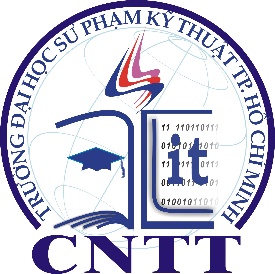


**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**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 KỸ SƯ CNTT**

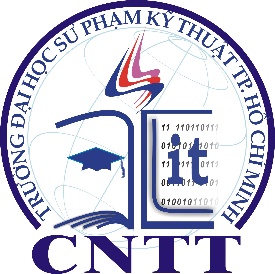
**GIẢNG VIÊN HƯỚNG DẪN**

**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 KỸ SƯ CNTT**

**GIẢNG VIÊN HƯỚNG DẪN**

**TS. NGUYỄN THÀNH SƠN**

**KHÓA 2017-2021**

**XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

**\*\*\*\*\*\*\*\*\***

**ĐH SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CNTT**

**\*\*\*\*\*\*\***

**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)*

**XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

**\*\*\*\*\*\*\*\*\***

**ĐH SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CNTT**

**\*\*\*\*\*\*\***

**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 kĩ 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](#_Toc74516997)

[LỜI MỞ ĐẦU 9](#_Toc74516998)

[CHƯƠNG 1: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU 12](#_Toc74516999)

[**1.1. Khai phá dữ liệu 12**](#_Toc74517000)

[**1.2. Kiến trúc data mining 13**](#_Toc74517001)

[CHƯƠNG 2: LUẬT KẾT HỢP VÀ THUẬT TOÁN APRIORI 15](#_Toc74517002)

[**2.1. Luật kết hợp: 15**](#_Toc74517003)

[**2.2. Giải thuật Apriori 16**](#_Toc74517004)

[**2.3. Cài đặt thuật toán Apriori 21**](#_Toc74517005)

[CHƯƠNG 3: TÌM HIỂU THUẬT TOÁN APRIORI TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG 22](#_Toc74517006)

[**3.1. Giới thiệu tổng quan 22**](#_Toc74517007)

[**3.2. Triển khai thuật toán Apriori cho tính toán song song 22**](#_Toc74517008)

[CHƯƠNG 4: TÌM HIỂU THUẬT TOÁN APRIORI CẢI TIẾN 26](#_Toc74517009)

[**4.1. Ý tưởng của thuật toán Apiori cải tiến 26**](#_Toc74517010)

[**4.2. Thuật toán Apriori cải tiến 27**](#_Toc74517011)

[**4.3. Vận dụng thuật toán Apriori cải tiến 27**](#_Toc74517012)

[CHƯƠNG 5: TÌM HIỂU VỀ HADOOP VÀ MAPREDUCE 29](#_Toc74517013)

[**5.1. Tìm hiểu về Hadoop 29**](#_Toc74517014)

[**5.2. Vai trò của Hadoop 29**](#_Toc74517015)

[**5.3. Nguyên tắc hoạt động của Hadoop 30**](#_Toc74517016)

[**5.4. Tìm hiểu về Hadoop framework 30**](#_Toc74517017)

[**5.5. Tìm hiểu về MapReduce 31**](#_Toc74517018)

[**5.6. Ưu điểm của MapReduce 32**](#_Toc74517019)

[**5.7. Nguyên tắc hoạt động 32**](#_Toc74517020)

[CHƯƠNG 6: THỰC HÀNH CÀI ĐẶT VÀ TRIỂN KHAI HADOOP VÀ MAPREDUCE 33](#_Toc74517021)

[**6.1. Yêu cầu về cấu hình máy, phần mềm cần chuẩn bị 33**](#_Toc74517022)

[**6.2. Các bước cài đặt Hadoop và Mapreduce 33**](#_Toc74517023)

[**6.3. Chạy ví dụ về Hadoop Mapreduce 45**](#_Toc74517024)

[CHƯƠNG 7: TRIỂN KHAI THUẬT TOÁN APRIORI CẢI TIẾN TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG. 48](#_Toc74517025)

[**7.1. Tóm tắt quá trình thực thi 48**](#_Toc74517026)

[**7.2. Cài đặt 50**](#_Toc74517027)

[CHƯƠNG 8: ĐÁNH GIÁ THỰC NGHIỆM 52](#_Toc74517028)

[**8.1. Đánh giá kết quả 52**](#_Toc74517029)

[**8.2. Thời gian thực thi giữa các thuật toán 52**](#_Toc74517030)

[**8.3. So sánh thời gian thực thi giữa thuật toán Apriori và Apriori cải tiến 54**](#_Toc74517031)

[**8.4. So sánh hiệu suất thực thi thuật toán Apriori trên môi trường xử lý song song. 57**](#_Toc74517032)

[KẾT LUẬN 60](#_Toc74517033)

[TÀI LIỆU THAM KHẢO 62](#_Toc74517034)

**DANH SÁCH HÌNH ẢNH**

[**Hình 1. Quá trình khai phá dữ liệu 12**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013966)

[**Hình 2. Kiến trúc data mining 13**](#_Toc75013967)

[**Hình 3. Mô tả quá trình của thuật toán Apriori 17**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013968)

[**Hình 4. Số lượng quy trình phân phối 24**](#_Toc75013969)

[**Hình 5. Phân phối Trie để tạo ra k-itemsets ứng viên 25**](#_Toc75013970)

[**Hình 6. Mô tả cách cải tiến thuật toán Apriori 26**](#_Toc75013971)

[**Hình 7. Tập giao dịch thực thi Apriori cải tiến 27**](#_Toc75013972)

[**Hình 8. Sinh ra 1-itemset thuật toán Aprior cải tiến 28**](#_Toc75013973)

[**Hình 9. Sinh ra 2-itemset thuật toán Apriori cải tiến 28**](#_Toc75013974)

[**Hình 10. Sinh ra 3-itemset thuật toán Apriori cải tiến 28**](#_Toc75013975)

[**Hình 11. Cài đặt Java 33**](#_Toc75013976)

[**Hình 12. Kiểm tra version java 34**](#_Toc75013977)

[**Hình 13. Cài đặt thông số 35**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013978)

[**Hình 14. Cấu hình hostname 36**](#_Toc75013979)

[**Hình 15. Cấu hình các biến môi trường 38**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013980)

[**Hình 16. Cấu hình đúng đường dẫn cho Java 39**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013981)

