

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN



PHẠM HOÀNG QUANG CẢNH: 17133003

NGUYỄN KHÁNH DUY: 17133007

PHẠM QUỐC ĐỆ: 17133013

Đề tài:

**KHAI PHÁ LUẬT
SỬ DỤNG THUẬT TOÁN APRIORY
TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG**

KHÓA LUẬN TỐT NGHIỆP

GIẢNG VIÊN HƯỚNG DẪN

TS. NGUYỄN THÀNH SƠN

KHÓA 2017-2021

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN



PHẠM HOÀNG QUANG CẢNH: 17133003

NGUYỄN KHÁNH DUY: 17133007

PHẠM QUỐC ĐỆ: 17133013

Đề tài:

**KHAI PHÁ LUẬT
SỬ DỤNG THUẬT TOÁN APRIORY
TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG**

KHÓA LUẬN TỐT NGHIỆP

GIẢNG VIÊN HƯỚNG DẪN

TS. NGUYỄN THÀNH SƠN

KHÓA 2017-2021

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên Sinh viên 1: Phạm Hoàng Quang Cảnh MSSV: 17133003

Họ và tên Sinh viên 2: Nguyễn Khánh Duy MSSV: 17133007

Họ và tên Sinh viên 3: Phạm Quốc Đệ MSSV: 17133013

Ngành: Kỹ thuật dữ liệu

Tên đề tài: *Khai phá luật sử dụng thuật toán Apriori trong môi trường xử lý song song.*

Họ và tên Giảng viên hướng dẫn: *TS. Nguyễn Thành Sơn*

NHẬN XÉT

1. Về nội dung đề tài khối lượng thực hiện:

.....

.....

.....

.....

.....

2. Ưu điểm:

.....

.....

.....

.....

.....

3. Khuyết điểm

.....

.....

.....

.....

.....

4. Đề nghị cho bảo vệ hay không ?

5. Đánh giá loại :

6. Điểm :

Tp. Hồ Chí Minh, ngày tháng 07 năm 2021

Giáo viên hướng dẫn

(Ký & ghi rõ họ tên)

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Họ và tên Sinh viên 1: Phạm Hoàng Quang Cảnh MSSV: 17133003

Họ và tên Sinh viên 2: Nguyễn Khánh Duy MSSV: 17133007

Họ và tên Sinh viên 3: Phạm Quốc Đệ MSSV: 17133013

Ngành: Kỹ thuật dữ liệu

Tên đề tài: *Khai phá luật sử dụng thuật toán Apriori trong môi trường xử lý song song.*

Họ và tên Giảng viên hướng dẫn: *TS. Nguyễn Thành Sơn*

NHẬN XÉT

1. Về nội dung đề tài khối lượng thực hiện:

.....
.....
.....
.....
.....

2. Ưu điểm:

.....
.....
.....
.....
.....

3. Khuyết điểm

.....
.....
.....
.....
.....
.....

4. Đề nghị cho bảo vệ hay không ?

5. Đánh giá loại :

6. Điểm :

Tp. Hồ Chí Minh, ngày tháng 07 năm 2021

Giáo viên hướng dẫn

(Ký & ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành tốt đề tài Khóa luận tốt nghiệp kỹ sư CNTT, ngoài sự nỗ lực của bản thân, sự góp sức các thành viên trong nhóm, chúng tôi còn nhận được sự quan tâm giúp đỡ của nhiều tập thể và cá nhân.

Trước hết, chúng tôi xin gửi tới toàn thể quý thầy, cô trong Khoa Công Nghệ Thông Tin, cùng thầy cô trong trường Đại học Sư phạm Kỹ Thuật TP.HCM lời cảm ơn chân thành nhất. Đặc biệt, chúng tôi xin bày tỏ lòng biết ơn sâu sắc tới Giảng viên hướng dẫn TS. Nguyễn Thành Sơn người đã tận tâm hướng dẫn chúng tôi trong suốt quá trình học tập và hoàn thiện đề tài. Cảm ơn các bạn sinh viên trong lớp đã hỗ trợ, góp ý chúng tôi trong quá trình thực hiện đề tài.

Do kiến thức còn hạn chế và khả năng áp dụng thực tế chưa nhiều nên bài báo cáo sẽ còn rất nhiều sai sót, kính mong sự góp ý đến từ quý thầy cô.

Cuối cùng, chúng tôi kính chúc quý thầy, cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý. Đồng kính chúc các bạn khoa Công Nghệ Thông Tin khóa 2017 ngành Kỹ Thuật Dữ Liệu sẽ luôn dồi dào sức khỏe, đạt được nhiều thành công tốt đẹp trong công việc.

TP. HCM, ngày tháng 07 năm 2021

Nhóm trưởng

Phạm Quốc Đệ

MỤC LỤC

LỜI CẢM ƠN	5
DANH MỤC VIẾT TẮT	8
DANH MỤC HÌNH ẢNH	9
DANH MỤC BẢNG BIỂU	10
LỜI MỞ ĐẦU	11
CHƯƠNG 1: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU	15
1.1. Khai phá dữ liệu	15
1.2. Kiến trúc data mining	16
CHƯƠNG 2: LUẬT KẾT HỢP VÀ THUẬT TOÁN APRIORI	19
2.1. Luật kết hợp	19
2.2. Bài toán phát hiện luật kết hợp	21
2.2.1. Phát biểu bài toán	21
2.2.2. Phát hiện tập mục phổ biến	22
2.3. Giải thuật Apriori	22
2.3.1. Nguyên tắc của thuật toán Apriori	22
2.3.2. Các thuật toán thuộc họ Apriori	27
CHƯƠNG 3: TÌM HIỂU THUẬT TOÁN APRIORI TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG	28
3.1. Giới thiệu tổng quan	28
3.2. Triển khai thuật toán Apriori cho tính toán song song	28
CHƯƠNG 4: TÌM HIỂU THUẬT TOÁN APRIORI CẢI TIẾN	32
4.1. Ý tưởng của thuật toán Apriori cải tiến	32
4.2. Thuật toán Apriori cải tiến	33
4.3. Vận dụng thuật toán Apriori cải tiến	34
CHƯƠNG 5: TÌM HIỂU VỀ HADOOP VÀ MAPREDUCE	36
5.1. Tìm hiểu về Hadoop	36
5.2. Vai trò của Hadoop	36
5.3. Nguyên tắc hoạt động của Hadoop	37
5.4. Tìm hiểu về Hadoop framework	37
5.5. Tìm hiểu về MapReduce	38
5.6. Ưu điểm của MapReduce	39
5.7. Nguyên tắc hoạt động	39
CHƯƠNG 6: THỰC HÀNH CÀI ĐẶT VÀ TRIỂN KHAI HADOOP VÀ MAPREDUCE	40

6.1. Yêu cầu về cấu hình máy, phần mềm cần chuẩn bị.....	40
6.2. Các bước cài đặt Hadoop và Mapreduce.....	40
6.3. Chạy ví dụ về Hadoop Mapreduce	52
CHƯƠNG 7: TRIỂN KHAI THUẬT TOÁN APRIORI CẢI TIẾN TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG.	55
7.1. Tóm tắt quá trình thực thi	55
7.2. Cài đặt	58
CHƯƠNG 8: ĐÁNH GIÁ THỰC NGHIỆM.....	61
8.1. Đánh giá kết quả	61
8.2. Thời gian thực thi giữa các thuật toán.....	61
8.3. So sánh thời gian thực thi giữa thuật toán Apriori và Apriori cải tiến	63
8.4. So sánh thời gian thực thi thuật toán Apriori trên môi trường xử lý song song.	66
KẾT LUẬN	69
TÀI LIỆU THAM KHẢO	71

DANH MỤC VIẾT TẮT

C_k	Tập các k-itemset ứng viên (Candidate sets)
CSDL	Cơ sở dữ liệu
Conf	Độ tin cậy (Confidence)
D	Cơ sở dữ liệu giao dịch
D_i	Phần thứ I của cơ sở dữ liệu D
Item	Mục
Itemset	Tập mục
k-itemset	Tập mục gồm k mục
L_k	Tập các k-itemset phổ biến
minconf	Ngưỡng tin cậy tối thiểu(minimum confidence)
minsup	Ngưỡng hỗ trợ tối thiểu(minimum)
σ	Số đếm hỗ trợ (Support count)
Sup	Độ hỗ trợ (Support)
T	Giao dịch(Transaction)
TID	Định danh của giao dịch(Unique Transaction Identifier)
Tid_list	Danh sách các định danh của giao dịch
$X \Rightarrow Y$	Luật kết hợp (Với X là tiền đề, Y là hệ quả)

DANH MỤC HÌNH ẢNH

Hình 1.1. Quá trình khai phá dữ liệu	15
Hình 1.2. Kiến trúc data mining	16
Hình 2.1. Mô tả quá trình của thuật toán Apriori	23
Hình 3.1. Số lượng quy trình phân phối	30
Hình 3.2. Phân phối Trie để tạo ra k-itemsets ứng viên.....	31
Hình 4.1. Mô tả cách cải tiến thuật toán Apriori	32
Hình 4.2. Tập giao dịch thực thi Apriori cải tiến.....	34
Hình 4.3. Sinh ra 1-itemset thuật toán Apriori cải tiến.....	34
Hình 4.4. Sinh ra 2-itemset thuật toán Apriori cải tiến.....	35
Hình 4.5. Sinh ra 3-itemset thuật toán Apriori cải tiến.....	35
Hình 6.1. Cài đặt Java.....	40
Hình 6.2. Kiểm tra version java.....	41
Hình 6.3. Cài đặt thông số	42
Hình 6.4. Cấu hình hostname.....	43
Hình 6.5. Cấu hình các biến môi trường.....	45
Hình 6.6. Cấu hình đúng đường dẫn cho Java.....	46
Hình 6.7. Cấu hình thuộc tính cho máy ảo	47
Hình 6.8. Cấu hình Map Reduce.....	48
Hình 6.9. Thêm các slave trên máy master.....	50
Hình 6.10. Kiểm tra các thành phần	51
Hình 6.11. Kiểm tra sự hoạt động của slave	52
Hình 6.12. Kết quả ví dụ WordCount	54
Hình 7.1. Quá trình tạo 1-itemset Apriori song song.....	56
Hình 7.2. Quá trình tạo 1-itemset Apriori song song.....	57
Hình 7.3. Tập dataset chạy ví dụ	58
Hình 7.4. Kết quả 1-itemset Apriori song song	59
Hình 7.5. Kết quả 2-itemset Apriori song song	59
Hình 7.6. Kết quả 2-itemset Apriori song song	60
Hình 8.1. Kết quả thực nghiệm về thời gian thực thi của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ thay đổi từ 1% đến 5%.	63
Hình 8.2. Kết quả thực nghiệm về số lượng transaction mà mỗi K-itemset phải duyệt trong quá trình thực thi chương trình ở thuật toán Apriori và thuật toán Apriori cải tiến.....	65
Hình 8.3. Kết quả thực nghiệm về thời gian xử lý ở mỗi K-itemset trong quá trình thực thi chương trình ở thuật toán Apriori và thuật toán Apriori cải tiến.....	66
Hình 8.4. Kết quả về thời gian thực thi chương trình trên từng cấu hình hadoop khác nhau	67

DANH MỤC BẢNG BIỂU

Bảng 8.1. Kết quả thực nghiệm về thời gian thực thi của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ thay đổi từ 1% đến 5%	62
Bảng 8.2. Kết quả về độ chính xác của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ 5%.	62
Bảng 8.3. Kết quả về số lượng duyệt các transaction khi thực hiện thuật toán Apriori và Apriori cải tiến ở mỗi K-Itemset với tập dữ liệu Mushroom.....	64
Bảng 8.4. Kết quả về thời gian thực thi chương trình trên từng cấu hình hadoop khác nhau	67
Bảng 8.5. Kết quả về trường hợp nhiều node hơn nhưng thời gian thực thi cao hơn ít node.....	68

LỜI MỞ ĐẦU

1. Giới thiệu

Ngày nay, các lĩnh vực khoa học kỹ thuật đang ngày một phát triển mạnh mẽ. Đặc biệt là ngành khoa học dữ liệu rất phát triển, nó được ứng dụng rất nhiều trong các lĩnh vực khác nhau của cuộc sống như: Giáo dục, Y tế, Kinh tế, Khoa học, Xây dựng,... Việc dùng các phương tiện tin học để tổ chức và khai thác các cơ sở dữ liệu đã phát triển từ những năm 60. Đặc biệt trong những năm gần đây vai trò của máy tính trong việc lưu trữ và xử lý thông tin ngày càng trở nên quan trọng. Bên cạnh đó các thiết bị thu thập dữ liệu tự động tương đối phát triển đã tạo ra những kho dữ liệu khổng lồ. Sự bùng nổ này đã dẫn tới một yêu cầu cấp thiết là cần có những kỹ thuật và công cụ mới để tự động chuyển đổi lượng dữ liệu khổng lồ kia thành các tri thức có ích. Mặt khác, trong môi trường cạnh tranh thì người ta ngày càng cần có thông tin với tốc độ nhanh để giúp cho việc ra quyết định và ngày càng có nhiều câu hỏi mang tính chất định tính cần phải trả lời dựa trên khối lượng dữ liệu khổng lồ đã có. Tiến hành các công việc như vậy chính là quá trình phát hiện tri thức trong cơ sở dữ liệu, trong đó kỹ thuật khai phá dữ liệu cho phép phát hiện tri thức tiềm ẩn ấy. Từ đó, các kỹ thuật khai phá dữ liệu đã trở thành một lĩnh vực thời sự của nền Công nghệ thông tin thế giới hiện nay nói chung và Việt Nam nói riêng. Rất nhiều tổ chức và công ty lớn trên thế giới đã áp dụng kỹ thuật khai phá dữ liệu vào các hoạt động sản xuất kinh doanh của mình và thu được những lợi ích to lớn.

Các kỹ thuật phát hiện tri thức và khai phá dữ liệu được thực hiện qua nhiều giai đoạn và sử dụng nhiều kỹ thuật: phân lớp (classification), phân cụm (clustering), phân tích sự tương tự (similarity analysis), tổng hợp (summarization), luật kết hợp (association rules), ... Một trong những nội dung cơ bản và phổ biến trong khai phá dữ liệu là phát hiện các luật kết hợp. Phương pháp này nhằm tìm ra các tập thuộc tính thường xuất hiện đồng thời trong cơ sở dữ liệu và rút ra các luật về ảnh hưởng của một tập thuộc tính dẫn đến sự xuất

hiện của một hoặc nhiều tập thuộc tính khác như thế nào? Do đó việc phát hiện ra các luật kết hợp là một bước rất quan trọng trong khai phá dữ liệu.

Mặt khác, hiện nay nhu cầu song hóa và xử lý phân tán là rất cần thiết bởi kích thước dữ liệu lưu trữ ngày càng lớn nên đòi hỏi tốc độ xử lý cũng như dung lượng bộ nhớ hệ thống phải đảm bảo. Vì vậy, yêu cầu cần có những thuật toán song hiệu quả cho việc phát hiện các luật kết hợp trong khai phá dữ liệu là rất cần thiết, góp phần thúc đẩy khả năng ứng dụng của việc phát hiện tri thức, hỗ trợ ra quyết định vào trong hoạt động thực tiễn.

2. Mục tiêu, đối tượng, phạm vi nghiên cứu

✓ Mục tiêu:

- Tìm hiểu bài toán khai phá luật kết hợp.
- Tìm hiểu bài toán khai phá luật kết hợp sử dụng thuật toán Apriori tuần tự và Apriori trên môi trường xử lý song song.
- Cài đặt demo khai phá luật kết hợp sử dụng các thuật toán Apriori trên các tập dữ liệu khác nhau.

✓ Đối tượng nghiên cứu:

- Luật kết hợp, khai phá luật kết hợp sử dụng thuật toán Apriori
- Thuật toán Apriori sử dụng giải thuật song song.

✓ Phạm vi nghiên cứu: Nghiên cứu thuật toán khai phá luật kết hợp và khai phá luật kết hợp sử dụng thuật toán Apriori giải thuật song song.

3. Nhiệm vụ và hướng tiếp cận của luận văn

- Nghiên cứu về thuật toán khai phá luật (*Đã thực hiện ở TLCN*).
- Nghiên cứu tìm hiểu thuật toán Apriori trên môi trường xử lý song song (*Đã thực hiện ở TLCN*).

- Nghiên cứu tìm hiểu thuật toán Apriori cải tiến.
- Cài đặt các thuật toán Apriori, Apriori cải tiến và Apriori trên môi trường xử lý song song.

