International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 2 431 - 437

Directed Graph based Distributed Sequential Pattern Mining Using Hadoop Map Reduce

Sushila S. Shelke, Suhasini A. Itkar,
PES's Modern College of Engineering, Shivajinagar, Pune

Abstract - Usual sequential pattern mining algorithms experiences the scalability problem when trade with very big data sets. In existing systems like PrefixSpan, UDDAG major time is needed to generate projected databases like prefix and suffix projected database from given sequential database. In DSPM (Distributed Sequential Pattern Mining) Directed Graph is introduced to generate prefix and suffix projected database which reduces the execution time for scanning large database. In UDDAG, for each unique id UDDAG is created to find next level sequential patterns. So it requires maximum storage for each UDDAG. In DSPM single directed graph is used to generate projected database and finding patterns. To improve the scanning time and scalability problem we introduce a distributed sequential pattern mining algorithm on Hadoop platform using MapReduce programming model. We use transformed database to reduce scanning time and directed graph to optimize the memory storage. Mapper is used to construct prefix and suffix projected databases for each length-1 frequent item parallel. The Reducer combines all intermediary outcomes to get final sequential patterns. Experiment results are compared against UDDAG, different values of minimum support, different massive data sets and with and without Hadoop platform which improves the scaling and speed performances. Experimental results show that DSPM using Hadoop MapReduce solves the scaling problem as well as storage problem of UDDAG.

Index Terms — *Distributed Environment, Directed Graph, Sequential Pattern Mining, Transformed Database.*

*****

I. INTRODUCTION

Sequential pattern mining is an vital data mining practice widely used in many applications, like Bioinformatics, customer behavior guesses in transaction history, web mining, intrusion detection in network attack. Usually, the main intention of sequential pattern mining is to find frequent sequences within a sequential database. The problem was first proposed in [1]. There is good progress towards mining sequential patterns like GSP [2], SPAM [3] Apriori-based algorithm , FreeSpan [4], PSPM [5] projection-based, SPADE [6] vertical data format based algorithm and pattern growth based approach in PrefixSpan [7], UDDAG [8], have been proposed. Pattern growth approach is significantly used in many algorithms owing to its advantages like Divide and conquer strategy, compact database and without candidate generation. Mainly execution of all above algorithms is on standalone environment which has some weaknesses like scalability problem, less efficient for huge dataset, large scanning time for database. To increase the performance and resolve the scalability issues of sequential pattern mining many researchers have provide different methods to work on distributed environment like parallel mining , distribute the mining computation over different nodes, apply different distributed environments like grid computing, cluster, cloud, Hadoop, etc.

In this paper we have proposed Directed Graph as a data structure to store the database efficiently which optimizes the memory and reduce the number of scans. Our algorithm is extended from UDDAG algorithm which is implemented using MapReduce programming model on hadoop distributed environment. Due to its execution on hadoop distributed environment, it solves the scalability problem.

The rest of the paper is structured as section 2 defines the problem statement and related works. Section 3 explains preliminary review of UDDAG and Hadoop MapReduce model. Section 4 gives algorithm for Directed Graph based distributed Sequential Pattern Mining on Hadoop MapReduce. Experiment results and performance evaluation are mentioned in section 5. Section 6 conclude the paper and discuss about issues that need to be consider in future work.

II. PROBLEM DEFINITION AND RELATED WORK

2.1 Problem Definition:

Let D_S be a sequence database, and $I = \{x_1, \ldots, x_m\}$ be a set of m different items, $S = \{s_1, \ldots, s_i\}$ is a sequence which contains an ordered list of itemsets. An itemset s_i is a subset of items $\subseteq$ I. A sequence $S_a = \{a_1, \ldots, a_n\}$ is a subsequence of sequence $S_b = \{b_1, \ldots, b_m\}$ where $1 \leq i_1 < \ldots < i_n \leq$ m such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}$.

Sequential pattern mining is to find out the complete set of sequential patterns whose frequency count $\geqslant$ min_sup * |D|, where min_sup is a minimum support threshold. Table 1 represents the example of sequence database.

Finding sequential patterns from massive dataset is not efficient on standalone system due to large scanning time and more memory storage. So, to discover the complete set of sequential patterns from massive datasets this paper proposed a system which uses MapReduce programming model on distributed environment Hadoop to increase the speed and scalability performance. Proposed system uses the directed graph for memory optimization and less scanning time.

