Frequent Item Set Mining for Data Streaming using Spark with Pincer Search Algorithm

Biman Giri, Sivagami M, Maheswari N.

Abstract: ***Data Streaming is the continuous flow of the vast volume of data which is transferred at high speed rate from one place to another using any network technique. Data streaming includes variety of data such as log files generated by web or mobile applications, ecommerce purchases, information from social networks, financial trading floors, or geospatial services, and telemetry from connected devices etc. The most challenging task is to find out frequent item set in ecommerce applications. It is not easy to find out the maximal frequent item set from online transactions due to high speed rate of the data transfer. In the current research context, set of applications such as market analysis, network security, sensors networks, web tracking are using association rules to find the frequent item set in data streams. Mining closed frequent item is the one step forward of mining association rules, which aims to find out the subset of frequent item set which could be frequent items. Pincer Search algorithm is one of the well-known algorithms to find closed frequent item sets and all subset of them. This algorithm uses approaches, top down as well as bottom up to find out the frequent item sets based on the threshold value. The proposed work adapts and tunes the Pincer Search algorithm for real time data streaming applications. The proposed system generates the streaming data using Apache Flume. Then data is selected randomly from real time streaming data and Pincer Search algorithm is applied on Apache Spark platform. The sample result screen of this approach is shown to ensure the use of Pincer Search algorithm in data streaming applications***

Keywords: Maximal frequent item set, Data Streaming, Web tracking, Pincer search algorithm, Apache Flume and Apache Spark

I. INTRODUCTION

Data Streaming is the sequence of data elements generated over time to time. This continuous flow of the data is running by infinite amount of time. Nowadays this large volume of data is generated by various sources like communication network, online transaction in a financial market and remote sensors in dynamic environment But these streaming data are fast changing, massive and infinite and it is impossible to store this data in some storage and the process. So, to process the streaming data, the following requirements should be addressed: (i) each item in the streaming should examine only once, (ii) the use of memory of the data streaming processing should be confined finitely and (iii) the new generated data items in the streaming should involve in some processing as fast as possible [1]

Revised Manuscript Received on March 25, 2019.

Biman Giri, SCSE, Vellore Institue of Technology, Chennai
Sivagami M, SCSE, Vellore Institue of Technology, Chennai
Maheswari N, SCSE, Vellore Institue of Technology, Chennai

In order to address these requirements, it is important to have less time complexity algorithm for streaming data to mine the frequent item set. Association rule mining is an important research to find out the frequent item set from the set of items. Frequent item set mining is the process of mining the item sets which occur more number of times in the input set. Nowadays very good range of applications uses association rules to extract the hidden patterns and the relationship between them for market basket analysis, text mining and web mining techniques [2]. Now in the current context many real time applications perform analysis using the data streaming model than the traditional static databases [3]. So, the database and knowledge extraction communities move to the data model which changes so rapidly. The association rule mining techniques find the frequent item sets based on the threshold value. This threshold value will vary application to application and domain to domain. There are several frequent item set mining techniques available like Apriori[10], FP-Growth[12], Dynamic Item Set Counting[13] and Pincer Search algorithm[11] etc. Most of the techniques uses either top-down or bottom-up, approaches to find out the frequent item set. So the time complexity of these algorithms is more and they may produce the redundant output. In order to do the analysis of streaming data, there is a need of an efficient algorithm which takes less amount of time and gives better result. Since Pincer Search algorithm uses top-down and bottom up (downward and upward properties), it finds the frequent item set in less time so Pincer Search algorithm is the better option to find out the frequent item set in streaming data.

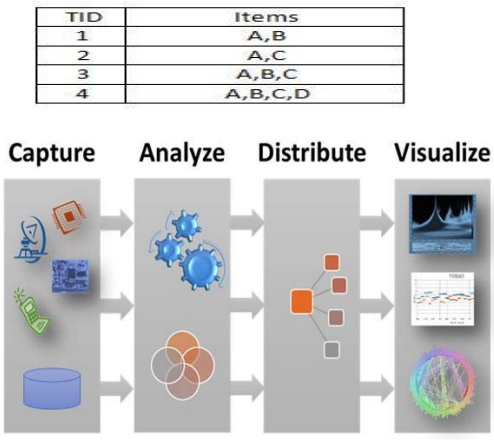

Fig. 1: Real Time Data Process, source: [15]