4. Kết quả đạt được

- Trình bày một cách khái quát về khai phá dữ liệu và phát hiện tri thức, quy trình khai phá dữ liệu, lựa chọn phương pháp khai phá dữ liệu.
- Xây dựng và cài đặt được chương trình thử nghiệm khai phá các luật kết hợp sử dụng giải thuật song song dựa vào thuật toán Apriori để ứng dụng cho bài toán khai phá dữ liệu.

5. Cấu trúc của luận văn: Luận văn chia làm 8 chương:

Chương 1: Tổng quan về khai phá dữ liệu.

Chương này giới thiệu khái niệm khai phá dữ liệu, kiến trúc của một hệ thống khai phá dữ liệu, một số kỹ thuật trong khai phá dữ liệu, lựa chọn phương pháp khai phá dữ liệu, một số khó khăn trong khai phá dữ liệu và ứng dụng.

Chương 2: Luật kết hợp và thuật toán Apriori.

Chương này nhóm tập trung giới thiệu khái niệm và cách tiếp cận khai phá luật kết hợp. Nội dung còn có luật kết hợp cơ sở, các hướng chính mở rộng và các công trình liên quan. Nội dung cốt lõi của thuật toán Apriori.

Chương 3: Tìm hiểu Apriori trong môi trường xử lý song song.

Chương này nhóm tập trung giới thiệu khái niệm và cách thức hoạt động của thuật toán Apriori trong môi trường song song.

Chương 4: Tìm hiểu thuật toán Apriori cải tiến.

Chương này nhóm tập trung phát triển cải tiến Apriori tuần tự.

Chương 5: Tìm hiểu về Hadoop và MapReduce.

Chương này nhóm tập trung tìm hiểu về Hadoop và cách thức hoạt động của MapReduce.

Chương 6: Thực hành cài đặt và triển khai Hadoop MapReduce.

Nhóm thực hiện cài đặt Hadoop và MapReduce và thực hành ví dụ.

Chương 7: Triển khai thuật toán Apriori trong môi trường song song.

Nhóm tiến hành triển khai thuật toán Apriori trong môi trường xử lý song song.

Chương 8: Đánh giá thực nghiệm.

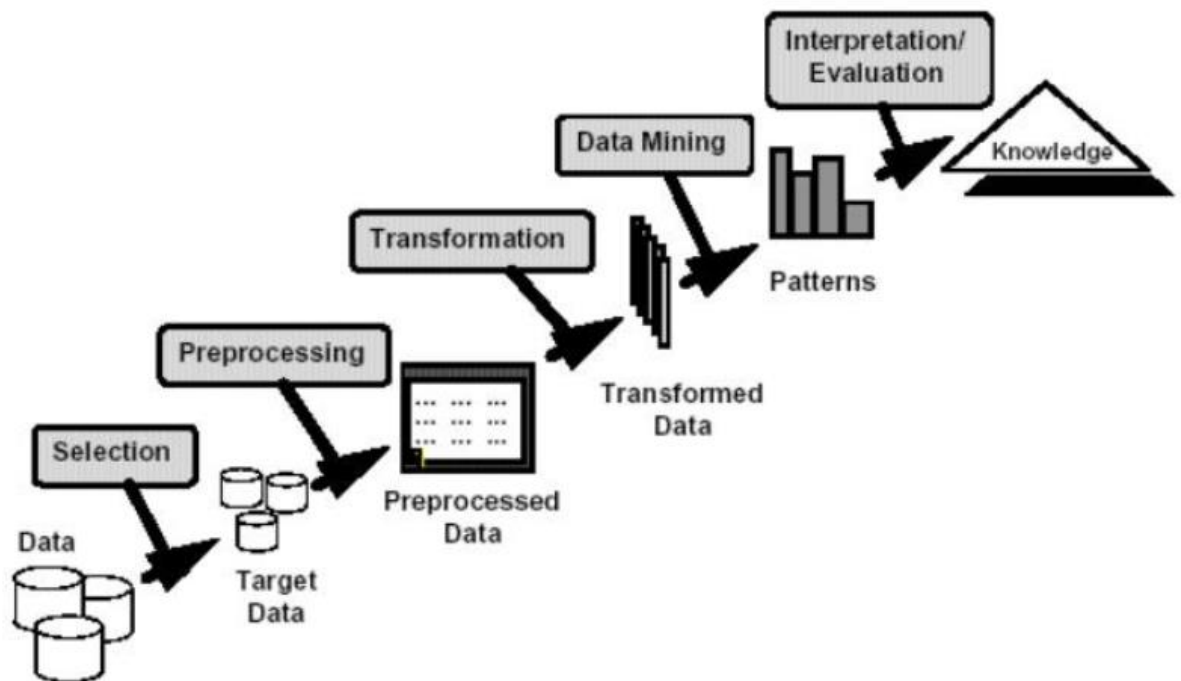
Chương này nhóm so sánh thời gian thực thi, độ chính xác giữa các thuật toán mà nhóm cài đặt trên các tập dữ liệu khác nhau.

CHƯƠNG 1: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU

1.1. Khai phá dữ liệu

Khai phá dữ liệu (Data Mining) là quá trình khám phá ra những thông tin tiềm ẩn được tìm thấy trong các cơ sở dữ liệu, góp phần cho việc khám phá tri thức từ cơ sở dữ liệu (Knowledge Discovery in Database - KDD) thông qua việc rút trích được các mẫu tri thức quan trọng từ một dữ liệu rất lớn[1].

Chức năng của khai phá dữ liệu gồm có gộp nhóm phân loại, dự báo, dự đoán và phân tích các thiết kế.



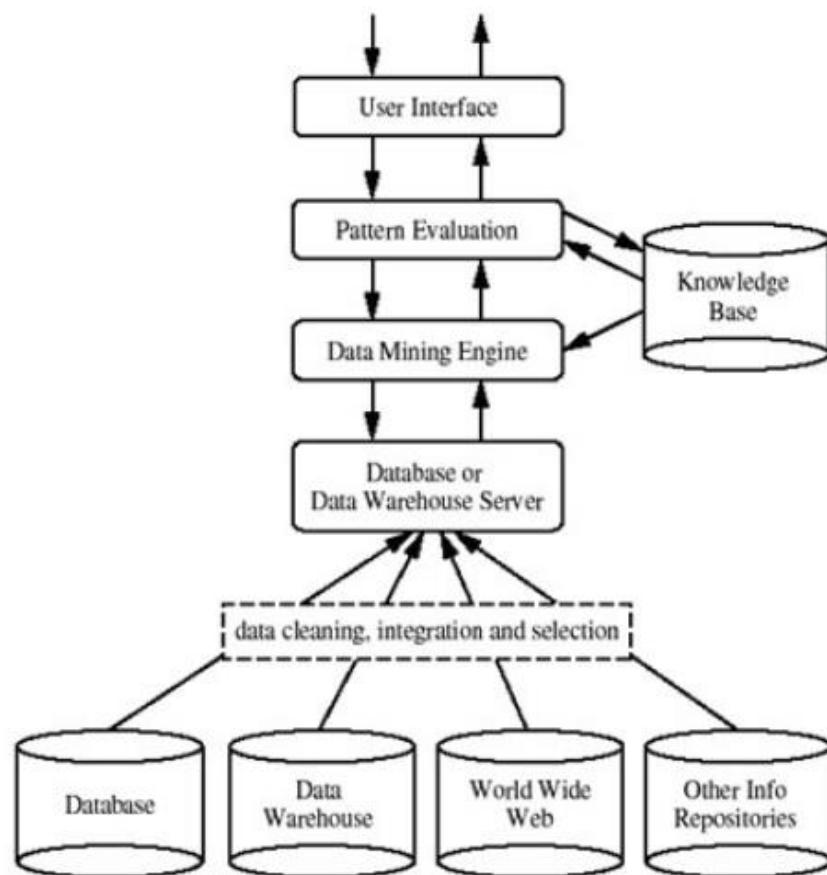
Hình 1.1. Quá trình khai phá dữ liệu

Quá trình được thực hiện qua các bước[2]:

- Tìm hiểu lĩnh vực của bài toán (ứng dụng): các mục đích của bài toán, các tri thức cụ thể của lĩnh vực.
- Tạo nên (thu thập) một tập dữ liệu phù hợp.
- Làm sạch và tiền xử lý dữ liệu.
- Giảm kích thước của dữ liệu, chuyển đổi dữ liệu: xác định tính quan trọng, giảm số chiều (số thuộc tính), biểu diễn bất biến.

- Lựa chọn chức năng khai phá dữ liệu: phân loại, gom cụm, dự báo, sinh ra các luật kết hợp.
- Lựa chọn/Phát triển các giải thuật khai phá dữ liệu phù hợp.
- Tiến hành khai phá dữ liệu.
- Đánh giá mẫu thu được và biểu diễn tri thức: hiển thị hóa, chuyển đổi, bỏ đi các mẫu dư thừa,...
- Sử dụng tri thức được khai phá

1.2. Kiến trúc data mining



Hình 1.2. Kiến trúc data mining

Dưới đây là các thành phần của một kiến trúc datamining[2]:

- Máy chủ kho dữ liệu (Database or Warehouse server): Là nơi lưu trữ dữ liệu hệ thống.
- Cơ sở tri thức (Knowledge base): Giá trị cốt lõi mong muốn.

- Máy khai phá dữ liệu (Data mining engine): Hệ thống được cài đặt bằng các modul có các thuật toán khai phá.
- Modul đánh giá mẫu (Pattern evaluation): Tích hợp các phương pháp đánh giá mẫu.
- Giao diện đồ họa cho người dùng (Graphic user interface): Giao diện web hoặc app để người dùng tương tác với hệ thống.

1.3. Thách thức trong khai phá dữ liệu

Khai phá dữ liệu gặp phải những thách thức sau đây [10]:

- Quy mô dữ liệu: trong thực tế kích thước tập dữ liệu dùng để khai phá dữ liệu thường khổng lồ thường ở mức tera-byte.
- Sự đa dạng và không đồng nhất về dữ liệu: dữ liệu đến từ nhiều nguồn khác nhau với các dạng khác nhau như có cấu trúc, bán cấu trúc, phi cấu trúc là vấn đề cực kì phức tạp.
- Tính chính xác và tin cậy: dữ liệu có thể đến từ nhiều nguồn không xác định có thể từ nguồn không tin cậy và không thể kiểm chứng.
- Bảo mật: dữ liệu riêng tư luôn là vấn đề cần xem xét trong khai phá dữ liệu. Lấy ví dụ như các giao dịch trong cuộc sống hằng ngày của chúng ta được đưa lên mạng và được lưu tại email, tin nhắn, blog, mua sắm, thanh toán trực tuyến.....Theo thời gian, các thông tin cá nhân sẽ được nằm rải rác trên mạng. Có thể một ngày nào đó sẽ bị khai thác. Vì vậy chúng ta cần phải có những chính sách đúng đắn và phương pháp tiếp cận hợp lý trong việc quản lý dữ liệu cá nhân nhưng vẫn tạo điều kiện cho hoạt động khai phá dữ liệu.

1.4. Ứng dụng của khai phá dữ liệu

- Lĩnh vực tài chính: ứng dụng trong lĩnh vực này dùng để tăng độ trung thành của khách hàng bằng cách thu thập và phân tích dữ liệu hành vi khách hàng.

- Lĩnh vực chăm sóc sức khỏe: tìm ra mối liên hệ giữa các loại bệnh và điều chỉnh phương pháp điều trị giúp thay đổi các loại thuốc đảm bảo cho bệnh nhân được chăm sóc kịp thời và phù hợp nhất.
- Lĩnh vực viễn thông: khai phá dữ liệu giúp các công ty viễn thông đạt lợi thế cạnh tranh và giảm các chi phí khách hàng bằng cách dự đoán hành vi khách hàng.
- Lĩnh vực marketing và sales: giúp các doanh nghiệp hiểu được các điều ẩn đằng sau dữ liệu giao dịch mua bán khách hàng. Phân tích thị trường và các sản phẩm thường được mua cùng nhau. Giúp quản bá sản phẩm có lợi nhuận cao nhất và tối đa hóa lợi nhuận.

Tóm lại khai phá dữ liệu có một ý nghĩa rất quan trọng trong nhiều lĩnh vực giúp cải thiện đời sống, tăng tính cạnh tranh trong kinh doanh.

CHƯƠNG 2: LUẬT KẾT HỢP VÀ THUẬT TOÁN APRIORI

2.1. Luật kết hợp

Luật kết hợp là một luật quan trọng khi ứng dụng trong khai phá dữ liệu.

Được thực hiện trên một tập các giao dịch cho trước để dự đoán khả năng xuất hiện của một giao dịch các mục item dựa trên sự xuất hiện của các mục trước đó.

Cho T là tập cơ sở dữ liệu các giao dịch t_1, t_2, \dots, t_n . $T = \{t_1, t_2, \dots, t_n\}$. Mỗi giao dịch t gồm n thuộc tính $I = \{i_1, i_2, \dots, i_n\}$.

Mục đích của luật kết hợp giữa các thuộc tính. Những luật kết hợp này có dạng $X \Rightarrow Y$. Có thể hiểu rằng những người mua mặt hàng trong tập X cũng thường mua mặt hàng trong tập Y . X được xem là biến độc lập còn Y được xem là biến phụ thuộc. Ví dụ: Nếu $X = \{\text{Bia, Nước Ngọt}\}$ và $Y = \{\text{Mực Khô, Bò Khô}\}$ và ta có luật kết hợp $X \Rightarrow Y$ thì ta có thể hiểu người mua Bia và Nước Ngọt cũng sẽ thường mua Mực Khô và Bò Khô.

Ví dụ:

Trong một cửa hàng tiện lợi, cứ khoảng 10 người mua coca cola thì có khoảng 4 người mua thêm bánh snack. Hoặc cứ 3 người mua bánh mì sẽ có 2 người mua thêm sữa tươi.

Vậy ở đây có 2 luật kết hợp là:

- Có 40% người mua coca cola sẽ mua thêm snack.
- Có 66% người mua bánh mì sẽ mua thêm sữa tươi.

Nhờ vậy mà cửa hàng sẽ biết được và đặt các mặt hàng cạnh nhau để phát triển chiến lược buôn bán của mình.

Các khái niệm cơ bản:

Đặt $I = \{i_1, i_2, \dots, i_n\}$ có n tập mục phân biệt.

D : CSDL giao dịch.

Mỗi giao dịch (transaction) $T \in D$ được định nghĩa như một tập con các mục trong I ($T \subseteq I$) và có một định danh duy nhất có dạng $\langle TID, i_1, \dots, i_k \rangle$.

Một số khái niệm trong luật kết hợp: [3]

Định nghĩa 1: Tập mục (itemset): là một tập hợp gồm một hoặc nhiều mục. Tập mục mức k (k -itemset) có k mục.

Ví dụ: 3-itemset là {Bánh mì, sữa, cafe}.

Định nghĩa 2: Luật kết hợp – kí hiệu $X \Rightarrow Y$, trong đó X, Y là các tập mục.

Ví dụ: bánh mì \Rightarrow sữa.

- Tổng số hỗ trợ (support count) - kí hiệu σ : là số lần xuất hiện của một tập mục.

- Ví dụ: $\sigma(\{\text{Bánh mì, sữa, cafe}\}) = 2$, số lần khách hàng mua cùng lúc 3 sản phẩm trong tập giao dịch D .

Định nghĩa 3: Độ hỗ trợ (support) - kí hiệu s : là tỷ lệ các giao dịch chứa cả X và Y đối với tất cả các giao dịch.

Ví dụ: $s(\{\text{Bánh mì, sữa, cafe}\}) = 2/5$. Tổng số lần khách hàng mua cả 3 sản phẩm trên tổng 5 giao dịch.

Công thức: $\text{Supp}(X) = |X| / |D|$

Độ hỗ trợ (Support) của luật kết hợp $X \Rightarrow Y$ là tần suất của giao dịch chứa tất cả các items trong cả hai tập X và Y

$$\text{Supp}(X \Rightarrow Y) = \frac{|\{T \in D \mid X \subseteq T\}|}{|D|}$$

Định nghĩa 4: Độ tin cậy (confidence) – kí hiệu c : là tỷ lệ các giao dịch chứa cả X và Y đối với các giao dịch chứa X . Ví dụ: $c(\{\text{Bánh mì, sữa,}$

cafe }) = 2/3. Tổng số lần khách hàng mua cả 3 sản phẩm trên tổng 3 giao dịch chứa sản phẩm Bánh mì

Độ tin cậy (*Confidence*) của luật kết hợp $X \Rightarrow Y$ là xác suất xảy ra Y khi đã biết X

$$\text{Conf}(X \Rightarrow Y) = \frac{\text{Supp}(X \Rightarrow Y)}{\text{Supp}(X)}$$

Định nghĩa 5: Tập mục thường xuyên (frequent/large itemset): là tập mục mà độ hỗ trợ lớn hơn hoặc bằng một giá trị ngưỡng minsup và nhiệm vụ của quá trình tìm kiếm:

- Độ hỗ trợ $s \geq$ giá trị ngưỡng **minsup**.
- Độ tin cậy $c \geq$ giá trị ngưỡng **minconf**.