TABLE 1

Example of Sequence Database

Seq_Id	Sequence
1	1 (1, 2, 3) (1, 3) 4 (3, 6)
2	(1, 4) 3 (2, 3)(1, 5)
3	(5, 6) (1, 2) (4, 6) 3 2
4	5 7 (1, 6) 3 2 3

2.2 Related Work:

Large research has been done and going on in the area of parallel and distributed sequential pattern mining on different environments like Grid, cluster, cloud , Hadoop, Distributed processors or nodes, etc. A frequent sequence mining algorithm based on variant of the parallel tree projection is designed by Guralnik and Karypis [11]. The Par-CSP algorithm (parallel closed sequential pattern mining) which runs on distributed memory system was introduced in [12]. Their parallel algorithm is based on the modern frequent closed sequence mining algorithm BIDE [13]. In [9] [10] by using Apriori method, GSP algorithm is executed on grid computing environment which is straightforward for loosely coupled methods. Grid computing platform requires a sophisticated method of data partitioning, stratagem to determine and allot the job to grid node dynamically and a smaller amount controlling grid can decrease the performance of entire structure. Cluster based algorithm is developed in [14], in which matching definition algorithm collects the sequence data into several clusters and then on parallel computers which are distributed memory modes clusters are distributed. SPAMC is nothing but Sequential pattern mining on the cloud algorithm which is extended from SPAM [15] is implemented on MapReduce framework in [16]. Pattern growth based PrefixSpan algorithm which performs pattern mining on huge data set on Hadoop environment by using MapReduce programming model has been projected in [17]. Multidimensional sequential pattern mining is performed on distributed sites in [18][21].

This paper is extended from UDDAG algorithm and concept of MapReduce programming model on Hadoop environment.

III. PRELIMINARIES

3.1 Up Down Directed Acyclic Graph:

In UDDAG, a new data structure, UpDown Directed Acyclic Graph for efficient mining of sequential patterns is used. DAG (Directed Acyclic Graph) represents patterns as vertexes with ids of transaction containing the pattern and directed edge as relationship of patterns. Up DAG represents DAG for patterns found in Prefix projected database while Down DAG represents DAG for patterns found in Suffix projected database of Frequent item x and by merging both DAG represents UDDAG with root vertex x. UDDAG allows bidirectional pattern growth from both the ends for detected sequential pattern. UDDAG algorithm uses transformed database where for each frequent item x, the algorithm creates a root vertex for x detects all the patterns from prefix and suffix projected database of x and then it creates directed acyclic graph to represent the enclose relationship of frequent items. UDDAG generates less projected database compare to PrefixSpan as UDDAG consider prefix and suffix projected database at the same time. UDDAG eliminates unnecessary candidate generation. Data structure in UDDAG requires more memory storage than PrefixSpan but it involves major cost of storing projected database. UDDAG has some challenging issues like independent frequent item detection, large memory usage for data structure UDDAG. The new system uses directed graph to store transformed database and it allows bidirectional pattern growth without creating projected database as like UDDAG.

3.2 Hadoop MapReduce:

Hadoop MapReduce is a software framework which allows distributed processing of huge amounts of data (multi-terabyte data-sets) on thousands of nodes of service hardware in a consistent, fault-tolerant manner [19]. A MapReduce *job* usually divides the input data-set into independent structures which are executed by the *map tasks* in a fully parallel manner. The outputs of the map functions are sorted by framework, which are then given as input to the *reduce tasks*. Typically inputs and the outputs of the job are stored in terms of file system. The framework takes care of tasks like scheduling, monitoring and re-execution of failure tasks. Map Reduce is trigger by the map and reduce operations in LISP like functional languages. This model solves the computation problem through two functions: map and reduce. Essentially, the MapReduce model permits users to write map and reduce components with functional-style code. Finally, these components are scheduled by the MapReduce system to scattered assets for execution while handling many tough problems such as network communication, parallelization and fault tolerance.

