Improved UP Growth Algorithm for Mining of High Utility Itemsets from Transactional Databases Based on Mapreduce Framework on Hadoop.

Vivek Jethe
Computer Department
MGM College of Engineering and Technology
Navi Mumbai, India
vivekj2612@gmail.com

Prof. Sachin Chavan
Computer Department
MGM College of Engineering and Technology
Navi Mumbai, India
ssschavan2003@gmail.com

Prof. Manoj Patil
Computer Department
Datta Meghe College of Engineering and Technology
Navi Mumbai, India
m_dpatil@yahoo.co.in

Abstract— **Now a days mining of high utility itemsets especially from the big transactional databases is required task to process many day to day operations in quick time. There are many methods presented for mining the high utility itemsets from large transactional datasets are subjected to some serious limitations such as performance of this methods needs to be investigated in low memory based systems for mining high utility itemsets from large transactional datasets and hence needs to address further as well. Another limitation is these proposed methods cannot overcome the screenings as well as overhead of null transactions; hence, performance degrades drastically. During this paper, we are presenting the new approach to overcome these limitations. We presented distributed programming model for mining business-oriented transactional datasets by using an improved MapReduce framework on Hadoop, which overcomes single processor and main memory-based computing, but also unexpectedly highly scalable in terms of increasing database size. We have used this approach with existing UP-Growth and UP-Growth+ with aim of improving their performances further. In experimental studies we will compare the performances of existing algorithms UP-Growth and UP-Growth+ against the improve UP-Growth and UP-Growth+ with Hadoop.**

Keywords- Dataset Mining, Hadoop, Itemsets, MapReduce Framework, Transactional Dataset, UP-Growth, UP-Growth+.

I. INTRODUCTION

Association rules mining (ARM) [1] is one of the most widely used techniques in data mining and knowledge discovery and has tremendous applications like business, science and other domains. Make the decisions about marketing activities such as, e.g., promotional pricing or product placements. A high utility itemset is defined as: A group of items in a transaction database is called itemset. There are two aspects in a transaction database: First one is itemset in a single transaction is called internal utility and second one is itemset in different transaction database is called external utility. The multiplication of external utility by the internal utility is the transaction utility of an itemset. By transaction utility, transaction weight utilizations (TWU) can be found. A utility is a high utility itemset only if its utility is not less than a user specified minimum support threshold utility value; otherwise itemset is treated as low utility itemset.

To generate these high utility itemsets mining recently in 2010, UP - Growth (Utility Pattern Growth) algorithm [2] was proposed by Vincent S. Tseng et al. for discovering high utility itemsets and a tree based data structure called UP - Tree (Utility Pattern tree) which efficiently maintains the information of transaction database related to the utility patterns. Four strategies (DGU, DGN, DLU, and DLN) used for efficient construction of UP - Tree and the processing in UP - Growth [11]. By applying these strategies, can not only

efficiently decrease the estimated utilities of the potential high utility itemsets (PHUI) but also effectively reduce the number of candidates. But more execution time for phase II (identify local utility itemsets) and I/O cost is taken by this algorithm.

Existing studies applied overestimated methods to facilitate the performance of utility mining. In these methods, potential high utility itemsets (PHUIs) are found first, and then an additional database scan is performed for identifying their utilities. However, a huge set of PHUIs are generated and their mining performance is degraded consequently by existing methods . When databases contain many long transactions or low thresholds are set, the situation may become worse. A challenging problem to the mining performance is the huge number of PHUIs since more the PHUIs the algorithm generates, the higher processing time it consumes.

To provide the efficient solution to mine the large transactional datasets, recently improved methods presented in [1]. In [1], authors presented propose two novel algorithms as well as a compact data structure for efficiently discovering high utility itemsets from transactional databases. Experimental results show that UP-Growth and UP-Growth+ outperform other algorithms substantially in terms of execution time. But these algorithms further needs to be extend so that system with less memory will also able to handle large datasets efficiently. The algorithms presented in [1] are practically implemented with memory 3.5 GB, but if memory size is 2 GB or below, the performance will again degrade in case of time. In this project we are presenting new approach which is extending these algorithms to overcome the limitations using the MapReduce framework on Hadoop.

II. LITERATURE SURVEY

R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," [3] as they discussed a well-known algorithms for mining association rules is Apriori, which is the pioneer for efficiently mining association rules from large databases.