Các tính chất liên quan luật kết hợp[2]:

- Nếu $X \Rightarrow Z$ và $Y \Rightarrow Z$ là thỏa mãn trên D thì không nhất thiết $X \cup Y \Rightarrow Z$ là đúng.
- Nếu luật $X \cup Y \Rightarrow Z$ thỏa mãn trên D thì không nhất thiết $X \Rightarrow Z$ và $Y \Rightarrow Z$ thỏa mãn trên D. Tuy nhiên, nếu $X \Rightarrow Y \cup Z$ thỏa mãn trên D thì suy ra được $X \Rightarrow Y$ và $X \Rightarrow Z$ cũng thỏa mãn trên D.
- Nếu $X \Rightarrow Y$ và $Y \Rightarrow Z$ thỏa mãn trên D thì không thể khẳng định $X \Rightarrow Z$ cũng thỏa mãn trên D.
- $X \Rightarrow (L - X)$ không có độ tin cậy tối thiểu thì không có luật nào trong các luật $Y \Rightarrow (L - Y)$ có độ tin cậy tối thiểu, trong đó $Y \subseteq X$; $X, Y \subset L$.

2.2. Bài toán phát hiện luật kết hợp

2.2.1. Phát biểu bài toán

Một cơ sở dữ liệu giao dịch D, cho tập các mục I, ngưỡng hỗ trợ minsup và ngưỡng tin cậy minconf. Tìm tất cả các luật kết hợp $X \Rightarrow Y$ trên CSDL D sao cho $\text{supp}(X \Rightarrow Y) \geq \text{minsup}$ và $\text{conf}(X \Rightarrow Y) \geq \text{minconf}$. Bài toán khai thác luật kết hợp có thể được chia làm 2 bài toán con được phát biểu như sau:

Nội dung thuật toán[2]

Đầu vào: I, CSDL D, minsup, minconf.

R: các luật kết hợp thỏa mãn.

Phương thức hoạt động:

(1) Tìm tập mục phổ biến từ D (tức là tập mục có độ hỗ trợ lớn hơn hoặc bằng minsup).

(2) Sinh ra các luật từ các tập mục phổ biến (large itemsets) sao cho độ tin cậy của luật lớn hơn hoặc bằng minconf.

2.2.2. Phát hiện tập mục phổ biến

Các thuật toán phát hiện tập mục phổ biến, phải thiết lập một số giai đoạn trên CSDL. Trong giai đoạn đầu, ta thực hiện tính độ hỗ trợ support cho mỗi mục riêng lẻ và xác định xem mục nào là phổ biến, nghĩa là có $\text{support} \geq \text{minsup}$.

Trong giai đoạn tiếp theo ta sinh các tập mục có khả năng là tập mục phổ biến (candidate itemset) từ tập mục phổ biến ở giai đoạn trước và tính độ hỗ trợ. Cuối mỗi giai đoạn, người ta xác định xem trong các tập mục phổ biến cho giai đoạn tiếp theo. Tiến trình này sẽ tiếp tục, cho đến khi không tìm được một tập các tập mục phổ biến mới hơn nữa.

2.3. Giải thuật Apriori

2.3.1. Nguyên tắc của thuật toán Apriori

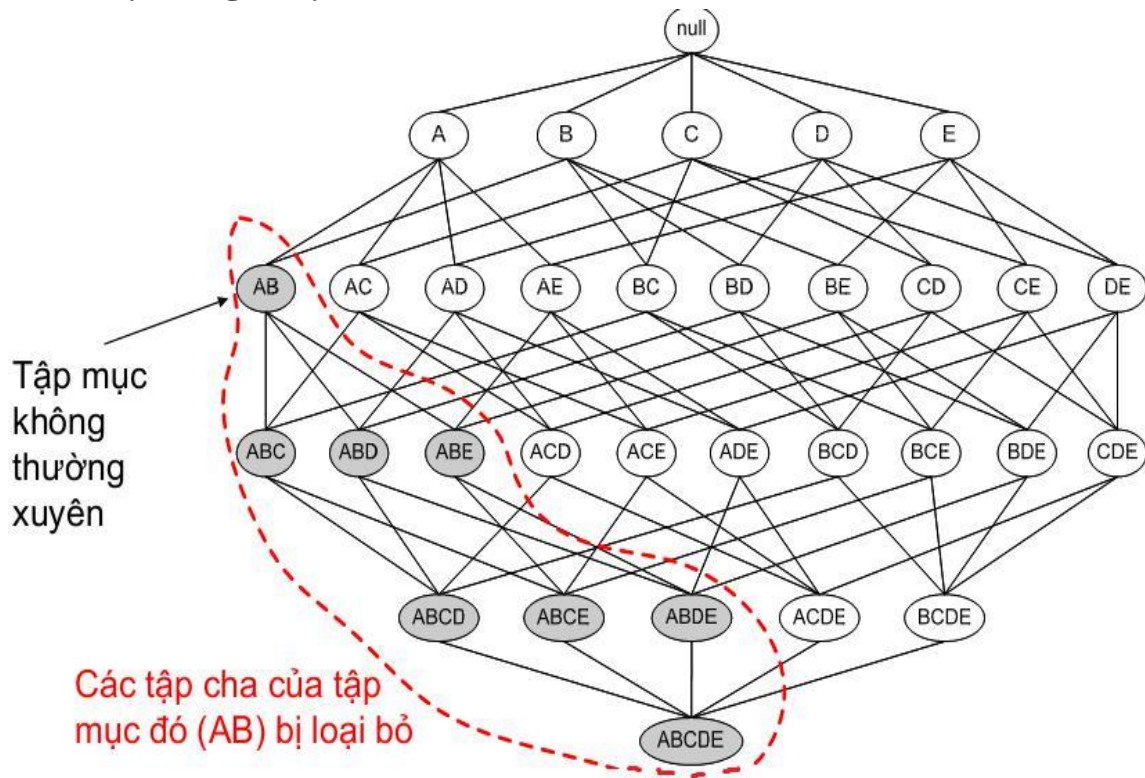
Nguyên tắc của giải thuật Apriori – Loại bỏ dựa trên độ hỗ trợ.

Nếu một tập mục là thường xuyên, thì tất cả các tập con (subsets) của nó đều là các tập mục thường xuyên. Nếu một tập mục là không thường xuyên (not frequent) thì tất cả các tập cha (supersets) của nó đều là các tập mục không thường xuyên.

Nguyên tắc của giải thuật Apriori dựa trên đặc tính không đơn điệu (anti-monotone) của độ hỗ trợ:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

Nội dung thuật toán



Hình 2.1. Mô tả quá trình của thuật toán Apriori

Dữ liệu vào: Tập các giao dịch D, ngưỡng hỗ trợ minsup.

Dữ liệu ra: Tập trả lời bao gồm các tập mục phổ biến trên D.

Phương pháp:

$L_1 = \{\text{large 1-itemset}\};$

for ($k = 2; L_{k-1} \neq \phi; k++$) **do begin**

$C_k = \text{apriori_gen}(L_{k-1});$ // sinh tập mục ứng cử mới C_k ;

forall giao dịch $t \in D$ **do begin**

$C_t = \text{subset}(C_k, t);$ // các tập mục ứng cử chứa trong t ;

forall tập mục ứng cử $c_i \in C_t$ **do**

$c_i.\text{count} ++;$

end;

$L_k = \{c_i \in C_k | c_i.\text{count} \geq \text{minsup}\}$

end;

Answer = $\cup_k L_k$;

Giải thuật Apriori:

- 1- Sinh ra tất cả các tập mục thường xuyên mức 1(frequent 1-itemsets).
- 2- Gán $k = 1$.
- 3- Lặp lại, cho đến khi không có thêm bất kỳ tập mục thường xuyên nào mới.
 - + Từ các tập mục thường xuyên mức k , sinh ra các tập mục mức $(k+1)$ cần xét.
 - + Loại bỏ các tập mục mức $k+1$ chứa các tập con là các tập mục không thường xuyên mức k .
 - + Tính độ hỗ trợ của các tập mục mức $k+1$, bằng cách duyệt qua tất cả các giao dịch.
 - + Loại bỏ các tập mục không thường xuyên mức $k+1$.
 - + Thu được các tập mục thường xuyên mức $k+1$.

Ví dụ 1:

Minsup: 0.4 (40%).

Tập cơ sở dữ liệu D:

TID (Mã giao dịch)	ItemSet (Tập các hạng mục)
T1	Sữa, bánh mì, café
T2	Coca, snack, đậu phộng
T3	Sữa, snack, bánh mì, đậu phộng
T4	Coca, đậu phộng

B1: Quét toàn bộ tập D lấy ra các tập 1-itemset và tìm kiếm trong D

Sữa: 2 – 50%

Bánh mì: 2 - 50%

Snack: 2 - 50%

Café: 1 – 25%

Coca: 2 -50%

Đậu phộng: 3- 75%

➔ Ta loại Café ra khỏi tập vì có $\text{Sup} < \text{Minsup}$ ($0.25 < 0.4$)

B2: Tạo tập 2 –itemset

Sữa, bánh mì

Sữa, snack

Sữa, coca

Sữa, đậu phộng

Bánh mì, snack

Bánh mì, coca

Bánh mì, đậu phộng

Snack, coca

Snack, đậu phộng

B3: Quét toàn bộ tập 2-itemset trong D

Sữa, bánh mì: 2 – 50%

Sữa, snack: 1 – 25%

Sữa, coca: 0 – 0%

Sữa, đậu phộng: 1 – 25%

Bánh mì, snack: 1 – 25%

Bánh mì, coca: 0 – 0%

Bánh mì, đậu phộng: 1 – 25%

Snack, coca: 1 – 25%

Snack, đậu phộng: 1 – 25%

➔ Ta loại những tập 2 –itemset < minsup

Cứ tiếp tục cho đến khi ta được kết quả toàn bộ các tập itemset
 $\geq \text{minsup}$

$L = \{\text{Sữa, bánh mì}\}$

Ví dụ 2: Ta có D là cơ sở dữ liệu giao dịch

TID (Mã giao dịch)	ItemSet (Tập các hạng mục)
T1	Táo, Nho, Mãng Cầu, Café, Sữa, Bánh Mì
T2	Táo, Nhân, Bánh Mỳ, Sữa, Cafe
T3	Ổi, Cam, Sữa, Bánh Mì
T4	Nho, Cam
T5	Café, Sữa, Bánh Mì
T6	Sữa
T7	Bánh Mì

Chọn minsupp = 30%

L1: k=1

ItemSet	Supp_Count
Táo	2
Nho	2
Mãng Cầu	1
Nhãn	1
Bánh Mì	5
Sữa	5
Café	3
Ổi	1
Cam	1

L2: k=2

ItemSet	Supp_Count
Bánh Mì, Sữa	4
Bánh Mì, Cafe	3
Café, Sữa	3

Luật	Độ tin cậy	Support
Bánh Mì => Sữa	80%	57%
Sữa => Bánh Mì	80%	57%
Bánh Mì => Cafe	60%	42,8%
Cafe => Bánh Mì	100%	42,8%
Cafe => Sữa	100%	42,8%
Sữa => Cafe	60%	42,8%

L3:k =3

ItemSet	Supp_Count
Bánh Mì, Sữa, Cafe	3

Luật	Độ tin cậy	Support
Bánh Mì, Cafe => Sữa	100%	42,8%
Bánh Mì, Sữa => Cafe	75%	42,8%
Sữa, Café => Bánh Mì	100%	42,8%

2.3.2. Các thuật toán thuộc họ Apriori

Dưới đây là một số thuật toán tiêu biểu thuộc họ Apriori và đặc điểm của chúng.

Thuật toán Apriori-TID:

Như đã biết, thuật toán Apriori quét toàn bộ CSDL trong mỗi giai đoạn để tính độ hỗ trợ. Việc quét toàn bộ CSDL có thể là không cần thiết đối với tất cả các giai đoạn. Do đó, Agrawal đã đề xuất thuật toán Apriori-TID.

Tương tự thuật toán Apriori, thuật toán Apriori-TID cũng xác định các tập mục ứng cử viên trước khi bắt đầu mỗi giai đoạn. Điểm khác nhau chủ yếu của thuật toán này so với thuật toán Apriori là nó không sử dụng CSDL để tính độ hỗ trợ trong các giai đoạn $k > 1$. Thay vào đó nó sử dụng mã khóa của các tập mục ứng cử đã sử dụng trong giai đoạn trước.

Nhiều thí nghiệm trên nhiều CSDL chỉ ra rằng thuật toán Apriori cần ít thời gian hơn giải thuật Apriori-TID trong các giai đoạn đầu, nhưng mất nhiều thời gian cho các giai đoạn sau.

Thuật toán Apriori -Hybrid

Thuật toán này dựa vào ý tưởng “không cần thiết phải sử dụng cùng một thuật toán cho tất cả các giai đoạn lên trên dữ liệu”. Như đã đề cập ở trên, thuật toán Apriori thực thi hiệu quả ở các giai đoạn đầu, thuật toán Apriori-TID thực thi hiệu quả ở các giai đoạn sau. Phương pháp của thuật toán Apriori-Hybrid là sử dụng thuật toán Apriori ở các giai đoạn đầu và chuyển sang sử dụng thuật toán Apriori-TID ở các giai đoạn sau.

Ngoài ra còn một số thuật toán thuộc họ Apriori như Dynamic Itemset Counting, thuật toán phân hoạch.

CHƯƠNG 3: TÌM HIỂU THUẬT TOÁN APRIORI TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG

3.1. Giới thiệu tổng quan

Tìm kiếm tập phổ biến là một trong những nghiên cứu các lĩnh vực khai phá dữ liệu. Thuật toán Apriori là thuật toán được thiết lập tốt nhất cho việc khai thác các tập mục phổ biến.

Một số triển khai thuật toán Apriori được báo cáo và đánh giá, trong đó việc triển khai tối ưu hóa cấu trúc dữ liệu được thực hiện bởi Bodon đã thu hút nhiều sự chú ý và kết quả triển khai để tìm kiếm các tập mục phổ biến dường như nhanh hơn so với Borgelt và Goethals.

Trong phần này, chúng ta triển khai một chương trình Apriori song song, thuật toán dựa trên công việc của Bodon và phân tích biểu diễn trên một máy tính song song. Ưu điểm việc triển khai của Bodon bằng cách sử dụng cấu trúc dữ liệu trie (cây tiền tố) hoạt động tốt hơn các triển khai khác sử dụng mảng băm.

3.2. Triển khai thuật toán Apriori cho tính toán song song

Thuật toán Apriori đã được sửa đổi theo một số cách, một trong những thay đổi của thuật toán là phân vùng cơ sở dữ liệu giao dịch thành các vùng riêng biệt $TDB_1, TDB_2, TDB_3, \dots, TDB_n$. Phân vùng cơ sở dữ liệu giao dịch có thể cải thiện hiệu suất khai thác các tập mục phổ biến bằng cách lắp từng phân vùng vào bộ nhớ chính hạn chế để truy cập nhanh và cho phép tạo thêm các tập mục phổ biến.