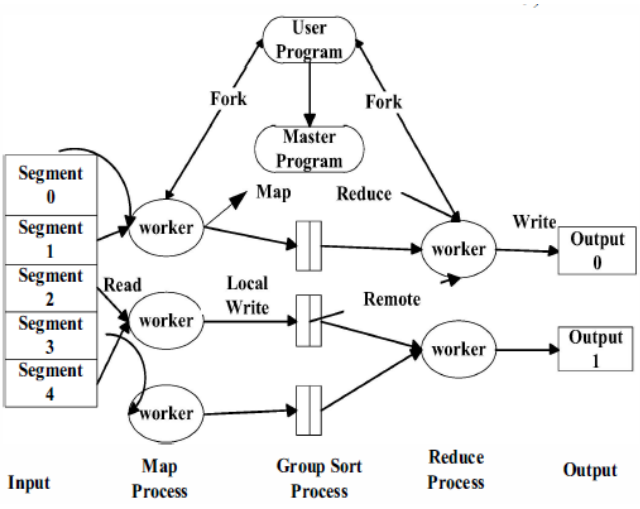

FIGURE 1 MAP REDUCE ARCHITECTURE

Input for the map function is a key-value pair which produces a list of key-value couples as output which is characterized as:

$$map :: (key_1, value_1) \Rightarrow list(key_2, value_2)$$

A reduce function acquires all the output with similar key values and produce a single list as output which is characterized as:

$$reduce :: (key_2, list(value_2)) \Rightarrow list(value_3)$$

The architecture of MapReduce model is shown in figure1. A MapReduce program mainly executes in two phases for parallel execution. In first phase, the input reader from master node splits the data into small parts and submits them to selected mapper programs randomly. In this process mappers perform their task in parallel processing stage and produces a group of [key, value] pairs. Each generated item is sorted and forwarded to the reducer. In second phase, reducer program combines all the items with given key and finds a single entity as a result. All reduce operations are performed independently similar to map operations. Input, final outputs are stored on a distributed file system whereas intermediate results are stored on local file system of map and reducer worker.

IV. Directed Graph Based Sequential Pattern Mining on Hadoop MapReduce

This section explains the directed graph based sequential pattern mining on Hadoop MapReduce, which transforms the original sequence database and then constructs the directed graph for storing transformed database. The construction of directed graph is done centrally and distributed at each node. Instead of projecting the database from transformed database and then constructing prefix and suffix projected database for finding sequential patterns as like UDDAG, directed graph based system make use of directed graph for construction of prefix and suffix database without constructing projected database. Construction of prefix and suffix database is done parallel by mapper and intermediate results are given as input to reducer to find the sequential patterns.

4.1 Database Transformation:

Definition 1 (Frequent item set): The absolute support for an item set in a sequence database is the frequency count of transactions whose sequences involve the item set. Frequent item (FI) set is an item set with a support count larger than minimum support frequency.s First we assign unique id to all frequent items in FI set in sequence database D. For example, Frequent items of length 1 for Table 1database are: [1], [1, 2], [2], [2, 3], [3], [4], [5], [6]. By assigning a unique id to each FI, we get [1] - 1, [1, 2] - 2, [2] - 3, [2, 3] - 4, [3] - 5, [4] - 6, [5] - 7, [6] - 8.

Definition 2 (Item pattern): An item pattern is a sequential pattern with accurately one item in each item set it contains.

Definition 3 (Transformed Database): Let D_S be a sequence database , by replacing item sets in each sequence with ids of FIs contained in the item set we get transformed database D_T. Transformed database is shown in Table 2.

TABLE 2

Transformed Database

Seq_Id	Sequence
1	1 (1, 2, 3, 4, 5) (1, 5) 6 (5, 8)
2	(1, 6) 5 (3, 4, 5)(1, 7)
3	(7, 8) (1, 2, 3) (6, 8) 5 3
4	7 (1, 8) 5 3 5

4.2 Construction of Directed Graph:

Construct Directed graph for D_T which contains vertex and directed edges. Vertex represents unique id assigned for frequent item. Each vertex also stores transaction ids where it appears in database and element id where frequent item appears. Figure 2 shows directed graph for table 2.