J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," [4] as they discussed Pattern growth-based association rule mining algorithms [4], such as FP-Growth [4] were afterward proposed. It is widely recognized that FP-Growth achieves a better performance than Apriori-based algorithms since it finds frequent itemsets without generating any candidate itemset and scans database just twice.

Cai et al. and Tao et al. first proposed the concept of weighted items and weighted association rules [5]. However, since the framework of weighted association rules does not have downward closure property, mining performance cannot be improved. To address this problem, Tao et al. proposed the concept of weighted downward closure property [12]. By using transaction weight, weighted support can not only reflect the importance of an itemset but also maintain the downward closure property during the mining process.

Liu et al. proposed an algorithm named Two- Phase [8] which is mainly composed of two mining phases. In phase I, it employs an Apriori-based level-wise method to enumerate HTWUIs. Candidate itemsets with length k are generated from length k-1 HTWUIs, and their TWUs are computed by scanning the database once in each pass. After the above steps, the complete set of HTWUIs is collected in phase I. In phase II, HTWUIs that are high utility itemsets are identified with an additional database scan.

Li et al. [7] proposed an isolated items discarding strategy (IIDS) to reduce the number of candidates. By pruning isolated items during level-wise search, the number of candidate itemsets for HTWUIs in phase I can be reduced. However, this algorithm still scans database for several times and uses a candidate generation-and-test scheme to find high utility itemsets.

Ahmed et al. [13] proposed a tree-based algorithm, named IHUP. A tree based structure called IHUP-Tree is used to maintain the information about itemsets and their utilities. Each node of an IHUP-Tree consists of an item name, a TWU value and a support count. IHUP algorithm has three steps: 1) construction of IHUP-Tree, 2) generation of HTWUIs, and 3) identification of high utility itemsets. In step 1, items in transactions are rearranged in a fixed order such as lexicographic order, support descending order or TWU descending order. Then the rearranged transactions are inserted into an IHUP-Tree.

III. PROBLEM STATEMENT

In the literature we have studied the different methods proposed for high utility mining from large datasets. But all this methods frequently generate a huge set of PHUIs and their mining performance is degraded consequently. Further in case of long transactions in dataset or low thresholds are set, then this condition may become worst. The huge number of PHUIs forms a challenging problem to the mining performance since the more PHUIs the algorithm generates, the higher processing time it consumes. Thus to overcome this challenges the efficient algorithms presented recently in [1]. These methods in [1] outperform the state-of-the-art algorithms almost in all cases on both real and synthetic data set. However this approach in [1] is still needs to be improved in case of less memory based systems.

These methods are further needs to be improved over their limitations presented below:

- Performance of this methods needs to be investigated in low memory based systems for mining high utility itemsets from large transactional datasets and hence needs to address further as well.
- These proposed methods cannot overcome the screenings as well as overhead of null transactions; hence, performance degrades drastically.

IV. PROPOSED SOLUTION

The recently methods presented for mining the high utility itemsets from large transactional datasets are subjected to some serious limitations such as performance of this methods needs to be investigated in low memory based systems for mining high utility itemsets from large transactional datasets and hence needs to address further as well. Another limitation is these proposed methods cannot overcome the screenings as well as overhead of null transactions; hence, performance degrades drastically. In this project we are presenting the new approach to overcome these limitations. We presented distributed programming model for mining business-oriented transactional datasets by using an improved MapReduce framework on Hadoop, which overcomes not only the single processor and main memory-based computing, but also highly scalable in terms of increasing database size. We have used this approach with existing UP-Growth and UP-Growth+ with aim of improving their performances further. In experimental studies we will compare the performances of existing algorithms UP-Growth and UP-Growth+ against the improve UP-Growth and UP-Growth+ with Hadoop. Also In this paper, UP-Tree (Utility Pattern Tree) is adopted, which scans database only twice to obtain candidate items and manage them in an efficient data structured way. Applying UP-Tree to the UP-Growth takes more execution time. Hence this paper presents modified algorithm named as IUPG(Improved UP-Growth) aiming to reduce the execution time by effectively identifying high utility itemsets.

