ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

- -

Tính toán song song

Nhóm: 09 — Báo cáo bài tập lớn

Viết chương trình Association Rules trên GPUs

Giảng viên hướng dẫn: TS. Nguyễn Quang Hùng
ThS. Nguyễn Mạnh Thìn

Sinh viên:	Vũ Hoàng Văn	1614063
	Đường Quang Huy	1611244
	Lê Trọng Phú	1612604
	Nguyễn Xuân Hiến	1652192

Thành phố Hồ Chí Minh, 5/2019

Mục lục

1 Bài toán

1.1 Lý thuyết

- Tìm hiểu về lập trình GPU (CUDA).
- Tìm hiểu về giải thuật cho Association Rules.

1.2 Hiện thực

- Viết chương trình
- Đánh giá hiệu năng (speedup) với số lượng core khác nhau.

2 Lý thuyết

2.1 Tìm hiểu về giải thuật cho Association Rules

2.1.1 Luật kết hợp trong khai phá dữ liệu (Association Rule in Data Mining)

Trong lĩnh vực Data Mining, mục đích của luật kết hợp (Association Rule - AR) là tìm ra các mối quan hệ giữa các đối tượng trong khối lượng lớn dữ liệu. Nội dung cơ bản của luật kết hợp được tóm tắt như dưới đây.
Cho cơ sở dữ liệu gồm các giao dịch T là tập các giao dịch t_1, t_2, ..., t_n.

$$T = \{t_1, t_2, \ldots, t_n\}$$

T gọi là cơ sở dữ liệu giao dịch (Transaction Database).
Mỗi giao dịch ti bao gồm tập các đối tượng I (gọi là itemset).

$$I = \{i_1, i_2, \ldots, i_n\}$$

Một itemset gồm k items gọi là k-itemset.
Mục đích của luật kết hợp là tìm ra sự kết hợp (association) hay tương quan (correlation) giữa các items. Những luật kết hợp này có dạng $X \Rightarrow Y$.
Trong Basket Analysis, luật kết hợp $X \Rightarrow Y$ có thể hiểu rằng những người mua các mặt hàng trong tập X cũng thường mua các mặt hàng trong tập Y. (X và Y gọi là itemset).
Ví dụ, nếu X = Apple, Banana và Y = Cherry, Durian và ta có luật kết hợp $X \Rightarrow Y$ thì chúng ta có thể nói rằng những người mua Apple và Banana thì cũng thường mua Cherry và Durian.
Theo quan điểm thống kê, X được xem là biến độc lập (Independent variable) còn Y được xem là biến phụ thuộc (Dependent variable).

2.1.2 Độ hỗ trợ (Support) và độ tin cậy (Confidence)

Độ hỗ trợ (Support) và độ tin cậy (Confidence) là 2 tham số dùng để đo lường luật kết hợp.
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. Ví dụ, support của luật $X \Rightarrow Y$ là 5% có nghĩa là 5% các giao dịch X và Y được mua cùng nhau. Công thức để tính support của luật $X \Rightarrow Y$ như sau:

$$support(X \to Y) = P(X \cup Y) = \frac{n(X \cup Y)}{N}$$

Trong đó:

- N là tổng số giao dịch.

Độ tin cậy (Confidence) của luật kết hợp $X \Rightarrow Y$ là xác suất xảy ra Y khi đã biết X. Ví dụ độ tin cậy của luật kết hợp {Apple} $\Rightarrow$ {Banana} là 80% có nghĩa là 80% khách hàng mua Apple cũng mua Banana.
Công thức để tính độ tin cậy của luật kết hợp X => Y là xác suất có điều kiện Y khi đã biết X như sau :

$$confidence(X \rightarrow Y) = P(X|Y) = \frac{n(X \cup Y)}{n(X)}$$

Trong đó:

- n(X) là số giao dịch chứa X.

Để thu được các luật kết hợp, ta thường áp dụng 2 tiêu chí: minimum support (min_sup) và minimum confidence (min_conf).
Các luật thỏa mãn có support và confidence thỏa mãn (lớn hơn hoặc bằng) cả Minimum support và Minimum confidence gọi là các luật mạnh (Strong Rle).
Minimum support và Minimum confidence gọi là các giá trị ngưỡng (threshold) và phải xác định trước khi sinh các luật kết hợp.
Một itemsets mà tần suất xuất hiện của nó $\geq$ min_sup goi là frequent itemsets.

2.1.3 Một số loại luật kết hợp

- Binary association rules (luật kết hợp nhị phân): Apple $\Rightarrow$ Banana
- Quantitative association rules (luật kết hợp định lượng): weight in [70kg – 90kg] $\Rightarrow$ height in [170cm – 190cm].
- Fuzzy association rules (Luật kết hợp mờ): weight in HEAVY $\Rightarrow$ height in TALL

Thuật toán phổ biến nhất tìm các luật kết hợp là Apriori sử dụng Binary association rules.

2.1.4 Thuật toán sinh các luật kết hợp Apriori

Tư tưởng chính của thuật toán Apriori là:

- Tìm tất cả frequent itemsets:
  - k-itemset (itemsets gồm k items) được dùng để tìm (k+1)- itemset.
  - Đầu tiên tìm 1-itemset (ký hiệu L_1). L_1 được dùng để tìm L_2 (2-itemsets). L_2 được dùng để tìm L_3 (3-itemset) và tiếp tục cho đến khi không có k-itemset được tìm thấy.
- Từ frequent itemsets sinh ra các luật kết hợp mạnh (các luật kết hợp thỏa mãn 2 tham số min_sup và min_conf).