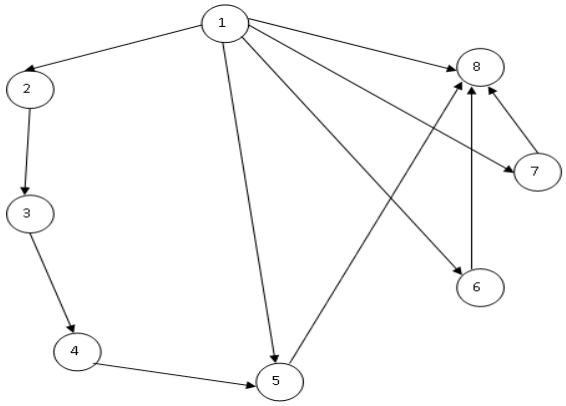

FIGURE 2 DIRECTED GRAPH

Let, set of unique ids for each FI contains n ids are $[I_1, I_2, \dots, I_n]$. Vertex I_1 stores transaction ids and each transaction id stores element id which is represented as:

$$I_1 \rightarrow [T_1, T_2, \dots, T_m] \quad \rightarrow (1)$$

m → number of transactions in the database

$$T_1 \rightarrow [T_{1_1}, T_{1_2}, T_{1_3}] \quad \rightarrow (2)$$

$$T_2 \rightarrow [T_{2_1}, T_{2_4}] \quad \rightarrow (3)$$

$$T_m \rightarrow [T_{m_1,\dots,} T_{m_n,}]$$

For example, for vertex 1 as per transformed database

$1 \rightarrow [1,2,3,4] \quad \rightarrow (4)$

$1 \rightarrow 1_1, 1_2, 1_3$

$2 \rightarrow 2_1, 2_4$

$3 \rightarrow 3_2$

$4 \rightarrow 4_2$

Edges in the graph are represented by item sequence in element in the sequence database which is used to traverse the graph while detecting the in between pattern.

 ISSN: 2321-8169
431 - 437

4.3 Prefix and Suffix Projected Database:

Prefix and suffix database is created for each vertex by using directed graph. Suppose for vertex I_1 transaction ids are as shown in equation 1 and element ids for each transaction are shown in equation 2 ,3,etc. While constructing Prefix database it ignore higher element id means ignore T_{13} *and* T_{24} from equation 2 and 3. However while constructing suffix database it ignore lower element id means ignore T_{11} *and* T_{21} from equation 2 and 3. For example let I_1= 1 and set of transaction ids and element ids are as shown in equation 4then prefix and suffix database for 1 is as shown below:

Pre (1_D) = 1. < 1, 1 >
2. < 1 >

Suf (1_D) = 1. < 1, 1 >
2. < 1 >

4.4 Sequential Pattern Mining:

Sequential pattern mining is performed by considering following three cases

1. Each vertex in directed graph is itself a sequential pattern.
2. Sequential pattern from Prefix and Suffix Projected Database
   Find the frequent items from prefix and suffix projected database by checking the support count value then detect the frequent patterns as below.

   To find the sequential patterns from prefix Projected database; if FI in Pre (a_D) for pattern a then append a after each FI.
   For example, a = 8, FI set for Pre(8_D) = 1, 2, 3, 7
   Then sequential patterns are $\{(1,8), (2,8), (3,8), (7,8)\}$

   To find the sequential patterns from suffix Projected database; if FI in Suf (a_D) for pattern a then append each FI after a.
   For example, a = 8, FI set for Suf (8_D) = 3, 5
   Then sequential patterns are $\{(8,3), (8,5)\}$

3. Discover Sequential patterns where length-1 pattern exist in between. Sequential patterns in this case are like length-1 pattern exist in between the beginning and ending of sequential patterns by considering its prefix and suffix database. First find the frequent pattern as in case 2.
   Let,
   $Pre(a_D) = [b, c, d, e] \rightarrow (P)$
   $Suf(a_D) = [d, f] \rightarrow (S)$
   Give unique ids to each frequent item in P and S then intersection of each element in P and a (the item pattern) with each element in S and find common transaction ids as $P_i \cap a \cap S_j$ where i and j are ids of frequent item in P and S.
   For example,
   $$a = 8P = [1,2,3,7]S = [3,5]$$

   Take intersection as $P_1 \cap 8 \cap S_1$ that is
   $$1 \cap 8 \cap 3 = 3,4$$

   Now check the sequence of element ids for 3rd and 4th transaction for vertex 1, 8 and 3 in directed graph. If the sequence for all above vertex is not in ascending order then there is 0 support count for this pattern else count the frequency count as 1. For vertex 1, 8 and 3 the sequence is $3_2 \rightarrow 3_1, 3_5 \rightarrow 3_2, 3_5$ no frequency count. For Vertex 7, 8, 3, the common transaction ids are $7 \cap 8 \cap 3 = \{3, 4\}$ and the sequence for 3rd is $3_1 \rightarrow 3_1, 3_3 \rightarrow 3_2, 3_5$. Consider $3_1 \rightarrow 3_3 \rightarrow 3_5$ as ascending order so frequency count for pattern 783 for 3rd transaction is 1.

