Apriori Algorithm

1. **Tổng quan thuật toán Apriori**

Khác với giải thuật FP-Growth, khám phá các mẫu thường xuyên bằng việc tạo ra cây FP-tree. Một thuật toán khác cho việc tạo ra các mẫu thường xuyên từ tập dữ liệu là Apriori, bởi Agrawal và Srikant 1994. Tư tưởng chính của thuật toán Apriori là tìm tất cả các tập dự tuyển và từ đó sinh ra các mẫu thường xuyên (frequent itemsets):

Lk: mẫu thường xuyên gồm K items.

Ck: tập dự tuyển gồm K items.

1. **Đặc điểm giải thuật**

Tạo ra nhiều tập dự tuyển. Một k-itemset cần ít nhất 2k-1 itemsets dự tuyển trước đó.

Kiểm tra tập dự tuyển nhiều lần. Chi phí càng lớn khi kích thước các itemsets tang dần.

1. **Giải thuật Apriori**

Step 1: Duyệt toàn bộ transaction database để tìm ra toàn bộ các item và tần suất F xuất hiện của các item. So sánh F với min\_support và loại bỏ các item không thỏa ngưỡng để được mẫu thường xuyên L1.

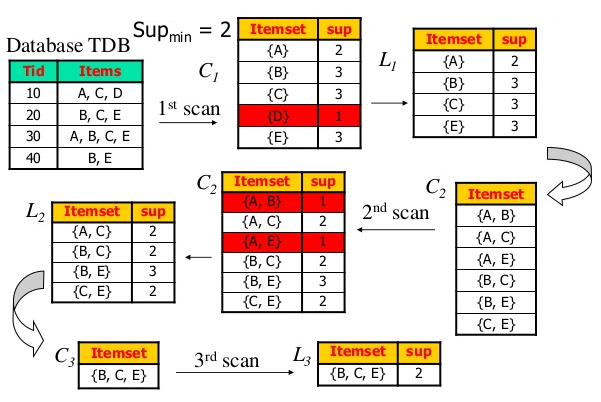
Step 2: Sử dụng Lk-1 để sinh ra các tập dự tuyển Ck. Sau đó xét các tập dự tuyển Ck và dùng tính chất nếu một tập mục là thường xuyên, thì tất cả các tập con (subsets) của nó đều là các tập mục thường xuyên để loại bớt các tập mục không thỏa trong Ck dựa vào Lk-1.

Step 3: Duyệt toàn bộ transaction database để tìm được tần suất F của mỗi tập dự tuyển trong Ck. So sánh F với min\_support để thu được tập k-itemsets (Lk).

Step 4: Lặp lại từ step 2 cho đến khi tập dự tuyển Ck rỗng.

1. **Ví dụ thuật toán**

Thuật toán Apriori được mô tả qua các bước sau:



Đặt vấn đề

1. Giải bài toán Association Rules trên node host 2 CPUs x 12 cores, 2 threads per cores (48 threads).
2. Giải bài toán Association Rules trên Xeon Phi 61 cores Infiniband 56Gbps.

Giải bài toán.

Input: tập các giao dịch với tên các item là các số. Mỗi dòng là một giao dịch.

Output: tập các mẫu thường xuyên với tần số xuất hiện trong toàn bộ giao dịch.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Platform | Input | | Ouput | Min\_support | Number of threads | | | | |
| 12 | 24 | 48 | 61 | 122 |
| Host | Big\_data\_1.txt |  |  |  |  |  |  |  |  |
| Big\_data\_2.txt |  |  |  |  |  |  |  |  |
| Big\_data\_3.txt |  |  |  |  |  |  |  |  |
| Xeon Phi | Big\_data\_1.txt |  |  |  |  |  |  |  |  |
| Big\_data\_2.txt |  |  |  |  |  |  |  |  |
| Big\_data\_3.txt |  |  |  |  |  |  |  |  |

Nhận xét: Với số liệu trên, ta thấy:

\_ Nhìn chung khi số thread càng tăng thì thời gian xử lí càng ngắn. Tuy nhiên nếu kích thước input quá nhỏ và số thread lớn thì thời gian cũng không giảm nhiều.(có khi tăng so với ít thread hơn)

\_ Khi đưa lên Xeon phi, nếu thời gian tính toán trên Xeon phi quá nhỏ so với thời gian truyền nhận dữ liệu thì hiệu suất đạt được cũng không cao(thời gian xử lí chung không giảm nhiều).