Apriori Algorithm:

- Duyệt (Scan) toàn bộ transaction database để có được support S của 1-itemset, so sánh S với min_sup, để có được 1-itemset (L_1)
- Sử dụng L_{k-1} nối (join) L_{k-1} để sinh ra candidate k-itemset. Loại bỏ các itemsets không phải là frequent itemsets thu được k-itemset
- Scan transaction database để có được support của mỗi candidate k-itemset, so sánh S với min_sup để thu được frequent k –itemset (L_k)
- Lặp lại từ bước 2 cho đến khi Candidate set (C) trống (không tìm thấy frequent itemsets)
- Với mỗi frequent itemset I, sinh tất cả các tập con s không rỗng của I
- Với mỗi tập con s không rỗng của I, sinh ra các luật s $\Rightarrow$ (I-s) nếu độ tin cậy (Confidence) của nó $\geq$ min_conf

2.2 Tìm hiểu về lập trình GPU (CUDA)

Cấu trúc phần cứng Nvidia GPU:

- Gồm nhiều processor được tổ chức thành các đơn vị multiprocessor.
- Sử dụng bộ nhớ chung global memory cho tất cả multiprocessors.

Ưu điểm của lập trình với NVIDIA CUDA:

- Phù hợp với bài toán có khả năng song song dữ liệu cao.
- Sử dụng mô hình lập trình và quản lý threads.
- Hỗ trợ tốt tương tác giữa các process.

Công cụ hỗ trợ lập trình:

- Sử dụng ngôn ngữ C/C++
- CUDA Toolkit và CUDA Library được cung cấp bởi NVIDIA với những thư viện và hàm đã được hiện thực để hỗ trợ cho việc giao tiếp với GPU.
- Hỗ trợ trên nhiều nền tảng OS, vd: window, linux,...
- Người lập trình không cần biết nhiều về cấu trúc phần cứng.

Các threads được chia thành các block và grid cho phù hợp với kiến trúc phần cứng.
Dữ liệu cần xử lý được đưa vào bộ nhớ của GPU.
Chương trình chính chạy trên CPU gọi hàm thực thi trên GPU.
Các threads cùng xử lý công việc trên dữ liệu chung.
Kết quả được trả về lại bộ nhớ cho chương trình chính.

3 Hiện thực

3.1 GPU capabilities

```
    Mon May 20 14:00:09 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.56       Driver Version: 418.56       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce 940M        Off  | 00000000:01:00.0 Off |                  N/A |
| N/A   45C    P0    N/A /  N/A |    504MiB /  2004MiB |      6%      Default |
+-------------------------------+----------------------+----------------------+
```

3.2 Hiện thực chương trình

Mẫu thiết kế cho Dataset

Dataset

```
int** data;
int *recordCount;
int *attrCount;
map<string, int> *attributesIndex;
vector<string> *attributesList;
```

```
// @maxRecords: Number of record to be initialized on GPU ram
DatasetCPU(int maxRecords);

// Insert new record
//@recordSet: set<string> of records, each string represents an attribute
//Return: true if record is successfully inserted, otherwise false
bool newRecord(set<string> &recordSet);

// Calculate the support rate of a record over the DatasetCPU
//@recordSet: set<string> of record, each string represents an attribute
double supportRate(set<string> &recordSet);

// Calculate the support rate of a record over the DatasetCPU
//@recordBit: binary bit of record, each string represents an attribute
double supportRate(int* recordSet);

// Calculate the confidendce rate of a rule over the DatasetCPU
//@lhsSet: set<string> of lhs attributes, each string represents an attribute
//@rhsSet: set<string> of rhs attributes, each string represents an attribute
double confidenceRate(set<string> &lhsSet, set<string> &rhsSet);

// Calculate the confidendce rate of a rule over the DatasetCPU
//@lhsSet: binary bit array of lhs attributes, each string represents an attribute
//@rhsSet: binary bit array rhs attributes, each string represents an attribute
double confidenceRate(int*lhsSet, int*rhsSet);

// Get a binary array represents for the index
//@recordIndex: index of the record in DatasetCPU
// THIS FUNCTION SHOULDN'T BE USED, JUST FOR TESTING
int* getRecord(int recordIndex);

// New attribute
//@attrName: a string represents for the attribute
//Return: index of the new attribute to be stored
int newAttribute(string attrName);

vector<string> *getAttributesSet() { return attributesList; }

// This function is used to convert a record represented as set to a binary array
//@recordSet: set of attribute references to the record
//Return: Pointer to the binary array
// HEAP VALUE SHOULD BE CLEAN AFTER USING
int* DatasetCPU::recordSetToBit(set<string> &recordSet);

set<string>* bitToRecordSet(int arr[]);

int getRecordCount() {
return *recordCount;
}

// Read data from a csv file, and return an Dataset object
static DatasetCPU* readCSV(string filename);
```

3.3 Đánh giá hiệu năng

Kích thước tập dữ liệu	Thời gian thực hiện	
	CPU	GPU
500	0,38s	1,43s
1000	1,69s	1,48s
5000	6,23s	1,66s
10000	15,76s	2,58s
20000	23,56s	3,23s
50000	97,38s	8,18s
100000	212,40s	16,53s

Link source: https://github.com/blackslender/PC182

Tài liệu

[1] http://archive.ics.uci.edu/ml/machine-learning-databases/plants/plants.data

[2] http://archive.ics.uci.edu/ml/machine-learning-databases/plants/stateabbr.txt

[3] Association rules mining example using Weka:
https://ongxuanhong.wordpress.com/2015/08/24/apriori-va-fp-growth-voi-tap-du-lieu-plants/

[4] Apriori theory:
http://ait.edu.vn/Hoc_thuat/Apriori.html

[5] Thuật toán Apriori khai phá luật kết hợp:
http://bis.net.vn/forums