4.5 Architecture of for Directed Graph based Distributed Sequential Pattern Mining on Hadoop MapReduce

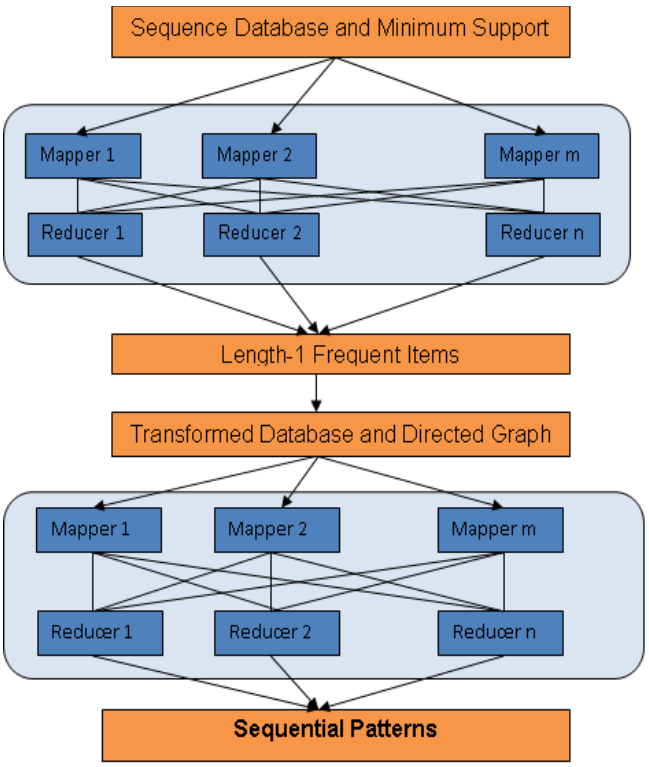

FIGURE 3 DSPM ARCHITECTURE

Architecture diagram for DSPM on hadoop Map reduce is shown in figure 3. MapReduce model is applied in two

phases. Phase 1 finds length-1 frequent items from sequence database and minimum support threshold value using map reduce model. Transformed database is generated from length-1 frequent items. Directed graph is constructed to store transformed database which is used as input for next map reduce model in phase 2. In this prefix and suffix databases are generated and frequent item which support minimum support count value are generated by mapper and reducer. Combined output of all reducer is nothing but final sequential patterns.

4.6 Algorithm for Directed Graph based Distributed Sequential Pattern Mining:

Input: Sequence Database D, Minimum Support value as min_sup

Output: P the complete set of patterns in D

Algorithm:

1. Input is given to mapreduce framework where it scans the database D and find length-1 FI as FISet=D.getFI(min_sup) .
2. Transform the sequence database by calling D.transform(D,FISet).
3. Construct the directed graph by calling D.constructDG(Transformed Database, FISet)
4. Assign task of generating prefix and suffix database to mapper function as genpresufdb(Dgraph,min_sup) parallelely on multiple nodes.
5. Intermediate result of step 5 is passed as input to reducer function to find the sequential patterns by following 3 ways
    a. Copy all length-1 frequent items to list of sequential pattern P.
    b. Find FI from prefix and suffix database by using case 2 and append it to pattern P.
    c. Find FI for in between pattern using case 3.

The algorithm first scans the database and generates length-1 FI set which is used to create transformed database using mapreduce. Next step in is to construct directed graph by using transformed database and FI set. Mapper function on different node generate prefix and suffix projected database parallel. Intermediate result of mapper function is passed as input to reducer function to find the sequential patterns as mentioned in section 4.4. Task of finding length-1 frequent item and finding patterns from prefix and suffix database is done on hadoop mapreduce platform to perform the task on distributed environment where it achieves maximum scalability.

