Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 45 (2015) 118 – 126

Procedia Computer Science

International Conference on Advanced Computing Technologies and Applications (ICACTA-2015)

A Deviant Load Shedding System for Data Stream Mining

Darshana Desai*,Dr. Abhijit Joshi

Student,Dwarkadas J Sanghvi College of Engineering,Mumbai, India

Vice Principal, Dwarkadas J Sanghvi College of Engineering, Mumbai, India

Abstract

Load shedding is imperative for data stream processing systems in numerous functions as data streams are susceptible to sudden spikes in volume. The proposed system is an attempt to seek and resolve four major problems associated with data stream, which include load shedding and anti-shedding time, number of transactions pruned and selecting predicate; using efficient mining system. The frequent pattern discovered in data stream used in the model exploits the synergy between scheduling and load shedding. This paper also proposes various load shedding strategies which reduce and lighten the workload of the system ensuring an acceptable level of mining accuracy using various parameters like transaction, priority and attributes of data mining. A majority chunk of workload in mining algorithm lies in the innumerable item sets, which are counted and enumerated. The approach is based on the frequent pattern matching principle of stream mining which involves reducing the workload to maintain smaller item sets.

Peer-review under responsibility of scientific committee of International Conference on Advanced Computing Technologies and Applications (ICACTA-2015).

*Keywords:*Data stream; Frequent pattern matching; Data overload; Load shedding.

* Corresponding author. Tel.: 919769862553; fax: +0-000-000-0000 .

*E-mail address:*ddarshu09@gmail.com, abhijit.joshi@djscoe.org

Peer-review under responsibility of scientific committee of International Conference on Advanced Computing Technologies and Applications (ICACTA-2015).
doi:10.1016/j.procs.2015.03.103

1. Introduction

Data streams have currently become a particular data type that raise enough consideration from the research community, feeding from entirety in data stream querying [18], clustering [7], classification [16] to frequent sets mining [8]. For many stream applications such as sensor networks, financial services, web tracking and personalization, the data arrival rate is fast and often uncertain. The evaluated results of these streams mostly crave delivery under time restraints to implement optimal outcomes.

In this unique setting, classical methods [9] that deal with ardently stored datasets become feeble in streaming situations. It is therefore vital to map new techniques that have the ability to interpret the comment in an online fashion on the streaming data while performing under the resource constraints e.g., CPU cycles, memory space, and bandwidth communication.

Consider, the scenario where, chunk of the data stream boosting unfortunately, which outstrips the system capacity to overburden position and the system may not be able to cart the results within the given time frame [1]. Accordingly, the quality of service can be debased without bound. To confront with such situations, one may add more resources to handle the increased load or distribute the calculations amongst multiple nodes which are costly and may be infeasible in installation. The other option may be, to prefer dropping raw data to curtail workload and continue the time lines of output results. Much work focuses on use of resources in a best effort way so that performance does not reduce. For instance, if the data aspect from a sensor exhibits a predictable trend, then the precision rules might be satisfied by transmitting only a fraction of the sensor data to the remote server [2].

In accordance with the scenario cited, it is observed that data stream possess following characteristics, i.e., the amount of data is not bounded with steep arrival rate entered and begins in adjacent to real time. Acknowledging these issues, it is difficult to evolve flexible algorithms that support the alteration of stream data in one-pass manner with constraints on system resources.

The rest of this paper is formulated as follows. In section 2, analogous work regarding data stream frequent pattern mining and load shedding is covered. Section 3, describes the architecture for load shedding system for data stream mining. In section 4, presents implementation and experimental result. Finally, ends with conclusion.

2. Related Work

This section describes the study of different methods presently being used for Load shedding on data stream mining.

For developing a design for mining the data stream, two conditions are needed. In the foremost condition, an algorithm which would process data in a data stream in accordance to three window models which are landmark window, sliding window and damped window, which defines the range of mining [3]. The second conditions is the processing time on each unit of stream data should be as precise as viable; otherwise the mining system would not be able to knob the stream data in time and causes the overload problem.

Window models are used to retrieve stored data from buffer. The characteristic of the preceding three window models are as follows [2, 10]. The landmark window model considers a time point is declared called as landmark, and the mining is done on the data among the landmark and the present time. The sliding window model has a non-changing size window is stated which slides in accordance of time, and the mining closures are always spawned based on the data within the present window. The damped window model, the mining is accomplished on data in a window in which present data is more crucial than earlier data.

