# Mining Data Stream for Load Shedding

RujutaVartak
*Department of Computer Engineering, Mumbai University, Shree L.R. Tiwari College of Engineering and Technology, Mira Road, India*
*rujutavvartak@gmail.com*

Dr.MadhuriRao
*Department of Information Technology, Mumbai University, Thadomal College of Engineering and Technology,Mumbai, India*
*my_rao@yahoo.com*

Prof. Jayamalini k
*Department of Computer Engineering, Mumbai University, Shree L.R. Tiwari College of Engineering and Technology ,Mira Road, India*
*jayamalini.k@slrtce.in*

***Abstract-*** Data stream is continuous flow of data, which necessitates load shedding for data stream processing system. Here we study overload handling for frequent pattern mining indata streams. Here in this paper load shedding use frequent pattern matching algorithm i.e priority, transaction and attribute in overload situation.
The heavy workload or continues stream of the mining algorithm lies mostly in the great deal of item sets, which need to be enumerated and by the mining algorithm. Therefore, our proposed scheme involves the maintenance of a smaller set of item sets, so the workload can be lessened accordingly.
***Keywords:*** *Data Stream, Frequent pattern mining, Load Shedding.*

*****

## I. INTRODUCTION

In many modern applications such as global retail chains, data is continuously streamed from each endpoint into the local analysis system. The characteristic of such data is that it is unchained in terms of continuity of data generation. This form of data has been termed as data stream[3].

Data streamlining is the method of extracting information structure of continuous, rapid data records. It is the sequential order of instances that seems in applications (e.g., sensor data, network traffic, click stream) of DataStream management [1].These applications read the instances only once or a small number of times using limited computing and storage capabilities. Here the goal is to predict the class(or) valuesof the new instances in the data stream resulted in desired information about the class.

Discovery of knowledge in the data-stream environment is an attractive yet challenging task, because data in a data stream is usually in finite in volume. Many traditional mining techniques such as scanning the entire data over and over(to produce mining answers) become in feasible in this case[6]. Asaresult, the methodologyto data stream mining differs considerably from database mining.

Nowadays, data stream is used, which is continuous, dynamic and has an enormous amount of data. The mining system, may sometimes be in capable of processing the arriving data promptly within a time unit. Such a situation is known as data overload[8]. Basically, when the transmission rate of data stream gets beyond the throughput (i.e., data processing rate)of the mining system, data overload happens
And causes some problems unavoidably. For example, the system always processes on early received data (and newly received data is queued up).More seriously, the buffer of the system may be filled up and newly incoming data gets lost. As a result, the technique of load management or overload handling has become an important and necessary issue in the area of data stream mining[7].

## II. RELATED WORKS

The concepts of Symmetric Real-time streams and Asymmetric Real-time streams are introduced [1].The system model described the RST capacities are derived for Symmetric Real-time streams and Asymmetric Real-time streams respectively, which can be used to guide the design of efficient real-time stream coding approaches. And show the Real-time stream transmission capacities of Symmetric Real-time streams and Asymmetric Real-time streams are asymptotically achievable by Proportional Time-invariant Intra-session Code and Maximum Time-invariant Intra-session Code respectively.

Demonstrated that graph-oriented approach is more efficient for load-shedding semantic data stream by RDF stream processing systems [2]. Indeed, this approach allows to improve the quality of queries results of these systems by protecting the semantics and consistency of the semantic data streams.

Proposed load adaptation techniques are designed considering various conditions of nodes and tasks [3], and classified as inter-node load balancing, inter-task load balancing, increasing processing capability, and finally load shedding for the last way to avoid entire system halt due to the data explosion.

Heuristics Miner algorithm is proposed, called StrProM [4], builds prefix–trees to extract sequential patterns of events from the stream. This algorithm uses batch based approach to continuously update and prune these prefix -trees. Effective Ensemble on Cloud with Load balancing method [5], to overcome the load shedding ,overhead on reprocessing the already processed item and peak loading.

