Apriori & FP Growth Report

I. Environment

OS: Ubuntu 16.04 KVM

Python version: 2.7.12

IBM data generator: Clone and modify from halfvim/quest

II. Experiment

➢ Data from IBM data generator

1. Generate testing data

   # Compile IBM data generator, which is modified to output one transaction per line

   $ cd ibm_datagen && make

   # Generate data with number of transactions from 10000, 20000 … to 100000.

   # Outputs are named as test_<# of transactions>.data located at data folder

   $ ./gen.sh

To generate adequate number of frequent items, some parameters of generator are fixed (tlen, nitems, npats = 20, 100, 100). If tlen too large, then the program will run for a long time. Besides, if the minimum support is constant, then the number of frequent items increases as nlen increase. The reason is that if nlen increase, then one item is more likely to appear in a transaction, so that the support of the item will increase and tend to surpass the minimum support.

2. Test Apriori & FP Grwoth Algorithm

   # Run two algorithm with all data generated with IBM data generator

   $ ./experiment.sh

For convenience, I wrote a script to run two algorithms with data generated with IBM data generator and output the information such as peak memory usage and execution time of the programs. The frequent items found by two algorithms were compared to verify the answers are consistent.

- **Data from Kaggle**

  # Run two algorithm with data downloaded from Kaggle. The data is located at data/kaggle

  ./kaggle_test.sh

The data downloaded from Kaggle contain about 10000 transactions from a bakery. Original data were converted to the custom csv format (kaggle_breadbasket.csv), which represent one transaction per line and separate items with comma.

III. Results

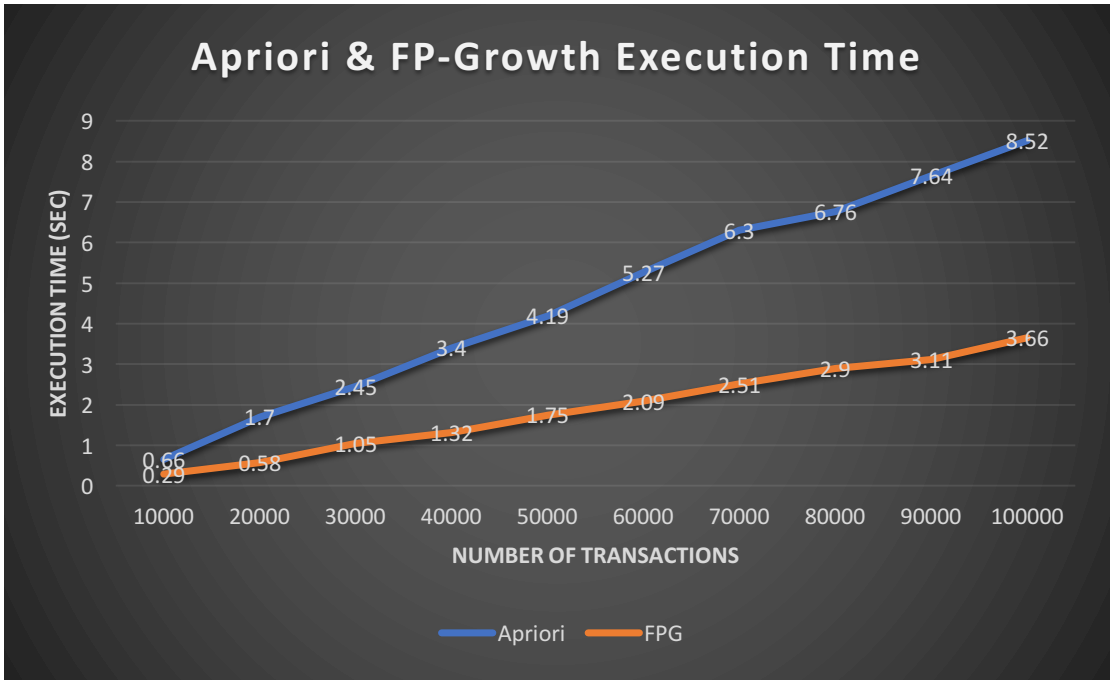

Figure 1. Relationship between number of transactions and execution time

As shown in Figure 1, the execution time is proportional to the number of transactions. Slopes of Apriori and FP-Growth are 0.786/90000 and 0.337/90000 respectively, which means the speed of FP-Growth is superior to that of Apriori. It is reasonable, since FP-Growth takes advantage of memory to avoid scanning database repeatedly. Next, we are curious how much additional memory FP-Growth consume to improve the performance.

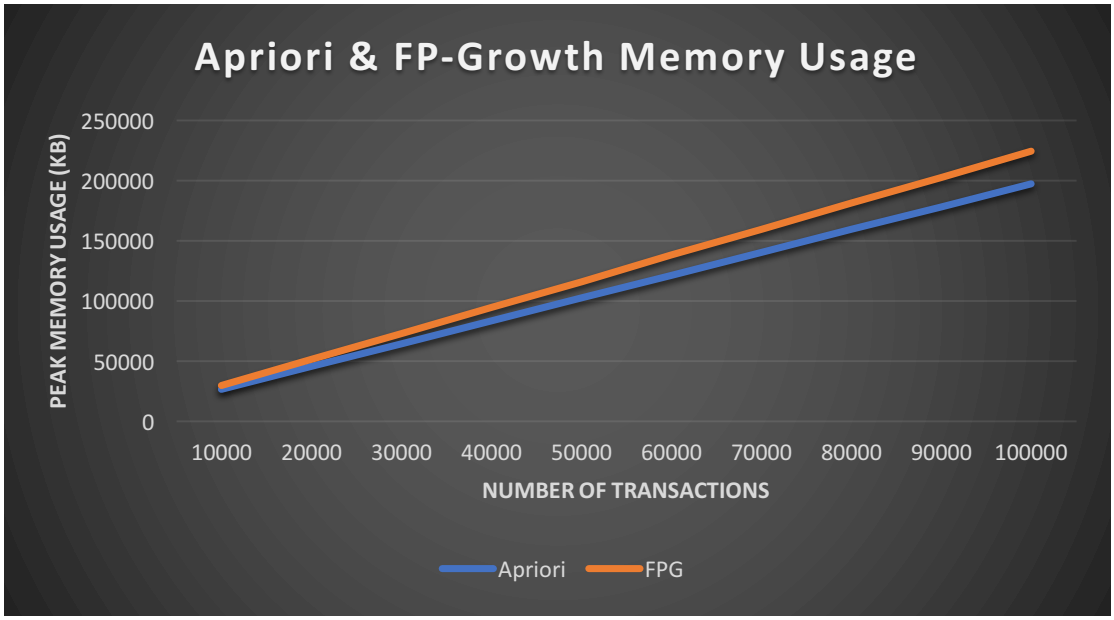

Figure 2. Relationship between number of transactions and memory usage

Figure 2 indicates that the peak memory usage is proportional to number of transactions. Slopes of Apriori and FP-Growth are 17028/90000 and 194308/90000 respectively. Clearly, FP-Growth takes a little bit more memory. To decide whether FP-Growth is better than Apriori algorithm in term of execution time and memory. We can focus on the point when number of transaction equal to 100000. Apriori needs 2.33 (=8.52/3.66) times execution time of FP-Growth, but FP-Growth only needs 1.14 (=224240/197156) times memory of Apriori. As a result, FP-Growth has better performance than Apriori algorithm generally.