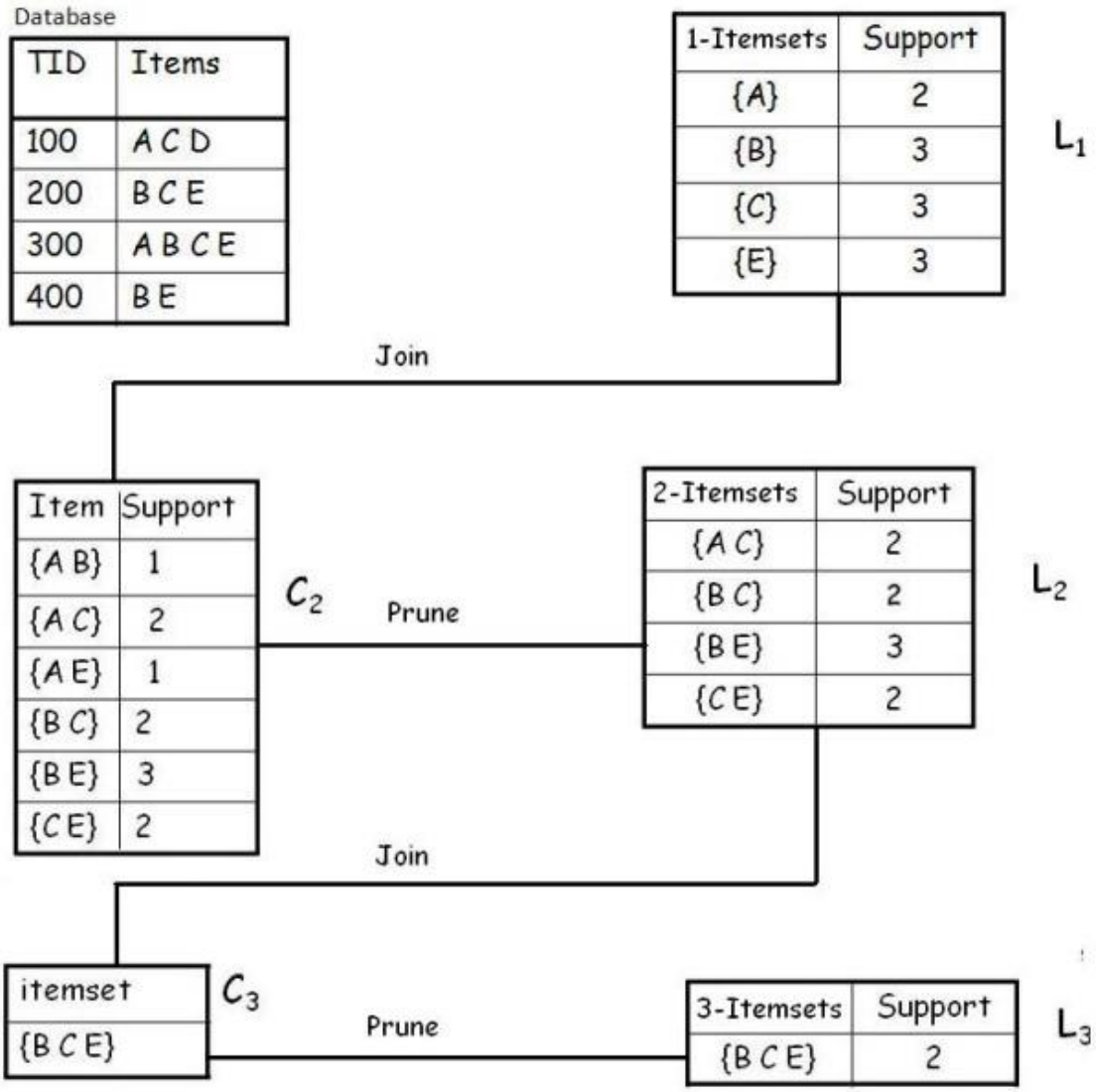
Report – Implementation on Apriori with Hadoop

Diagram:

(min support = 2)



Concept:

由前一階段產生的 candidate (C)再重新 scan 整個 Database，並刪除那些小於 threshold support 產生此一階段的 frequent itemset (L)。接著用(L)產生下一階段的(C)，由(L)產生(C)時需要”剪枝”。所謂的剪枝就是當前長度為 N 的(L)要產生長度是 N+1 的(C)時，所有(C)長度為 N 的 subset 都需要存在於(L)，也就是必須要是 N-frequent itemset。

由上例，注意當 L2 要產生 C3 時，雖然有多種可能性例如{A B C}，但是由於 L2 並沒有{A B}，所以把{A B C}從 C3 刪去。

Design Detail:

- Input:
  - Items: 所有販賣的物品清單
  - Transactions: 交易紀錄，是 items 的集合
- Output:
  - 各階段的 N-frequent itemset
- Parameter:
  - Min Support: 最小支持度
  - Iteration Number: 迴圈次數

[Algorithm]

1. Read Items.txt to get initial items
2. Counting Support to generateL1 and set i = 1
3. Li X Li (cartesian product) to generate Ci+1
4. Prune Ci+1 by checking subset with Li
5. Counting Support to generateLi+1
6. Go to 3 until condition satisfied

[Map Reduce]

Mapper: (產生 Ci)

[setup]:

從上一次結果產生這次要用的 frequent itemset。

1. Read 上一次結果
2. 執行 carsetian_and_prune()
3. 更新 frequent itemset 變數

[map]:

如果此 transaction 有某項 frequent itemset，輸出:

<key, value> = <frequent itemset, 1>

Reducer: (產生 Li)

算出有幾個同 key(frequent itemset)的 piar 數，如果>= min support 則輸出此 key。

<key, value> = <NULL, frequent itemset >

最後如果輸出的 Li 為空則停止，但我在這邊是把 iteration 數寫死。

Command Line:

arg[0]: transactions arg[1]: output dir arg[2]: items

arg[3]: min support　　arg[4]: Iteration number

Experiments:

A. Small Dataset

[case 1]

Source: https://wizardforcel.gitbooks.io/dm-algo-top10/content/apriori.html

transactions1.txt

```
A,C,D
B,C,E
A,B,C,E
B,E
```

items1.txt:

```
A
B
C
D
E
```

min support = 2

L1:

```
[root@sandbox target]# hadoop fs -cat /user/root/output1/out_0/*
A,
B,
C,
E,
```

L2:

```
[root@sandbox target]# hadoop fs -cat /user/root/output1/out_1/*
A,C,
B,C,
B,E,
C,E,
```

L3:

```
[root@sandbox target]# hadoop fs -cat /user/root/output1/out_2/*
B,C,E,
```

這其實就是 Report 地一張圖的結果。

[case 2]

Source: http://www.code2learn.com/2015/02/frequent-itemsets-apriori-algorithm-and.html

transactions2.txt

```
1,2,3,4,5,6
7,2,3,4,5,6
1,8,4,5
1,9,0,4,6
0,2,4,5
```

items2.txt:

```
0
1
2
3
4
5
6
7
8
9
```

min support = 3

L1:

```
[root@sandbox target]# hadoop fs -cat /user/root/output2/out_0/*
1,
2,
4,
5,
6,
```

L2:

```
[root@sandbox target]# hadoop fs -cat /user/root/output2/out_1/*
1,4,
2,4,
2,5,
4,5,
4,6,
```

L3:

```
[root@sandbox target]# hadoop fs -cat /user/root/output2/out_2/*
2,4,5,
```

And the Ground-truth:

- **Frequent Itemsets of Size 1:** 1, 2, 4, 5, 6
- **Frequent Itemsets of Size 2:** 14, 24, 25, 45, 46
- **Frequent Itemsets of Size 3:** 245

B. Big Dataset

Source: https://wiki.csc.calpoly.edu/datasets/wiki/apriori

這組大筆測資總共有四種 size: 1000, 5000, 20000, 75000

然後我會先用 parse.py 把資料前處理過。

這邊先敘述一下我設計的吃資料格式

[transactions.txt]:

A,B,C,D

Milk,Beer,Diaper

…

輸入的可以是數字或是字串，唯一要注意是要用逗號分開且不能有重複，即 A,A,A,B,C 是不被允許的。

[items.txt]:

A

B

C

…

輸入的可以是數字或是字串，所以東西都要唯一且換行。

經過以上前處理步驟，就可以來看看這組測資的特性。雖然 transaction 數目可以多 75000 筆，但是 item 數卻僅僅有 50 項，可以看到是不成比例的。然後根據經驗，最多候選項目的時候是 L1 X L1 時，但這樣也僅僅產生 50 的次方項目，所以記憶體是絕對足夠的。然後他的一筆典型交易像是這樣:

7,15,44,49

由這些我們可以看看設計不同參數來看看結果。

[case 1] 1000 transactions

min support = 20

L1, L2: (省略)

L3:

```
[root@sandbox target]# hadoop fs -cat /user/root/output1000/out_2/*
0,2,46,
11,37,45,
11,37,7,
11,45,7,
12,31,36,
12,31,48,
12,36,48,
15,49,7,
16,32,45,
18,3,35,
23,24,40,
23,24,41,
23,40,41,
24,40,41,
31,36,48,
37,45,7,
```

L4:

```
[root@sandbox target]# hadoop fs -cat /user/root/output1000/out_3/*
11,37,45,7,
12,31,36,48,
23,24,40,41,
```

L5: (Empty)

[case 2] 5000 transactions

min support = 20

L1, L2: (省略)

