CẢI TIẾN THUẬT TOÁN APRIORI TRÊN MÔ HÌNH MAP/REDUCE

Trần Thiên Thành[1], Phan Đình Sinh[2]

[1]Khoa Công nghệ thông tin, trường Đại học Quy Nhơn, thanhtranthien@gmail.com
[2]Khoa Công nghệ thông tin, trường Đại học Quy Nhơn, sinhphandinh@yahoo.com

Tóm tắt: Tìm kiếm các tập mục phổ biến là một bước rất quan trọng khai phá luật kết hợp, để từ đó phát sinh luật kết hợp. Một trong những thuật toán khai phá luật kết hợp kinh điển và nổi tiếng là thuật toán Apriori. Tuy nhiên, thuật toán Apriori có yếu điểm là với trường hợp cơ sở dữ liệu lớn thì thời gian tính toán cũng sẽ rất lớn. Vì vậy, trong bài báo này chúng tôi đề xuất một cải tiến của thuật toán Apriori trên mô hình lập trình và tính toán song song Map/Reduce. Cải tiến được thực hiện thông qua việc xây dựng thuật toán Apriori trên mô hình Map/Reduce, qua đó đưa ra giải pháp tìm tất cả tập mục phổ biến thông qua hai pha: pha thứ nhất tìm các tập mục phổ biến 1_itemsets; pha thứ hai tìm tất cả tập mục phổ biến k_itemsets (k>=2). Với phương pháp này việc tính toán đã cải thiện đáng kể thời gian tính toán khi ứng dụng thuật toán Apriori khai phá cơ sở dữ liệu lớn. Bước đầu thử nghiệm cho thấy thuật toán Apriori phát triển trên mô hình Map/Reduce cho kết quả nhanh hơn trên cơ sở dữ liệu lớn so với thuật toán Apriori gốc.

Từ khóa: Apriori, Map/Reduce

1 Mục mở đầu

Khai phá dữ liệu đã và đang được ứng dụng rộng rãi trong rất nhiều lĩnh vực. Đặc biệt, trong những năm gần đây với số lượng dữ liệu đã vượt quá khả năng xử lý của những hệ thống cơ sở dữ liệu truyền thống thì việc khai phá và xử lý dữ liệu lớn một cách hiệu quả là một thách thức thực sự.

Nhiều thuật toán khai phá tập mục phổ biến đã được công bố với kết quả rất thành công như thuật toán Apriori, FP-Growth,… Thuật toán Apriori là một thuật toán nền tảng cho nhiều thuật toán khai phá luật kết hợp sau này. Tuy nhiên, thuật toán này lại không thực sự hiệu quả với cơ sở dữ liệu lớn vì thời gian thực hiện lớn do phải duyệt nhiều lần cơ sở dữ liệu.

Mô hình lập trình Map/Reduce được đề xuất là mô hình lập trình và thực thi song song các xử lý trên các tập dữ liệu lớn. Map/Reduce trên nền tảng Hadoop có thể chạy trên các phần cứng thông thường và giúp làm đơn giản hóa các thuật toán tính toán phân tán. Do đó, việc cải tiến thuật toán Apriori khai phá tập mục phổ biến trên mô hình Map/Reduce là một giải pháp hợp lý.

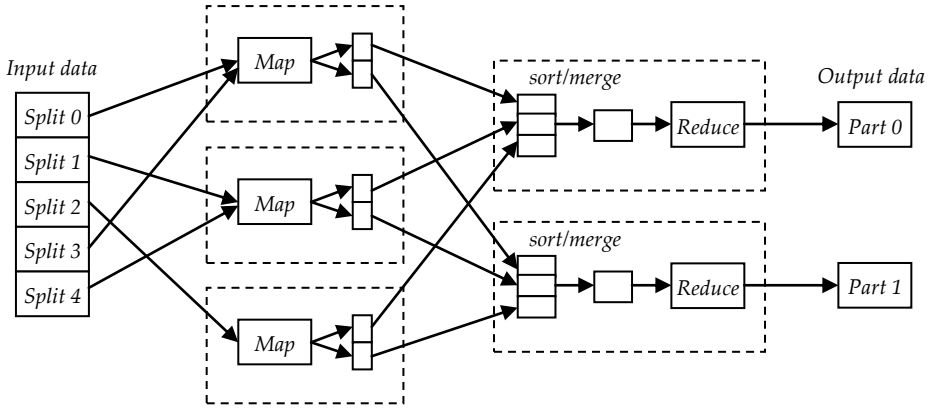
Trong bài báo này, chúng tôi đề xuất cải tiến thuật toán Apriori dựa trên mô hình Map/Reduce. Thuật toán này thực hiện tìm kiếm tất cả các tập mục phổ biến qua hai pha: pha thứ nhất tìm kiếm các tập mục phổ biến 1_itemsets và pha thứ hai tìm kiếm các tập mục phổ biến k_itemsets (k>=2). Thông qua mô hình Map/Reduce chia việc xử lý thành nhiều công việc