TID	Items
1	A,B
2	A,C
3	A,B,C
4	A,B,C,D

Fig. 2: Transaction Database T

L1	A:4 B:3 C:3 D:1
L2	AB:3 AC:3 BC:2
L3	ABC:2

Fig. 3: Find Frequent Itemset Using Normal Frequent Mining Algorithm (Apriori) where Support Count=2

Here in the Figure 2.2 the final result is ABC which is the frequent item in the transaction T. And by the characteristics of the frequent set if super set is frequent and all subset is also frequent. But the algorithm apriori not following this rule and here not only the final result is ABC with this, A,B,C,AB,AC,BC are also the part of the result. So here we can see the redundancy. If we will delete all the redundancy the sometimes we will lose the information. The support count of the maximal frequent item set is known but the support count of their subset is also lost. This algorithm is traveling from one side called bottom up so if we use this for the data streaming purpose then the performance issue also occurs and as a result we will also lose some information from the streaming data also. To overcome the disadvantage of the maximal frequent item set, the Pincer Search algorithm is considered.

Level	Bottom-Up	Top-Down
L1	A:4 B:3 C:3 D:1	ABCD-1
L2	AB:3 AC:3 BC:2	ABC:2

Fig. 4: Find Frequent Itemset Using Pincer Search Algorithm Where Support Count=2

Here from Figure 4 we can see that Pincer Search algorithm completed in the level-2 itself. The final output of this algorithm is ABC and from this we can get the entire result set by generating the sub set of ABCs along with the result ABC. Because Pincer Search uses downward closure property along with upward closure property. Downward closure property, confirms that all subsets of frequent item sets must be frequent. So this algorithm greatly reduces the number of iterations, without losing any information.

I. PROCEDURE FOR PAPER SUBMISSION

A.Related work

There are various techniques available to find the frequent item set in data streaming. In this section we will discuss three well known algorithms estMAX [1], variable window size [2] and lattice algorithm [3].

B. estMAX Algorithm

Ho jin Woo et al., [1] proposed a method for tracing the set of maximum frequent item set from the data steaming. The method called estMAX which maintains the frequent item set by prefix tree and extract the entire maximal frequent item set without using additional superset or subset checking. When the new data come, it matched with the previously prefix tree node with the newly coming item and at the same time if any subset of the newly marked item already has been set previously then it is cleared as well.

C. Variable window size by CVA

V. Sidda Reddy et al., [2] solved the challenge with respect to the minimum frequent item set over the data streaming engaging the variable window size and low memory space. To check the various structures of the data streaming, the authors have introduced a two level of the windows which fix the window size instantly and control the heterogeneous and homogeneous among the transactions added to the window. Here this algorithm proposes the incremental mining of frequent item-sets from the window and a context variation analysis approach are being used. This algorithm provides the clear boundary between the maximum frequent item set and infrequent item set in specific item.

D. Lattice algorithm

Ye-Ln Chang et al., [3] embedded the property of subsets into the lattice structure efficiently to mine closed frequent item set over a data stream sliding window. The algorithm consists of two parts, one is called lattice node which contains the information of transaction item set and transaction ID. Another one, relation between the superset and subset, here is one component called pointer which points to the subset from the superset. Since the data in the sliding window incrementally updated subset lattice, the algorithm need not reconstruct the structure

II. PROPOSED SYSTEM

The main goal of the application is to find out the frequent item set from the continuous flow of the data in data streaming. Pincer Search algorithm is tuned to find the frequent item set in data streaming. Pincer Search algorithm has tuned in such a way that, the algorithm selects the random data within the specific time duration and apply the Pincer Search algorithm to retrieve frequent item set. Also, in this section, it is explained that how Pincer Search algorithm is integrated with data streaming. An application is built to capture the data from the data streaming source and gives these data as a input to our algorithm, processes the input data and generates result. Then the result is displayed via the user interface of the application to the end user. The proposed application is divided in three modules as follows: A) Data streaming B) Pincer Search Algorithm C) Result display. The system architecture of proposed system is given below in Fig-5.