Proposed an efficient algorithm DSM-Miner for mining maximal frequent patterns over data streams [6]. SWM-Tree can effectively maintain the latest pattern's information that users need in data stream. Also perform mining maximal frequent and appropriate pruning operations.

Using Lossy counting algorithm load shedding schema is proposed for mining system [7]. The load shedding schema is compatible with the mining algorithm on the same data

structure and the cost of switch between both of them is found to be small. By running the scheme (in place of its own mining algorithm) when necessary, the mining system can obtain an increase in its throughput and thus protect itself from the overload.

Using Loadstar dynamically adjusts to changes in available system resources [8]. This proves that an intelligent load shedding scheme can make a stream management system more robust to external disturbances.

Proposed an algorithm called INSTANT, based on operators on itemsets, intended to discover frequent itemsets from data streams [9]. INSTANT is a single-phase mining algorithm with a solid theoretical foundation. In comparison with traditional two-phase data mining approaches, which store meta-patterns and re-mine internal data structures to finalize mining results, INSTANT can produce the mining results in an instant way with one data scan only. INSTANThas good time and space efficiencies both withincreases of the data stream scale and decreases of the minimum support value.

The Count Sketch algorithm with the Sampling algorithm is used [10]. So far, neglected the space cost of actually storing the elements from the stream.This is because different encodings can yield very different space use. Count Sketch algorithm achieves better space bounds for many natural distributions on the item frequencies.

Novel algorithms for computing approximate frequency counts of elements in a data stream [11]. Solve iceberg queries, frequent item sets, association rules, and packet flow identification

## III. Proposed System

The proposed mining method is entirely based on a minimum-support count calculated with respect to system load for frequent pattern matching to discard the items from the item sets hence, shedding the system load. The discarded item sets are considered again in existing system.

Architecture of Load shedding for frequent pattern Mining

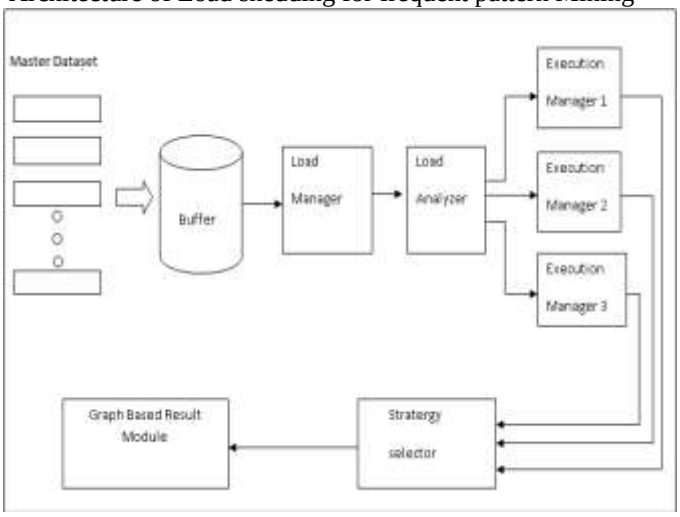


Fig.1 Architecture of load shedding system

### A. Load Manager

The functions of this module are to monitor the incoming stream, calculate the maximum item sets the system can process to make it 100% busy. Depending on maximum item sets, minimum support count is suggested (which can be changed). Load Manager acquires JSON file as an input stream. The data from the file is stored in the buffer till the buffer is full. Considering, the system cannot process all the transactions at a time, the system is given the transaction one after the other depending on the window size. The window size is kept dynamic and it changes as per the system load, at that instance of time.
The system load is calculated for three mining algorithms, namely, transaction based, priority based and attribute based by providing same no of transactions to all three. Then it calculates the average of system load obtained for these mining algorithms. To ensure the accuracy of calculated system load, the average system load is compared with system load calculated previously. If they are not same, then this step is repeated until the accuracy is attained.