nhỏ, phân tán khắp các nút tính toán, rồi thu thập các kết quả từ các nút để tổng hợp thành kết quả toàn thể.

Bài báo gồm các phần như sau: Sau phần mở đầu là mục 2 dành cho việc mô tả mô hình và phương pháp lập trình dựa trên Map/Reduce. Mục 3 trình bày thuật toán Apriori gốc. Mục 4 chúng tôi đề xuất cải tiến thuật toán MRApriori trên mô hình Map/Reduce. Mục 5 là kết quả thử nghiệm thuật toán MRApriori và so sánh với thuật toán gốc Apriori. Cuối cùng, trong mục 6 là kết luận và một số ý kiến nhận xét về vấn đề đã được nghiên cứu.

2 Mô hình Map/Reduce

Mô hình lập trình Map/Reduce được Google đề xuất năm 2004 để xử lý những tập dữ liệu lớn [2]. Map/Reduce là mô hình được áp dụng trên một hệ thống các máy tính được kết nối với nhau và kèm theo là một hệ thống chia sẻ tập tin phân tán. Với mô hình này, sẽ chia nhỏ một công việc thành các công việc con giống nhau và dữ liệu đầu vào cũng được chia thành các mảnh dữ liệu nhỏ. Đặc biệt nhất, để thực hiện các thao tác xử lý một cách song song và đồng thời [4]. Map/Reduce sử dụng hai thao tác cho việc thực thi công việc ban đầu từ người dùng là hàm Map và hàm Reduce. Hàm Map tiếp nhận mảnh dữ liệu input và thực hiện xử lý nào đó để chuẩn bị dữ liệu làm đầu vào cho hàm Reduce. Hàm Reduce thực hiện xử lý riêng của nó và trả ra một phần nhỏ kết quả cuối cùng của công việc. Sau khi tất cả các hàm Reduce thực hiện ta sẽ có được toàn bộ kết quả công việc.



Hình 1. Mô hình Map/Reduce [6]

2.1 Hàm Map

Hàm Map nhận dữ liệu đầu vào là một cặp dữ liệu (key, value) và tùy vào mục đích của người dùng mà hàm Map sẽ trả ra danh sách các cặp dữ liệu (intermediate key, value).

2.2 Hàm Reduce

Hệ thống sẽ gom nhóm tất cả value theo intermediate key từ các output của hàm Map, tạo thành tập các cặp dữ liệu với cấu trúc là (key, tập các value cùng key). Dữ liệu input của

hàm Reduce là từng cặp dữ liệu được gom nhóm ở trên và sau khi thực hiện xử lý nó sẽ trả ra cặp dữ liệu (key, value) output cuối cùng cho người dùng.

3 Thuật toán Apriori

Apriori là thuật toán được đề xuất bởi R. Agrawal và R. Srikant vào năm 1994 để khai thác cơ sở dữ liệu để tìm các tập mục phổ biến [5]. Thuật toán Apriori dùng cách tiếp cận lặp, với các tập mục k_itemsets được dùng để thăm dò các tập (k+1)_itemsets. Đầu tiên, các tập mục phổ biến 1_itemsets được tìm thấy bằng cách quét cơ sở dữ liệu để đếm số lượng từng item, và thu thập những item thỏa mãn độ hỗ trợ cực tiểu, tập kết quả đặt là L_1. Tiếp theo, L_1 được dùng để tìm L_2, là các tập mục phổ biến 2_itemsets, nó lại được dùng để tìm L_3, và cứ thế tiếp tục cho tới khi tập mục phổ biến k_itemsets không thể tìm thấy. Việc tìm kiếm cho mỗi L_k đòi hỏi một lần quét toàn bộ cơ sở dữ liệu.