A. Data Streaming

Initially, the data analytics is done using static data in different ways to retrieve some decision-making information. But instead of working with static data, if the analysis is done for dynamic data in data streaming then we may get the updated result and hence the obtained results will be suitable for real time applications. Streaming data analysis can be done in parallel to achieve the result faster and it is appropriate to take decision in e-business applications faster

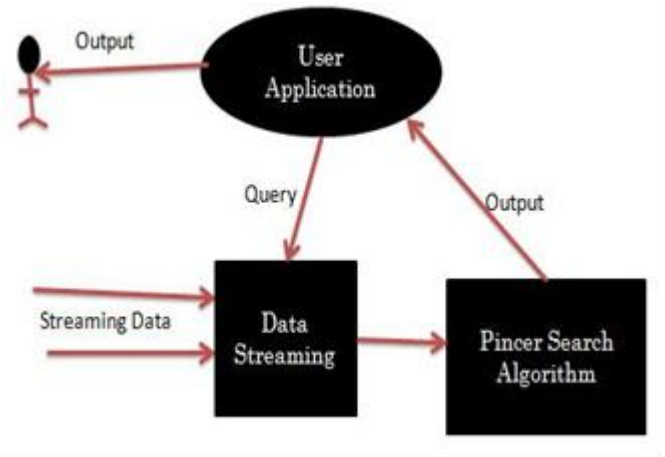

Fig. 5: System Architecture

So, the Apache Flume platform is used to capture the data from various sources. Apache flume is distributed, reliable and available system for efficiently collecting aggregating and moving large amount of data from different source to one centralized source. After storing the data in centralized data store (HDFS5 here we are using), the data to be processed. Hadoop cannot process the data immediately first it will store the data then from HDFS (Hadoop distributed file system) and it will process the data. But the platform called Apache Spark instantly processes the data and gives the output. Since the Apache Spark is more appropriate to do the real time analysis, Apache Spark is used to process the streaming data in the application.

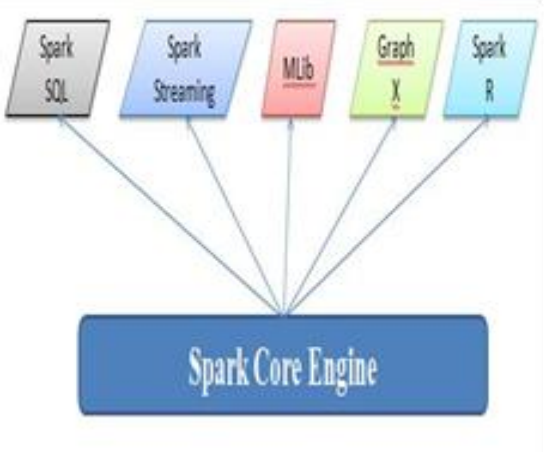

Fig. 6: Spark Eco-System, source: [14]

Spark was introduced by the apache software foundation for speeding up the hadoop computational process. Spark is not the Modified version of the hadoop but for storage purpose it uses hadoop. There are two ways by which we can build the spark on hadoop these are, First Standalone deployments where the Spark work on HDFS which is used for data storage purpose where the spark and Map Reduce run side by side. Second is Hadoop Yarn where Spark run on Yarn or Mesos which integrates the Spark with Hadoop and allows other components to run on spark. To work with spark, we need to know the features of spark or the ecosystem of the spark which is as follows. Here we are using spark streaming. Apache Spark core is the execution engine for the spark platform on which all the functionality builds upon this. Spark SQL introduces new data abstraction called Schema RDD which supports structure and semi structure data. Spark Streaming is the unique feature of the Spark which allows Spark to play with real time data to do analysis. It divides the streaming data into batch data then create RDD for each and every data and do the analysis.

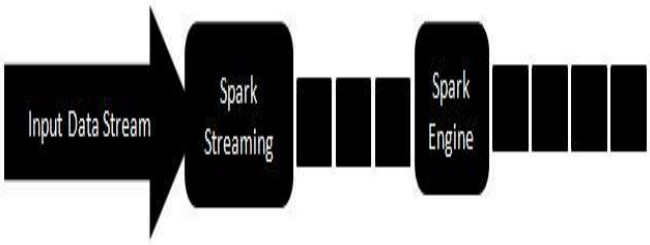

Fig. 7: Spark Streaming, source: [14]