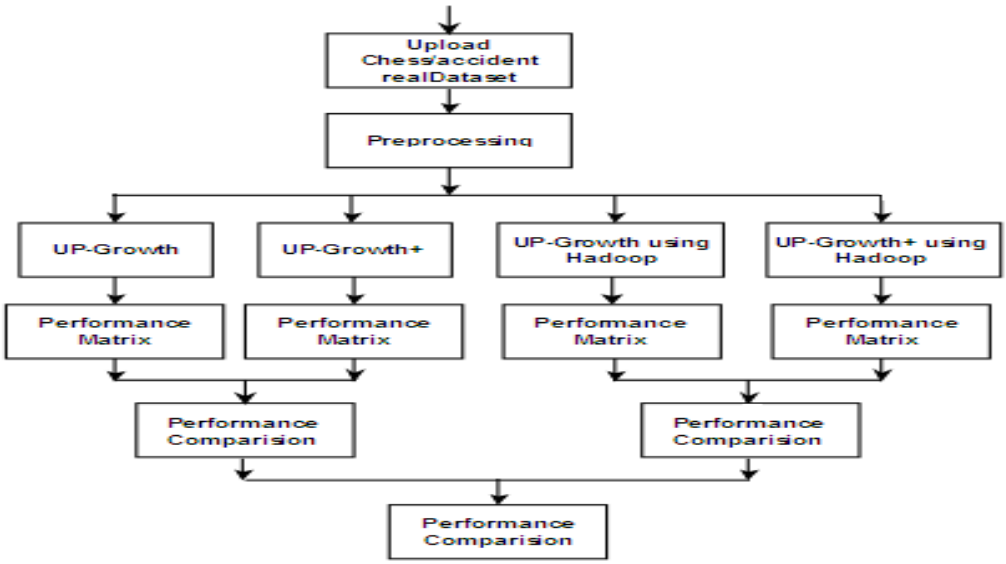

Figure 1: Flowchart of Performance comparison

A. The MapReduce Framework for Handling Big Datasets

Google's MapReduce [22] was first proposed in 2004 for massive parallel data analysis in shared-nothing clusters. Literature [23] evaluates the performance in Hadoop/HBase for Electroencephalogram (EEG) data and saw promising performance regarding latency and throughput. Karim et al. [24] proposed a Hadoop/MapReduce framework for mining maximal contiguous frequent patterns (which was first introduced at literature in RDBMS/single processor-main memory based computing) from the large DNA sequence dataset and showed outstanding performance in terms of throughput and scalability [21].

B. Hadoop Overview:

When data sets go beyond a single storage capacity, then distributing them to multiple independent computers becomes important. Trans-computer network storage file management system is called A distributed file system is Trans-computer network storage file management system . A typical Hadoop distributed file system contains thousands of servers, where each server stores partial data of file system.

C. MapReduce Overview:

In distributed data storage, when parallel processing the data, a lot should be considered, such as synchronization, concurrency, load balancing and other details of the underlying system which makes the simple calculation become very complex. MapReduce programming model was proposed in 2004 by the Google, which

is used in processing and generating large data sets implementation. Such a framework solves many problems, such as job scheduling,data distribution, fault tolerance, machine to machine communication, etc. MapReduce is applied in Google's Web search. Programmers are needed to write many programs for the specific purpose to deal with the massive data distributed and stored in the server cluster, such as crawled documents, web request logs, etc., which gives the results of different data, such as web document, inverted indices, different views, worms collected the number of pages for each host a summary of a given date within the collection of the most common queries and so on.

We use Two algorithms, named utility pattern growth (UPGrowth) and UP-Growth+, and a compact tree structure, called utility pattern tree (UP-Tree), for high utility itemsets discovery and to maintain important information related to utility patterns within databases.

UP-Growth algorithm

Input: UP-Tree T_x, Header Table HT_x, minimum utility threshold t, Item set $I = \{i_1, i_2, .., i_k\}$.

Process:

1. For each entry i_k in HT_x do
2. Trace links of each item. And calculate sum of node utilitynu_{sum}.
3. If $nu_{sum} \geq$ t
4. Generate Potential High Utility Itemset (PHUI)
   $Y = X \cup i_k$
5. Put Potential Utility of i_k as approximated utility of Y
6. Construct Conditional Pattern Based HT_Y.
7. Put local promising items intoHT_Y.
8. Apply Discarding Local Unpromising (DLU) to minimize path utilities of paths.
9. Apply DLU with $Insert\_Recognized\_Path$ to insert path intoT_Y.
10. If $T_Y \neq \emptyset$ then call to UP-Growth.
11. End if
12. End for.