Data overloading problem occurs as the data stream is endless, changing and has huge amount of data, because of which mining system process the incoming data properly within a given time i.e., when the transmission rate of data stream gets beyond data processing rate of the mining system, causes some problems like undesired gain etc. The most serious is that the buffer of the system will be brimmed with incoming data and this data gets adrift. As a result, the approach of overload handling becomes an extensive and basic for mining the data stream [2].

To address the overload issue in data stream mining, most algorithms described, works on finite memory space [11, 12, 13, 14] i.e., they avoid the case that CPU is also a bounded source. It is critical to characterize that the main complication in frequent set mining is the huge number of item sets whose frequencies need to be followed. Given a transaction of size *n*, the number of item sets is exponentially proportional to its size; i.e., fora transaction of size *n*= 5, the number of item sets may be up to 2^5-1. This certainly makes the frequent set mining problem impressionable to overloading.

Zhang and et al., analyses that employed with the overload conduction for frequent-pattern mining in live data streams, is to deals with the frequent item sets which has to be calculated and casted by the mining[3]. Therefore, load shedding scheme is associated by the preservation of a smaller set of item sets, so the workload can be conical.

The issue of dropping a few data in order to confront with high agility data streams is often known as load shedding. Presently, this problem has been intensively considered in data stream querying [3, 15]. However, it has not been well-forwarded in the context of data streamlining [1], exclusively for the complication of finding frequent sets from transactional data streams.

Charikar and et. al., mentioned that Frequent-pattern mining in data streams initially focuses on single instance and predefine notations items and lossy count [4]. Lossy Counting is the basic method that uses €-deficient mining methods on frequent item sets from transactional data streams having only one instance and user specified minimum support count.

Keeping in mind the various issues associated with exiting techniques a load adjustable mining system for determining frequent patterns in data streams is proposed. The proposed mining method 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. Using various mining algorithms namely, apriori, least frequently used, FP – tree etc., four distinct schemes for load shedding are arranged and unified in the system, proving it efficient for shedding the workload during peak times.

3. Proposed System

Consider, data stream over Y is ceaseless sequence of elements where each item set is a transaction. Let I={x1, x2 ... xz} be a set of items from the source of data stream, which is continuously entering the local system. An item set is the part of I and written as Y = xi, xj …, xm. . The threshold of minimum support (st) is calculated in accordance with system load, which is used by frequent pattern mining algorithm for searching of the item sets whose numbers of occurrences are above st each time. An item set is a candidate item if its count is not known but its items are accepted to be rapid.

The amount of the data stream is noted as the count of transmitted item sets during a time unit, which may differ in respect of time. In accordance to threshold, the size of the window, ws is calculated by the system. The motive of frequent pattern mining in data streams is to search from the stream data, the item sets whose occurrence are more than st * ws every time.

Consider, the A amount of data components of the data stream to be transmitted in unit of time and B is processing rate of mining on the stream elements in unit time. The system load is stated as the ratio of transmission period to processing period at given point, i.e., A/B but the value of A exceed B then the system gets busier with processing the continuously incoming data and overloading situation is encountered when the transmission amount of data stream becomes more than the processing rate of mining method (i.e. A>B) [2]. So, it is necessary to calculate the maximum transactions the system can process, making it 100% busy. Depending on this calculation, the system should allow only that number of transactions and shed the rest of them, so that the desired output is achieved and avoiding overloading issue.

3.1 Architecture of Deviant Load Shedding System for Data Stream Mining

Based on system load condition mentioned above, the deviant load shedding model will reduce the amount of data being processed to coordinate the system capacity. Now, let us see the framework of deviant load shedding system shown in figure 1.