B. Pincer search algorithm

Dao-I Lin Zvi M. Kedem of New York University proposed a Pincer Search algorithm in 1997. It is the modification of Apriori algorithm which is proposed by R. Aggarwal Srikant. The Apriori algorithm uses bottom-up search and restricts to top-down search to find the frequent item set. All the frequent mining techniques travel from either top-down or bottom-up to find the frequent item set, whereas the Pincer search algorithm uses both top-down and bottom-up approaches for association rule mining. The Pincer Search algorithm use two properties for top-down Maximal Frequent Item Set for confirmed item set and Maximum Frequent Candidate Set for not confirmed item set, for bottom-up and it uses two parameters one is L for confirmed item set and C for not confirmed item set. If some item set is found in top down approach, then this will be uses for eliminating the item for bottom up approach. And the item in top-down is frequent means all the subset of this is also frequent which follow the downward property. If some infrequent item is found in bottom-up approach this can be used to eliminate the item from top-down result because if item set is infrequent means all the superset is also infrequent it follows the upward property also.

By using these two upward and downward closure properties, the algorithm works faster, and which is suitable for data streaming. The algorithm starts with by generating the item sets and it also uses Maximum Frequent Candidate Set for top down approach. L is confirmed set of bottom-up and MFS is confirmed set for top-down approaches. The recursive call of this algorithm will be terminated when this two set will match. In each pass we are counting the support count for candidates in bottom-up direction, and also top-down direction. Consider at pass K, the length of the item set must be K, and it should be classified. If some item set which is an element of Maximum Frequent Candidate Set is found frequent, then the all the subsets will frequent. Then all its subsets of cardinality K are pruned from the set of candidates considered in bottom-up approach in this pass. They and their superset never be candidates again. Similarly when some new item set is found infrequent in bottom-up direction, the algorithm makes use of it to update Maximum Frequent Candidate Set, so that no subset of any item set in MFCS should have this infrequent item set as its subset. So, this algorithm obtains the result faster than any other frequent item set mining technique and remove the redundancy in the final output. Hence this algorithm is chosen for frequent item set mining in real time data streaming applications. This algorithm has been tuned to select the random transactions from real time data streaming as input data and applies the Pincer Search algorithm to achieve the result faster

C. Pincer search method

```
L0 = ∅ ; k=1; C1 = {{i} | i ∈ I }; S0 = ∅
MFCS = {{ 1,2,…, n}}; MFS = ∅ ;
  do until Ck = ∅ and Sk-1 = ∅
      read the database and count support for Ck & MFCS.
    MFS = MFS ∪ {frequent itemsets in MFCS};
    Sk = {infrequent itemsets in Ck };
    call MFCS_gen algorithm if Sk ≠ ∅;
     call MFS_pruning procedure;
    generate candidates Ck+1 from Ck ;
if any frequent itemset in Ck is removed from
 MFS_pruning procedure
call recovery procedure to recover candidates to Ck+1 .
call MFCS_prune procedure to prune candidates in Ck+1 .
    k=k+1;
    return MFS
    MFCS_gen
    for all itemsets s ∈ Sk
        for all itemsets m ∈ MFCS
            if s is a subset of m
          MFCS = MFCS \ {m}
            for all items e ∈ itemsets
  if m \ {e} is not a subset of any itemset in MFCS
              MFCS = MFCS ∪ {m \ {e}}
     return MFCS
    Recovery
    for all itemsets l ∈ Lk
          for all itemsets m ∈ MFS
        if the first k-1 items in l are also in m
                  for i from j+1 to |m|
Ck+1 = Ck+1 ∪ {{l.item1, …, l.itemk,   m.itemi}}
        MFS_prune
       for all itemsets c in Lk
      if c is a subset of any itemset in the current MFS
      MFCS_prune
      for all itemsets in Ck+1
     if c is not a subset of any itemset in the current MFCS
  delete c from Ck+1;
```

Fig 8: Pincer search algorithm

To implements this algorithm in the proposed frame work, the language called Scala is used. Scala is scalable, and it is a hybrid functional programming language. It is very smooth integration with object-oriented programming language. Since Spark is built using Scala language, if the Pincer Search algorithm is implemented using Scala, it will run fast and as well as there will be flexibility to manipulate the Spark internal features based on requirements.

D. Result display

Result display module acts as an interface between the user and application. In this application user can see one window where he or she can specify (i) the duration of the processing (ii) the support count which is the threshold value and (iii) the output location and format. The application takes these inputs as a user query and based on this it will process the streaming data, display the result and store the output of the streaming output in the specified location in the user specified format. Also, the application stores the result in some external storage for future processing.