Output: All PHUI's in T_x

UP-Growth+ algorithm

In UP-Growth, minimum item utility table is used to reduce the overestimated utilities. In UP-Growth+ algorithm we replace Discarding Local Unpromising (DLU) with Discarding Node Utility (DNU), DLN is replace with Decreasing local Node utilities for the nodes of local UP-Tree (DNN) and Insert_Recognized_Path is replace by ⟦Insert_Recognized_Path⟧_miu When a path is retrieved, minimal node utility of each node in the path is also retrieved in the data mining process . Thus, the minimum item utility can be simply replaced with minimal node utility as follows

Assume, p is the path in item $\{i_m\} - CPB$ and $UI_{\{i_m\}} - CPB$ is the set of unpromising items in$\{i_m\} - CPB$. The path utility of p in$\{i_m\} - CPB$, i.e.,$pu(p, \{i_m\} - CPB)$, $\{i_m\} - CPB$, is recalculated as:

$$pu(p, \{i_m\} - CPB) = p.\{i_m\}.nu - \sum_{\forall_i \in UI_{\{i_m\}} - CPB \wedge i \subseteq p} miu(i) \times p.count$$

Where $p.count$ is the support count of p in $\{i_m\} - CPB$.

Assume, a reorganized path $p = < N'_{i_1}, N'_{i_2}, \ldots, N'_{i_m} >$ in $\{i_m\} - CPB$ inserted into $< N'_{i_1}, N'_{i_2}, \ldots, N'_{i_m} >$ path in $\{i_m\} - tree$. Thus node utility of item node is recalculated as:

$$N_{ik}.nu_{new} = N_{ik}.nu_{old} + pu(p, \{i_m\} - CPB) - \sum_{j=k+1}^{m'} miu(i_j) \times p.count$$

Where $N_{ik}.nu_{old}$ is the node utility of N_{ik} in $\{i_m\} - tree$before adding p.

V. PRACTICAL RESULT AND ENVIRONMENT

In this section, we introduce practical environment and results.

A. *Input Dataset*

Following table 1 sows the dataset used for implementation. We compare both dataset to find accuracy by calculating precision and recall.

NO.	Dataset Name	Dataset Type
1.	Accident	Dense
2.	Cheese	Dense

Table 1: Experimental Input Dataset.

B. *Hardware and Software Used*

Hardware Requirement:

Processor	- Pentium –IV
Speed	- 1.1 Ghz
RAM	- 256 MB(min)
Hard Disk	- 20 GB
Key Board	- Standard Windows Keyboard
Monitor	- SVGA

Software Requirement:

Operating System	- Windows XP/7/8
Programming Language	- Java
Tool	- Eclipse, Hadoop

C. *Metrics Computed*

Results are computed as referred in [1] and performance is compared using hadoop.

D. *Results of Practical Work*

Practical work done is as shown in figure given below.

Following figure shows output of dataset transactions implementation.

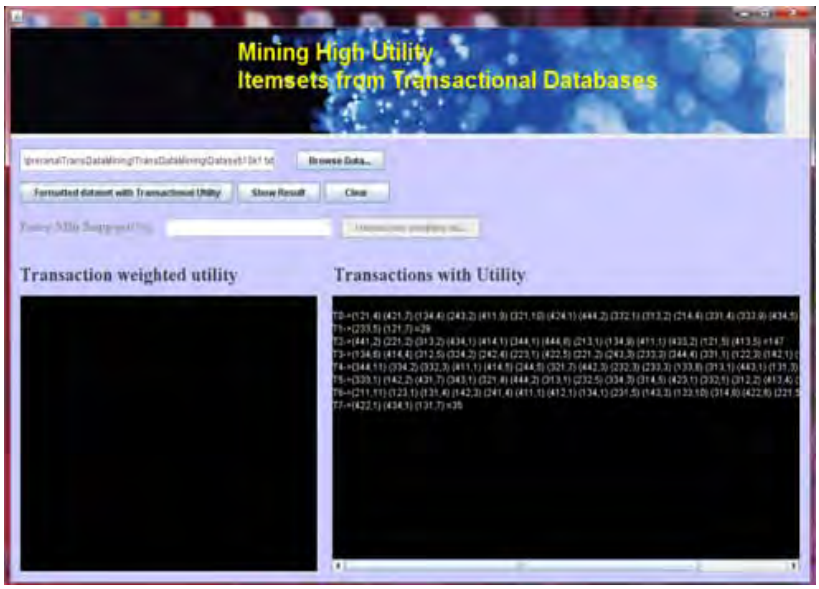

Figure 2: Output screen 1.