Since, we are using mining algorithm, minimum support count needs to be calculated. The minimum support count depends on the system load. The load Manager calculates the minimum support count. Let the resulted minimum support count be ‘x’ and it indicates that the system would retain the maximum ‘x’ transactions. Users may use the support count recommended by the system or he may select the minimum support count as per his/her requirements. As per the value provided by the user for minimum support count, the load monitoring module selects the transactions and passes to the load analyzer module.

### B. Load Analyzer

This module helps the user in selecting the strategy means execution manager (transaction based, attribute based or priority based mining algorithms) for retaining maximum transactions, which are to be processed by the system. To compare the various strategies, the parameters such as precision value, execution time, processing time and percentage of transactions retained are used. To get the values of these parameters, we will use a JSON file having 10,000 transactions. The load analyzer calculates the value of these parameters for the above mentioned three strategies at the system load ranging between 100% and 200%.

### C. Strategy Selector

The main job of this module is shed the load of the system. This module sheds the load using stream mining process. The values obtained for all the four parameters for three strategies (mining algorithms) are compared. This helps the user in deciding the appropriate strategy as per the system load.

### D. Algorithm - Attribute Based Strategy

Start
Step 1: The set of transactions are read from the JSON file.
Step 2: These transactions are stored in the buffer

Step 3: The calculated values of minimum support count and system load is given to the pre-analyser module as an input.
Step 4:Depending on the minimum support count, the pre-analyser module executes the attribute based strategy.
Step 5: stores set of item set in array (L).
Step 6:All the selected item sets are kept in array (C) and frequently occurring items are kept in frequent item sets.
Step 7: Repeat Steps 5 and 6.
Step8: After one transaction is processed, the next transaction is taken in L.
Step9: the selected transactions are arranged in descending order of their hit count.

Step 10:Now, the N number of transactions are sent to stream mining module.
Step11: The pruned transactions are sent back to the buffer in load monitoring module for next processing cycle.
Step12: Finally, the precision value and percentage of transaction retained are calculated.
End

### *E. Algorithm - Priority Based Strategy*

The load shedding can be also done on the base of priority. Priority based system prunes the item sets on the base of their priority assigned. Here, the priority is assigned using first come first serve basis.
Start
Step 1: The data stream is taken as an input by load monitoring module from JSON file.
Step 2:Depending on the transactions present in the buffer, the system load, minimum support count is calculated by load monitoring module.
Step3: The first transaction is selected.
Step4:It is stored in an array (L) and the incoming transactions are added to it.
Step5: The selected transactions are checked and the count of reoccurring items are stored as a support count.
Step 6:If(minimum support < support count)
then
a) Remove all the item from the L, which does not satisfy the minimum support count
b) These pruned items are sent back to buffer for next processing cycle
else
a) Sort the items in L in decreasing order of support count
b) These sorted items stored in L
step7: Increment the hit count for reoccurrence of item set and record the maximum hits.
step 8: Calculate percentile hit.
step9: Assign priority to selected items using first come first serve basis.
step10: The highest priority transactions are send for processing.
step11:Send these pruned transactions to the buffer.
step12:The retained transactions are sent to the next module.
Step 13: stepcalculates precision value.

### *F. Algorithm - Transaction Based Strategy*

The third criteria in proposed scheme is transaction based, which works on the simple concept of the pruning the longest transactions, .i.e., if the longest transaction is of 10 item sets and the system can process only 8, then we discard the last 2 items from the item sets, allowing the others to process.
Start
Step1: Json data stream is given as input to load monitoring module and stored in the buffer.
Step 2: calculates the system load.
Step3: Depending on the system load, a set of transaction is sent to strategy selector module. This set of transaction is stored in an array (L).
Step 4: The support count of all the reoccurring items are calculated for transaction based strategy by strategy selector module.
Step 5: Repeat step 4
Step 6: Items in L are sorted in increasing order of support count.
Step7: Create duplicate of all the item in the item sets. The clone of these item sets are created for sending to the buffer. The cloning operation is carried out for those item sets, which are pruned.
Step8: Send these pruned items back to buffer for next processing cycle.
Step9: The load monitoring module checks the system load. Depending on the system load, the module checks, if the system can process more transactions, then got to step 4, else stop.
Step 10: calculates the precision value.
End.