Final sequential patterns for database in Table 1 by using above algorithm are discovered for each vertex in directed graph are listed below as Sequential Pattern (SP_i) where i indicates vertex in graph.

$$SP_1 = \{\langle 1\rangle, \langle 1,1\rangle\}$$

$$SP_2 = \{\langle 2\rangle\}$$

$$SP_3 = \{\langle 3\rangle, \langle 1,3\rangle, \langle 3,1\rangle, \langle 1,3,1\rangle\}$$

$$SP_4 = \{\langle 4\rangle, \langle 1,4\rangle, \langle 4,1\rangle, \langle 1,4,1\rangle\}$$

$$SP_5 = \{\langle 5\rangle, \langle 1,5\rangle, \langle 2,5\rangle, \langle 3,5\rangle, \langle 5,5\rangle, \langle 1,3,5\rangle, \langle 1,5,5\rangle, \langle 5,1\rangle, \langle 5,3\rangle, \langle 1,5,1\rangle, \langle 1,5,3\rangle\}$$

$$SP_6 = \{\langle 6\rangle, \langle 1,6\rangle, \langle 2,6\rangle, \langle 3,6\rangle, \langle 6,3\rangle, \langle 6,5\rangle, \langle 6,5,3\rangle, \langle 3,6,5\rangle, \langle 2,6,5\rangle, \langle 1,6,5\rangle\}$$

$$SP_7 = \{\langle 7\rangle, \langle 7,1\rangle, \langle 7,3\rangle, \langle 7,1,3\rangle, \langle 7,5\rangle, \langle 7,1,5\rangle, \langle 7,3,5\rangle, \langle 7,5,3\rangle\}$$

$$SP_8 = \{\langle 8\rangle, \langle 1,8\rangle, \langle 2,8\rangle, \langle 3,8\rangle, \langle 7,8\rangle, \langle 8,3\rangle, \langle 8,5\rangle, \langle 8,3,5\rangle, \langle 8,5,3\rangle, \langle 7,8,3\rangle, \langle 7,8,5\rangle, \langle 7,8,5,3\rangle\}$$

In Proposed system task of constructing prefix and suffix database is done on multiple nodes parallel so, it reduces the execution time as compare to old algorithms. Also, it uses directed graph to store database which gives better memory optimization and less scanning time.

V. Experimental Results

The proposed system is tested on synthetic data sets which are generated by IBM Quest Synthetic Data Generator [21]. Experimental result shows the scalability performance between Directed Graph based system verses existing sequential pattern algorithm UDDAG, comparison of Directed Graph with and without Hadoop platform. Performance of the system will be calculated against different sizes of datasets, different values of minimum support and number of nodes in distributed environment.

The data sets were generated for parameters shown in table 3.

TABLE 3

Parameters for generating Datasets

Symbol	Name
C	Number of Sequences
N	Number of Different Items
S	Average number of items per sequence
T	Average number of items in transactions
I	Average number of items in transaction in pattern

Experiment Result1 shows the performance for dataset C5N100S8T3I3 having 5k sequences, 100 distinct items and 50% min_support in figure 4. It highlights the scalability performance for UDDAG, Directed Graph

 ISSN: 2321-8169
431 - 437

and Directed Graph on Hadoop.

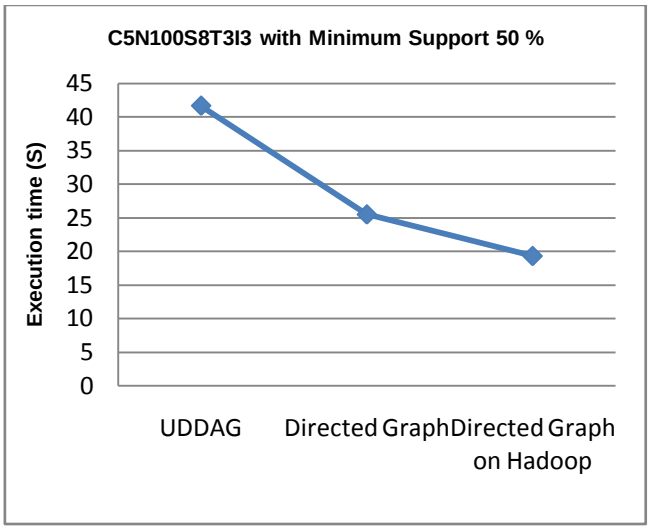

FIGURE 4 EXECUTION TIME COMPARISONS WITH MIN_SUP 50%

Experiment 2 display the performance for same dataset but with different min_sup values as 5%, 10% and 20% for all three algorithms. It is given table 4 as well as in figure 5.