L3:

```
[root@sandbox target]# hadoop fs -cat /user/root/output5000/out_2/*
0,2,46,
11,37,45,
11,37,7,
11,45,7,
12,31,36,
12,31,48,
12,36,48,
15,49,7,
16,32,45,
17,29,47,
18,3,35,
23,24,40,
23,24,41,
23,24,43,
23,40,41,
23,40,43,
23,41,43,
24,40,41,
24,40,43,
24,41,43,
31,36,48,
37,45,7,
40,41,43,
```

L4:

```
[root@sandbox target]# hadoop fs -cat /user/root/output5000/out_3/*
11,37,45,7,
12,31,36,48,
23,24,40,41,
23,24,40,43,
23,24,41,43,
23,40,41,43,
24,40,41,43,
```

L5:

```
[root@sandbox target]# hadoop fs -cat /user/root/output5000/out_4/*
23,24,40,41,43,
```

L6: (Empty)

[case 3] 20000 transactions

min support = 20

L1, L2, L3: (省略)

L4:

```
[root@sandbox target]# hadoop fs -cat /user/root/output20000/out_3/*
11,37,45,7,
12,31,36,48,
16,2,32,45,
18,3,33,35,
23,24,40,41,
23,24,40,43,
23,24,41,43,
23,40,41,43,
24,40,41,43,
```

L5:

```
[root@sandbox target]# hadoop fs -cat /user/root/output20000/out_4/*
23,24,40,41,43,
```

[Discussion]

由於這個 dataset 不同大小間只是子集，因此我們可以發現如果 min support 其實找到的 pattern 差不多的，且越大筆在 L2 和 L3 會運算越來越久。

但如果筆數到達 75K 其實如果 min support 還是設太低是沒有意義的，因此最後我們在 75K 再做一次並用更大的 min support 看看找到的東西是甚麼。

[case 4] 75000 transactions

min support = 1500

L1, L2: (省略)

L3:

```
[root@sandbox target]# hadoop fs -cat /user/root/output75000/out_2/*
0,2,46,
11,37,45,
11,37,7,
11,45,7,
12,31,36,
12,31,48,
12,36,48,
15,49,7,
16,32,45,
17,29,47,
18,3,35,
23,24,40,
23,24,41,
23,24,43,
23,40,41,
23,40,43,
23,41,43,
24,40,41,
24,40,43,
24,41,43,
31,36,48,
37,45,7,
40,41,43,
```

L4:

```
[root@sandbox target]# hadoop fs -cat /user/root/output75000/out_3/*
11,37,45,7,
12,31,36,48,
23,24,40,41,
23,24,40,43,
23,24,41,43,
23,40,41,43,
24,40,41,43,
```

L5:

```
[root@sandbox target]# hadoop fs -cat /user/root/output75000/out_4/*
23,24,40,41,43,
```

L6: (Empty)

[case 5] 75000 transactions

min support = 2500

L3:

```
[root@sandbox target]# hadoop fs -cat /user/root/output75000_2/out_2/*
0,2,46,
11,37,7,
18,3,35,
```

[Discussion]

產生這些 frequent itemset 後，我們可以藉由他提供的 SQL 檔來知道這些數字背後的真實意義是甚麼。

首先他會建立 good table，而這張 table 等同於我們的 items.txt 內容，其欄位是:

```
create table goods (
    Id int,
    Flavor varchar2(15),
    Food varchar2(15),
    Price float,
    Type varchar2(5),
    constraint gpid PRIMARY KEY (Id)
);
```

(EB-setup.sql)

然後在另外的一份 SQL 檔，我們可以知道有哪些項目被建立:

```
insert into goods values (0,'Chocolate','Cake',8.95,'Food');
insert into goods values (1,'Lemon','Cake',8.95,'Food');
insert into goods values (2,'Casino','Cake',15.95,'Food');
insert into goods values (3,'Opera','Cake',15.95,'Food');
insert into goods values (4,'Strawberry','Cake',11.95,'Food');
insert into goods values (5,'Truffle','Cake',15.95,'Food');
insert into goods values (6,'Chocolate','Eclair',3.25,'Food');
insert into goods values (7,'Coffee','Eclair',3.5,'Food');
insert into goods values (8,'Vanilla','Eclair',3.25,'Food');
insert into goods values (9,'Napoleon','Cake',13.49,'Food');
```

總共有 50 項。

75000 transactions with min support = 1500 的結果: [23, 24, 40, 41, 43]分別是以下項目:

ID	Flavour	Food	Price	Type
23	Raspberry	Cookie	1.09	Food
24	Lemon	Cookie	0.79	Food
41	Raspberry	Lemonade	3.25	Drink
40	Lemon	Lemonade	3.25	Drink

43	Green	Tea	1.85	Drink

所以可以知道熱門組合會是這組，進而可以決定銷售方針。