Thuật toán Apriori có độ phức tạp về thời gian là $O(k * (k^2 + t * n))$ với k là kích thước tập mục phổ biến, t là kích thước cơ sở dữ liệu và n là số tập mục của t. Vậy, độ phức tạp về thời gian sẽ là $O(k^3 + k* t * n)$. Hay là, $O(k* t * n)$ khi $t >> k$, $n >> k$.

Thuật toán Apriori [1]

Input: cơ sở dữ liệu các giao dịch DB và độ hỗ trợ cực tiểu ξ.

Output: Tất cả các tập mục phổ biến.

Method: Các bước của thuật toán Apriori như sau:

```
L1 = find_frequent_1itemsets(DB);
for (k = 2; Lk-1 ≠ ϕ; k++) {
    Ck = apriori_gen(Lk-1);
    for each transaction t ∈ DB {
        Ct = subnet(Ck, t);
        for each candidate c ∈ Ct
            c.count++;
    }
}
return L = ∪k Lk;
```

procedure apriori_gen(L_{k-1}: frequent (k-1)_itemsets)

```
for each itemset l1 ∈ Lk-1
    for each itemset l2 ∈ Lk-1
        if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧…∧ (l1[k-2] = l2[k-2]) ∧(l1[k-1] < l2[k-1])
        then {
```

$c = l_1 \oplus l_2$;

if has_infrequent_subset (c, L_{k-1}) then delete c;

else add c to C_k;

}

return C_k;

procedure has_infrequent_subset (c: candidate k_itemsets;L_{k-1}: frequent (k-1)_itemsets)

for each (k-1)_subset s of c

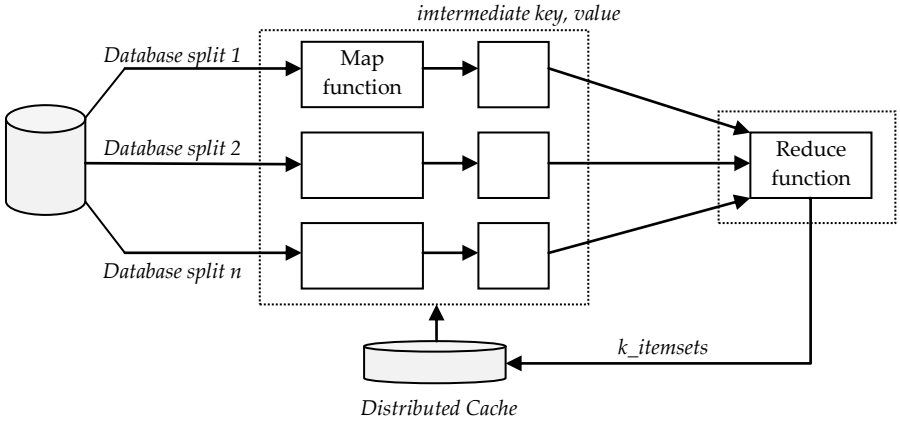
if s ∉ L_{k-1} then

return True;

return False;

4 Triển khai thuật toán Apriori trên Map/Reduce

Trong phần này chúng tôi trình bày thuật toán Apriori trên mô hình Map/Reduce (MRApriori), kế thừa từ ý tưởng của thuật toán Apriori. Thuật toán MRApriori cũng thực hiện các bước giống như thuật toán Apriori, thông qua hai pha thực hiện để tìm thấy tất cả tập mục phổ biến. Thuật toán duyệt cơ sở dữ liệu theo cách của mô hình Map/Reduce, dựa vào hai hàm được định nghĩa là Map và Reduce để tính độ hỗ trợ của các tập mục. Việc tính toán được thực hiện song song và phân tán cho nên việc tính toán nhanh hơn, đặc biệt là trên các cơ sở dữ liệu thực sự lớn. Mô hình luồng dữ liệu của thuật toán MRApriori được thiết kế như hình 2.