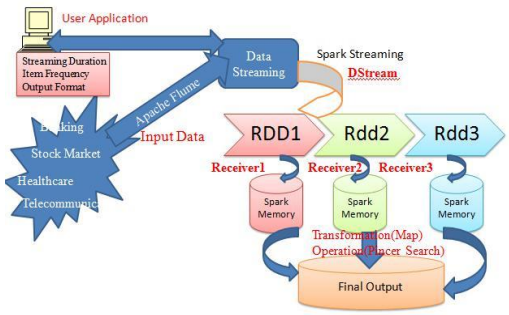

Fig. 9: Application Flow

The proposed system captures the data using apache flume channel and stores these data in HDFS. Then the Spark streaming engine fetches the data from the HDFS and then divides these data in to separate DStream and the size of DStream will depend on the amount of data. The data Dstream is stored in temporary storage space called spark memory and then algorithm fetches the data from the spark memory and applies the frequent mining process to find the frequent item set inside the data then display the data in user specified location. In the proposed framework, user application is implemented in java platform and the algorithm is implemented using the Scala language and the data streaming module, captures the data from Apache Flume and stores data temporarily in HDFS which is called Hadoop Distributed File System. Then Apache Spark's unique feature Spark Streaming fetches the data from the HDFS and then process the data, here processing is based on frequent item set mining Pincer Search technique to find the frequent item set from the data streaming, and then the output will be provided based on user's requirements. To build this proposed framework, the environment is setup, using the hardware and software. The required hardware and software details are as mentioned below,

III. Implementation details

A. Software Requirements

Apache Flume 6.0
Apache Hadoop 2.6
Apache Spark 2.6
Apache Maven
Scala Platform
Ecllipse Neon
OS: Linux

B.Hardware requirements

Proessor: Intel i3 processor
RAM: Minimum 2GB
Hardisk: 128GB

To build this system we are have to first environment setup, to do this first we have to install Apche Hadoop in the Linux operating system, Then to get the data streaming features we Apache Spark have to be installed on the Hadoop System with Scala environment. For data streaming, Apache Flume is integrated, and it uses the Netcat source for generating the data. Then the data is processed by the frequent mining algorithm which find the frequent item set from the streaming of the data and send these data based on user requirements. Here is snapshot of data generating and the output of the pincer search algorithm is shown.

IV. CONCLUSION

Pincer Search Algorithm is tuned to handle real time data to find the frequent item set in data streaming. The algorithm is tested with (i) the generated data in the local host (ii) dataset simulation using Apache Flume and netcat and got the frequent item set in both cases correctly. In future the algorithm will be tested in real time with data generated from any IOT (Internet of things) applications to find the frequent item set. Various domains like bio-informatics, e-commerce, e-purchase, this tuned Pincer-Search algorithm may be used to find the dependency across the data items to unveil the association between the various attributes of the data set.

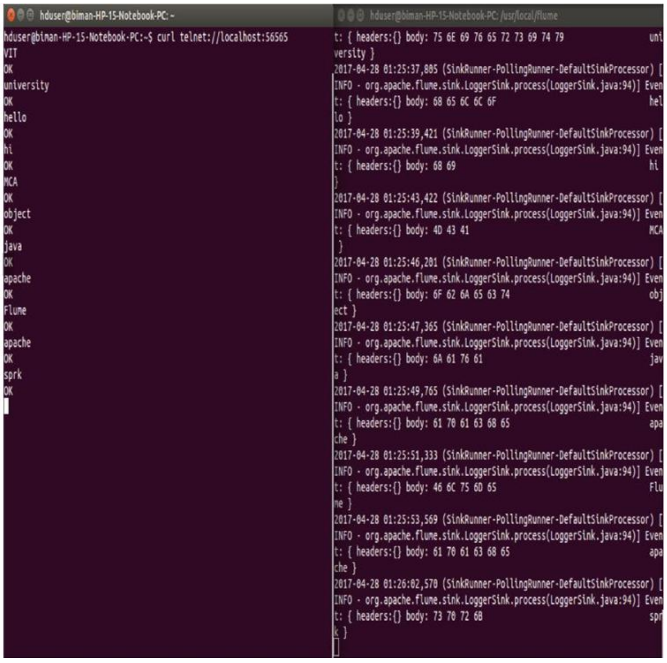

Fig. 10: Flume Data Capturing.

