Department of Physics and Computer Science
Wilfrid Laurier University
75 University Ave W
Waterloo N2L 3C5, Canada

3SUM Application on Big data set

by Wenbo Han, BSc Computer Science
supervised by Dr. Ilias S. Kotsireas, Department of Physics and Computer Science, WLU
26th April 2018

Abstract

Data mining is an interdisciplinary sub-field of computer science, which is the process of discovering some patterns in a large data set. It involved multimethods such as statistical analysis, database construction, and machine learning. It is one of the most popular subjects in the field of modern technology.

In this project, we are going to figure out a solution for a specific computational question. There are three text files called A, B and C. They are provided with the same column numbers, but the row numbers may vary. Besides, a constant λ is given, we should find three line numbers m, n and p from A, B and C files respectively which satisfies the condition such that the sum of every corresponding entry in m, n and p is equal to the constant λ. This project aims to design an efficient and reliable algorithm to implement this question. In the following sections, the design of the algorithm and some fundamental implemented methods are going to be introduced.

1 Data Pre-Processing

Before applying the particular algorithm on this data set, some pre-processing tools will be used on this step. Since this is a Linux-based project, the Shell script will be involved as a fundamental script tool and some advanced text processing language such as AWK and SED will be generally used.

AWK, a powerful programming language, designed for processing and analyzing text files such as extracting data from the original data file which is extremely suitable for this object since our text files are organized by lines and columns.

SED, another utility based on the scripting editor which is used to parse and transforms text by using this compact language.

Both of these languages are very powerful and useful. The following functions are used in the project.

1.1 Re-Organizing The Data Set

Three primary functions are generally applied to process the original data set:

- Split the original data set into some small chunks based on the characteristics of the data.
- Sort the original or the processed data set to optimize the algorithm.
- Re-classify the data sets based on the some features and organize them together for some purpose.

1.2 Filtering The Data Set

The data volume can be compressed by applying some specific algorithms on the original data set. In this project, 3SUM [2] is widely used to filter the data. The traditional 3SUM algorithm is only applicable to a single list, the modification was made by extending its functionality into some different lists in the project.

Two steps are followed in this algorithm:

- Distinguish the data from different files, since all data are decimal based number, so they can be tenfold expanded firstly. In order to keep their features (the sum of three numbers is a constant), the numbers from the first list are added to 1, the numbers from the second list are added to 2, a number 3 is subtracted from the third list's numbers. The sum of these new numbers is ten time of the original sum.

- Put all these new distinct data together, sort them and then follow the traditional 3SUM algorithm:

```
Sort(List);
while i from 0 to n-2 do
    a = List[i];
    pos_start = i + 1;
    pos_end = n - 1;
    while pos_start < pos_end do
        b=List [pos_start];
        a=List [pos_end];
        if a+b+c==λ*10 then
            /** check which list the number comes from if the
              number is end with 1, then it is from the 1st list;
            if the number is end with 2, then it is from the 2nd list;
            if the number is end with 7, then it is from the 3rd list;
            change the new data back to the original data;
            if b==List [pos_start + 1] then
                pos_start ++;
            else
                pos_end –;
            end
        else
            if a+b+c<λ*10 then
                pos_start ++;
            else
                pos_end –;
            end
        end
    end
end
```

Algorithm 1: Pseudopod for modified 3SUM algorithm

2 Algorithm Implementation

There are two methods designed to resolve this problem, one is implemented by using multithread and the alternative method is implemented by constructing some tables based on their properties in a single thread processor

2.1 Method One: Multithreading O(n^3)

In this part, multithreading implementation is involved and the method is implemented by combining the results from modified 3SUM algorithm

Firstly, the 3SUM algorithm is applied to generate a bunch of results which are all satisfied A[m]+B[n]+C[p]=λ.

Secondly, a AWK program is used to extra the indexes of the numbers which satisfy the condition, and put them into three separate sub files.

Thirdly, a multithreading program is applied by going through all satisfied conditions to find the value of m, n and p.

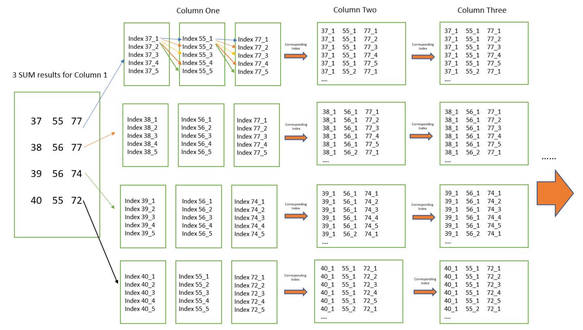

Figure 1: Multithread implementation

The arrows with different color refer to the different threads, which means multiple progresses running at the same time. This program can be scaled by running at multiple machines or a powerful machine with a multicores CPU.

2.2 Method Two: Single-thread O(n^2)

In this method, several types of tables (lists) will be built based on their properties, and some relationship can be constructed among these tables.

The dimension of these tables can be determined by checking the upper bound of the data set, such as the number of files, the number of lines and the maximum number of index for a single number.

By observing the data set and applying the utility in Linux, some parameters can be obtained:

The maximum number of index for a single number: 200,000
The maximum number of distinct numbers occurrence: 30
The line number: 1,000,000
The column number: 37
The maximum number of 3SUM result: 343

The number of files: 3

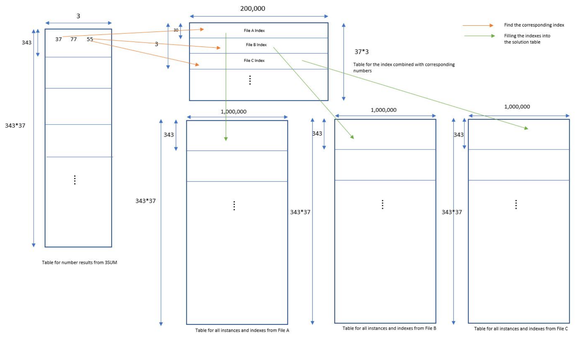

Figure 2: Single process implementation

In this method, we are going to compare the 3SUM result of column 1 in File A with the result of column 2 in File A to see if the line number in both of them. If yes, we go to the same group (row) of the table for indexes for File B to compare the common elements, and go to the same row for indexes for File C. If the common elements exists in all three tables, then we store these three line numbers and compare this result with column 3 and so on. If not, then we go to the next 3SUM result of column 1 in File A. If any combination of line numbers can be found in column 37 (last column), then this combination is the solution of this problem.
Besides, we assign 1,000,000 slots for every 3SUM results and put the line number into the corresponding index of array directly. It is really convenient for searching, the running time complexity should be O(n). Since we are going to search all possible solutions in File A, the total running time should be O(n^2).

Conclusion

Based on the experiments, we can figure that the efficiency of the running time is not all depends on the resources (memory capacity and CPU cores), the optimization of the algorithm is the vital part of an excellent solution.
In addition, we should take extra care of the memory usage when handling the problem about the Big Data.

References

[1] solutionmachine
https://soulmachine.gitbooks.io/algorithm-essentials/java/linear-list/array/3sum.html
[2] 3SUM
https://en.wikipedia.org/wiki/3SUM