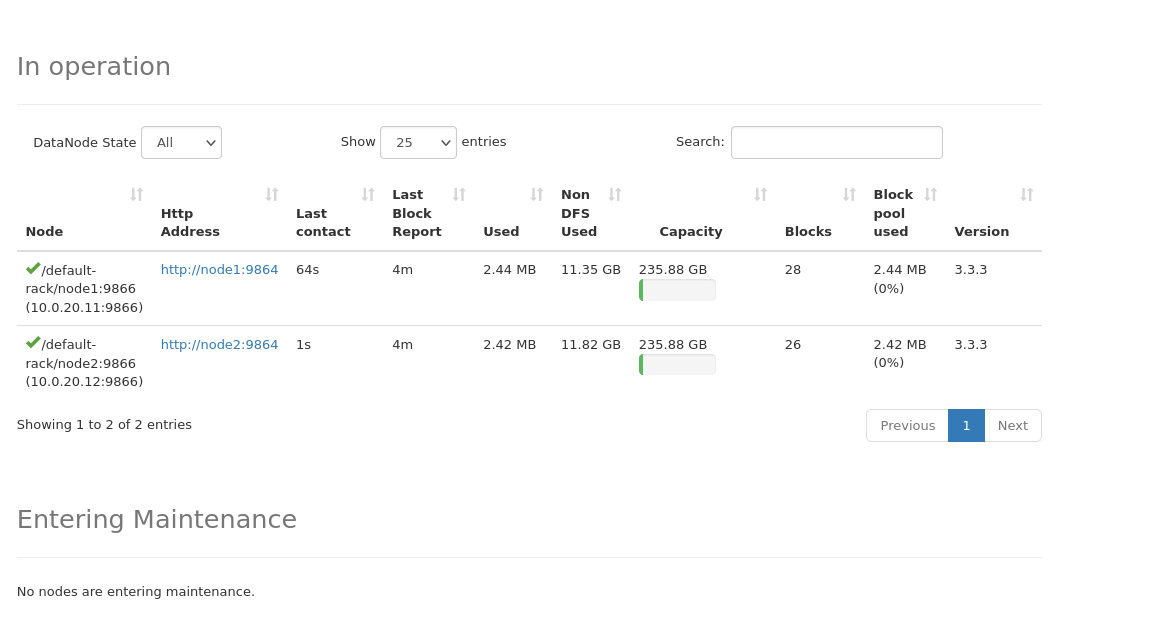
Thực hiện upfile bằng cách lên Web UI thực hiện:



Sau khi upfile thành công ta sẽ có được cửa sổ thông báo:



Kiểm tra việc tắt node, có thể thấy phần last contact của node1 đã trễ không gửi tín hiệu heartbeat đến namenode và coi node đó đã chết.



## Thực nghiệm MongoDB

Khi ta đã khởi tạo xong các node, ta đi xem trạng thái cài đặt các node trong MongoDB replication

|  |
| --- |
| rs.status |

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Thực hiện việc xem dữ liệu trong node ta sẽ có các lệnh:

|  |
| --- |
| show dbs  show collections |

Ảnh có chứa văn bản

Mô tả được tạo tự động

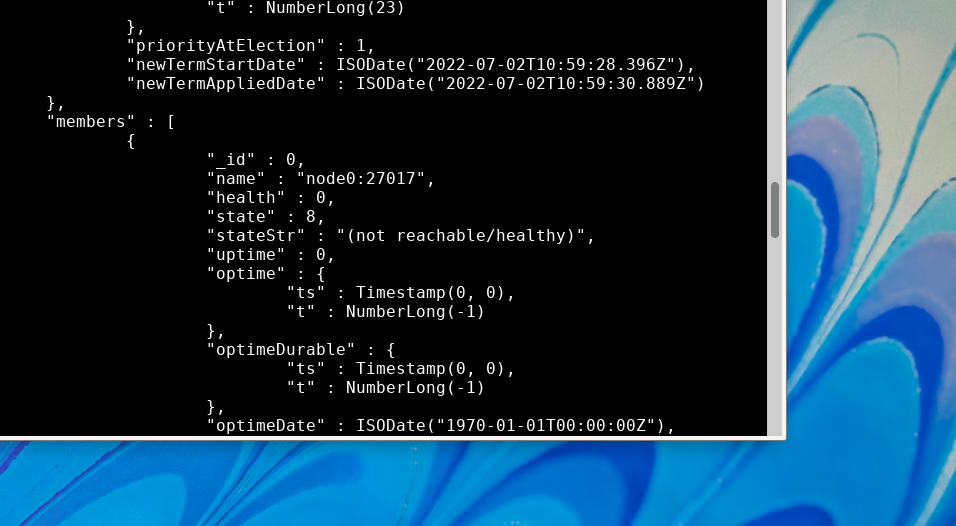
Sau khi xem dữ liệu trong Primary node ta đi đăng nhập vào xem dữ liệu trong Secondary node bằng lệnh:

|  |
| --- |
| db.getMongo().setSecondaryOk() |

Ảnh có chứa văn bản

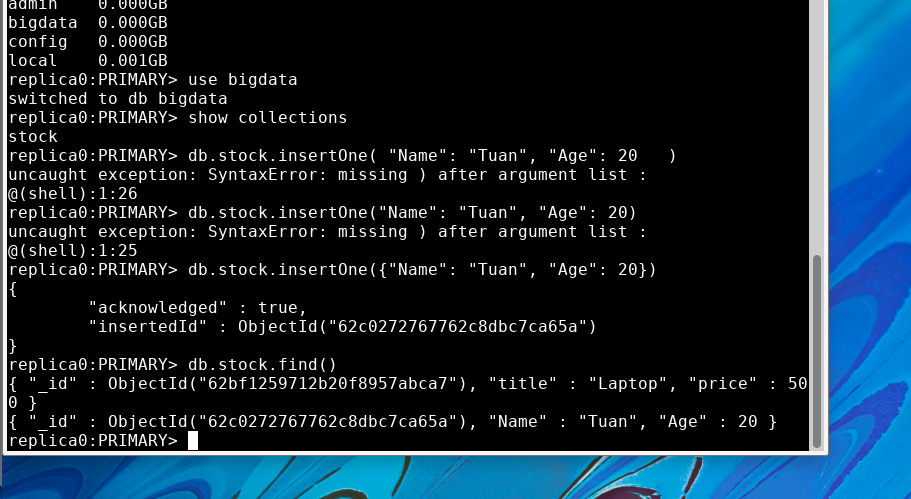
Mô tả được tạo tự động

Sau khi thực hiện việc tắt node primary thì thấy một node secondary lên thay thế và thành công.



Tiếp đến xem node mới thay thế primary có thực hiện thêm dữ liệu không?

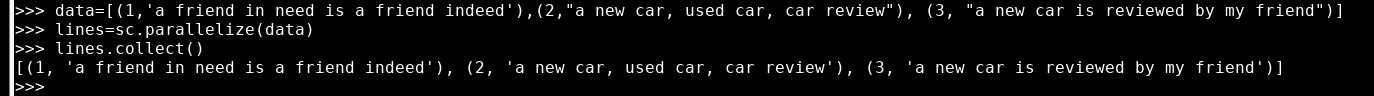
Có thể thấy việc thêm dữ liệu sau khi tắt node thành công.



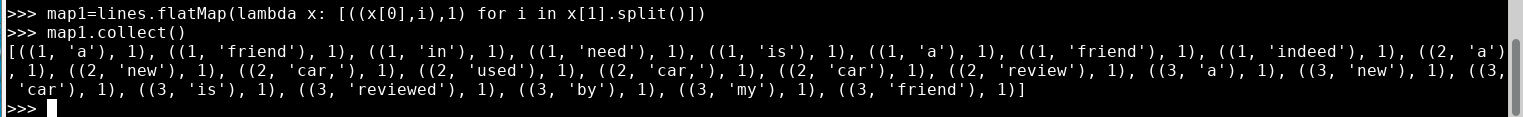
## Thuật toán TF-IDF trong MapReduce

### Tính Term Frequency (TF)

TF: Term Frequency(Tần suất xuất hiện của từ) là số lần từ xuất hiện trong văn bản. Đầu tiên ta sẽ đi tạo dữ liệu như sau:

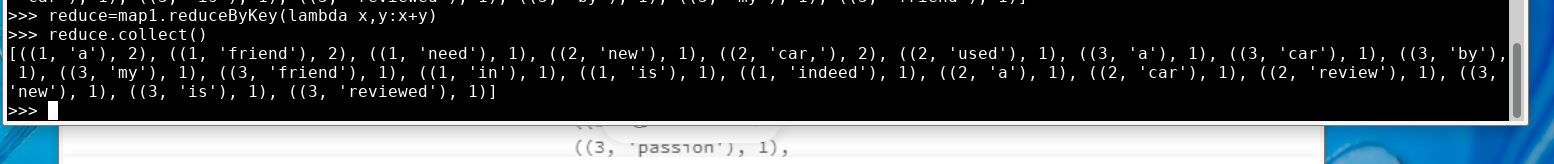


Bước 1: Mapping key/value thành một cặp key/value mới  
Ở bước này, ta sẽ ánh xạ key/value hiện có của mình thành một cặp key/value mới bao gồm tài liệu-id và mã thông báo làm khóa và 1 (đại diện cho số lượng) là giá trị. Sử dụng phép chuyển đổi flatMap ở đây để kết hợp tất cả các mã thông báo trong một danh sách duy nhất.



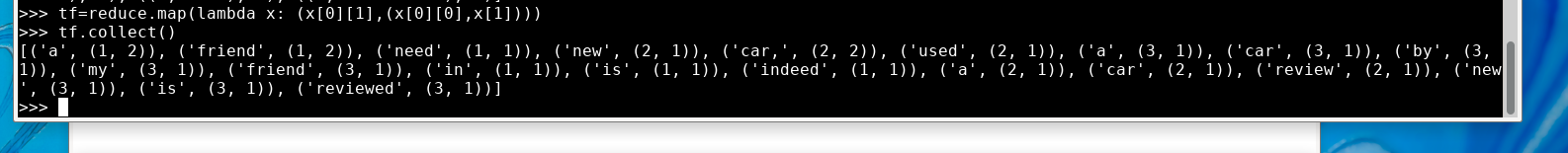
Bước 2: Reducing key/value

Bước này, ta sẽ nhóm các cặp key/value với khóa chung và tổng hợp thêm các giá trị cho cùng một khóa để có được tần suất thuật ngữ cho một từ cụ thể tương ứng với tài liệu-id của nó.



Bước 3: Mapping lại

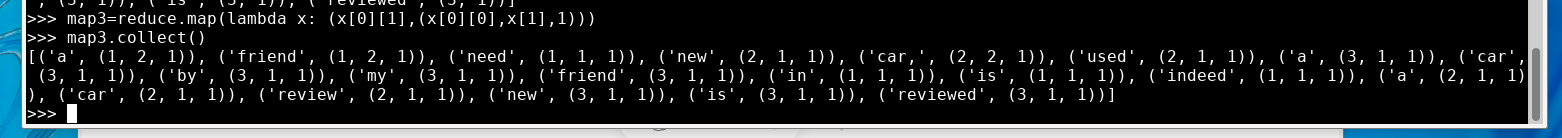
Trong bước này, chúng ta sẽ thay đổi các cặp key/value thành một tập hợp key/value mới với mã thông báo là khóa và id tài liệu và tần suất thuật ngữ (TF) tương ứng làm giá trị.



### Tính Inverse Document Frequency (IDF)

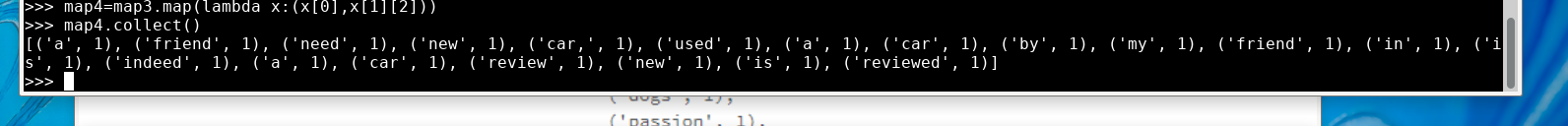
Bước 1: Mapping

Bước này, ta sẽ ánh xạ (mapping) cặp key/value trước đó thành một cặp key/value mới. Ở đây, khóa sẽ là mã thông báo và giá trị của nó sẽ là id tài liệu TF cho mã thông báo đó cùng với bộ đếm là 1. Cái này ở đây cho biết sự hiện diện của một từ trong id tài liệu được liên kết với cái đó.