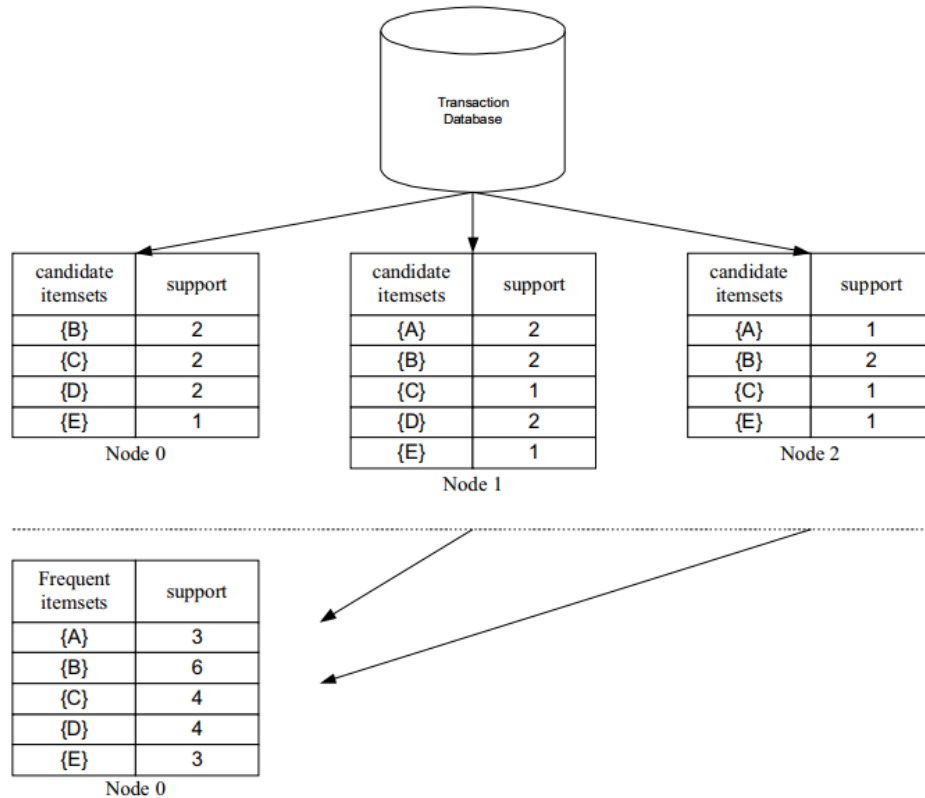
Một thuật toán Apriori dựa trên phân vùng để phân vùng cơ sở dữ liệu giao dịch thành N phân vùng và sau đó phân phối N phân vùng cho N nút mà ở đó mỗi nút tính toán k -itemsets cục bộ từ phân vùng của nó. Khi mỗi nút kết thúc k -itemsets ứng viên cục bộ, nó gửi k -itemsets ứng viên cục bộ của nó tới nút 0. Sau đó, nút 0 tính tổng của tất cả k -itemsets ứng viên và lược bỏ những k -itemsets ứng viên thành k -itemsets phổ biến[4].

Thực hiện việc tính toán song song, giả sử ta có N nút trong một máy tính song song:

- Quét song song Cơ sở dữ liệu giao dịch (TDB) để lấy tập 1-itemsets phổ biến, L_1 .
 - Chia tập TDB thành N phần;
 - Bộ xử lý P_i đọc TDB [i] để tạo ứng viên cục bộ 1-itemsets từ phân vùng của nó;
 - P_i chuyển các tập 1-itemsets ứng viên cục bộ của nó cho P_0 ;
 - P_0 tính toán số lượng của tất cả các tập 1-itemsets ứng viên cục bộ và lược bỏ nó để tạo ra 1-itemsets phổ biến, L_1 ;
- Quét song song TDB để lấy bộ 2-itemsets phổ biến, L_2
 - P_i quét TDB để lấy bộ 2-itemsets ứng viên cục bộ;
 - P_i chuyển tập 2-itemsets ứng viên cục bộ của nó cho P_0 ;
 - P_0 tính tổng số của tất cả các tập 2-itemsets ứng viên cục bộ và cắt bớt nó để lấy L_2 ;
- $K = 3$
 - While ($L_{k-1} \neq \text{empty}$)
Begin
 - Xây dựng cây ứng viên Trie[i] cho C_k trên Bộ xử lý dựa trên P_i trên L_{k-1} ;
 - Quét TDB [i] trên Bộ xử lý P_i để cập nhật số lượng Trie [i] cho C_k ;
 - Cắt Trie [i] để C_k lấy L_k ;
 - $k = k + 1$;

End;

Để tính toán các tập 1-itemsets phổ biến, mỗi nút đếm các tập 1-itemsets cục bộ của nó từ phân vùng của nó và chuyển kết quả đến nút 0. Vì nút 0 nhận tất cả kết quả từ N-1 nút khác, nó tạo ra các tập 1-itemsets chung, L_1 .

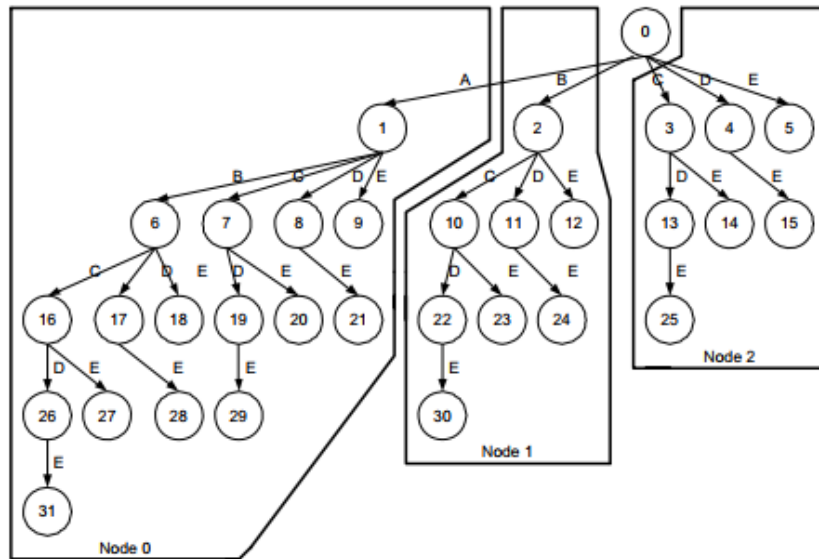


Hình 3.1. Số lượng quy trình phân phối

Quy trình tương tự được áp dụng để tìm tập 2-itemsets ứng viên. Mỗi nút tính toán các tập 2-itemsets ứng viên cục bộ của nó dựa trên tập dữ liệu được phân phối cho nó. Kết quả được gửi trở lại nút 0 để tạo ra tập 2-itemsets phổ biến chung. Trong quá trình triển khai, cũng giống như triển khai của Bodon, chúng ta không sử dụng cấu trúc dữ liệu trie để lưu trữ tập 1-itemsets và tập 2-itemsets. Thay vào đó, một vector đơn giản được sử dụng để lưu trữ các tập 1-itemsets ứng viên và một mảng hai chiều được sử dụng để lưu trữ các tập 2-itemsets ứng viên.

Để tạo tập k-itemsets phổ biến trong đó $k \geq 3$, một cấu trúc dữ liệu trie được sử dụng để lưu trữ k-itemsets ứng viên. Mỗi nút xây dựng trie cục bộ của riêng nó tăng dần cho các tập k-itemsets ứng viên cục bộ của nó dựa trên L_{k-1} . Số lượng của mỗi tập hợp vật phẩm được thực hiện trên trie. Sau đó, mỗi nút sẽ lược bỏ các

tập k-itemsets ứng viên cục bộ của chính nó để lấy k-itemsets phổ biến. Sau đó, tập k-itemsets phổ biến chung đạt được bằng cách thu thập tất cả các tập k-itemsets phổ biến cục bộ từ mỗi nút. Quá trình này được lặp lại cho đến khi không còn tập hợp mục ứng viên nào được tạo ra.



Hình 3.2. Phân phối Trie để tạo ra k-itemsets ứng viên

Việc triển khai này về cơ bản là một thuật toán Apriori dựa trên phân vùng. Cấu trúc dữ liệu được sử dụng để lưu trữ các tập mục ứng viên và số lượng của chúng là một trie. Cấu trúc dữ liệu trie cung cấp một kỹ thuật hiệu quả để lưu trữ, truy cập và đếm các tập phổ biến. Một trie được tạo tăng dần phụ thuộc vào số lượng các tập mục ứng viên. Nếu một tập ứng viên là không phổ biến, đường dẫn với tập hợp đó sẽ không được mở rộng thêm trong trie. Do đó, một nút trong bộ ba chỉ chứa các tập phổ biến với số lượng tương ứng.

CHƯƠNG 4: TÌM HIỂU THUẬT TOÁN APRIORI CẢI TIẾN

4.1. Ý tưởng của thuật toán Apriori cải tiến

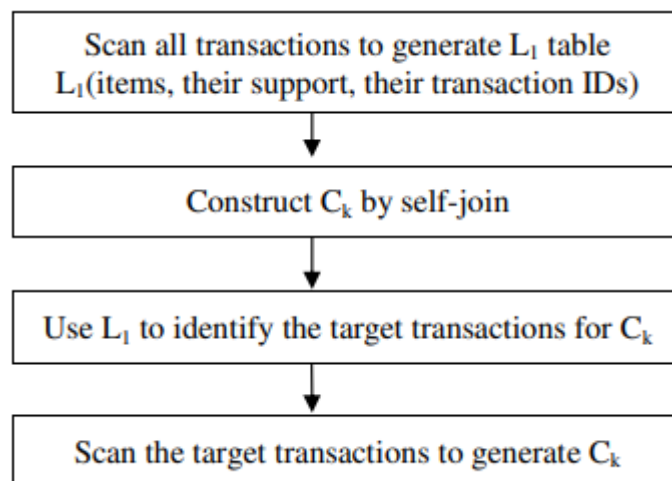
Để hiểu rõ được ý tưởng của thuật toán Apriori cải tiến, chúng ta cần phải có các định nghĩa sau:

Định nghĩa 1: Giả sử $T = \{T_1, T_2, \dots, T_m\}$, ($m \geq 1$) là tập các giao dịch, $T_i = \{I_1, I_2, \dots, I_n\}$, ($n \geq 1$) là tập các item và $k\text{-itemset} = \{i_1, i_2, \dots, i_k\}$, ($k \geq 1$) cũng là tập k item và $k\text{-itemset} \subseteq I$.

Định nghĩa 2: Giả sử σ (itemset) là số support count của itemset hoặc tần suất xuất hiện của tập hợp mục trong các giao dịch.

Định nghĩa 3: Giả sử C_k là tập phổ biến ứng viên (candidate itemset) có kích thước k và L_k là tập phổ biến có kích thước k .

Trong cách tiếp cận này, chúng ta nâng cao thuật toán Apriori để giảm thời gian tạo tập phổ biến ứng viên (candidate itemset). Trước tiên, chúng ta quét tất cả các giao dịch để tạo L_1 chứa item, support count, transactionID nơi item được tìm thấy.



Hình 4.1. Mô tả cách cải tiến thuật toán Apriori

Trong cách tiếp cận này, chúng ta nâng cao thuật toán Apriori để giảm thời gian tạo các tập phổ biến ứng viên. Trước tiên, chúng ta quét tất cả các giao dịch để tạo L_1 chứa các item, support count, transactionID nơi các item được tìm thấy.

Và sau đó, chúng ta sử dụng L1 như một trình trợ giúp để tạo ra L2, L3...Lk. Khi muốn tạo C2, chúng ta thực hiện L1*L1 để tạo ra 2-itemset C(x,y) trong đó x và y là các item của C2. Trước khi quét tất cả các giao dịch để đếm support count cho từng ứng viên, hãy sử dụng L1 để lấy ID giao dịch với support count tối thiểu giữa x và y và do đó chỉ quét C2 trong các giao dịch cụ thể này. Làm điều tương tự đối với C3, chúng ta xây dựng tập 3-itemset C(x,y,z) trong đó x, y và z là các item của C3 và sử dụng L1 để lấy transactionID ứng với support count tối thiểu giữa x, y và z, sau đó chỉ quét C3 trong các giao dịch cụ thể này và lặp lại những bước này cho đến khi không còn tập phổ biến mới nào được hình thành.

4.2. Thuật toán Apriori cải tiến

Thuật toán Apriori cải tiến được mô tả như sau:

Bước 1: Tạo items, items support, transaction ID

(1) $L1 = \text{find_frequent_1_itemsets}(T)$;

(2) For ($k = 2$; $L_{k-1} \neq \Phi$; $k++$) {

Bước 2: Tạo C_k từ L_{k-1}

(3) $C_k = \text{candidates generated from } L_{k-1}$;

Bước 3: Lấy item I_w với support tối thiểu trong C_k bằng L1, ($1 \leq w \leq k$).

(4) $x = \text{Get_item_min_sup}(C_k, L1)$;

Bước 4: Lấy transactionID có chứa x

(5) $Tgt = \text{get_Transaction_ID}(x)$;

(6) For each transaction t in Tgt Do

(7) Increment the count of all items in C_k that are found in Tgt;

(8) $L_k = \text{items in } C_k \geq \text{min_support}$;

(9) End;

(10) }

4.3. Vận dụng thuật toán Apriori cải tiến

Trong phần này, chúng ta áp dụng thuật toán Apriori cải tiến đối với tập giao dịch D có 9 giao dịch như sau:

T_ID	Items
T ₁	I ₁ , I ₂ , I ₅
T ₂	I ₂ , I ₄
T ₃	I ₂ , I ₄
T ₄	I ₁ , I ₂ , I ₄
T ₅	I ₁ , I ₃
T ₆	I ₂ , I ₃
T ₇	I ₁ , I ₃
T ₈	I ₁ , I ₂ , I ₃ , I ₅
T ₉	I ₁ , I ₂ , I ₃

Hình 4.2. Tập giao dịch thực thi Apriori cải tiến

Đầu tiên, chúng ta quét tất cả các giao dịch để tìm tập ứng viên 1-itemset cùng với support count (support) và transactionID (TIDs). Sau đó, chúng ta loại bỏ tập ứng viên không thỏa mãn minsup ($< \text{minsup}$). Ta được L1 như bảng sau:

Items	support	T_IDs	
I ₁	6	T ₁ , T ₄ , T ₅ , T ₇ , T ₈ , T ₉	
I ₂	7	T ₁ , T ₂ , T ₃ , T ₄ , T ₆ , T ₈ , T ₉	
I ₃	5	T ₅ , T ₆ , T ₇ , T ₈ , T ₉	
I ₄	3	T ₂ , T ₃ , T ₄	
I ₅	2	T ₁ , T ₈	deleted

Hình 4.3. Sinh ra 1-itemset thuật toán Apriori cải tiến

Bước tiếp theo, chúng ta tạo tập ứng viên 2-itemset từ L1. Để tìm support count cho mỗi itemset, hãy chia từng itemset trong 2-itemset thành hai phần tử, sau đó sử dụng bảng L1 để xác định các giao dịch nơi có thể tìm thấy itemset, thay vì tìm kiếm chúng trong tất cả các giao dịch.

Ví dụ: Chúng ta lấy mục đầu tiên trong bảng L2 bên dưới (I₁, I₂), trong Apriori ban đầu, chúng tôi quét tất cả 9 giao dịch để tìm mục (I₁, I₂); nhưng trong thuật toán cải tiến chúng ta sẽ chia mục (I₁, I₂) thành I₁ và I₂ và tìm support tối

thiểu giữa chúng bằng cách sử dụng L1, ở đây I_1 có hỗ trợ tối thiểu nhỏ nhất. Sau đó, chúng tôi tìm kiếm tập phổ biến (I_1, I_2) chỉ trong các giao dịch T_1, T_4, T_5, T_7, T_8 và T_9 .

Items	support	Min	Found in	
I_1, I_2	4	I_1	$T_1, T_4, T_5, T_7, T_8, T_9$	
I_1, I_3	4	I_3	T_5, T_6, T_7, T_8, T_9	
I_1, I_4	1	I_4	T_2, T_3, T_4	deleted
I_2, I_3	3	I_3	T_5, T_6, T_7, T_8, T_9	
I_2, I_4	3	I_4	T_2, T_3, T_4	
I_3, I_4	0	I_4	T_2, T_3, T_4	deleted

Hình 4.4. Sinh ra 2-itemset thuật toán Apriori cải tiến

Chúng ta làm tương tự cho 3-itemset và được bảng như sau:

Items	support	Min	Found in	
I_1, I_2, I_3	2	I_3	T_5, T_6, T_7, T_8, T_9	deleted
I_1, I_2, I_4	1	I_4	T_2, T_3, T_4	deleted
I_1, I_3, I_4	0	I_4	T_2, T_3, T_4	deleted
I_2, I_3, I_4	0	I_4	T_2, T_3, T_4	deleted

Hình 4.5. Sinh ra 3-itemset thuật toán Apriori cải tiến

Sau bước này ta thấy các 3-itemset ứng viên đều không thỏa min_sup , nên ta dừng tại đây và lấy bảng L2 làm kết quả cuối cùng.

CHƯƠNG 5: TÌM HIỂU VỀ HADOOP VÀ MAPREDUCE

5.1. Tìm hiểu về Hadoop

Hadoop là một Apache framework mã nguồn mở cho phép phát triển các ứng dụng phân tán (distributed processing) để lưu trữ và quản lý các tập dữ liệu lớn. Những năm 2000, Google công bố tài liệu nghiên cứu cách tiếp cận và nguyên tắc thiết kế để xử lý khối lượng lớn dữ liệu đã được đánh chỉ mục trên web. Những nguyên tắc cơ bản thiết kế là[5]:

Thứ nhất, thực tế nếu ta có đến hàng trăm hay thậm chí hàng ngàn ổ máy lưu trữ thì lỗi xảy ra là điều hiển nhiên chứ không phải ngoại lệ, do đó giám sát liên tục, phát hiện lỗi, kháng lỗi và tự động phục hồi phải được tích hợp với hệ thống.