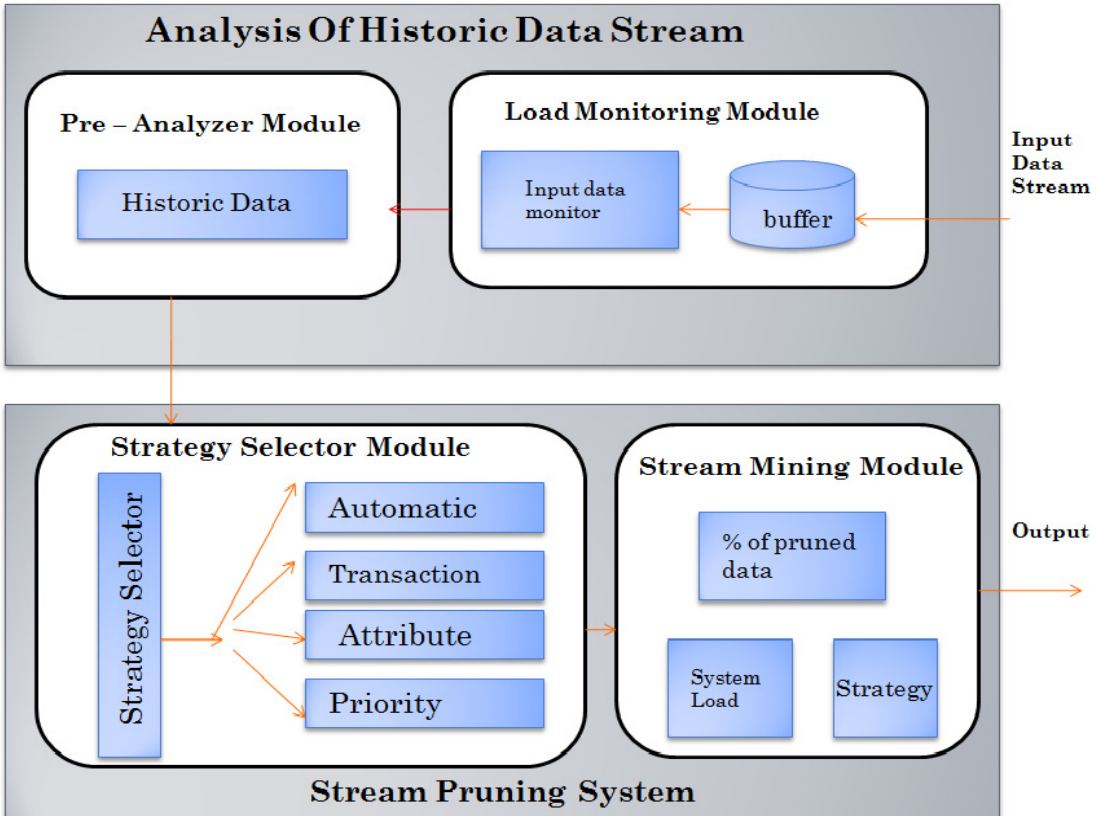

Fig.1. Architecture of Load Shedding System for Data stream Mining

The architecture is basically divided into two main sub- systems i.e. Analysis of Historic Stream and Stream Pruning System.

3.1.1 Analysis of Historic stream

This system basically deals with the calculations of system load considering various parameters (on historic data) and minimum support count depending on the system load. This helps the user to select the strategy for processing on live data stream as per requirements of processing system. This system is divided into two sub- modules – load monitoring module and pre- analyser module.

a) Load Monitoring Module

This module is basically deals with monitoring the input data stream to sending it to next system. It first takes the data stream in the form of JSON file, sends it to the buffer and then using sliding window protocol, it is input to pre- analyzer module. Since, module has to deal with monitoring of system load every time the stream enters, the stream has to go through different phases which are as follows.

The phase starts with the data stream which is taken in the form of JSON file (test data). The input file can also be XML type but it requires more processing complexity with respect to time resulting in more CPU usage as compared to JSON file. Since, data stream are busty with explicit in timestamp, it is difficult to update the stream continuously, hence, buffering of stream is required. But, due to limitation in the resources available, the buffered stream has to be processed continuously with calculation of system load. Keeping in mind these requirements, we have divided the load monitoring module into sub-modules which are as follows.

1) *Buffer* : The JSON file which is input data stream has to be stored in buffer as it needs to be updated with time before it is given for processing. But the size of the buffer for any system is limited. Implementation of the buffer should be such that updating of elements should be fast and easy, which can be done either by using linked list or dynamic array. We have used linked list, as insertion or deletion of the elements is much easier and faster in it as compared to dynamic array and it is the only data structure in JAVA which can be implemented as a queue or stack (i.e. FIFO or LIFO). Considering, the data stream being dynamic in nature, the system can only process some item sets depending on the systemload. So, it is necessary to calculate the load of the system every time the stream enters to avoid crashing of the system due to overload. Hence the input monitoring module is required to send the amount of data that the system can process at any instance of time.