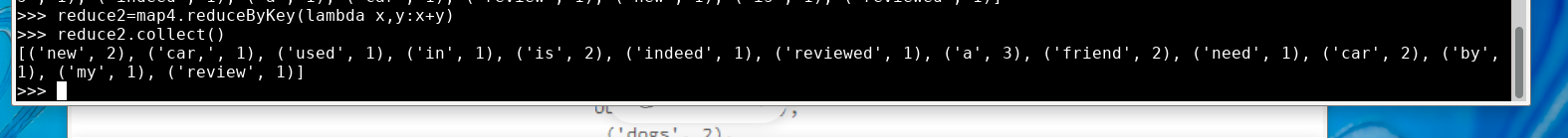
Bước 2: Mapping

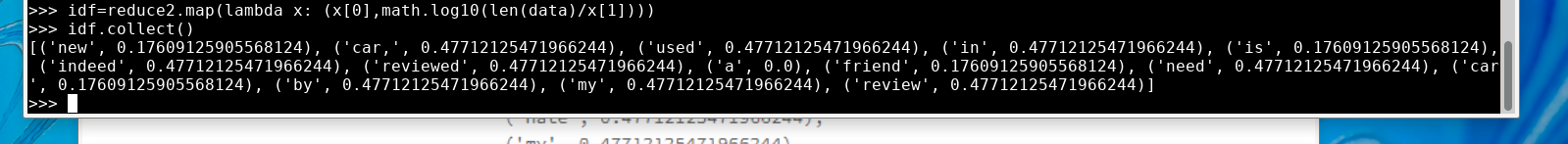
Trong bước này, sẽ trích xuất mã thông báo và số bộ đếm là 1 đại diện cho sự xuất hiện của nó trong một số tài liệu nhất định.



Bước 3: Reducing theo key

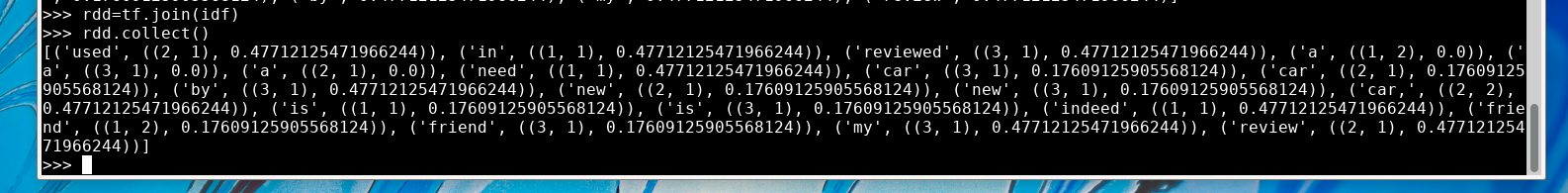
Ta sẽ giảm theo từng key để có được số lượng tài liệu có chứa một mã thông báo cụ thể w.



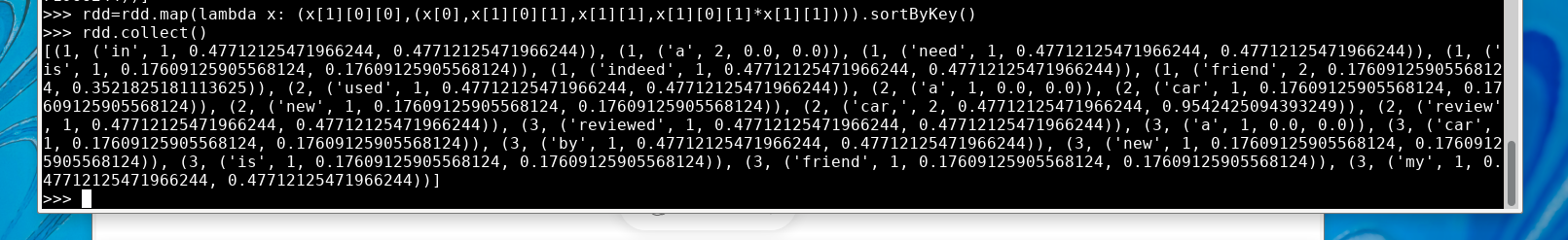
Bây giờ vì đã có số lượng tài liệu chứa mỗi mã thông báo w, chúng ta sẽ chỉ ánh xạ đầu ra cuối cùng này với phép biến đổi logarit để tính điểm IDF.