TABLE 4

EXECUTION TIME COMPARISONS FOR UDDAG, DG AND DG ON HADOOP

Minimum Support %	UDDAG (Seconds)	Directed Graph (Seconds)	Directed Graph on Hadoop (Seconds)
5	61.621	44.178	28.475
10	65.592	46.021	26.926
20	47.59	25.598	16.579

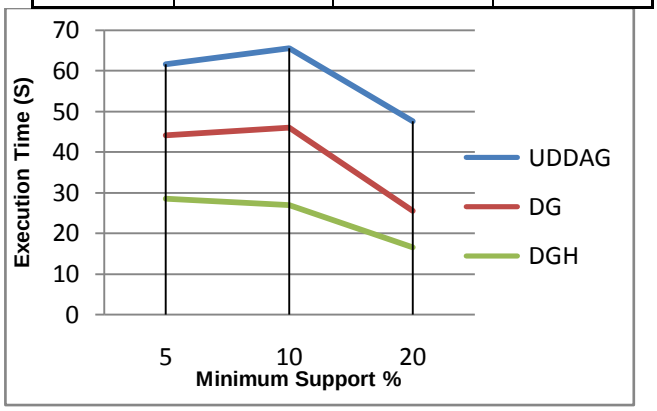

FIGURE 5 EXECUTION TIME COMPARISONS FOR UDDAG, DG AND DG ON HADOOP

Experiment 3 shows the performance for C800S8I8 dataset having 800k sequences, 1000 distinct items with min_sup 10%, 20% and 50% and the result is taken on 2 and 4 nodes. It is shown in figure 6.

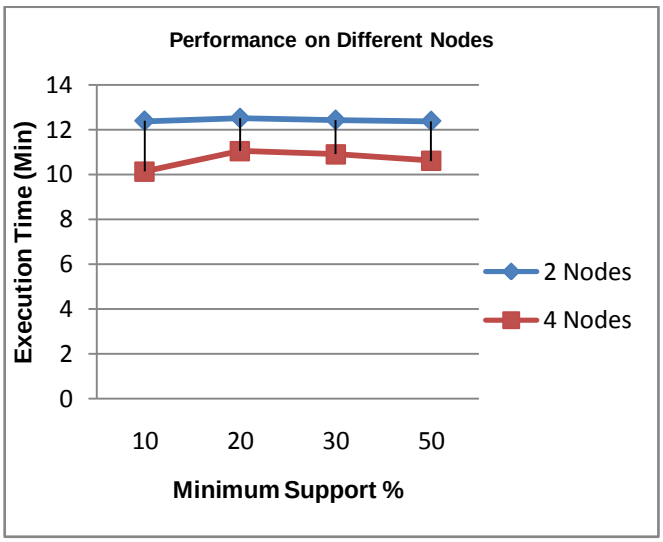

FIGURE 6 COMPARISONS ON DIFFERENT NODES

Experiment 4 shows the performance on different nodes like 2 and 4 for two different datasets as C800S8I8and C1000S8T5I8 with minimum support 50%. It is displayed in figure 7. By increasing the different number of nodes system gives better speed and scalability performance.

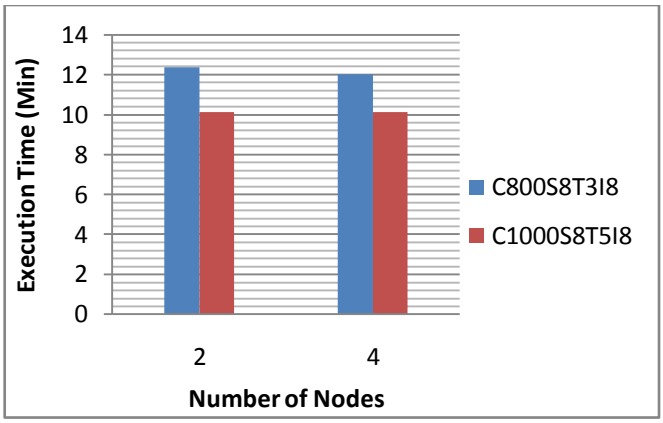

FIGURE 7 COMPARISONS ON DIFFERENT NODES WITH DIFFERENT DATASETS

VI. CONCLUSION AND FUTURE WORK

This paper studies many existing sequential pattern mining algorithm which are implemented on distributed environment like Grid, Cloud, Cluster and Hadoop MapReduce. New system is implemented in this paper which uses Hadoop MapReduce environment to find sequential patterns. DSPM first converts the sequence database into transformed database which is represented by directed graph as data structure to reduce memory storage and scanning time of database frequently. Because of distributed environment like Hadoop the scaling problem is improved in the new system. DSPM is compared with UDDAG on single system and it gives better performance using directed graph data structure.