Hình 2. Mô hình luồng dữ liệu của thuật toán MRApriori

4.1 Thuật toán MRApriori

Input: cơ sở dữ liệu các giao dịch DB và độ hỗ trợ cực tiểu ξ.

Output: Tất cả các tập mục phổ biến.

Method: Các bước của thuật toán MRApriori như sau:

Pha 1: Tìm các tập mục phổ biến 1_itemsets

Ta định nghĩa hàm Map của mô hình Map/Reduce, sẽ nhận dữ liệu đầu vào là các giao dịch của DB. Tại mỗi nút, đầu ra của hàm Map là *output(a_i, part_count)* trả về các cặp (*imtermediate key, value*), trong đó *imtermediate key* ứng với các tập mục và *value* là số *imtermediate key* tương ứng với các giao dịch. Ở bước này, hàm Reduce nhận đầu vào là các cặp (*imtermediate key, value*), sẽ tập hợp và tính tổng các *value* theo *imtermediate key,* đây là độ hỗ trợ của các tập mục. Đầu ra của hàm Reduce là *output(a_i, C)* trả về là các tập mục phổ biến 1_itemsets, có độ hỗ trợ *C* lớn hơn hoặc bằng độ hỗ trợ cực tiểu.

procedure Mapper_MRApriori;

input: S_i

output: <key, value>

foreach transaction t in S_i

    foreach item a_i in t

        output(a_i, part_count);

    end foreach

end foreach

procedure Reducer_MRApriori;

input: <key2 = a_i, value2 = $S(a_i)$>

output: <key3, value3>

C = sum($S(a_i)$);

if (C >= ξ)

    output(a_i,C);

end if

Pha 2: Tìm các tập mục phổ biến k_itemsets (k>=2)

Đây là bước lặp, hàm Map sẽ nhận dữ liệu đầu vào là các giao dịch và các tập mục phổ biến L_{k-1} (k>=2). Ở bước này, tìm kiếm các tập mục ứng viên C_k bằng hàm *apriori_gen*, tức là sắp xếp và kết nối từng tập mục trong L_{k-1}, loại bỏ các tập mục trùng lặp. Với các tập mục k_itemsets (k>=3), thực hiện hàm *prune(C_k)* xén tỉa các tập mục không phổ biến. Duyệt cơ sở dữ liệu, tính độ hỗ trợ của các tập mục. Đầu ra hàm Map là *output(a_i, part_count)* trả về các cặp (*imtermediate key, value*) với *imtermediate key* là các tập mục và *value* là số *imtermediate key* trên các giao dịch. Hàm Reduce ở bước này thực hiện giống như bước trên.

procedure Mapper2_MRApriori;

input: S_i, L_{k-1} (k>=2)

output: <key, value>

C_k = apriori_gen(L_{k-1});

if (k>=3)

prune(C_k);

end if

foreach transaction t in S_i

C_t = subset(C_k,t);

foreach item a_i in C_t

output(a_i, part_count);

end foreach

end foreach

procedure Reducer2_MRApriori; (Xây dựng như hàm Reducer_MRApriori)

Với mỗi nút, độ phức tạp về thời gian của thuật toán để tính toán các tập mục là $O(t * n/p)$ trong đó, t là kích thước cơ sở dữ liệu và n là số tập mục của t và p số nút trên một hệ thống. Thực hiện kết nối và xén tỉa k tập mục phổ biến, nên có độ phức tạp về thời gian $O(k^2/p)$. Vậy, độ phức tạp về thời gian của thuật toán MRApriori là $O(k * (k^2 + t * n)/p)$, hay $O((k^3 + k * t * n)/p)$. Suy ra, $O((k* t * n)/p)$ khi $t >> k$, $n >> k$, chưa tính thời gian truyền dữ liệu giữa các máy.

4.2 Ví dụ minh họa

Cho một cơ sở dữ liệu DB gồm 6 giao dịch: T_1: {A, B}; T_2: {C, D}; T_3: {A, B, E}; T_4: {A, E}; T_5: {B, C, E}; T_6: {C, E} và độ hỗ trợ cực tiểu ξ = 2/6. Giả sử có 3 nút (m=3), với nút 1 xử lý S_1: giao dịch T_1 và T_2; nút 2: xử lý S_2: giao dịch T_3 và T_4; nút 3 xử lý S_3: giao dịch T_5 và T_6.