## IV. Implementation and Experimental Results

In this section proposed mining system to examine the effectiveness of load controlling function. Json file used to generate test dataset.
Load shedding is not required when system load is below 100%. When system load is greater than 100% then these three strategies are applied.

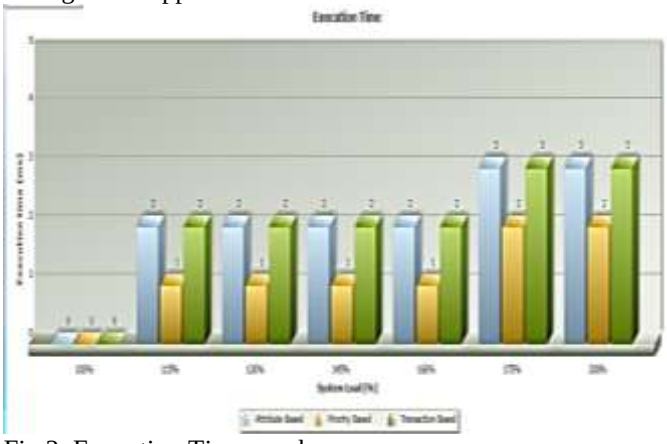

Fig.2: Execution Time graph

The fig.2 analyses that Execution time verses System load. The x-axis contain execution time ranging from 0 ms to 5 ms and y-axis contain 100% to 200% system load. From the result priority strategy contain less time compare to other two strategy.

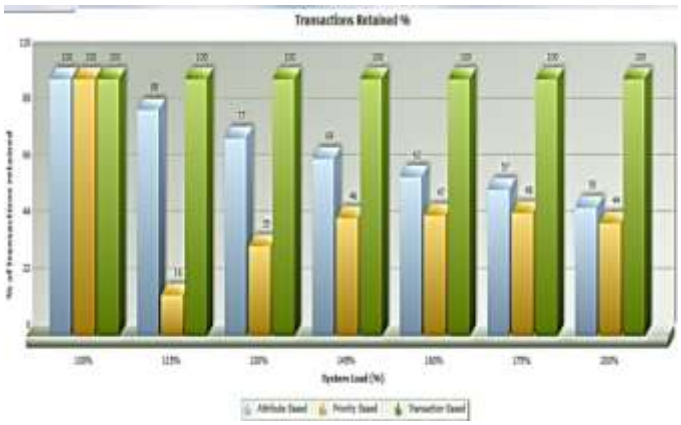


Fig.3: Transaction Retained graph

This graph represent Transaction retained verses system load. In fig.3 shows that transaction based strategy contain retain value is 100% in all system load. Therefore, transaction strategy is selected when the user want to retain all the transaction.

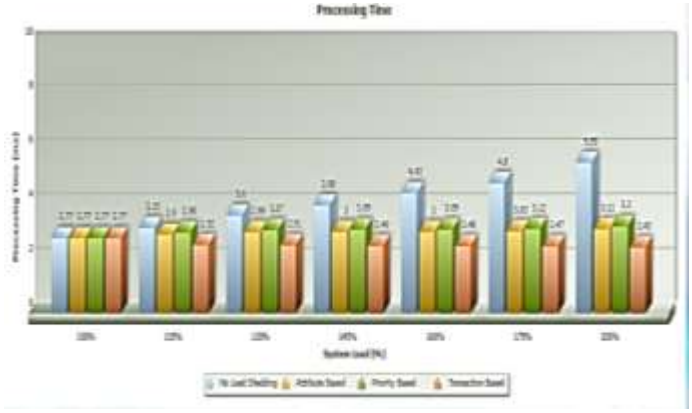


Fig. 4: Processing time graph

In this fig.4 , compare all processing time in between all three strategies and no load shedding. Result show that processing time of no load shedding is greater than other three strategies.