Following figure 3 shows the output screen for transaction weighted utility and transaction utility implementation.

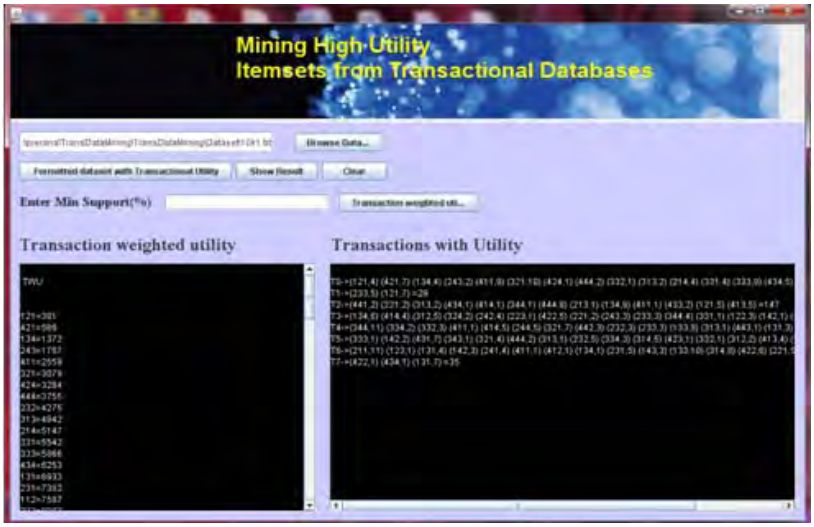

Figure 3: Output Screen 2.

VI. WORK DONE

Use either SI (MKS) or CGS as primary units. (SI units are strongly encouraged.) English units can also be used as secondary units (in parentheses). This also applies to papers in data storage. For example, write "15 Gb/cm2 (100 Gb/in2)." there could be an exception when English units are used as identifiers in trade, such as "3½-in disk drive." Avoid combining SI and CGS units, such as magnetic field in oersteds. Equations do not balance dimensionally as, this often leads to confusion. If you must use mixed units, the units for each quantity in an equation should be clearly stated.

The SI unit for magnetic field strength H is A/m. However, if you want to use units of T, either refer to magnetic flux density B or magnetic field strength symbolized as µ0H. Use the center dot to separate compound units, e.g., "A•m2."

VII. CONCLUSION

In this paper we have presented new enhanced framework of two recently presented algorithms namely UP-Growth and UP-Growth+ with aim of improving the processing time performance and mining performance under the less system memory environment as well. We have used the concept of Hadoop with MapReduce Framework along with these two algorithms. The proposed system block diagram is depicted in this paper with the details of using Hadoop Framework. In the results section, we presented the work done so far over this proposed approach with the datasets used. In the future completely evaluate this proposed architecture and compare its performance against existing methods in order to claim the effectiveness and efficiency of this proposed network.

ACKNOWLEDGEMENT

I would like to take this opportunity to thank MGM CET and Principal Dr. S.K Naranyankhedkar, MGMCET for giving me an opportunity to carry out my seminar work. I express my deep sense of gratitude towards Prof. V. Balachandran, Head of the Department of Computer for his valuable guidance and encouragement. I wish to acknowledge my extreme gratitude to my guide Prof. Manoj Patil for guiding me throughout the work on project. I also like to thank Prof. Sachin Chavan for his guidance.

REFERENCES

[1] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu, Fellow, IEEE, "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 25, NO. 8, AUGUST 2013.

[2] V incent . S. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu.: UP – Growth : An Efficient Algorithm for High Utility Itemset Mining. In Proc. of ACM-KDD, Washington, DC, USA, pp. 253- 262, July 25– 28, 2010.

[3] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th Int'l Conf. Very Large Data Bases (VLDB), pp. 487-499, 1994.

[4] C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases," IEEE Trans. Knowledge and Data Eng., vol. 21, no. 12, pp. 1708-1721, Dec. 2009.