### Tính TF-IDF

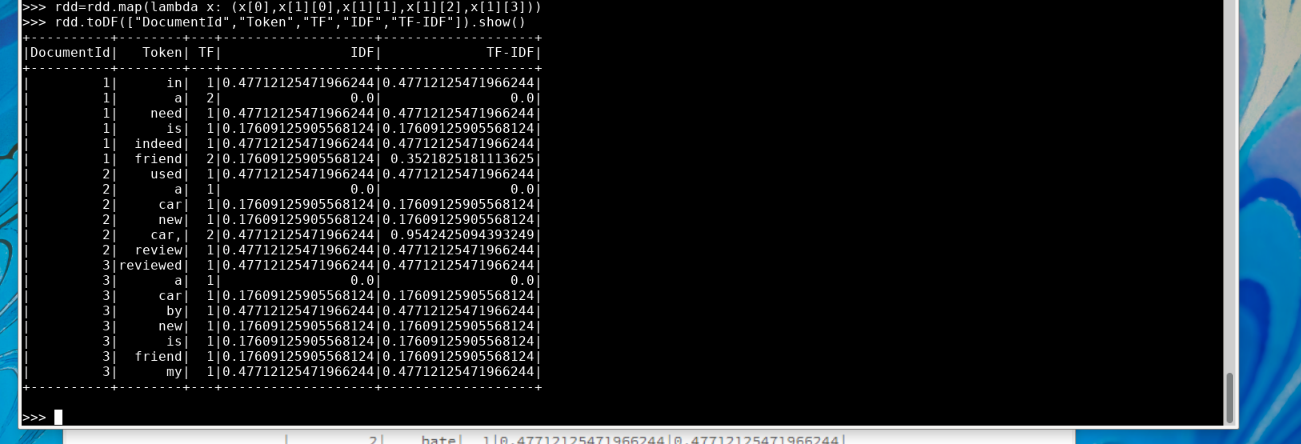
Ta sẽ thực hiện một phép nối bên trong để chỉ định mỗi mã thông báo với một id tài liệu, TF và điểm IDF.



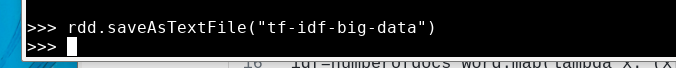
Giờ chúng ta sẽ ánh xạ hàm này để nhân các giá trị TF và IDF của mỗi mã thông báo được liên kết với id tài liệu tương ứng.



Sau đó, ta chuyển đổi kết quả sang khung dữ liệu Pyspark để trực quan hóa điểm số rõ ràng hơn và so sánh với điểm số TF-IDF thực tế được tính toán theo lý thuyết.



Cuối cùng, ta đi lưu kết quả ra file.



# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Slides môn Nhập môn dữ liệu lớn - Ths.Nguyễn Đình Quý |
| [2] | [TF-IDF là gì? Code demo thuật toán TF-IDF với dữ liệu tiếng Việt (nguyenvanhieu.vn)](https://nguyenvanhieu.vn/tf-idf-la-gi/#tf-idf-la-gi) |
| [3] | https://towardsdatascience.com/tf-idf-calculation-using-map-reduce-algorithm-in-pyspark-e89b5758e64c?gi=ed37763bbda1 |
| [4] | Điện toán đám mây và Bài toán xử lí dữ liệu lớn theo mô hình ánh xạ - rút gọn - Tạp chí Khoa học Trường Đại học Cần Thơ - Khoa Công nghệ Thông tin & Truyền thông, Trường Đại học Cần thơ - Trần Cao Đệ |