[**Hình 17. Cấu hình thuộc tính cho máy ảo 40**](#_Toc75013982)

[**Hình 18. Cấu hình Map Reduce 41**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013983)

[**Hình 19. Thêm các slave trên máy master 43**](#_Toc75013984)

[**Hình 20. Kiểm tra các thành phần 44**](#_Toc75013985)

[**Hình 21. Kiểm tra sự hoạt động của slave 45**](#_Toc75013986)

[**Hình 22. Kết quả ví dụ WordCount 47**](#_Toc75013987)

[**Hình 23. Quá trình tạo 1-itemset Apriori song song 49**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013988)

[**Hình 24. Quá trình tạo 1-itemset Apriori song song 49**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013989)

[**Hình 25. Tập dataset chạy ví dụ 50**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013990)

[**Hình 26. Kết quả 1-itemset Apriori song song 51**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013991)

[**Hình 27. Kết quả 2-itemset Apriori song song 51**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013992)

[**Hình 28. Kết quả 2-itemset Apriori song song 51**](file:///E:\HKII_Nam4\BAO_CAO_KLTN_FINAL.docx#_Toc75013993)

[**Hình 29. 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%. 54**](#_Toc75013994)

[**Hình 30. 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 56**](#_Toc75013995)

[**Hình 31. 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 57**](#_Toc75013996)

[**Hình 32. Kết quả về thời gian thực thi chương trình trên từng cấu hình hadoop khác nhau 58**](#_Toc75013997)

**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 vị nghiên cứu**

* Mục tiêu:

- Tìm hiểu bài toán khai phá luật kết hợp.

- Tìm hiểu bài toán khai phá luật kết hợp sử dụng giải thuật song song (parallel mining of association rules).

- Cài đặt demo khai phá luật kết hợp sử dụng giải thuật song song.

* Đối tượng nghiên cứu

Luật kết hợp, khai phá luật kết hợp sử dụng giải thuật song song

- Một số thuật toán khai phá luật kết hợp 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 giải thuật song song.

**3. Nhiệm vụ và hướng tiếp cận của luận văn**

Nghiên cứu về thuật toán khai phá luật.

Nghiên cứu tìm hiểu các thuật toán khai phá luật kết hợp sử dụng giải thuật song song trong khai phá dữ liệu.

Cài đặt một số thuật toán song song khai phá dữ liệu và phát hiện luật kết hợp.

**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 5 chương:

Chương 1: Tổng quan về khai phá dữ liệu.

Chương 2: Luật kết hợp và thuật toán Apriori.

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

Chương 4: Tìm hiểu thuật toán Apriori cải tiến.

Chương 5: Tìm hiểu về Hadoop và MapReduce.

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

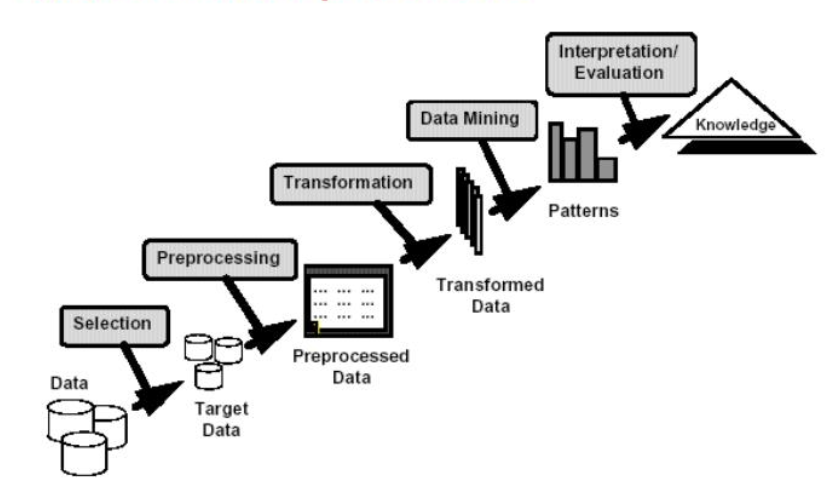
Chương 7: Triển khai thuật toán Apriori trong môi trường song song.

Chương 8: Đánh giá thực nghiệm.

# **CHƯƠNG 1: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU**

* 1. **Khai phá dữ liệu**

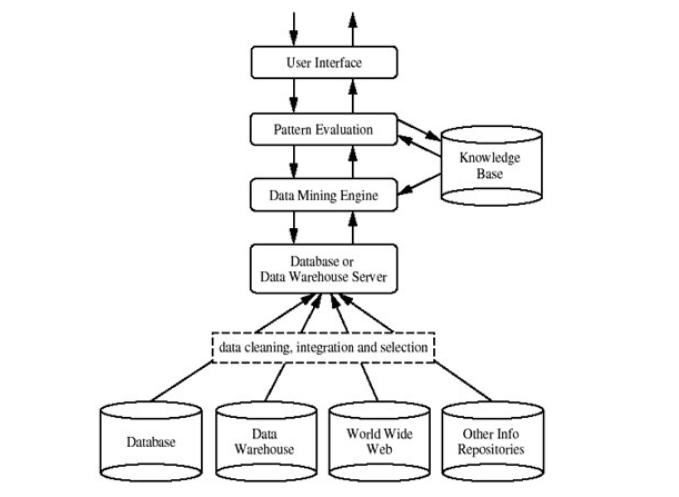
Khai phá dữ liệu (Data Mining) là quá trình khám phá ra những thông tin tìm ẩn được tìm thấy trong các cơ sở dữ liệu, góp phần cho việc khám phá tri thức từ cơ sở dữ liệu (Knowledge Discovery in Database - KDD) thông qua việc rút trích được các mẫu tri thức quan trọng từ một dữ liệu rất lớn[1].

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. 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. **Kiến trúc data mining[2]**



Hình 2. Kiến trúc data mining

* 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 lỗi mong muốn.
* Máy khai phá dữ liệu (Data mining engine): Hệ thống được cài đặt bằng các modun có các thuật toán khai phá.
* Modun 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.

# **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 sỡ dữ liệu các giao dịch t1,t2,…tn. T = {t1,t2,…,tn}. Mỗi giao dịch t gồm n thuộc tính I= {i1,i2…in}.

Mục đích của luật kết hợp giữa các thuộc tính. Những luật kết hợp này có dạng X => 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 = {Bia, Nước Ngọt} và Y ={Mực Khô, Bò Khô} và ta có luật kết hợp X => 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 cocacola 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 cocacola 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 lượt buôn bán của mình.

**Các định nghĩa cơ bản[3]:**

- Tập mục (itemset): là một tập hợp gồm một hoặc nhiều mục. Tâp mục mức k (k-itemset) có k mục. Ví dụ: 3-itemset là {Bánh mì, sữa, cafe}.

- Luật kết hợp – kí hiệu X -> Y, trong đó X, Y là các tập mục. Ví dụ: bánh mì -> sữa.

* Tổng số hỗ trợ (support count)- kí hiệu σ : là số lần xuất hiện của một tập mục. Ví dụ: σ({Bánh mì, sữa, cafe}) = 2, số lần khách hàng mua cùng lúc 3 sản phẩm trong tập giao dịch D.
* 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({Bánh mì, sữa, cafe}) = 2/5. Tổng số lần khách hàng mua cả 3 sản phẩm trên tổng 5 giao dịch.

Công thức : Supp(X) = |X| / |D|

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

Supp(X=>Y) =

* 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({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 =>Y là xác suất xảy ra Y khi đã biết X

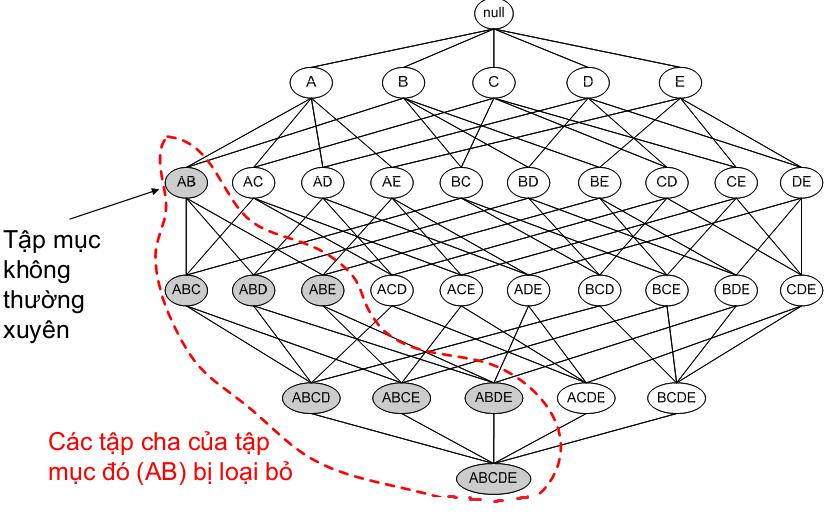
Conf(X⇒Y) =

* 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 ≥ giá trị ngưỡng **minsup.**
* Độ tin cậy c ≥ giá trị ngưỡng **minconf**.
  1. **Giải thuật 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:

**Nội dung thuật toán**

Hình 3. 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:**

L1 = {large 1-itemset};

**for** (k = 2; Lk-1 ≠ 𝜙; k++) **do begin**

Ck = apriori\_gen(Lk-1); // sinh tập mục ứng cử mới Ck;

**forall** giao dịch t ∈ D **do begin**

Ct = subset(Ck, t); // các tập mục ứng cử chứa trong t;

**forall** tập mục ứng cử ci ∈ Ct **do**

ci.count ++ ;

**end;**

Lk = {ci ∈ Ck|ci.count ≥ minsup}

**end;**

**Answer** = ∪kLk ;

**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, báng 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 Sup<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 phông: 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 >=minsup

L= {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, Nhãn, 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 |

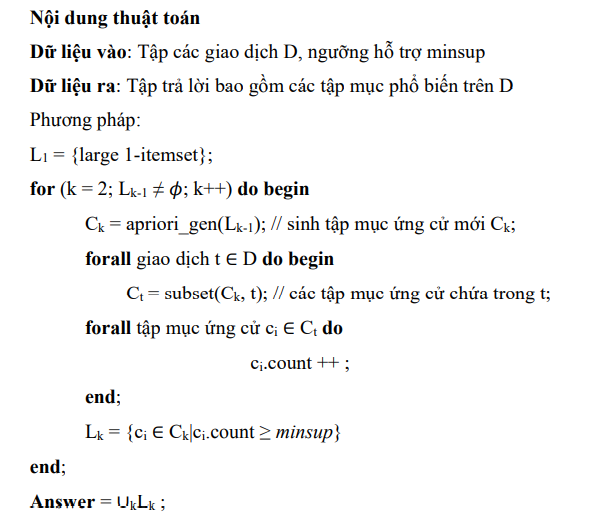
|  |  |  |
| --- | --- | --- |
| **Luật** | **Độ tin cậy** | **Support** |
| Bánh Mì => Sữa | 80% | 57% |
| Sữa => Bánh Mì | 80% | 57% |
| Bánh Mì =>Cafe | 60% | 42,8% |
| Cafe => Bánh Mì | 100% | 42,8% |
| Cafe => Sữa | 100% | 42,8% |
| Sữa => Cafe | 60% | 42,8% |

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. Cài đặt thuật toán Apriori**



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

## **3.1. Giới thiệu tổng quan[4]**

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[4]**

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 TDB1, TDB2, TDB3,…,TDBn. 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 cụ 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.

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, L1.

- Chia tệp TDB thành N phần;

- Bộ xử lý Pi đọc TDB [i] để tạo ứng viên cục bộ 1-itemsets từ phân vùng của nó;

- Pi chuyển các tập 1-itemsets ứng viên cục bộ của nó cho P0;

- P0 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, L1;

* Quét song song TDB để lấy bộ 2-itemsets phổ biến, L2

- Pi quét TDB để lấy bộ 2-itemsets ứng viên cục bộ;

- Pi chuyển tập 2-itemsets ứng viên cục bộ của nó cho P0;

- P0 tính tổng số của tất cả các tập 2-itemsets ứng viên cục bộ và cắt bớt nó để lấy L2;

* K = 3

- While (Lk-1 <> empty)

Begin

- Xây dựng cây ứng viên Trie[i] cho Ck trên Bộ xử lý dựa trên Pi trên Lk-1;

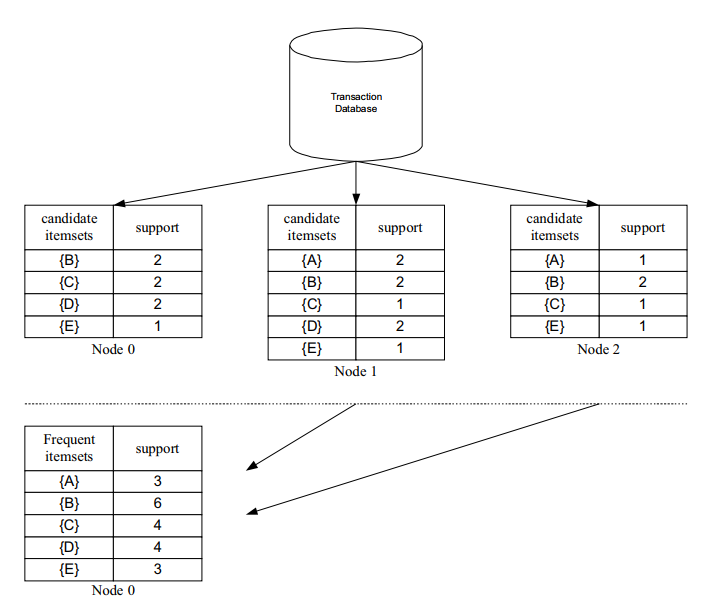
- Quét TDB [i] trên Bộ xử lý Pi để cập nhật số lượng Trie [i] cho Ck;

- Cắt Trie [i] để Ck lấy Lk;

- k = k + 1;

End;

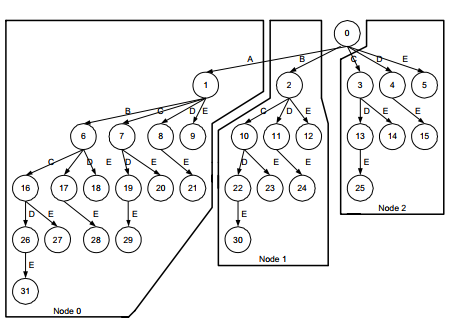
Để tính toán các tập 1-itemsets phổ biến, mỗi nút đếm các tập 1-itemsets cục bộ của nó từ phân vùng của nó và chuyển kết quả đến nút 0. Vì nút 0 nhận tất cả kết quả từ N-1 nút khác, nó tạo ra các tập 1-itemsets chung , L1.



Hình 4. 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 vectơ 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 ≥ 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 Lk-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-itemsests 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 5. 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 Apiori cải tiến**

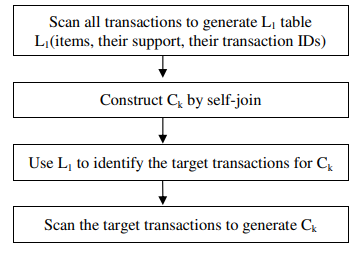
Để hiểu rõ được ý tưởng của thuật toán Apriori cải tiến, chúng ta cần phải có các định nghĩa sau:

*Định nghĩa 1:* Giả sử T = {T1, T2,…,Tm}, (m ≥ 1) là tập các giao dịch, Ti = {I1, I2,…,In}, (n ≥ 1) là tập các item và k-itemset = {i1, i2,…,ik), (k ≥ 1) cũng là tập k item và k-itemset ⊆ 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 Ck là tập phổ biến ứng viên (candidate itemset) có kích thước k và Lk 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 L1 chứa item, support count, transactionID nơi item được tìm thấy.



Hình 6. 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 L1 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-itemser 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 = find\_frequent\_1\_itemsets (T);

(2) For (k = 2; Lk-1 ≠Φ; k++) {

Bước 2: Tạo Ck từ LK-1

(3) Ck = candidates generated from Lk-1;

Bước 3: Lấy item Iw với support tối thiểu trong Ck bằng L1, (1≤w≤k).

(4) x = Get \_item\_min\_sup(Ck, L1);

Bước 4: Lấy transactionID có chứa x

(5) Tgt = get\_Transaction\_ID(x);

(6) For each transaction t in Tgt Do

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

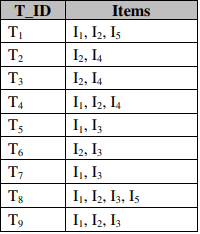
(8) Lk= items in Ck ≥ min\_support;

(9) End;

(10) }

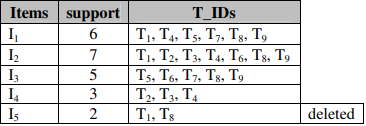
## **4.3. Vận dụng thuật toán Apriori cải tiến**

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



Hình 7. Tập giao dịch thực thi Apriori cải tiến

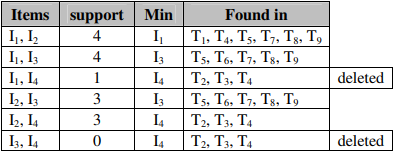
Đầu tiên, chúng ta quét tất cả các giao dịch để tìm tập ứng viên 1-itemset cùng với support count (support) và transactionID (T\_IDs). Sau đó, chúng ta loại bỏ tập ứng viên không thỏa mãn min\_sup (<min\_sup). Ta được L1 như bảng sau:



Hình 8. Sinh ra 1-itemset thuật toán Aprior cải tiến

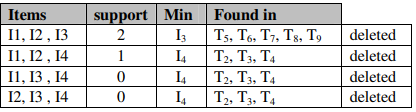
Bước tiếp theo, chúng ta tạo tập ứng viên 2-itemset từ L1. Để tìm support count cho mỗi itemset, hãy chia từng itemset trong 2-itemset thành hai phần tử, sau đó sử dụng bảng L1 để xác định các giao dịch nơi có thể tìm thấy itemset, thay vì tìm kiếm chúng trong tất cả các giao dịch.

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 I1 có hỗ trợ tối thiểu nhỏ nhất. Sau đó, chúng tôi tìm kiếm tập phổ biến (I1, I2) chỉ trong các giao dịch T1, T4, T5, T7, T8 và T9.



Hình 9. Sinh ra 2-itemset thuật toán Apriori cải tiến

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



Hình 10. Sinh ra 3-itemset thuật toán Apriori cải tiến

Sau bước này ta thấy các 3-itemset ứng viên đều không thõa 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 cỗ 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 đền 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 NodeManage

* ResourceManager: Quản lý toàn bộ tài nguyên tính toán của cluster.
* NodeManger: Giám sát việc sử dụng tài nguyên của container và báo cáo với ResourceManger. 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 khổng lồ 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.

*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 nguồn 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](https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html)

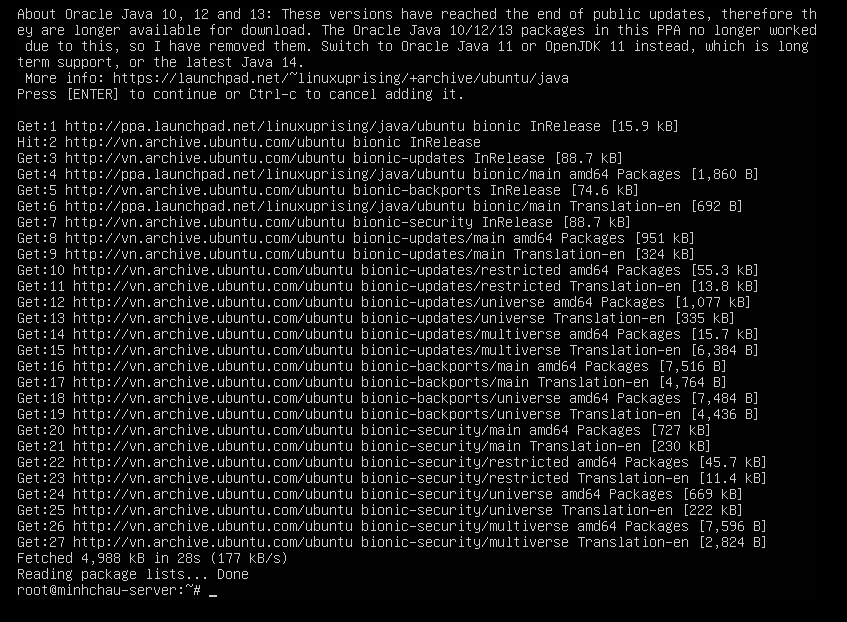
Hệ điều hành Ubuntu-server 18.04 [Link download](https://releases.ubuntu.com/18.04/)

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



Hình 11. Cài đặt Java

**Update và khởi động lại máy**

# apt update

# reboot

**Tiến hành cài đặt**

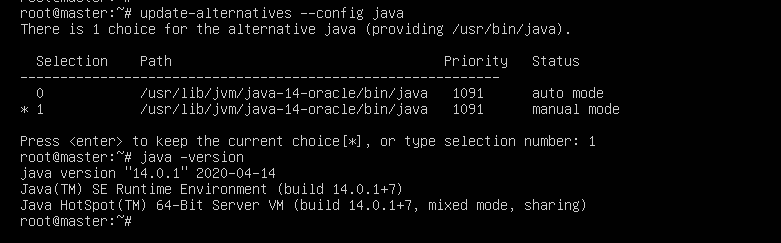
# apt install -y oracle-java14-installer

(Chấp nhận Liscence và chọn bản Oracle Manual Mode)

# update-alternatives –-config java

(Kiểm tra kết quả)

# java -version



Hình 12. Kiểm tra version java

**Cài đặt SSH**

# apt-get install ssh

# apt install openssh-server

**Cấu hình SSH**

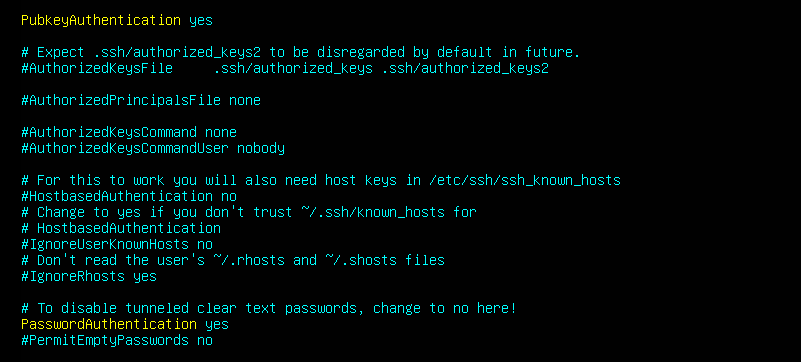
# vim /etc/ssh/sshd\_config (Cài đặt lại các thông số như bên dưới)

...

PubkeyAuthentication yes

PasswordAuthentication yes

...



Hình 13. 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

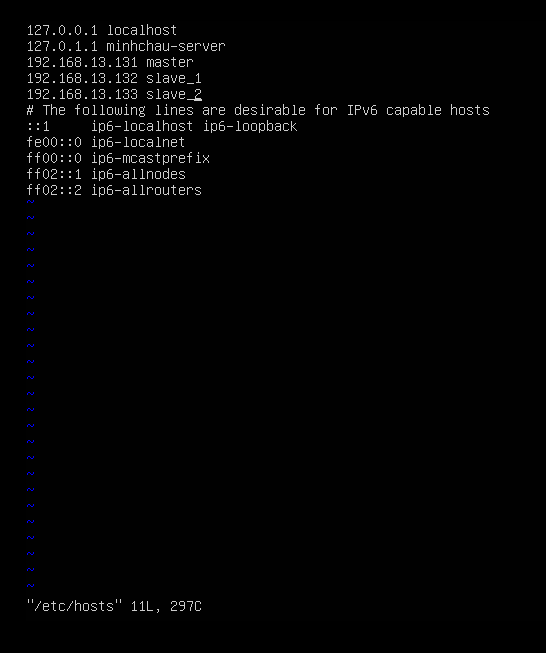
IP slave\_2: 192.168.13.133

# 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



Hình 14. 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

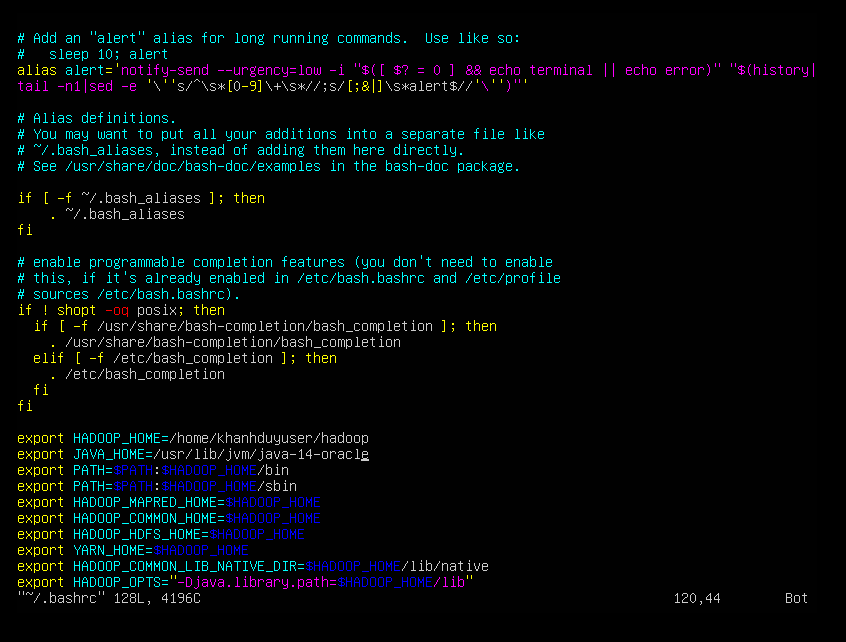
export YARN\_HOME=$HADOOP\_HOME

export

HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export

HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib"

 Nhấn Esc, nhập :wq để lưu và thoát file.

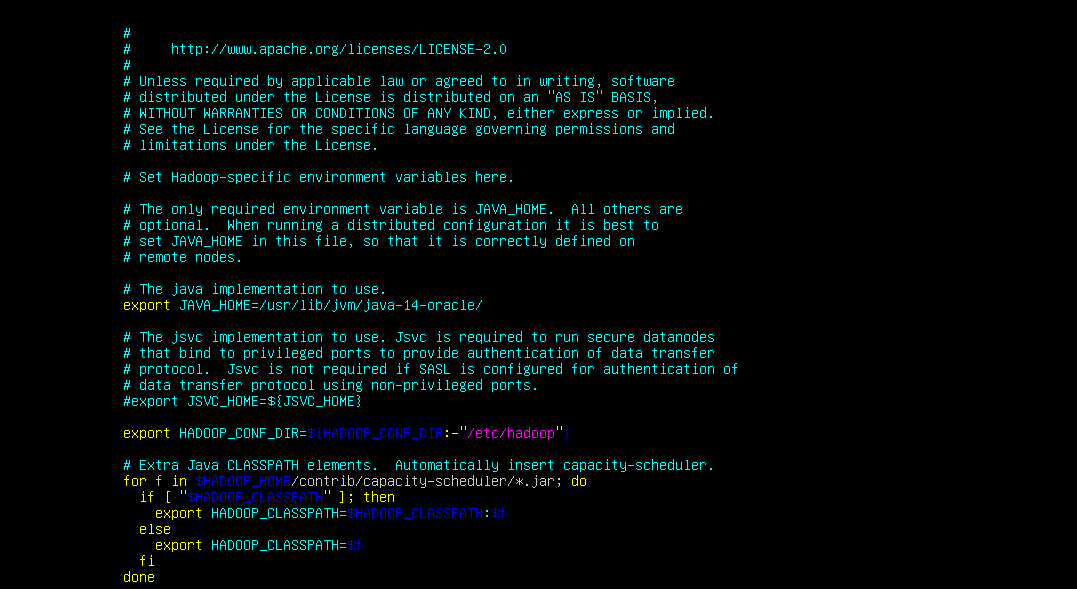
Hình 15. Cấu hình các biến môi trường

# source ~/.bashrc (Cập nhập lại file hệ thống)

***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/

Hình 16. Cấu hình đúng 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>



Hình 17. 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>

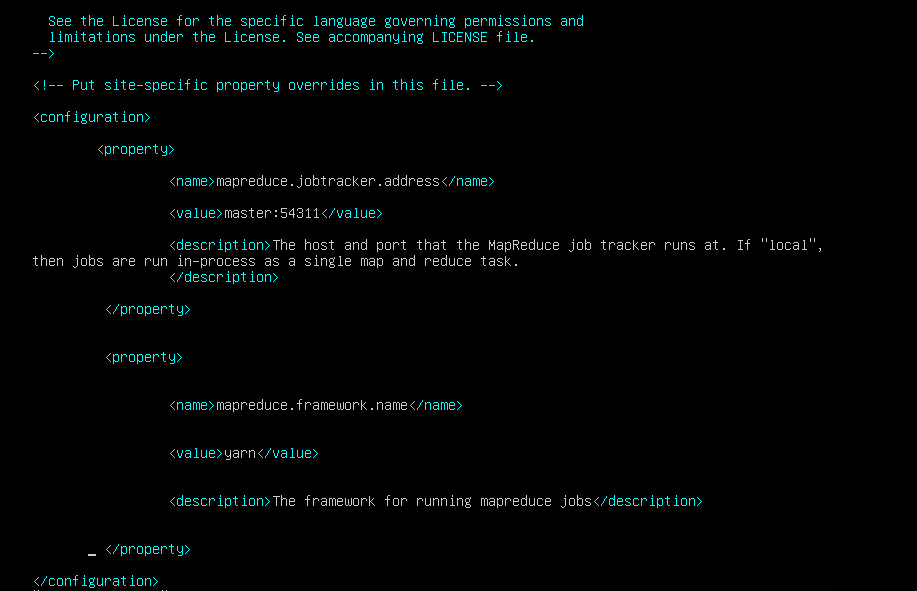
<property>

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

<value>yarn</value>

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

</property>

</configuration>

Hình 18. 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>

<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 19. 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)



Hình 20. Kiểm tra các thành phần

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



Hình 21. 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 dụng 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/hadoop-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

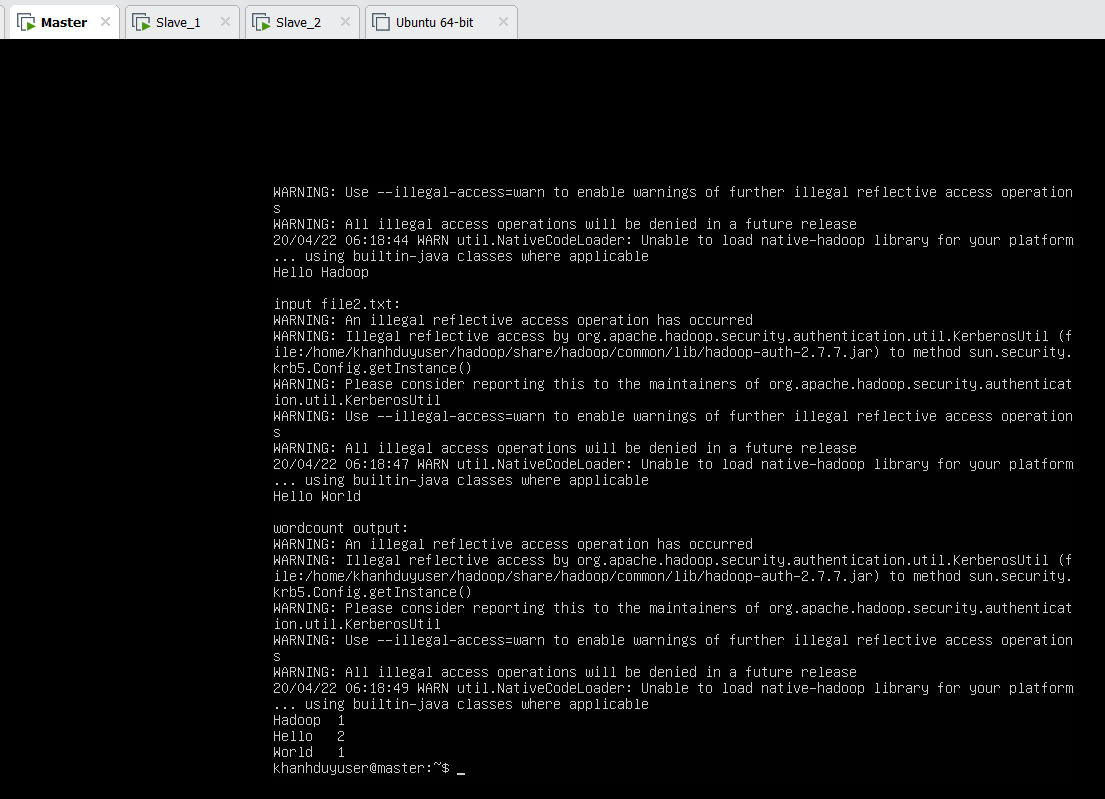
Kết quả:

...

World 1

Hadoop 1

Hello 2



Hình 22. 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:

Lk: tập phổ biến có độ dài k.

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

Để tìm 1-itemsets (tập phổ biến có kích thước 1, L1) 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 Lk trong MapReduce, chúng tôi thực hiện như sau:

*Giai đoạn Pre-Map*

Bước 1: Kết hợp Lk 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 item đặt 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 Lk không. Chúng tôi thực hiện việc kiểm tra này bằng cách giữ mã băm của các tập vật phẩm trong Lk trong một tập hợp và kiểm tra đối chiếu với tập hợp này. Nếu tất cả các tập con nằm trong Lk 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*

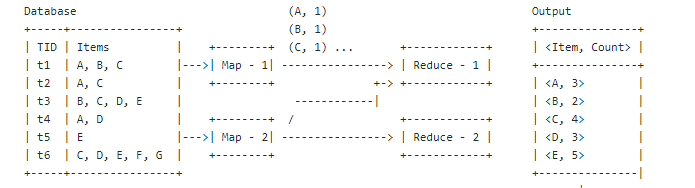
Bước 6: Thêm các số lên và nếu nó ở trên minsup hãy thêm tập phổ biến vào L(k + 1).

Ví dụ mô tả thuật toán trên:

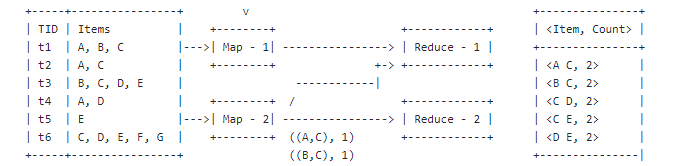
Số giao dịch = 6.

min\_sup = 40%.

min\_number of item = 2.

Tìm 1-itemsets:

Hình 23. Quá trình tạo 1-itemset Apriori song song

Tìm 2-itemsets:

Hình 24. Quá trình tạo 1-itemset Apriori song song

Tìm k-itemsets: Lặp lại đến khi nào L\_k rỗng.

* 1. **. 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

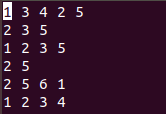
AprioriPassKMapper : Tái thực thi qua map từ 2 đến k.

AprioriReducer: Giai đoạn reduce.

ItemSet: Đại diện cho itemset.

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

AprioriUtils Chưa các phương thức thực thi của Apriori.

Tập dataset: 6 transaction

Hình 25. Tập dataset chạy ví dụ

Đẩy data lên hdfs

*hdfs dfs -put dataSet/sample*

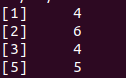
Câu lệnh thực thi

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

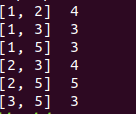
Chọn k-itemset = 3, support = 0.5, num-transaction = 6

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

Kết quả thực thi:

output1/part-r-00000

Hình 26. Kết quả 1-itemset Apriori song song

output2/part-r-00000

Hình 27. Kết quả 2-itemset Apriori song song

output3/part-r-00000

Hình 28. 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/**](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 |

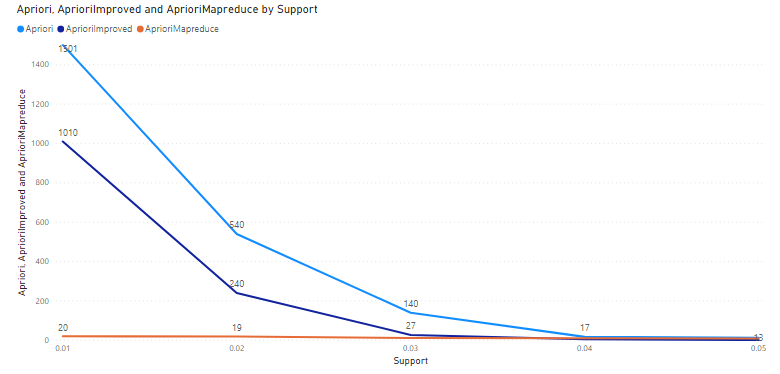
*Bảng 8.1. Kết quả thực nghiệm về thời gian thực thi của các thuật toán trên bộ dữ liệu T10I4D100K với độ hỗ trợ thay đổi từ 1% đến 5%*

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%)  {829} (s:6.81%)  {766} (s:6.26%)  {722} (s:5.85%)  {354} (s:5.84%)  {684} (s:5.41%)  {217} (s:5.38%)  {494} (s:5.1%)  {419} (s:5.06%) | {529} (s:7.06%)  {829} (s:6.81%)  {766} (s:6.26%)  {722} (s:5.85%)  {354} (s:5.84%)  {684} (s:5.41%)  {217} (s:5.38%)  {494} (s:5.1%)  {419} (s:5.06%) | {529} (s:7.06%)  {829} (s:6.81%)  {766} (s:6.26%)  {722} (s:5.85%)  {354} (s:5.84%)  {684} (s:5.41%)  {217} (s:5.38%)  {494} (s:5.1%)  {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 29. 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 mực 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**

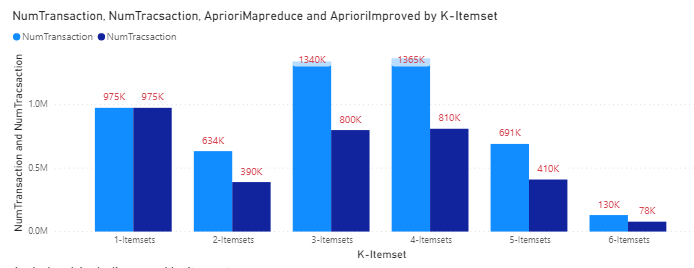
Như chúng ta đã đề cập ở phần tìm hiểu lý thuyết về thuật toán Apriori và Apriori cải tiến thì ở thuật toán Apriori khi đếm số lượng xuất hiện của các Itemset, thì chương trình phải duyệt lại toàn bộ tất cả các transaction có trong tập dữ liệu. Nhưng đối với thuật toán Apriori cải tiến, khi đếm số lượng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

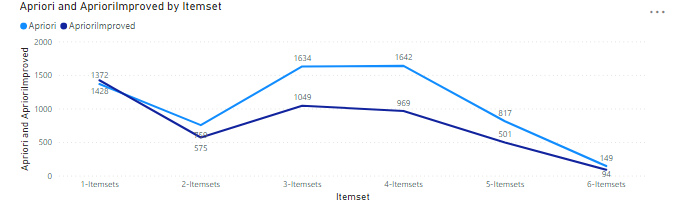
Dựa trên kết quả chúng ta dễ dàng thấy được số lần thực hiện duyệt transaction ở mỗi thuật toán là rất nhiều, với tập dữ liệu Mushroom chỉ với 8124 transaction nhưng số lượng duyệt transaction đối với mỗi K-Itemset lên tới con số triệu lần (nhiều nhất là **1364832** lần ở 4-Itemset). Ta dễ dàng quan sát thấy ở thuật toán Apriori số lượng duyệt là rất nhiều so với Apriori cải tiển 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 30. 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 Apirori 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 31. 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

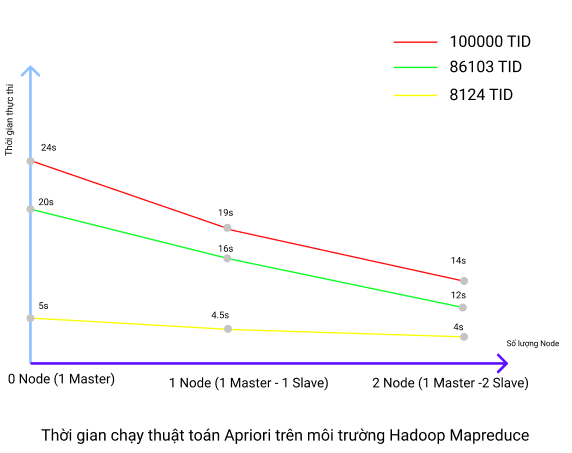
Quan sát biểu đồ ta thấy, khi cùng xử lý ở 1-Itemset thì thời gian thực thi của thuật toán Apriori sẽ nhỉnh hơn thời gian thực thi của thuật toán Apriori cải tiến, lý do xảy ra sự chênh lệch này trong khi cả 2 đều cùng nhau duyệt chung một số lượng transaction như nhau (974880) có lẽ do ở thuật toán Apriori cải tiến, chương trình phải xử lý phần 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 nhỉnh 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 hiệu suất 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 kì 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 |

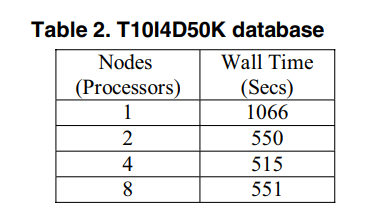
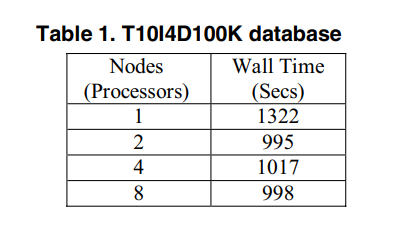
*Bảng 8.3. Kết quả về thời gian thực thi chương trình trên từng cấu hình hadoop khác nhau*



Hình 32. 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.



*Bảng 8.4. 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.

**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.

Giới thiệu khái quát về khai phá luật kết hợp: các khái niệm trong khai phá luật kết hợp. Môi trường Hadoop MapReduce và thuật toán Apriori trong môi trường này.

Thiết lập môi trường Hadoop Mapreduce, cài đặt thuật toán khai phá luật kết hợp và thuật toán khai phá luật kết hợp sử dụng giải thuật song song là Apriori để ứng dụng cho bài toán khai phá dữ liệu.

Đánh giá thực nghiệm của thuật toán Apriori qua các môi trường, các cách cải tiến thuật toán.

**2. Những hạn chế:** Luận văn vẫn còn có những hạn chế:

Tìm hiểu và cài đặt thuật toán khai phá luật kết hợp sử dụng giải thuật song song còn hạn chế.

Dữ liệu nhóm chọn thực hiện chưa đủ lớn.

1. **Hướng phát triển**

Triển khai trên máy thực với tập dữ liệu lớn, có thể ứng dụng được những tri thức sinh ra từ luật kết hợp.

Các giải thuật khai phá dữ liệu (ví dụ như các giải thuật sinh luật kết hợp sử dụng giải thuật song song) 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.

# **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 TPHCM, 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