Thứ hai, các tập tin rất lớn so với tiêu chuẩn truyền thống. Tập tin có dung lượng hàng GB và hàng tỷ đối tượng là rất phổ biến. Do đó, những giả định về thiết kế và các thông số như vận hành I/O hay kích thước khối phải xem xét lại.

Thứ ba, hầu hết các tập tin được cập nhật bằng cách thêm dữ liệu mới hơn là ghi đè lên dữ liệu hiện có. Việc ghi dữ liệu ngẫu nhiên trong một tập tin trên thực tế là không xảy ra. Khi ghi, các tập tin chỉ đọc và thường đọc theo thứ tự. Vì đây kiểu truy cập vào các tập tin lớn, nên sự bổ sung thêm trở thành tiêu điểm của việc tối ưu hóa hiệu suất và bảo đảm hoàn tất giao dịch.

5.2. Vai trò của Hadoop

Xử lý và làm việc khối lượng dữ liệu khổng lồ tính bằng Petabyte.

Xử lý trong môi trường phân tán, dữ liệu lưu trữ ở nhiều phần cứng khác nhau, yêu cầu xử lý đồng bộ.

Khắc phục các lỗi xuất hiện thường xuyên.

Giải quyết vấn đề băng thông giữa các phần cứng vật lý chứa dữ liệu phân tán có giới hạn.

5.3. Nguyên tắc hoạt động của Hadoop

Giai đoạn 1: Một người dùng hay một ứng dụng có thể đưa (submit) một công việc (Job) lên Hadoop (Hadoop Job Client) với yêu cầu xử lý cùng các thông tin cơ bản.

Giai đoạn 2: Hadoop Job Client submit job (file jar, file thực thi) và các thiết lập cho JobTracker. Sau đó, master sẽ phân phối tác vụ đến các máy slave để theo dõi và quản lý tiến trình các máy này, đồng thời cung cấp thông tin về tình trạng và chẩn đoán liên quan đến job-client.

Giai đoạn 3: TaskTrackers trên các node khác nhau thực thi tác vụ MapReduce và trả về kết quả output được lưu trong hệ thống file.

5.4. Tìm hiểu về Hadoop framework

Hadoop framework gồm 4 module [6]:

Hadoop Common: Đây là các thư viện và tiện ích cần thiết của Java để các module khác sử dụng. Cung cấp hệ thống file và lớp OS trừu tượng, đồng thời chứa các mã lệnh Java để khởi động Hadoop.

Hadoop YARN: 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.

Bên trong YARN, chúng ta có hai trình quản lý ResourceManager và NodeManager

- ResourceManager: Quản lý toàn bộ tài nguyên tính toán của cluster.
- NodeManager: Giám sát việc sử dụng tài nguyên của container và báo cáo với ResourceManager. Các tài nguyên ở đây là CPU, memory, disk, network,...

Hadoop Distributed File System (HDFS): Đây là hệ thống file phân tán cung cấp truy cập băng thông cao cho ứng dụng khai thác dữ liệu.

Là hệ thống file phân tán, cung cấp khả năng lưu trữ dữ liệu không lỗi và tính năng tối ưu hoá việc sử dụng băng thông giữa các node. HDFS có thể được sử dụng để chạy trên một cluster lớn với hàng chục ngàn node.

Cho phép truy xuất nhiều ổ đĩa như là 1 ổ đĩa. Nói cách khác, chúng ta có thể sử dụng một ổ đĩa mà gần như không bị giới hạn về dung lượng. Muốn tăng dung lượng chỉ cần thêm node (máy tính) vào hệ thống.

Có kiến trúc Master-Slave

NameNode chạy trên máy chủ Master, có tác vụ quản lý Namespace và điều chỉnh truy cập tệp của client

DataNode chạy trên các nút Slave. có tác vụ lưu trữ business data thực tế

Một tập tin với định dạng HDFS được chia thành nhiều block và những block này được lưu trữ trong một tập các DataNodes

Kích thước 1 block thông thường là 64MB, kích thước này có thể thay đổi được bằng việc cấu hình

Hadoop MapReduce: Đây là hệ thống dựa trên YARN dùng để xử lý song song các tập dữ liệu lớn. Cho phép lập trình tạo ra các chương trình song song dùng để xử lý dữ liệu.

5.5. Tìm hiểu về MapReduce

Là thành phần quan trọng của góp phần làm nên sức mạnh của Hadoop. MapReduce được chia thành hàm là Map và Reduce. Những hàm này được định nghĩa bởi người dùng là hai quá trình liên tiếp khi xử lý dữ liệu[9].

Map nhận input là tập các cặp khóa/giá trị và output là tập các cặp khóa/giá trị trung gian và ghi xuống đĩa cứng và thông báo cho Reduce nhận dữ liệu đọc.

Reduce sẽ nhận khóa trung gian I và tập các giá trị ứng với khóa đó, ghép nối chúng lại để tạo thành một tập khóa nhỏ hơn. Các cặp khóa/giá trị trung gian sẽ được đưa vào cho hàm reduce thông qua một con trỏ vị trí (iterator). Điều này cho phép ta có thể quản lý một lượng lớn danh sách các giá trị để phù hợp với bộ nhớ.

Ở giữa Map và Reduce thì còn 1 bước trung gian đó chính là Shuffle. Sau khi Map hoàn thành xong công việc của mình thì Shuffle sẽ làm nhiệm vụ chính là thu thập cũng như tổng hợp từ khóa/giá trị trung gian đã được map sinh ra trước đó rồi chuyển qua cho Reduce tiếp tục xử lý.

5.6. Ưu điểm của MapReduce

MapReduce có khả năng xử lý dễ dàng mọi bài toán có lượng dữ liệu lớn nhờ khả năng tác vụ phân tích và tính toán phức tạp. - Nó có thể xử lý nhanh chóng cho ra kết quả dễ dàng chỉ trong khoảng thời gian ngắn.

Mapreduce có khả năng chạy song song trên các máy có sự phân tán khác nhau.

MapRedue có khả năng thực hiện trên nhiều ngôn ngữ lập trình khác nhau như: Java, C/ C++, Python, Perl, Ruby,...

5.7. Nguyên tắc hoạt động

Phân chia các dữ liệu cần xử lý thành nhiều phần nhỏ trước khi thực hiện.

Xử lý các vấn đề nhỏ theo phương thức song song trên các máy tính rồi phân tán hoạt động theo hướng độc lập.

Tiến hành tổng hợp những kết quả thu được để đề ra được kết quả sau cùng.

CHƯƠNG 6: THỰC HÀNH CÀI ĐẶT VÀ TRIỂN KHAI HADOOP VÀ MAPREDUCE

6.1. Yêu cầu về cấu hình máy, phần mềm cần chuẩn bị

Máy ảo VM-ware Workstation Pro version 15.0.2 [Link download](#)

Hệ điều hành Ubuntu-server 18.04 [Link download](#)

Cung cấp tài nguyên: Ram 2Gb, HDD 40Gb

6.2. Các bước cài đặt Hadoop và Mapreduce

Cài đặt Oracle Java 14[8]

```
# add-apt-repository ppa:linuxuprising/java
```

```
About Oracle Java 10, 12 and 13: These versions have reached the end of public updates, therefore they are no longer available for download. The Oracle Java 10/12/13 packages in this PPA no longer worked due to this, so I have removed them. Switch to Oracle Java 11 or OpenJDK 11 instead, which is long term support, or the latest Java 14.
More info: https://launchpad.net/~linuxuprising/+archive/ubuntu/java
Press [ENTER] to continue or Ctrl-c to cancel adding it.

Get:1 http://ppa.launchpad.net/linuxuprising/java/ubuntu bionic InRelease [15.9 kB]
Hit:2 http://vn.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://vn.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://ppa.launchpad.net/linuxuprising/java/ubuntu bionic/main amd64 Packages [1,860 B]
Get:5 http://vn.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:6 http://ppa.launchpad.net/linuxuprising/java/ubuntu bionic/main Translation-en [692 B]
Get:7 http://vn.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:8 http://vn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [951 kB]
Get:9 http://vn.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [324 kB]
Get:10 http://vn.archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [55.3 kB]
Get:11 http://vn.archive.ubuntu.com/ubuntu bionic-updates/restricted Translation-en [13.8 kB]
Get:12 http://vn.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1,077 kB]
Get:13 http://vn.archive.ubuntu.com/ubuntu bionic-updates/universe Translation-en [335 kB]
Get:14 http://vn.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [15.7 kB]
Get:15 http://vn.archive.ubuntu.com/ubuntu bionic-updates/multiverse Translation-en [6,384 B]
Get:16 http://vn.archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [7,516 B]
Get:17 http://vn.archive.ubuntu.com/ubuntu bionic-backports/main Translation-en [4,764 B]
Get:18 http://vn.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [7,484 B]
Get:19 http://vn.archive.ubuntu.com/ubuntu bionic-backports/universe Translation-en [4,436 B]
Get:20 http://vn.archive.ubuntu.com/ubuntu bionic-security/main amd64 Packages [727 kB]
Get:21 http://vn.archive.ubuntu.com/ubuntu bionic-security/main Translation-en [230 kB]
Get:22 http://vn.archive.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [45.7 kB]
Get:23 http://vn.archive.ubuntu.com/ubuntu bionic-security/restricted Translation-en [11.4 kB]
Get:24 http://vn.archive.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [669 kB]
Get:25 http://vn.archive.ubuntu.com/ubuntu bionic-security/universe Translation-en [222 kB]
Get:26 http://vn.archive.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [7,596 B]
Get:27 http://vn.archive.ubuntu.com/ubuntu bionic-security/multiverse Translation-en [2,824 B]
Fetched 4,988 kB in 28s (177 kB/s)
Reading package lists... Done
root@minhchau-server:~# _
```

Hình 3. Cài đặt Java

Update và khởi động lại máy

```
# apt update
```

```
# reboot
```


Tiến hành cài đặt

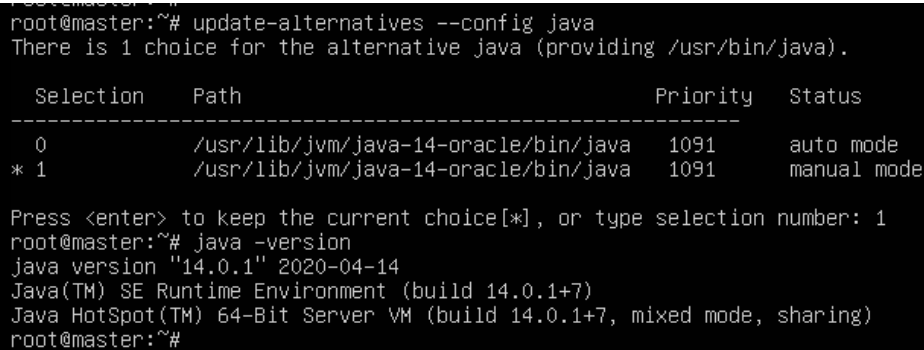
```
# apt install -y oracle-java14-installer
```

(Chấp nhận Liscence và chọn bản Oracle Manual Mode)

```
# update-alternatives --config java
```

(Kiểm tra kết quả)

```
# java -version
```



```
root@master:~# update-alternatives --config java
There is 1 choice for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority  Status
  -----
    0           /usr/lib/jvm/java-14-oracle/bin/java          1091    auto mode
*   1           /usr/lib/jvm/java-14-oracle/bin/java          1091    manual mode

Press <enter> to keep the current choice[*], or type selection number: 1
root@master:~# java -version
java version "14.0.1" 2020-04-14
Java(TM) SE Runtime Environment (build 14.0.1+7)
Java HotSpot(TM) 64-Bit Server VM (build 14.0.1+7, mixed mode, sharing)
root@master:~#
```

Hình 4. Kiểm tra version java

Cài đặt SSH

```
# apt-get install ssh
```

```
# apt install openssh-server
```

Cấu hình SSH

```
# vim /etc/ssh/sshd_config (Cài đặt lại các thông số như bên dưới)
```

...

```
PubkeyAuthentication yes
```

```
PasswordAuthentication yes
```

...

```

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
PermitEmptyPasswords no

```

Hình 5. Cài đặt thông số

(Sau khi sửa thì nhấn phím ESC, nhập :wq để lưu và thoát khỏi vim).

Khởi động lại SSH

```
# service sshd restart
```

Cấu hình host/hostname

ifconfig (Kiểm tra và cấu hình lại IP các máy Master và các cụm Cluster)

```
IP master: 192.168.13.131
```

```
IP slave_1: 192.138.13.132
```

```
IP slave_2: 192.168.13.133
```

vim /etc/hosts (Cấu hình IP của các máy Master và các cụm Cluster)

```
192.168.13.131 master
```

```
192.168.13.132 slave_1
```

```
192.168.13.133 slave_2
```

```

127.0.0.1 localhost
127.0.1.1 minhchau-server
192.168.13.131 master
192.168.13.132 slave_1
192.168.13.133 slave_2
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
~
~
~
~

```

Hình 6. Cấu hình hostname

vim /etc/hostname (Đặt lại tên Master – chỉ thực hiện trên máy Master).

Trong file này sẽ xuất hiện hostname mặc định của máy, xóa đi và đổi thành master.

vim /etc/hostname (Đặt lại tên của máy Slave – thực hiện trên máy Slave).

Trong file này sẽ xuất hiện hostname mặc định của máy, xóa đi và đổi thành slave_1(trên máy slave_1), slave_2(trên máy slave_2).

reboot (Reset máy).

Tạo user (Tạo user hadoopuser để quản lý các permission cho đơn giản)

```

# addgroup hadoopgroup
# adduser khanhduyuser
# usermod -g hadoopgroup khanhduyuser
# groupdel khanhduyuser

```

Cài đặt Hadoop 2.7.7

```

# su khanhduyuser (Chuyển qua thư mục /home/khanhduyuser)
# wget

```

<https://archive.apache.org/dist/hadoop/common/hadoop-2.7.7/hadoop-2.7.7.tar.gz>

```
# tar -xvf hadoop-2.7.7.tar.gz (Giải nén file)
# mv hadoop-2.7.7 hadoop (Đổi tên file thành hadoop để dễ quản lý).
```

Cấu hình các thông số cho hệ thống Hadoop

File .bashrc

```
# vim ~/.bashrc (Cấu hình trên file hệ thống)
```

Thêm vào cuối file .bashrc nội dung như sau:

```
export HADOOP_HOME=/home/khanhduyuser/hadoop
export JAVA_HOME=/usr/lib/jvm/java-14-oracle
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export
HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

```
# Add an "alert" alias for long running commands. Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -i "${?} = 0" && echo terminal || echo error' "${history}
tail -n1|sed -e '\`s/\`s*[0-9]\`+\`s*//;s/[:&|]\`s*alert$/\`'\`'"

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

export HADOOP_HOME=/home/khanhduyuser/hadoop
export JAVA_HOME=/usr/lib/jvm/java-14-oracle
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
"~/.bashrc" 128L, 4196C
```

File `hadoop-env.sh`

```
# vim ~/hadoop/etc/hadoop/hadoop-env.sh
```

Tìm đoạn `export JAVA_HOME=...` sửa thành như sau:

```
# export JAVA_HOME=/usr/lib/jvm/java-14-oracle/
```

```
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# Set Hadoop-specific environment variables here.
#
# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.
#
# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-14-oracle/
#
# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol. Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}
#
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}
#
# Extra Java CLASSPATH elements. Automatically insert capacity-scheduler.
for f in $(find $HADOOP_HOME/contrib/capacity-scheduler/*.jar); do
  if [ "$HADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else
    export HADOOP_CLASSPATH=$f
  fi
done
```

Hình 8. Cấu hình đường dẫn cho Java

File `core-site.xml`

vim ~/hadoop/etc/hadoop/core-site.xml (Mở file `core-site.xml`).

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/khanhduyuser/tmp</value>
    <description>Temporary Directory.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:54310</value>
    <description>Use HDFS as file storage engine</description>
  </property>
</configuration>
```

```

<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>

    <name>hadoop.tmp.dir</name>

    <value>/home/khanhduyuser/tmp</value>

    <description>Temporary Directory.</description>

  </property>

  <property>

    <name>fs.defaultFS</name>

    <value>hdfs://master:54310</value>

    <description>Use HDFS as file storage engine</description>

  </property>
</configuration>

```

Hình 9. Cấu hình thuộc tính cho máy ảo

File mapred-site(chỉ cấu hình mở master)

```

# cd ~/hadoop/etc/hadoop/
# cp mapred-site.xml.template mapred-site.xml
# vim mapred-site.xml

```

Chỉnh sửa lại thông tin như sau:

```

<configuration>
  <property>

    <name>mapreduce.jobtracker.address</name>

    <value>master:54311</value>

    <description>The host and port that the MapReduce
job tracker runs at. If "local", then jobs are run in-
process as a single map and reduce task.

    </description>

  </property>

  <property>

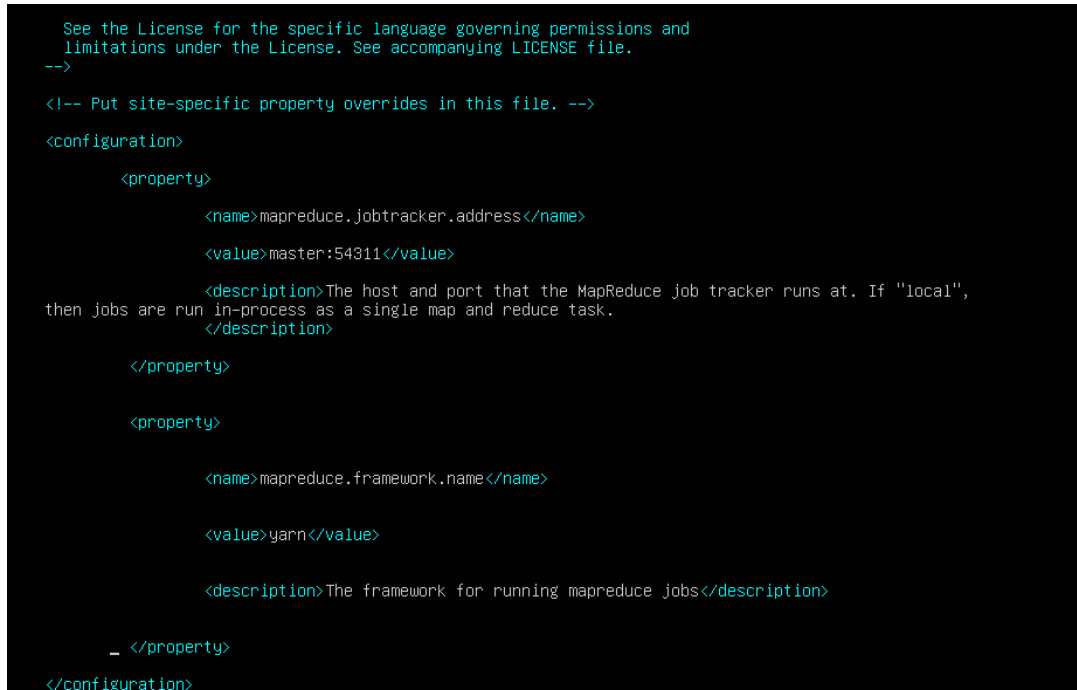
    <name>mapreduce.framework.name</name>

```

```

        <value>yarn</value>
        <description>The framework for running mapreduce
jobs</description>
    </property>
</configuration>

```



```

    See the License for the specific language governing permissions and
    limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
        <name>mapreduce.jobtracker.address</name>
        <value>master:54311</value>
        <description>The host and port that the MapReduce job tracker runs at. If "local",
then jobs are run in-process as a single map and reduce task.
        </description>
    </property>

    <property>

        <name>mapreduce.framework.name</name>

        <value>yarn</value>

        <description>The framework for running mapreduce jobs</description>

    </property>
</configuration>

```

Hình 10. Cấu hình Map Reduce

File hdfs-site.xml

```

# vim ~/hadoop/etc/hadoop/hdfs-site.xml

<configuration>
    <property>
        <name>dfs.replication</name>
        <value>2</value>
        <description>Default block replication. The
actual number of replications can be specified when the
file is created. The default is used if replication is not
specified in create time.
        </description>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/home/khanhduyuser/hadoop/hadoop_data/hdfs
/namenode</value>
    </property>

```



```
<description>Determines where on the local
filesystem the DFS name node should store the name
table(fsimage). If this is a comma-delimited list of
directories then the name table is replicated in all of
the directories, for redundancy.
```

```
</description>
```

```
</property>
```

```
<property>
```

```
<name>dfs.datanode.data.dir</name>
```

```
<value>/home/khanhduyuser/hadoop/hadoop_data/hdfs/
datanode</value>
```

```
<description>Determines where on the local
filesystem an DFS data node should store its blocks. If
this is a comma-delimited list of directories, different
devices. Directories that do not exist are ignored.
```

```
</description>
```

```
</property>
```

```
</configuration>
```

File yarn-site.xml

```
# vim ~/hadoop/etc/hadoop/yarn-site.xml
```

```
<configuration>
```

```
<property>
```

```
<name>yarn.nodemanager.aux-services</name>
```

```
<value>mapreduce_shuffle</value>
```

```
</property>
```

```
<property>
```

```
<name>yarn.resourcemanager.scheduler.address</name>
```

```
<value>master:8030</value>
```

```
</property>
```

```
<property>
```

```
<name>yarn.resourcemanager.address</name>
```

```
<value>master:8032</value>
```

```
</property>
```

```
<property>
```

```
<name>yarn.resourcemanager.webapp.address</name>
```

```
<value>master:8088</value>
```

```
</property>
```

```
<property>
```

```
<name>yarn.resourcemanager.resource-
tracker.address</name>
```

```

        <value>master:8031</value>
    </property>
</property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>master:8033</value>
</property>
</configuration>

```

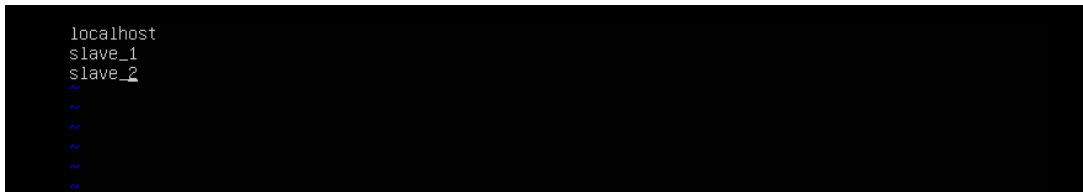
Thêm các máy slaves vào master (chỉ cấu hình ở master)

```
# vim ~/hadoop/etc/hadoop/slaves
```

Thêm hostname của các máy slave: mỗi máy slave đặt trên 1 dòng

```
# slave_1
```

```
# slave_2
```



Hình 11. Thêm các slave trên máy master

Cài đặt ssh key giữa các node: (Thực hiện trên master)

Đăng nhập với khanhduyuser

```
# sudo su - khanhduyuser
```

Tạo ssh key

```
# ssh-keygen -t rsa -P ""
```

```
# cat /home/khanhduyuser/.ssh/id_rsa.pub >>
```

```
    /home/khanhduyuser/.ssh/authorized_keys
```

```
# chmod 600
```

```
    /home/khanhduyuser/.ssh/authorized_keys
```

Share ssh key giữa master – master

```
# ssh-copy-id -i ~/.ssh/id_rsa.pub master
```

Share ssh key giữa master – slave_1

```
# ssh-copy-id -i ~/.ssh/id_rsa.pub slave_1
```

Share ssh key giữa master – slave_2

```
# ssh-copy-id -i ~/.ssh/id_rsa.pub slave_2
```

Test kết nối ssh

Test kết nối tới master

```
# ssh khanhduyuser@master
```

Đăng xuất

```
# logout
```

Test kết nối tới slave_1

```
# ssh khanhduyuser@slave_1
```

Đăng xuất

```
# logout
```

Test kết nối tới slave_2

```
# ssh khanhduyuser@slave_2
```

Đăng xuất


```
# logout
```

Khởi động thành phần hadoop và kiểm tra hoạt động của các thành phần

```
#hadoop namenode -format (Chỉ chạy 1 lần duy nhất)
```

```
#start-all.sh (Khởi động các thành phần)
```

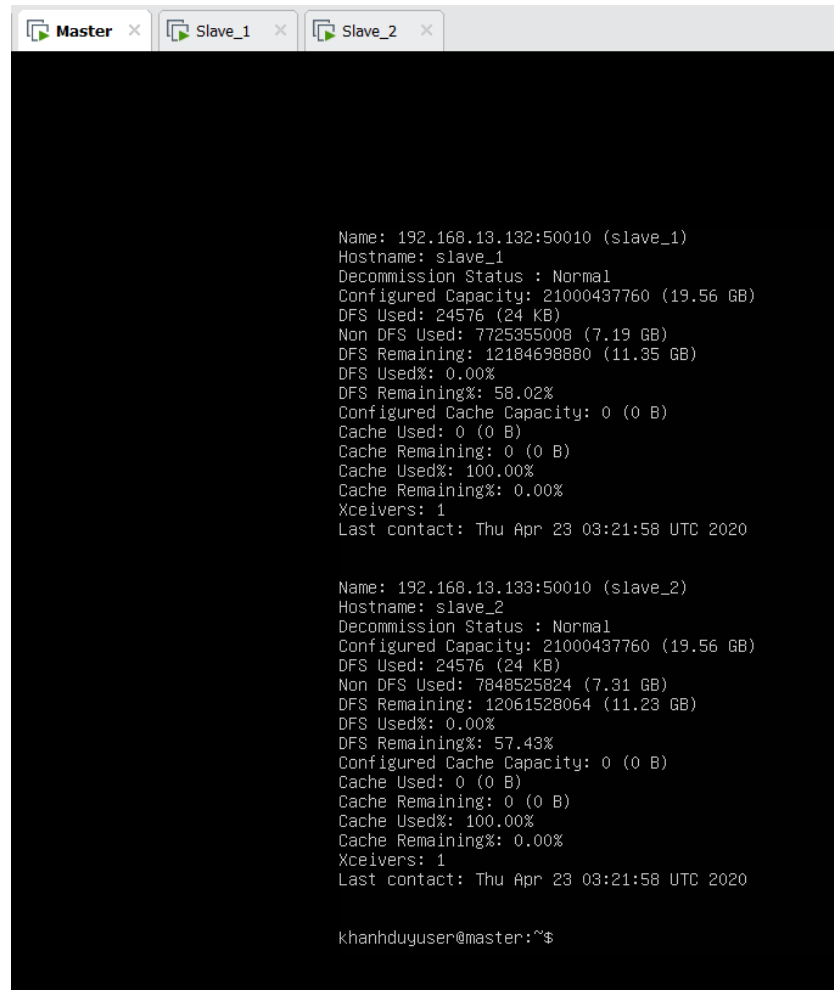
```
#jps (kiểm tra các thành phần)
```



```
khanhduyuser@master:~$ jps
2161 DataNode
1971 NameNode
3209 Jps
2413 SecondaryNameNode
2670 ResourceManager
2863 NodeManager
khanhduyuser@master:~$
```

Hình 12. Kiểm tra các thành phần

```
#hdfs dfsadmin -report (Kiểm tra hoạt động của các máy Slave)
```



```
Name: 192.168.13.132:50010 (slave_1)
Hostname: slave_1
Decommission Status : Normal
Configured Capacity: 21000437760 (19.56 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 7725355008 (7.19 GB)
DFS Remaining: 12184698880 (11.35 GB)
DFS Used%: 0.00%
DFS Remaining%: 58.02%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Apr 23 03:21:58 UTC 2020

Name: 192.168.13.133:50010 (slave_2)
Hostname: slave_2
Decommission Status : Normal
Configured Capacity: 21000437760 (19.56 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 7848525824 (7.31 GB)
DFS Remaining: 12061528064 (11.23 GB)
DFS Used%: 0.00%
DFS Remaining%: 57.43%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Apr 23 03:21:58 UTC 2020

khanhduyuser@master:~$
```

Hình 13. Kiểm tra sự hoạt động của slave

6.3. Chạy ví dụ về Hadoop Mapreduce

Ra thư mục gốc

```
# cd ~
```

Tạo file test

```
# vim test.sh
```

Tạo với nội dung như sau:

```
#!/bin/bash
```

```
# test the hadoop cluster by running wordcount
```

```
# create input files
```

```
mkdir input
```

```
echo "Hello World" >input/file2.txt
```

```

        echo "Hello Hadoop" >input/file1.txt

# create input directory on HDFS

        hadoop fs -mkdir -p input

#put input files to HDFS

        hdfs dfs -put ./input/* input

# run wordcount

        hadoop jar

        $HADOOP_HOME/share/hadoop/mapreduce/sources/had
        oop-mapreduce-examples-2.7.7-sources.jar
        org.apache.hadoop.examples.WordCount input
        output

# print the input files

        echo -e "\ninput file1.txt:"

        hdfs dfs -cat input/file1.txt

        echo -e "\ninput file2.txt:"

        hdfs dfs -cat input/file2.txt


# print the output of wordcount

        echo -e "\nwordcount output:"

        hdfs dfs -cat output/part-r-00000

```

Giải thích: Test trên sẽ tạo ra 2 file *file2.txt* và *file1.txt* có nội dung lần lượt là *Hello World* và *Hello Hadoop*. 2 file này lần lượt được đưa vào trong HDFS, sau đó sẽ chạy một job có nhiệm vụ đếm số lần xuất của hiện của mỗi từ có trong *file1.txt* và *file2.txt*.

Chạy test: # ./test.sh

Kết quả:

```
...  
  
World      1  
  
Hadoop     1  
  
Hello     2
```

```
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operation
s
WARNING: All illegal access operations will be denied in a future release
20/04/22 06:18:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
Hello Hadoop

input file2.txt:
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (f
ile:/home/Khanhduyuser/hadoop/share/hadoop/common/lib/hadoop-auth-2.7.7.jar) to method sun.security.
krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authenticat
ion.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operation
s
WARNING: All illegal access operations will be denied in a future release
20/04/22 06:18:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
Hello World

wordcount output:
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (f
ile:/home/Khanhduyuser/hadoop/share/hadoop/common/lib/hadoop-auth-2.7.7.jar) to method sun.security.
krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authenticat
ion.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operation
s
WARNING: All illegal access operations will be denied in a future release
20/04/22 06:18:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
Hadoop 1
Hello 2
World 1
Khanhduyuser@master:~$ _
```

Hình 14. Kết quả ví dụ WordCount

Trong trường hợp bạn muốn test lại thì phải xóa kết quả cũ bằng lệnh sau

```
# rm -rf input  
  
# hadoop fs -rm -r input  
  
# hadoop fs -rm -r output
```

Rồi chạy lại: # ./test.sh

CHƯƠNG 7: TRIỂN KHAI THUẬT TOÁN APRIORI CẢI TIẾN TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG.

7.1. Tóm tắt quá trình thực thi

Một số kí hiệu:

L_k : tập phổ biến có độ dài k .

C_k : tập phổ biến ứng viên có độ dài k .

Để tìm 1-itemsets (tập phổ biến có kích thước 1, L_1) trong MapReduce, chúng tôi chỉ cần tìm ra (item, 1) trong giai đoạn Map, sau đó thêm số lượng lên trong giai đoạn Reduce.

Để tạo $C_{(k+1)}$ từ L_k trong MapReduce, chúng tôi thực hiện như sau:

Giai đoạn Pre-Map

Bước 1: Kết hợp L_k để tạo các tập phổ biến có kích thước $k + 1$, chúng ta chỉ có thể tạo $(k+1)$ -itemsets với k -itemsets có cùng một mục.

Ví dụ:

Trường hợp 1:

Giả sử chúng ta có 2 itemset có độ dài 3 (list [0..2])

[1, 2, 3], [1, 2, 4]

Trong trường hợp này, chúng tôi sẽ kiểm tra xem itemsets có giống nhau từ chỉ mục 0 đến chỉ mục 1 hay không và tạo 4-newitemsets và lược bỏ.

Trường hợp 2:

Giả sử chúng ta có 2 itemset có độ dài 3 (list [0..2])

[1, 2, 3], [2, 3, 4]

Trong trường hợp này, chúng ta chỉ nên kiểm tra chỉ số 0 và ngắt, không cần tạo $(k + 1)$ -itemsets.

Bước 2: Đối với mỗi tập hợp con của các tập phổ biến trên, chúng ta kiểm tra xem nó có nằm trong L_k không. Chúng tôi thực hiện việc kiểm tra này bằng cách giữ mã băm của các tập vật phẩm trong L_k trong một tập hợp và kiểm tra đối chiếu với tập hợp này. Nếu tất cả các tập con nằm trong L_k thì chúng ta thêm tập phổ biến vào $C_{(k+1)}$.

Bước 3: Thêm các tập phổ biến trong $C_{(k+1)}$ vào Trie.

Trong pha n-Map($n > 1$)

Bước 4: Đọc một phần của tệp giao dịch và đối với mỗi giao dịch, gọi hàm findItemsets() để nhận các tập hợp itemsets xuất hiện trong giao dịch.