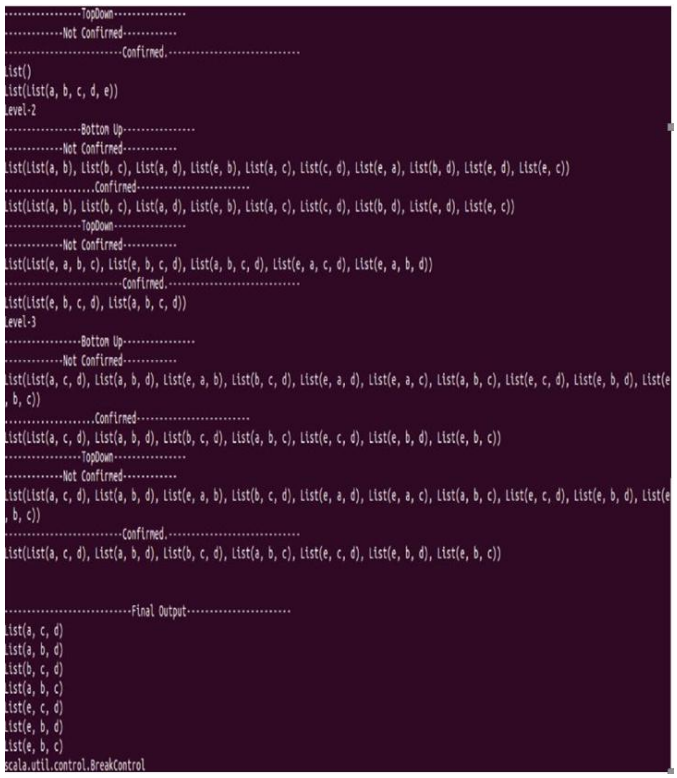

Fig. 11: Pincer search algorithm output

REFERENCES

1. V.Sidda Reddy T.V.Rao A.Govardhan. "Mining Frequent item-sets(mfi) Over Datastreams: Variable Window Size(vws) By Context Variation Analysis (cva) Of The Streaming Transactions".International Journal of Data Mining Knowledge Management Process (IJDKP).
2. R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules in Large Databases". Proc. of the 20th Int. Conf. on Very Large Data Bases, 1994.
3. Pei X. Yan C. Giannella, J. Han and P. Yu. "Mining Frequent Pattern in Data Streams at Multiple Time Granu-larities". Proc. of the NSF Workshop on Next Generation Data Mining, 2002.
4. H. Chang and W. S. Lee. "A Sliding Window Method for Finding Recently Frequent Itemsets over Online Data Stream". Journal of Information Science and Engineering, 2004.
5. Gurmeet Singh Manku and Rajeev Motvani. "Approximate frequency counts over data streams".
6. Kevin Chen Moses Charikar and Martin Farach-Colton." Finding frequent items in data streams". Journal of Theoretical Computer Science Special issue on automata, languages and programming, 2004.
7. M.Narendra V.Sidda Reddy and K.Helini. "Knowledge Discovery from Static Datasets to Evolving Data Streams and Challenges". International Journalof Computer Applications, 2014.
8. Ho Jin Woo and Won Suk Lee. estMax: "Tracing Maximal Frequent Item Sets Instantly over Online Transactional Data Streams". IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING.
9. Chia-En Li Ye-In Chang and Wei-Hau Peng. "An Efficient Subset-Lattice Algorithm for Mining Closed Frequent Itemsets in Data Streams". Conference on Technologies and Applications of Artificial Intelligence.
10. Jugendra Dongre, Gend Lal Prajapati, S.V. Tokekar, "The Role Of Apriori Algorithm For Finding The Association Rules In Data Mining", Issue and Challenges in Intelligent Computing Techniques(ICICT), 2014 International Conference.
11. Dao-l Lin, Z.M. Kedem, "Pincer Search – An Efficient Algorithm For Discovering The Maximum Frequent Set",IEEE Transections on knowledge and data engineering Vol:14, 03.05.2002.
12. Min Chen, XueDong Gao,HuiFei Li, "An efficient parallel FP-Growth algorithm , Cyber-Enabled Distributed Computing and Knowledge Discovery", CyberC '09. International Conference on,2009.
13. Mikhail Zymbler,South Ural State University, "Accelerating Dynamic Itemset Counting on Intel many-core system"s , Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention .