УДК 009.912

STUDY OF THE EFFICIENCY OF PARALLELIZATION FOR THE MOST FREQUENT PATTERN SEARCH ALGORITHMS USING THE EXAMPLE OF APRIORI

Khovrat A.V.
Scientific Supervisor – Cand. of Tech. Sc., Sen. Res. Off. Kobziev V.H.
Kharkiv National University of Radio Electronics, dept. of AM
Kharkiv, Ukraine
phone: +38 (096) 424 48 36, email: artem.khovrat@nure.ua

Робота присвячена визначенню ефективності застосування принципів розпаралелювання, зокрема технології MapReduce задля прискорення обробки текстової інформації за допомогою алгоритму Apriori для виявлення частотних шаблонів. Визначено, що при реалізації зазначеного алгоритму з використанням фреймворку Hadoop прискорення досягає 3.1, що дозволяє стверджувати про доцільність використання паралелізованої версії обраної моделі у Big Data.

Every day, the amount of data related to Big Data is increasing, and one of the tasks that arises during their analysis is to identify the most frequent patterns in order to create a set of associative rules. Classical sequential algorithms, according to numerous studies (Subhani et al., 2018), work rather slowly on large volumes of information. In order to overcome this problem, it was decided to consider the effectiveness of using the MapReduce (Dancik, 2020) parallelization technology to speed up the work of the most popular (given the queries on Google) algorithm for finding regularities in data – Apriori (Subhani et al., 2018).

The selected algorithm has only four steps. First: finding support for each element. Support is the frequency of occurrence of an element in a certain data set. After finding this parameter, it is enough to choose those elements that satisfy a certain pre-set restriction. As a result, all the most frequent patterns will be found, on the basis of which a set of associative rules can be built. The final step is to sort the received values in descending order of elevator (ascent). Below is a fragment of pseudocode for the implementation of the algorithm.

```
while Lk−1 is not empty
  Ck ← Apriori_gen(Lk−1, k)
  for transactions t in T
    Dt ← {c in Ck : c ⊆ t}
    for candidates c in Dt
      count[c] ← count[c] + 1
  Lk ← {c in Ck : count[c] ≥ ε}
  k ← k + 1
return Union(Lk)
```

Having clarified the essence of Apriori, let's move on to the possibilities of

its parallelization. First of all, the process of determining support for each set from a given set can be performed in parallel. Although it is worth noting here that there may be a problem of having shared data for different streams.

On the other hand, this problem does not occur in the check for passing the threshold value. Provided that a copy of the threshold is stored in separate processes. The above parallelization mechanisms are combined in MapReduce technology. Its essence lies in the distribution of processing of a common set of data to individual nodes. This procedure is performed using the mapping function, with subsequent application of the necessary algorithms, and a reducer that collects data from all nodes and unifies them.

In the current case, we distribute the information read from the database in different blocks. Each of these blocks executes a sequential form of the Apriori algorithm. Next, the combiner combines the values on each block and passes them to the reducer, which in turn filters the data.

It is worth noting that the ordering process necessary for the correct operation of the algorithm is performed automatically by the available mapper.

The current study examines MapReduce with the Hadoop framework.

We will increase the number of nodes from 1 to 4. Similarly, we will increase the volume of data, for which we will consider 4 options: 10, 20, 40, 80 lines. We will set the memory on each node to 4 GB, and the clock frequency to 2.4 GHz.

The maximum speedup that was obtained for four nodes with 80,000 rows of data was 3.1. The minimum is 1.5. The relatively small acceleration is explained by the general simplicity of the basic algorithm. For one core, the set acceleration is equal to 1, because the difference between the sequential version and the application of MapReduce technology in this case is insignificant. Thus, after analyzing the existing algorithms for searching for the most frequent patterns, choosing Apriori as a basic method and finding out the possibilities for its implementation, building a solution based on MapReduce technology, it can be stated that this technology is definitely effective. Even with insignificant, by Big Data standards, volumes of data, it provides a significant acceleration, which allows you to search for patterns in the shortest possible time.

References:

1. Subhani, S., Devarakonda, N., & Nagamani, C. (2018). Parallel Computing Algorithms for Big data frequent pattern mining. *International Conference on Intelligence & Data Engineering ICCIDE-2017*. Springer Nature Singapore.

2. Dancik, G. (2020). The MapReduce framework. https://gdancik.github.io/CSC-343/data/notes/MapReduce.pdf

3. Anastasiu, D., Iverson, J., Smith, S., & Karypis, G. (2014). Big Data Frequent Pattern Mining. In Frequent Pattern Mining (pp. 225–259). Springer.