Bước 5: Gửi các tập phổ biến ở trên dưới dạng (itemset, 1).

Trong giai đoạn Reduce

Bước 6: Thêm các số lên và nếu nó ở trên minsup hãy thêm tập phổ biến vào $L_{(k+1)}$.

Ví dụ mô tả thuật toán trên:

Số giao dịch = 6.

min_sup = 40%.

min_number of item = 2.

Tìm 1-itemsets:

Database		(A, 1) (B, 1) (C, 1) ...		Output	
TID	Items			<Item, Count>	
t1	A, B, C	Map - 1	Reduce - 1	<A, 3>	
t2	A, C			<B, 2>	
t3	B, C, D, E			<C, 4>	
t4	A, D	/		<D, 3>	
t5	E	Map - 2	Reduce - 2	<E, 5>	
t6	C, D, E, F, G				

Hình 7.1. Quá trình tạo 1-itemset Apriori song song

Tìm 2-itemsets:

		v		
TID	Items			<Item, Count>
t1	A, B, C	Map - 1	Reduce - 1	
t2	A, C			<A C, 2>
t3	B, C, D, E			<B C, 2>
t4	A, D	/		<C D, 2>
t5	E	Map - 2	Reduce - 2	<C E, 2>
t6	C, D, E, F, G	((A,C), 1)		<D E, 2>
		((B,C), 1)		

Hình 7.2. Quá trình tạo 1-itemset Apriori song song

Tìm k-itemsets: Lặp lại đến khi nào L_k rỗng.

7.2. Cài đặt

Ngôn ngữ sử dụng : Java.

Link Github:

https://github.com/Nguyen-Khanh-Duy-1007/Apriori_Algorithm_KLTN2021

Một số class chính:

AprioriDriver thiết lập thông tin mapreduce.

AprioriPass1Mapper đọc dataset theo dòng và cập nhật theo <Individual item, 1>.

Ví dụ:

1 1 3 4 2 5 -> line number : 0 Items : 1 3 4 2 5

2 2 3 5 -> line number : 1 Items : 2 3 5

3 1 2 3 5 -> line number : 2 Items : 1 2 3 5

4 2 5 -> line number : 3 Items : 2 5

AprioriPassKMapper : Tái thực thi qua map từ 2 đến k.

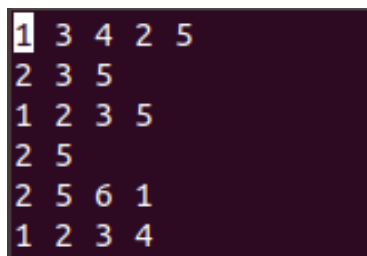
AprioriReducer: Giai đoạn reduce.

ItemSet: Đại diện cho itemset.

Transaction: Đại diện cho transaction sử dụng List(interger).

AprioriUtils Chưa các phương thức thực thi của Apriori.

Tập dataset: 6 transaction



```
1 3 4 2 5
2 3 5
1 2 3 5
2 5
2 5 6 1
1 2 3 4
```

Hình 7.3. Tập dataset chạy ví dụ

Đẩy data lên hdfs

```
hdfs dfs -put dataSet/sample
```

Câu lệnh thực thi

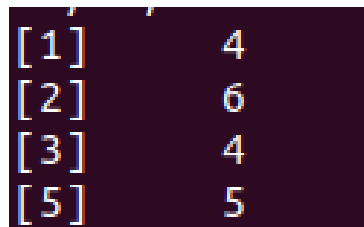
```
hadoop jar HadoopBasedApriori.jar apriori.AprioriDriver sample output  
k-itemset support sum-transaction
```

Chọn k-itemset = 3, support = 0.5, num-transaction = 6

```
hadoop jar HadoopBasedApriori.jar apriori.AprioriDriver sample output  
3 0.5 6
```

Kết quả thực thi:

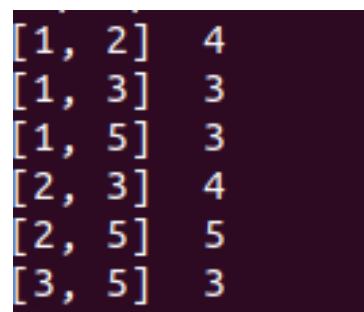
output1/part-r-00000



[1]	4
[2]	6
[3]	4
[5]	5

Hình 7.4. Kết quả 1-itemset Apriori song song

output2/part-r-00000



[1, 2]	4
[1, 3]	3
[1, 5]	3
[2, 3]	4
[2, 5]	5
[3, 5]	3

Hình 7.5. Kết quả 2-itemset Apriori song song

output3/part-r-00000

```
[1, 2, 3]      3  
[1, 2, 5]      3  
[2, 3, 5]      3
```

Hình 7.6. Kết quả 2-itemset Apriori song song

CHƯƠNG 8: ĐÁNH GIÁ THỰC NGHIỆM

8.1. Đánh giá kết quả

Để đánh giá các thuật toán khai phá luật kết hợp được nghiên cứu trong luận văn này chúng tôi tiến hành thực nghiệm trên các bộ dữ liệu mushroom (8124 giao dịch), bộ dữ liệu retail (86103 giao dịch), và T10I4D100K (100000 giao dịch) và được lấy từ các trang <http://fimi.uantwerpen.be/data/>.

Chúng tôi cài đặt các thuật toán Apriori, Apriori cải tiến, Apriori trên môi trường song song, so sánh các thuật toán dựa vào thời gian thực thi và số lượng tài nguyên mà thuật toán sử dụng. Việc so sánh các thuật toán dựa vào 2 yếu tố thời gian thực thi và độ chính xác. Độ chính xác được đánh giá dựa trên số tập mục phổ biến giống nhau mà các thuật toán phát hiện được. Ngoài ra chúng tôi còn đánh giá dựa trên số lượng tài nguyên mà thuật toán sử dụng.

Ngoài ra chúng tôi còn đánh giá kết quả thực thi giữa thuật toán Apriori và Apriori cải tiến trên cùng một dataset để quan sát thời gian thực thi giữa hai thuật toán và điều khác biệt giữa hai toán và số lượng duyệt transaction ở mỗi K-itemset.

Còn đối với môi trường song song Hadoop Mapreduce chúng tôi thực hiện chạy thuật toán trên nhiều tập datasets với nhiều cách setup các máy master và slave khác nhau như: 1 master – 0 slave, 1 master – 1 slave và 1 master – 2 slave để xem thời gian thực thi giữa các cấu hình này.

8.2. Thời gian thực thi giữa các thuật toán

Để so sánh thời gian thực thi giữa các thuật toán, chúng tôi thực nghiệm trên nhiều dataset và trình bày một trong những dataset đem lại kết quả khách quan nhất và đồ thị hóa và chúng tôi quyết định chọn tập dataset T10I4D100K với số TID là 100000.

Trình bày kết quả thực nghiệm về thời gian thực thi của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ thay đổi từ 1% đến 5%. Kết quả thực nghiệm trong bảng cho thấy thời gian thực thi của các thuật toán:

Độ support	Apriori	Apriori Cải tiến	Apriori song song
0.01	1501s	1010s	20s
0.02	540s	240s	19s
0.03	140s	27s	11s
0.04	17s	5s	10s
0.05	13s	1.3s	9s

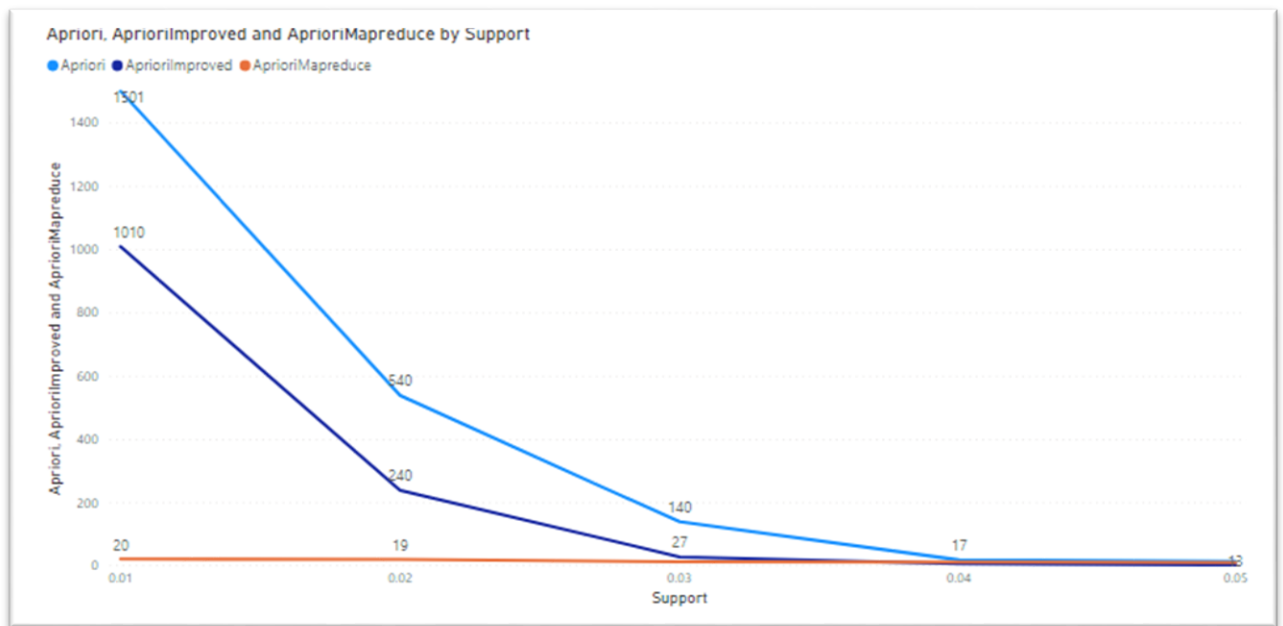
Bảng 8.1. Kết quả thực nghiệm về thời gian thực thi của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ thay đổi từ 1% đến 5%

Trình bày kết quả về độ chính xác của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ từ 1% đến 5%, các thuật toán đều sinh số lượng tập mục giống nhau.

Support (%)	Apriori	Apriori cải tiến	Apriori song song
5	{529} (s:7.06%)	{529} (s:7.06%)	{529} (s:7.06%)
	{829} (s:6.81%)	{829} (s:6.81%)	{829} (s:6.81%)
	{766} (s:6.26%)	{766} (s:6.26%)	{766} (s:6.26%)
	{722} (s:5.85%)	{722} (s:5.85%)	{722} (s:5.85%)
	{354} (s:5.84%)	{354} (s:5.84%)	{354} (s:5.84%)
	{684} (s:5.41%)	{684} (s:5.41%)	{684} (s:5.41%)
	{217} (s:5.38%)	{217} (s:5.38%)	{217} (s:5.38%)
	{494} (s:5.1%)	{494} (s:5.1%)	{494} (s:5.1%)
	{419} (s:5.06%)	{419} (s:5.06%)	{419} (s:5.06%)

Bảng 8.2. Kết quả về độ chính xác của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ 5%.

Ngoài ra, để quan sát sự chênh lệch về thời gian thực thi của các thuật toán trên bộ dữ liệu T10I4D100K, chúng tôi đã đưa số liệu của thời gian thực thi vào phần mềm Power BI để so sánh thời gian thực thi giữa 3 thuật toán Apriori, Apriori cải tiến và Apriori trên môi trường song song.



Hình 8.1. Kết quả thực nghiệm về thời gian thực thi của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ thay đổi từ 1% đến 5%.

Kết quả cho thấy, thuật toán Apriori thuần có thời gian thực thi chậm nhất trong cả ba thuật toán, thuật toán Apriori cải tiến có thời thực thi nhanh hơn gần như gấp đôi so với thuật toán Apriori, còn đối với Apriori trên môi trường song song có tốc độ nhanh nhất hầu như đều giao động từ 10s – 20s, đối với các độ support, tuy nhiên từ độ support 5% trở đi, thì tốc độ thực thi của thuật toán này đều ở mức 8s-9s và không thể thấp hơn bởi vì đó là khoảng thời gian tối thiểu để khởi động các Job chạy Map-Reduce.

8.3. So sánh thời gian thực thi giữa thuật toán Apriori và Apriori cải tiến

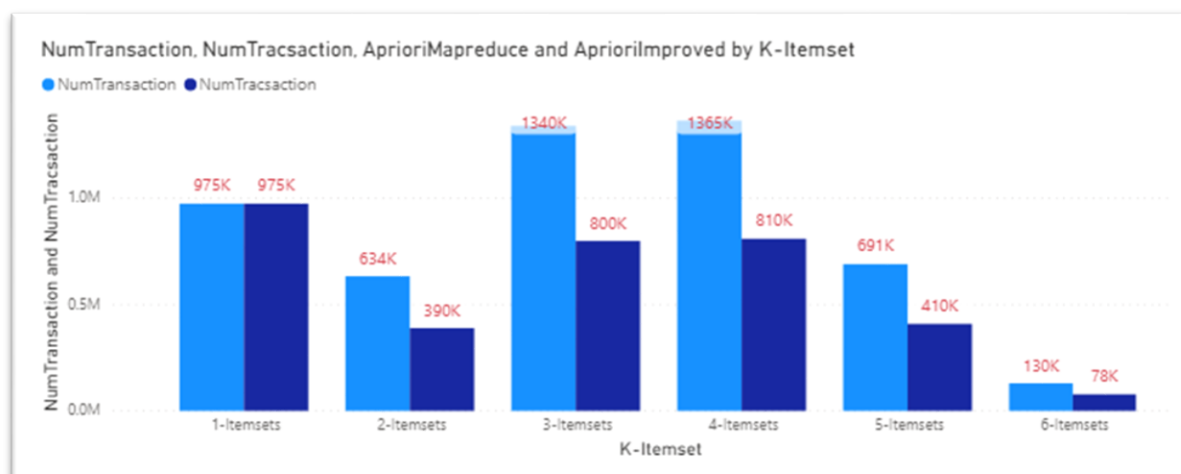
Như chúng ta đã đề cập ở phần tìm hiểu lý thuyết về thuật toán Apriori và Apriori cải tiến thì ở thuật toán Apriori khi đếm số lượng xuất hiện của các Itemset, thì chương trình phải duyệt lại toàn bộ tất cả các transaction có trong tập dữ liệu. Nhưng đối với thuật toán Apriori cải tiến, khi đếm số lượng xuất hiện của các Item thì chương trình chỉ duyệt trên những transaction nào có sự xuất hiện của các Itemset đó nên số lượng transaction mà chương trình duyệt sẽ ít hơn so với thuật toán Apriori.

Để dễ dàng hình dung về sự chênh lệch số lượng transaction mà chương trình phải duyệt ở mỗi Itemset chúng tôi đã làm thực hiện so sánh số lượng transaction mà cả hai thuật toán Apriori và Apriori cải tiến cùng duyệt ở mỗi K-Itemset trên tập dữ liệu Mushroom (8124 TID).

K-Itemset	Apriori	Apriori Cải tiến
1-Itemset	974880	974880
2-Itemset	633672	389823
3-Itemset	1340460	799502
4-Itemset	1364832	809974
5-Itemset	690540	410003
6-Itemset	129984	77961

Bảng 8.3. Kết quả về số lượng duyệt các transaction khi thực hiện thuật toán Apriori và Apriori cải tiến ở mỗi K-Itemset với tập dữ liệu Mushroom.

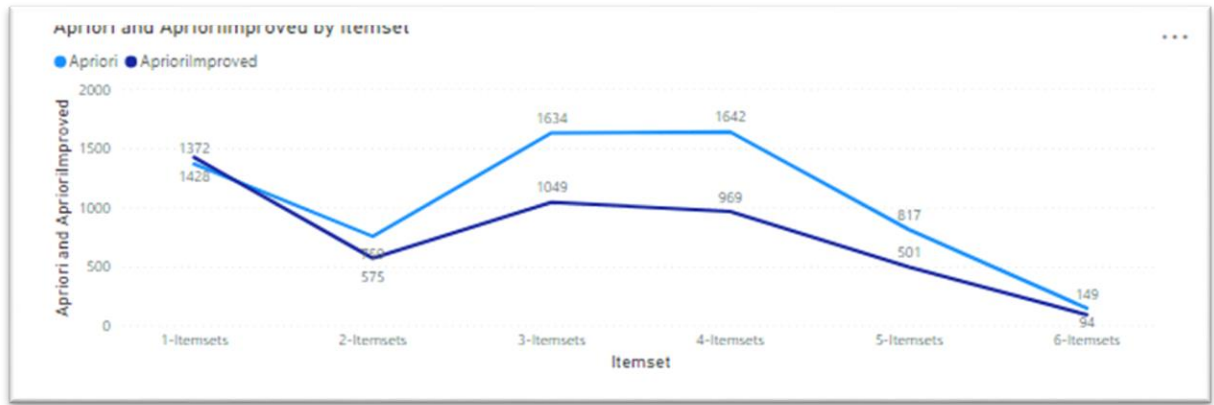
Dựa trên kết quả chúng ta dễ dàng thấy được số lần thực hiện duyệt transaction ở mỗi thuật toán là rất nhiều, với tập dữ liệu Mushroom chỉ với 8124 transaction nhưng số lượng duyệt transaction đối với mỗi K-Itemset lên tới con số triệu lần (nhiều nhất là **1364832** lần ở 4-Itemset). Ta dễ dàng quan sát thấy ở thuật toán Apriori số lượng duyệt là rất nhiều so với Apriori cải tiến và để dễ dàng so sánh, chúng tôi đưa dữ liệu trên lên PBI để dễ dàng nhận thấy sự chênh lệch giữa hai thuật toán này.