2) *Input Monitoring Module* : This module deals with controlling of the data send to system depending on the load on it at particular instance of time. Keeping this in mind and after the study of the different window models, sliding window protocol can be used. In sliding window protocol, the sliding time of fixed length window can be changed. Since, we have to control the input data to the system, which can done by changing the sliding time of the window every time with respect to the system load.

After controlling the incoming data stream, now we have to check which of the items from the item sets are required and which have to be pruned, depending on the system load. This can be done on the bases of minimum support count as we are using frequent pattern matching algorithm. The threshold of minimum support count is calculated depending on maximum transactions the system can process making the system completely busy. Minimum support count (st) is on which the transaction is pruned, i.e., if the occurrence of the item set is less than ‘st’, they are discarded. Considering the minimum support count calculated, the user has to select the strategy. The selection of the strategy depends on processing system, performance of the algorithm at specific system load and considering minimum item sets are pruned which are discussed in the next module.

b) Pre- Analyzer Module

This module helps the user in selection of strategy as it gives performance measures for all the three strategy i.e. transaction based, attribute based and priority based. The performance measures such as execution time, percentage of data pruned, processing time and precision value of each strategy are calculated using historic data, i.e., on 10,000 item sets at system load ranging from 100 to 200 percentage. It is basically a comparative study of the strategies on which helps user in selecting the appropriate strategy depending on the need. The performance measures which are taken into consideration are as follows.

I.*Execution time:* The time taken by the algorithm to prune items from the item sets depending on the system load. It is basically the total time for the execution of algorithm i.e. finish time after calling the mining algorithm start time before calling the mining algorithm.

II. *Percentage of items pruned (%):* It gives the percentage of number of item set retained by the mining algorithms with respect to system load. Percentage of items pruned is required to know how items are removed from the original item set depending on the system load.

III. *Processing time:* The time taken by the algorithm to process all items in an item set with respect to system load at that instance. This measure is required to know how much time the algorithm takes to process one item set (assuming one ticket takes 10 millisecond for persistency).

IV. *Precision value:* It defines the accuracy in matching obtained by the algorithm. It is ratio of number of item sets retained by the mining algorithm to the number of item sets the system can process, making it 100% busy.

Depending on the value of these performance measure, the user would be able to select any of the three strategy. After comparing performance value obtained for each strategy, a dynamic or automatic strategy is suggested. This strategy dynamically selects one of the strategy from transaction based, attribute based and priority based, depending on system load. Considering the performance measures for the different strategies, the user selects on of them depending on the need.

3.1.2 Stream Pruning System

This system gives option for the selecting strategy depending on the processing system and requirements. This system basically gives the pruning results depending on user selected strategy and system load. This system is further divided into two modules, i.e., strategy selector module and load mining module.

*(i) Strategy Selector Module***:** This module allows the user to select one of the strategies given, i.e., Transaction, Attribute, Priority or Automatic. Depending on the strategy selected the output is generated by next module i.e., load mining module. The working of the strategies is explained below.

a) Transaction-Based: Among the accepted and raw item sets, the individual items are discarded by the transaction based algorithm, from the individual item set depending on the system load i.e., if the system can allow 10 items from an item set, which is of 12 items, the transaction based would remove the 2 items from the item set .

b) Attribute-based: In this case, items within the item sets are mined and then are ordered by mining algorithm based on mining results. This process discards those items in an item sets having the lowest frequencies. These discarded items are lined back in the buffer and therefore, handled accordingly by the mining system with respect to rate of processing depending on system load.

c) Priority-based: A priority is assigned to each incoming item. So, from the accepted item sets, the item set with lowest priorities are elected to be pruned, so the remaining count of item sets can be taken care by the mining system on system load.

Based on the three basic schemes defined above, a system generated default scheme is suggested for user.

d) Automatic: In this case, the systems dynamically selects the strategy on the basis of analysis given by pre-analyzer module analysis(see table 1).The pre-analysis module computes various values, such as precision, processing time, execution time and percentage of item sets pruned by varying the system load for all three strategies.