DSPM with directed graph is tested on 2 as well as 4 nodes and it shows good performance. There are some challenging issues that need to be solved in future like finding constraint based , closed sequences sequential patterns on distributed environment.

REFERENCES

[1] R. Agrawal and R. Srikant,"Mining Sequential Patterns", Proc. of the 11th Int'l Conf. on Data Engineering (ICDE'95), 1995.

[2] Srikant R. and Agrawal R., "Mining sequential patterns: Generalizations and performance improvements", In Proceedings of the 5th International Conference Extending Database Technology, 1996, pp 1057, 3-17.

[3] J. Ayres, J. Gehrke, T. Yu, and J. Flannick, "Sequential Pattern Mining Using a Bitmap Representation", Proc. Int'l Conf. knowledge Discovery and Data Mining 2002, pp. 429-435, 2002.

[4] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining", Proc. of the 6th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD'00), 2000.

[5] Taoshen Li, Weina Wang, Qingfeng Chen, "On the Sequential Pattern Mining Algorithm Based on Projection position", The 8th International Conference on Computer Science & Education (ICCSE 2013) April 26-28, 2013. Colombo, Sri Lanka.

[6] ZakiM, "SPADE: an efficient algorithm for mining frequent sequences", Mach Learn, 2001.

[7] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Mei-Chun "Mining Sequential Patterns by Pattern – Growth: The PrefixSpan Approach", IEEE Transaction on Knowledge and Data Engineering, Vol. 16, No. 10, Oct 2004.

[8] Jinlin Chen, "An UpDown Directed Acyclic Graph Approach for Sequential Pattern Mining", IEEE Transaction on Knowledge and Data Engineering, Vol. 22, No. 7, Oct 2010.

[9] Chih-Hung Wu, Yu-Chieh Lo, "Mining Sequential Patterns on a Grid-Computing Environment", IEEE International Conference on Systems, Man, and Cybernetics October 8-11, 2006, Taipei, Taiwan.

[10] Chih-Hung Wu, Chih-Chin Lai, Yu-Chieh Lo, "An empirical study on mining sequential patterns in a grid computing environment", Expert Systems with Applications, 2012 5748–5757.

[11] Guralnik V, Karypis G.,"Parallel tree-projection-based sequence mining algorithms", Parallel Computing, 2004.

[12] Cong S, Han J, Padua D., "Parallel mining of closed sequential patterns", In Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, USA, 2005: 562-567.

[13] Wang J, Han J.,"BIDE: Efficient mining of frequent closed sequences", In: Proceedings of the 20th International Conference on Data Engineering. Boston, USA, 2004: 79-91.

[14] Shaochun Wu, Genfeng Wu, Shenjie Jin, "Pre-Clustering based Sequential Pattern mining", IEEE, 2004.

[15] J. Ayres, J. Gehrke, T. Yu, and J. Flannick, "Sequential Pattern Mining Using a Bitmap Representation," Proc. Int'l Conf. knowledge Discovery and Data Mining , pp. 429-435, 2002.

[16] Chun-Chieh Chen, Chi-Yao Tseng,Ming-Syan Chen, "Highly Scalable Sequential Pattern Mining Based on MapReduce Model on the Cloud", IEEE International Congress on Big Data, 2013.

[17] Wei Yong-qing, Liu Dong, Duan Lin-shan," Distributed PrefixSpan Algorithm Based on MapReduce", International Symposium on Information Technology in Medicine and Education, 2012.

[18] Kong-Fa-Hu, Chang-Hai Zhang, Ling Chen, "A Scalable Method of Mining Approximate Multidimensional Sequential Patterns on Distributed Systems", Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007.

[19] Apache Hadoop http://hadoop.apache.org/

[20] IBM Quest Synthetic Data Generator www.phillipe-fournier-viger.com/spmf/datasets/IBM_Quest_data_generator.zip

[21] S. Itkar, S. Shelke, "Sequential Pattern Mining and Distributed sequential pattern mining- A Survey and Future Scope",ICACCE-2014, PP 112-117.