Pha 1: Tìm các tập mục phổ biến 1_itemsets

Với mỗi S_i, hàm Map sẽ trả về danh sách các cặp (*imtermediate key, value*) như sau:

C_{11}={<A, 1>, <B, 1>, <C, 1>, <D, 1>}; C_{21}={<A,2>, <B, 1>, <E, 2>}; C_{31}={<B, 1>, <C, 2>, <E, 2>}

Hàm Reduce với đầu vào là các cặp (*imtermediate key, value*), sẽ tập hợp và tính tổng các *value* theo *imtermediate key*. Ta có, C_1 = {<A, 3>, <B, 3>, <C, 3>, <D, 1>, <E, 4>}

L_1 chứa các tập mục phổ biến 1_itemsets thỏa mãn độ hỗ trợ cực tiểu ξ = 2/6.

L_1 = {<A, 3>, <B, 3>, <C, 3>, <E, 4>}

Pha 2: Tìm các tập mục phổ biến k_itemsets (k>=2). Đây là bước lặp:

Ở bước này, hàm apriori_gen(L_{k-1}) tạo ra các tập mục 2_itemsets C_2 bằng cách sắp xếp và kết nối từng tập mục trong L_{k-1} (k=2). Ta có: C_2 = ({A,B}, {A,C}, {A,E}, {B,C}, {B,E}, {C,E})

Thực hiện phân tán C_2 đến các nút: C_{12}=({A,B}, {A,C}); C_{22}=({A,E}, {B,C}); C_{32}=({B,E}, {C,E})

Duyệt từng giao dịch, tính toán độ hỗ trợ và tập hợp các tập mục 2_itemsets C_2:

C_{12} = (<{A,B},2>, <{A,C},0>); C_{22} = (<{A,E},2>, <{B,C},1>); C_{32} = (<{B,E},2>, <{C,E},2>)

Hàm Reducer, cũng sẽ tập hợp và tính tổng các *value* theo *imtermediate key*. Ta có,

C_2 = (<{A,B},2>, <{A,C},0>, <{A,E},2>, <{B,C},1>, <{B,E},2>, <{C,E},2>)

L_2 chứa các tập mục phổ biến 2_itemsets thỏa mãn độ hỗ trợ cực tiểu ξ = 2/6.

L_2 = (<{A,B},2>, <{A,E},2>, <{B,E},2>, <{C,E},2>)

Làm tương tự các bước trên, ta có các tập mục 3_itemsets C_3 bằng cách sắp xếp và kết nối từng tập mục trong L_{k-1} (k=3), loại bỏ các tập mục trùng lặp, ta có:

C_3 = ({A,B,E}, {A,B,C}, {A,C,E}, {B, C, E})

Với các tập mục có kích thước k>= 3, thì xén tỉa các tập mục không phổ biến bởi hàm *prune(C_k)*. Tức là, tất cả các tập con của một tập mục phổ biến cũng phải là phổ biến. Do đó, ta có C_3 chứa tập mục có độ hỗ trợ như sau: C_3 = <{A,B,E},1> và L_3={} vì tập mục {A,B,E} không thỏa độ hỗ trợ cực tiểu ξ.

Vậy ta có các tập mục phổ biến sau:

L_1 = {<A, 3>, <B, 3>, <C, 3>, <E, 4>}; L_2 = (<{A,B},2>, <{A,E},2>, <{B,E},2>, <{C,E},2>)

5 Thử nghiệm và đánh giá

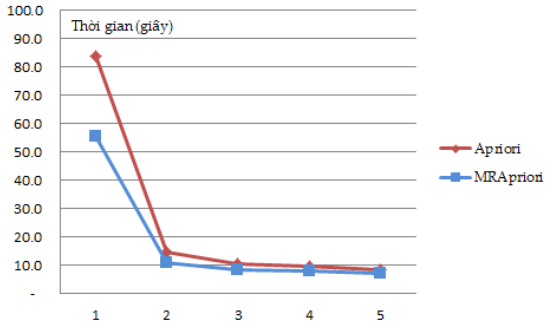
Chúng tôi chạy thuật toán Apriori theo hướng tiếp cận tính toán song song [3] không triển khai trên mô hình Map/Reduce và thực thi thuật toán MRApriori trên máy chủ IBM 2.9GHz cài đặt hệ điều hành Ubuntu 14 làm Master và 5 máy Dell core 2 2.4GHz cũng cài đặt Ubuntu 14 làm Slave, ngôn ngữ Java, Hadoop 2.2.0, chạy với các ngưỡng cực tiểu và thời gian tương ứng trong các bảng so sánh sau với bộ dữ liệu mẫu trên *http://fimi.ua.ac.be/data*. Hai tập dữ liệu được sử dụng để đánh giá kết quả thử nghiệm là tập dữ liệu T10I4D100K có 10000 giao dịch với 954 item và tập dữ liệu T40I10D100K có 100000 giao dịch với 942 item.

Bảng 1. Bảng so sánh của hai thuật toán trên tập dữ liệu T40I10D100K

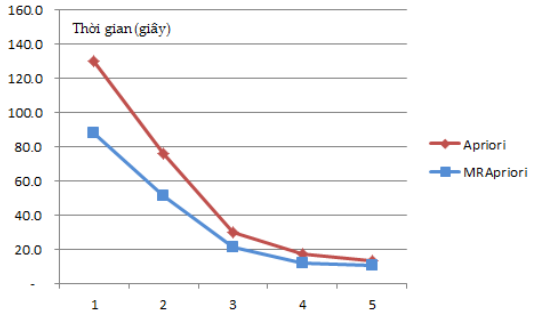
ξ	**Apriori (giây)**	**MRApriori (giây)**
0.01	83.8	55.4
0.02	14.6	11.0
0.03	10.4	8.2
0.04	9.5	7.9
0.05	8.4	7.3

Bảng 2. Bảng so sánh của hai thuật toán trên tập dữ liệu T10I4D100K

ξ	**Apriori (giây)**	**MRApriori (giây)**
0.001	130.0	87.9
0.002	76.2	51.4
0.003	30.3	21.7
0.004	17.3	12.3
0.005	13.4	10.5



Hình 3. So sánh thời gian thực hiện hai thuật toán trên tập dữ liệu T40I10D100K



Hình 4. So sánh thời gian thực hiện hai thuật toán trên tập dữ liệu T10I4D100K

6 Kết luận

Trong bài báo này, chúng tôi đã đề xuất cải tiến thuật toán Apriori trên mô hình Map/Reduce, cải tiến này là một giải pháp khi ứng dụng thuật toán Apriori trên cơ sở dữ liệu có số lượng giao dịch lớn. Thực hiện đánh giá độ phức tạp về thời gian của thuật toán MRApriori là tốt hơn thuật toán Apriori. Kết quả thực nghiệm cho thấy tính khả thi của việc cải tiến thuật toán MRApriori trên mô hình Map/Reduce so với thuật toán Apriori theo hướng tiếp cận song song không triển khai trên môi trường Map/Reduce. Trong thời gian tới, chúng tôi sẽ tiếp tục tiến hành thử nghiệm thuật toán MRApriori với các thuật toán song song khác trong khai phá tập mục phổ biến để có những đánh giá khách quan.

Tài liệu tham khảo

1. Han J. & Kamber M, *Data mining Concepts and Techniques*, San Francisco, CA, Elsevier Inc, 2006.
2. Hadoop Map/Reduce Tutorial, Apache™. Retrieved November 12, 2012 from *http://hadoop.apache.org/docs/r0.20.2/mapred_tutorial.html*, 2010.
3. Nilesh.S.Korde & Shailendra.W.Shende, *Parallel Implementation of Apriori Algorithm*, IOSR Journal of Computer Science (IOSR-JCE), e-ISSN: 2278-0661, p-ISSN: 2278-8727, PP 01-04, 2014.
4. Jeffrey D. & Sanjay G, *Mapreduce: Simplified data processing on large cluster*, In OSDI, pages 137 – 150, 2004.
5. Agrawal R. and Srikant R, *Fast algorithm for mining associantion rules in large databases*, In Proc, VLDB, pages 487 – 499, 1994.
6. Middleton A.M, *Hanbook of Cloud Computing*, Data-Intensive Technologies for Cloud Computing, FL, USA, Springer: 83-136, 2010.