Hình 8.2. Kết quả thực nghiệm về số lượng transaction mà mỗi K-itemset phải duyệt trong quá trình thực thi chương trình ở thuật toán Apriori và thuật toán Apriori cải tiến

Quan sát trên biểu đồ, ta dễ dàng nhận thấy được sự chênh lệch và thay đổi giữa số lượng transaction được duyệt ở mỗi K-Itemset. Và dựa trên số liệu từ chương trình trả về ta dễ dàng nhận thấy, ở 1-Itemset thì số lượng transaction được duyệt giữa cả hai thuật toán hầu như là tương tự bởi vì theo lý thuyết thì lần ở 1-Itemset thì cả hai thuật toán đều có cách xử lý và đếm số lượng xuất hiện của các Item là như nhau. Nhưng số lượng transaction từ 2-Itemsets trở đi hầu như đều giảm đi khá nhiều (có những K-itemset, số lượng giao dịch gần như giảm đi một nửa), điều đó cho thấy thuật toán Apriori cải tiến đã giúp chương trình duyệt cơ sở dữ liệu ít hơn nhiều so với thuật toán Apriori thông thường.

Và để nhận định kỹ hơn về thời gian thực thi giữa thuật toán Apriori thông thường và Apriori cải tiến, chúng ta cùng xem thời gian thực thi của chương trình ở từng K-Itemset để xem hiệu năng và hiệu suất của thuật toán Apriori cải tiến nổi trội hơn như thế nào.



Hình 8.3. Kết quả thực nghiệm về thời gian xử lý ở mỗi K-itemset trong quá trình thực thi chương trình ở thuật toán Apriori và thuật toán Apriori cải tiến

Quan sát biểu đồ ta thấy, khi cùng xử lý ở 1-Itemset thì thời gian thực thi của thuật toán Apriori sẽ nhanh hơn thời gian thực thi của thuật toán Apriori cải tiến, lý do xảy ra sự chênh lệch này trong khi cả 2 đều cùng nhau duyệt chung một số lượng transaction như nhau (974880) có lẽ do ở thuật toán Apriori cải tiến, chương trình phải xử lý phân tạo một mảng trống để lưu các vị trí xuất hiện của các Itemset thuật các transaction nào nên dẫn tới chênh lệch như trên hình. Tuy nhiên sau khi đã có được mảng lưu các vị trí xuất hiện thì từ 2-Itemsets trở đi, thời gian thực thi của Apriori cải tiến hầu như nhanh hơn rất nhiều so với Apriori thông thường, đây là điều dễ hiểu khi số lượng duyệt transaction của Apriori cải tiến ít hơn rất nhiều so với Apriori thông thường.

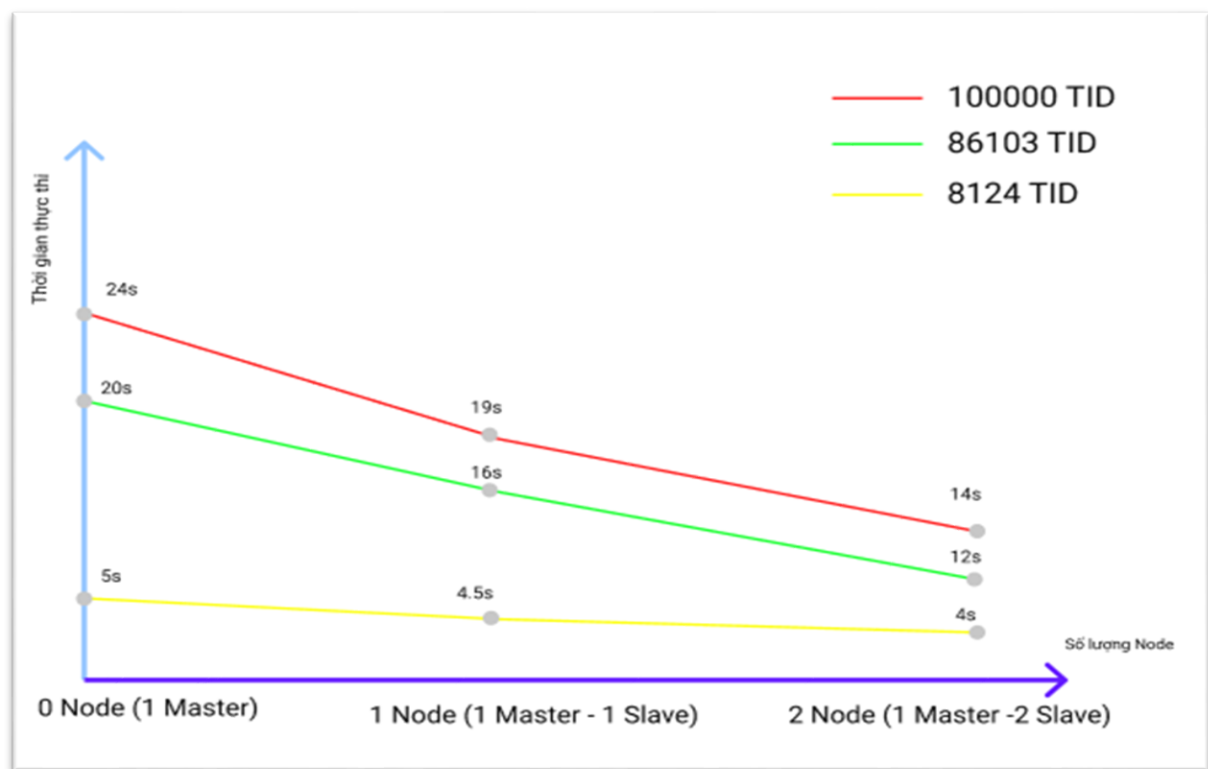
8.4. So sánh thời gian thực thi thuật toán Apriori trên môi trường xử lý song song.

Như đã đề cập ở phần đầu tiên, so sánh hiệu suất giữa các thuật toán cho thấy rằng thuật toán Apriori trên môi trường xử lý song song có sự chênh lệch hẳn so với hai thuật toán còn lại (1500s so với 20s) khi xử lý ở các tập dữ liệu cực lớn cũng như ở các độ support cực kỳ nhỏ, cho thấy sự vượt trội rất nhiều của thuật toán Apriori trên môi trường song song. Tuy nhiên, để cài đặt môi trường xử lý song song, không những phải đòi hỏi cấu hình của máy tính cũng như cách thức cài đặt về node master và mode slave trên hệ thống hadoop. Và để kiểm định về hiệu suất thực thi chương trình đối với các cách cài đặt khác nhau đối với các node master và các node slave, chúng tôi cùng thực hiện thuật toán Apriori trên nhiều

tập dữ liệu khác nhau với nhiều các cài đặt khác nhau như: 1 master – 0 slave, 1 master – 1 slave và 1 master – 2 slave vì cấu hình thiết bị của chúng tôi có giới hạn nên không thể thực nghiệm trên những trường hợp nhiều node hơn.

	1 master – 0 slave	1 master – 1 slave	1 master – 2 slave
100000 TID	24s	19s	14s
86103 TID	20s	16s	12s
8421 TID	5s	4.5s	4s

Bảng 8.4. Kết quả về thời gian thực thi chương trình trên từng cấu hình hadoop khác nhau



Hình 8.4. Kết quả về thời gian thực thi chương trình trên từng cấu hình hadoop khác nhau

Dựa trên kết quả đo được, chúng ta dễ dàng đưa ra kết luận với nhiều node hơn, nhiều slave hơn thì tốc độ xử lý dữ liệu của hệ thống sẽ nhanh hơn khoảng 25% khi bật thêm một node, đối với những tập dữ liệu có số lượng transaction nhỏ thì sự chênh lệch này rất khó nhận biết được bởi vì thời gian xử lý đối với những tập dữ liệu này ra rất nhanh dù đối với 1 Node hay nhiều Node thì tốc độ cũng không có sự khác biệt.

Tuy nhiên trong nhiều tài liệu tham khảo, thì có nhiều trường hợp khi xử dụng nhiều node hơn thì có khi không tối ưu và hiệu quả hơn so với ít node, đây là một trong những vấn đề nan giải mà nhóm chúng tôi đang tìm cách thực nghiệm là kiểm chứng, tuy nhiên vì cấu hình hệ thống có hạn, rất khó để chúng tôi có thể thực hiện kiểm nghiệm này một cách chính xác và hiệu quả cũng như đem lại một kết quả trực quan nhất có thể.

Table 1. T10I4D100K database

Nodes (Processors)	Wall Time (Secs)
1	1322
2	995
4	1017
8	998

Table 2. T10I4D50K database

Nodes (Processors)	Wall Time (Secs)
1	1066
2	550
4	515
8	551

Bảng 8.5. Kết quả về trường hợp nhiều node hơn nhưng thời gian thực thi cao hơn ít node

Nguyên nhân gây ra việc với nhiều Node hơn những hiệu suất lại kém hơn so với ít Node phần lớn là do cấu hình giữa các Node không đồng đều dẫn tới việc khi phân chia tập dữ liệu vào các node không có sự thống nhất, node thì quá ít dữ liệu, node thì quá nhiều dữ liệu làm cho chương trình không thể tối ưu. Một trong những lý do được đưa ra là do yếu tố tức thời của hệ thống (có thể là hệ thống windown, cũng có thể là hệ thống hadoop, RAM, CPU, ..) tuy nhiên chênh lệch ở đây là không nhiều, và kết luận được đưa ra là đối với những tập dữ liệu phù hợp nên dùng phù hợp các Node để không bị lãng phí tài nguyên cũng như đem lại một hiệu suất tốt nhất cho hệ thống.

KẾT LUẬN

Khai phá dữ liệu là một lĩnh vực nghiên cứu về việc phát hiện ra tri thức trong một cơ sở dữ liệu rộng lớn bằng các phương thức thông minh đang thu hút các nhà nghiên cứu và người dùng trong lĩnh vực công nghệ thông tin. Nghiên cứu ở lĩnh vực này đòi hỏi sự tích hợp các kết quả nghiên cứu ở nhiều lĩnh vực của khoa học máy tính và việc áp dụng nó trong từng nhiệm vụ của khai phá dữ liệu.

Tìm hiểu về thuật toán Apriori cũng như triển khai thuật toán Apriori trên môi trường xử lý song song mang lại kiến thức vô cùng đa dạng về khai phá dữ liệu, hệ thống xử lý dữ liệu song song, dữ liệu phân tán và dữ liệu lớn. Ngoài ra các ứng dụng của thuật toán Apriori còn có thể áp dụng vào thực tiễn giúp các doanh nghiệp bán lẻ có thể thống kê các thông tin về nhu cầu sử dụng sản phẩm của khách hàng từ đó nâng cao doanh thu và đáp ứng các nhu cầu của khách hàng một cách cấp thiết nhất.

1. Kết quả đạt được

Trình bày một cách khái quát về khai phá dữ liệu và phát hiện tri thức, quy trình khai phá dữ liệu, lựa chọn phương pháp khai phá dữ liệu. Các khái niệm cũng như các cách thức căn bản nhất về lĩnh vực data mining, từ đó có thể phát triển lên nghiên cứu các tính nhất, thuộc tính của dữ liệu.

Tìm hiểu tổng quan về lý thuyết của thuật toán Apriori, ưu nhược điểm từ đó tiếp tục nghiên cứu thuật toán Apriori cải tiến giúp giải quyết các nhược điểm của thuật toán trước đó. Ngoài ra để tối ưu về hiệu suất hoạt động của thuật toán, chúng tôi còn tìm hiểu được các kiến thức về hệ thống Hadoop mapreduce để tiến hành triển khai thuật toán Apriori trên môi trường xử lý song song.

Cài đặt được thuật toán Apriori, thuật toán Apriori cải tiến, cài đặt hệ thống Hadoop mapreduce, cài đặt thuật toán Apriori trên môi trường hadoop 1 Node master và 2 Node slave.

Chạy chương trình và đánh giá hiệu suất giữa các thuật toán với nhau, đánh giá độ hiệu quả của thuật toán Apriori cải tiến so với thuật toán Apriori thông thường, ngoài ra chúng tôi còn thực nghiệm để đánh giá độ ảnh hưởng của cấu hình hệ thống Hadoop đối với hiệu suất thực thi chương trình trên môi trường song song.

2. Những hạn chế: Khóa luận vẫn còn có những hạn chế:

Lý thuyết về khai phá dữ liệu còn ở mức cơ bản, chưa đi sâu nhiều vào kỹ thuật cũng như những tính chất, đặc điểm của từng loại dữ liệu. Chỉ tìm hiểu một số luật kết hợp cơ bản, chưa nắm vững hoàn toàn cách thức cũng như phương pháp khai phá luật kết hợp.

Cài đặt thuật toán Apriori và Apriori cải tiến còn chưa tối ưu được các câu lệnh dẫn tới thời gian thực thi chương trình còn kém xa so với một số chương trình Apriori và Apriori cải tiến đã có trước đó.

Tập dữ liệu còn mang tính đặc thù (dạng số) cũng như độ lớn về tập dữ liệu cũng chưa nhiều cũng như chưa có độ thực tế so với các tập giao dịch bán hàng hiện có.

Hệ thống còn hạn chế về cấu hình nên còn nhiều thực nghiệm chưa thể kiểm chứng, dẫn đến các số liệu đánh giá còn mang nhiều tính chủ quan về hạn chế.

3. Hướng phát triển

Các giải thuật khai phá dữ liệu chỉ là một phần công việc trong phát hiện tri thức. Trong tương lai, nhóm chúng tôi cũng cần phải quan tâm hơn đến các giai đoạn khác: lựa chọn dữ liệu, làm sạch và tiền xử lý dữ liệu, ... Khai phá dữ liệu, phát hiện tri thức được thực hiện lặp đi lặp lại, có sự tương tác với nhau vì vậy ta cũng cần tìm hiểu thêm sự tác động lẫn nhau đó, kết hợp nhiều thuật toán khai phá dữ liệu với nhau để tạo ra kết quả tốt hơn nữa. Cũng như nâng cấp về cấu hình của hệ thống máy tính để có thể thực nghiệm thêm nhiều trường hợp cũng như chạy trên những tập dữ liệu có độ lớn lên tới hơn một triệu giao dịch.

TÀI LIỆU THAM KHẢO

- [1] Võ Thị Ngọc Châu, Luật kết hợp, tại Đại học Bách Khoa Tp.Hồ Chí Minh, 2012.
- [2] Nguyễn Đăng Cẩm, Luận văn thạc sĩ, Ứng dụng bài toán khai phá song song luật kết hợp để phân tích đơn thuốc bệnh viện đa khoa Phú Riềng, 2018, pp 20-25.
- [3] Nguyễn Đăng Cẩm, Nguyễn Thành Sơn, Tạp chí khoa học giáo dục kỹ thuật số 53 (07/2019), Trường đại học sư phạm kỹ thuật TP HCM, Đánh giá hiệu quả của thuật toán khai phá luật kết hợp môi trường xử lý song song.
- [4] Chia-Chu Chiang, Yanbin Ye, A Parallel Apriori Algorithm for Frequent Itemsets Mining, 2017, p 87-89
- [5] <https://viblo.asia/p/tim-hieu-ve-hadoop-bJzKmOBXl9N>
- [6] <https://kipalog.com/posts/Tong-quan-mo-hinh-lap-trinh-MapReduce>
- [7] <https://topdev.vn/blog/hadoop-la-gi>
- [8] Lê Thị Minh Châu, Bài giảng hướng dẫn cài đặt Hadoop nhiều Node, Môn BigData Essentials, 2019, Tuần 4
- [9] <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>
- [10] <https://kcntt.duytan.edu.vn/Home/ArticleDetail/vn/170/2615/loi-the-va-thach-thuc-cua-khai-pha-du-lieu>