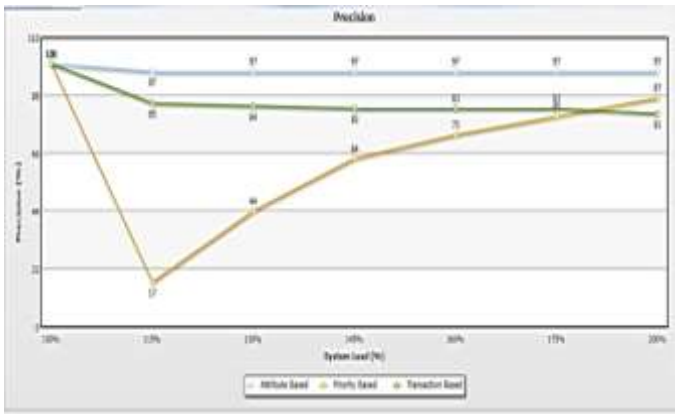


Fig. 5: Precision graph

In this last figure calculate the precision value. The precision value is ratio of number of item sets retained by the mining algorithm to the number of item sets system can process, making in 100% busy.

## V. CONCLUSION

The load-shedding schemes can indeed lighten the workload of an overloaded system as well as preserve the mining accuracy at an acceptable level. User can make choice depending on the mining and load administrable is proposed. Moreover, three load shedding strategies are constructed and unified in the system based on the various concepts. The pruned data are later send back to the buffer and are mined by the strategies.

## VI. REFERENCES

[1] Yunkai Wei, Yuming Mao, SupengLengand Tracey Ho, “Priority-based Real-time Stream Coding under Burst Erasures” ,IEEE Global Communication Conference,(San Diego, CA), December 2015.

[2] FethiBelghaouti, AmelBouzeghoub, ZakiaKaziAoul,RajaChiky, “Graph-Oriented Load-Shedding For Semantic Data Stream Processing”,Computational Intelligence for Multimedia Understanding (IWCIM), October 2015.

[3] Myungcheol Lee, Miyoung Lee, Sung Jin Hur, Ikkyun Kim, “Load Adaptive Distributed Stream Processing System for Explosive Stream Data”, ICACT ,pp.753-757, 2015.

[4] Marwan Hassani, Sergio Siccha, Florian Richterand Thomas Seidl, “Efficient Process Discovery From Event Streams Using Sequential Pattern Mining” IEEE Computational Intelligence, IEEE Symposium Series ,pp.1366-1373,2015.

[5] JagadheeswaranKathirvel, Dr. ElangoParasuraman, “Effective Data Stream Mining using Ensemble on Cloud with Load balancing (E2CL)” , International Conference on Computing and Communications Technologies IEEE, pp .383-386, 2015.

[6] Junrui Yang , Yanjun Wei , Fenfen Zhou, “An efficient algorithm for mining maximal frequent patterns over data streams” 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, IEEE, pp.444-447,2015.

[7] Kuen-Fang Jea, Chao-Wei Li, Chih-Wei Hsu, Ru-Ping Lin, Ssu-Fan Yen, “A Load Shedding Scheme for Frequent Pattern Mining in Transactional Data Streams” , IEEE Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp.1294-1299,2011.

[8] Y. Chi, H. Wang, and P. S. Yu, “Loadstar: Load Shedding in Data Stream Mining,” Proceedings of the 31st VLDB Conference, Trondheim, Norway, pp.1302-1305, 2005.

[9] G Mao, X Wu, X Zhu, et al, “Mining maximal frequent itemsets form data streams,” Journal of Information Science, vol. 33(3), pp. 251-262, 2007.

[10] M. Charikar, K. Chen, and M. Farach-Colton, “Finding Frequent Items in Data Streams,” Proceedings of the 29th ICALP Colloquium, Malaga, Spain, pp. 693-703, 2002.

[11] G. S.Manku and R. Motwani, “Approximate Frequency Counts over Data Streams,” Proceedings of the 28th VLDB Conference, Hong Kong, pp. 346-357, China, 2002.