[5] C.H. Cai, A.W.C. Fu, C.H. Cheng, and W.W. Kwong, "Mining Association Rules with Weighted Items," Proc. Int'l Database Eng. and Applications Symp. (IDEAS '98), pp. 68-77, 1998.

[6] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proc. ACM-SIGMOD Int'l Conf. Management of Data, pp. 1-12, 2000.

[7] S.C. Lee, J. Paik, J. Ok, I. Song, and U.M. Kim, "Efficient Mining of User Behaviors by Temporal Mobile Access Patterns," Int'l J. Computer Science Security, vol. 7, no. 2, pp. 285-291, 2007.

[8] H.F. Li, H.Y. Huang, Y.C. Chen, Y.J. Liu, and S.Y. Lee, "Fast and Memory Efficient Mining of High Utility Itemsets in Data Streams," Proc. IEEE Eighth Int'l Conf. on Data Mining, pp. 881- 886, 2008.

[9] Y.-C. Li, J.-S. Yeh, and C.-C. Chang, "Isolated Items Discarding Strategy for Discovering High Utility Itemsets," Data and Knowledge Eng., vol. 64, no. 1, pp. 198-217, Jan. 2008.
[10] C.H. Lin, D.Y. Chiu, Y.H. Wu, and A.L.P. Chen, "Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window," Proc. SIAM Int'l Conf. Data Mining (SDM '05), 2005.
[11] Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Itemsets Mining Algorithm," Proc. Utility-Based Data Mining Workshop, 2005.
[12] F. Tao, F. Murtagh, and M. Farid, "Weighted Association Rule Mining Using Weighted Support and Significance Framework," Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '03), pp. 661-666, 2003.
[13] J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," Proc. 21th Int'l Conf. Very Large Data Bases, pp. 420-431, Sept. 1995.
[14] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proc. ACM-SIGMOD Int'l Conf. Management of Data, pp. 1-12, 2000.
[15] S.C. Lee, J. Paik, J. Ok, I. Song, and U.M. Kim, "Efficient Mining of User Behaviors by Temporal Mobile Access Patterns," Int'l J. Computer Science Security, vol. 7, no. 2, pp. 285-291, 2007.
[16] H.F. Li, H.Y. Huang, Y.C. Chen, Y.J. Liu, and S.Y. Lee, "Fast and Memory Efficient Mining of High Utility Itemsets in Data Streams," Proc. IEEE Eighth Int'l Conf. on Data Mining, pp. 881-886, 2008.
[17] Y.-C. Li, J.-S. Yeh, and C.-C. Chang, "Isolated Items Discarding Strategy for Discovering High Utility Itemsets," Data and Knowledge Eng., vol. 64, no. 1, pp. 198-217, Jan. 2008.
[18] C.H. Lin, D.Y. Chiu, Y.H. Wu, and A.L.P. Chen, "Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window," Proc. SIAM Int'l Conf. Data Mining (SDM '05), 2005.
[19] Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Itemsets Mining Algorithm," Proc. Utility-Based Data Mining Workshop, 2005.
[20] R. Martinez, N. Pasquier, and C. Pasquier, "GenMiner: Mining nonredundant Association Rules from Integrated Gene Expression Data and Annotations," Bioinformatics, vol. 24, pp. 2643-2644, 2008.
[21] Md. Rezaul Karim1, Chowdhury Farhan Ahmed2, Byeong-Soo Jeong1, Ho-Jin Choi3, "An Efficient Distributed Programming Model for Mining Useful Patterns in Big Datasets", IETE TECHNICAL REVIEW | Vol 30 | ISSUE 1 | JAN-FEB 2013.
[22] J. Dean, and S. Ghemawa, "MapReduce: Simplified Data Processing on Large Clusters," OSDI, 2004.
[23] H. Dutta, and J. Demme, "Distributed Storage of Large Scale Multidimensional EEG Data using Hadoop/HBase," Grid and Cloud Database Management, New York City: Springer; 2011.
[24] M.R. Karim, B.S. Jeong, and H.J. Choi, "A MapReduce Framework for Mining Maximal Contiguous Frequent Patterns in Large DNA Sequence Datasets", IETE Technical Review, Vol. 29, no. 2, pp. 162-8, Mar-Apr, 2012.