Table 1.System selector condition – Automatic

System Load	Strategy to be selected	Justification
0 to 100%	No Pruning needed	System can handle all the transactions
100 to 130 %	Transaction Based	Execution takes slightly more time, however all the transactions are retained.
130 to 160%	Attribute Based	Processing time is less as compared to transaction based, but some transaction will be pruned.
160 % to infinite	Priority Based	Processing time is least but more transaction are pruned as compared to above two strategies

ii)Load Mining Module: The strategy selected by the user is given to this module, which mines the data item set depending on minimum support count, system load, maximum transaction the system can process i.e. system is completely busy and the precision value. The mining output also depends on system configuration. The mining output is given with respect to system load, percentage of data pruned, number of transaction present in buffer and strategy selected (i.e., in case of automatic strategy).

4. Implementation and Experimental Results

In this section, the examination of the proposed mining system is conducted using an analysis to evaluate the effectiveness for controlling load function. Personal computer platform was used to carry out experiment which has an Intel core i5-3230M@2.60GHz. The operating system is Windows 7 Ultimate Service Pack 1, and the programs of the mining and the load shedding schemes are implemented in Java EE (compiled and deployed on Apache Tomcat 7file Server).

The test dataset is generated from a JSON file which is prepared using a JSON generator script. The file has 10,000 transactions of the average length of 10 items each. This item set is employed to simulate the mining. For minimum requirement, minimum support is set to 0.1.

The maximum processing rate of the system on the input data is first calculated(W) taking into consideration various factors like system load, system performance etc. Once this is calculated, it will be persisted into the application. If more than a transactions (say B) arrive in a unit time, the system is overloaded.

The experiment was carried many times on the input data, by varying the system load from 100% (normal) to 200% (overloaded), and observed the efficiency and mining accuracy of the four load-shedding schemes.

When the system load is below 100%, load shedding is not required, and approaching item sets are precisely processed. The data stream module is used when the system load is above 100%, a load shedding system is applied. Observations shows that the proposed load shedding system are all efficacious in shedding the system load. Also if the load-shedding process is effective, the total execution time required by the system for frequent pattern mining will be lessen than that the time required by mining algorithm on the arriving data

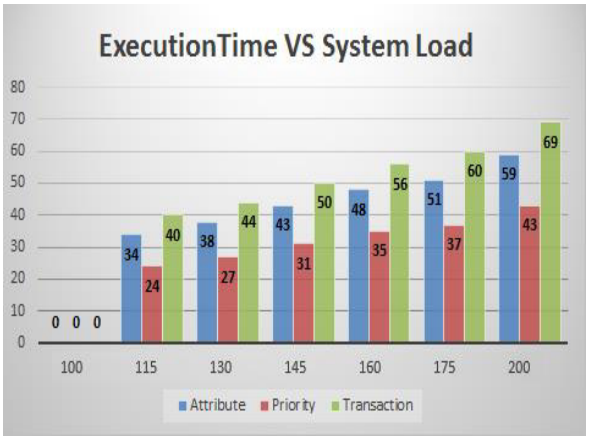

Fig.2. Execution Time vs. System Load

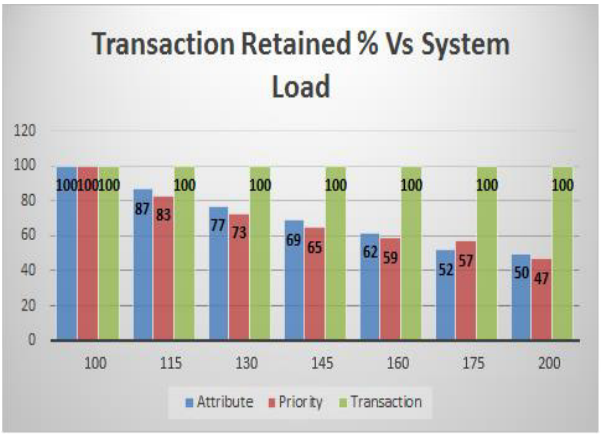

Fig.3. Transaction Retained vs. System Load

The first part of experiment, analyses the execution time required by load shedding algorithm. The x-axis specifies the execution time and on y-axis specifies the system load ranging from 100% to 200%. The chart shows in figure 2 the execution time each algorithm takes to execute the item set at that specific system load. From the result it is evident that, the priority-based strategy is most adequate as it can curtail the execution time for the system for excess loaded system to a great extent. The attribute based and transaction-based strategies, on the other side, although less significant as compared to priority based system.

In the second part of experiment, checks the result of mining accuracy taking transaction retained (%) vs. system load, i.e., number of transactions retained by each algorithm of load shedding schemes is checked. The transaction-based strategy retains the entire transaction set as it removes the individual item sets from within the transaction. The attribute-based and priority-based have low retention rate comparatively. The experimental values are presented in Figure 3.

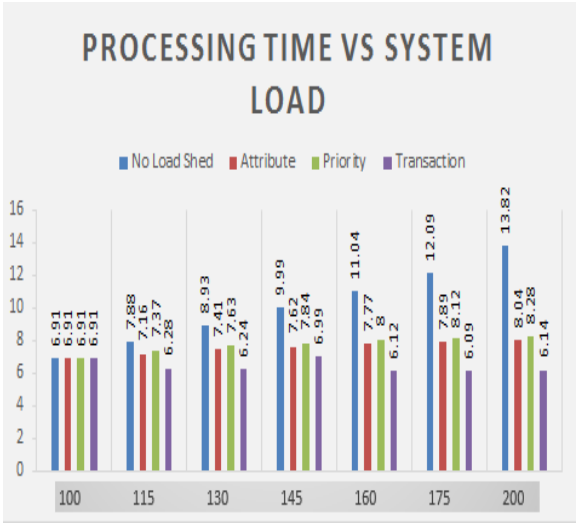

Fig.4. Processing Time vs System Load

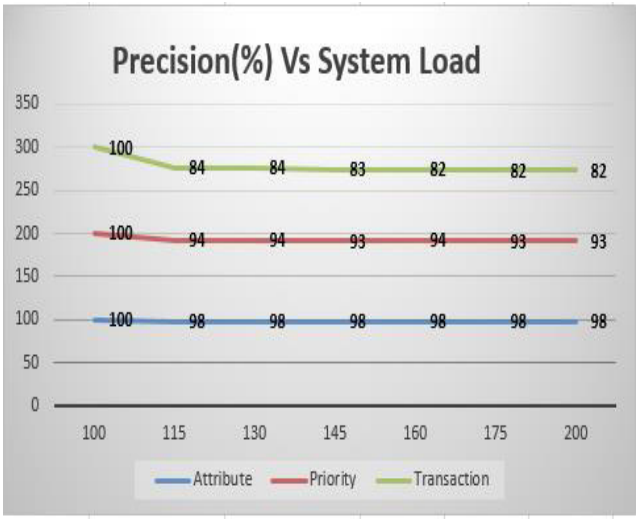

Fig.5. Precision (%) vs System Load

In the third part, figure 4. , experiment compares the processing time vs. system load i.e. ,assuming that the transactions will need at most 10 ms of time for processing. The graph shows that the processing time required for pruning item in the item set pruned taken by priority based strategy and attribute based needs the almost same while items pruned by transaction-based strategy needs more time.

In the fourth part carries the experiment of the precision values for strategies in accordance with the system load is shown in figure 5. The precision value is calculated on number of the transaction selected by the mining algorithm by the number of the transactions the system can process. Accordingly, the precision value of attribute value is high as compared to other two schemes as the mining is done on the attribute based pattern matching.

From the experimental results it is found that, among the three shedding strategies, the priority based scheme is most obvious in condensing the system load taking least amount of time, while the transaction-based scheme best preserves all the transactions though it needs some extra time to process the transactions.

These results are used to devise the election of strategies for the loaded system. The priority based strategy is endorsed as it needs significantly less time to prune data so, if the user requirement is to lessen the excessive load. On the other hand, if the quality of mining is required, then the attribute-based strategy is proposed to both reduce the system load and mine with high accuracy. This scheme takes slightly more time than priority-based strategy, but

it gives accurate results. Transaction-based mining can be selected when the user wants to retain all the transactions, but there is no requirement to process and persistence of individual data items within the transactions, i.e., the item sets within the transactions can be dropped permanently.

The experimental results show that shedding of load is a last effort made in accordance to restrict crashing of mining system.

5. Conclusion

In this investigation, frequent pattern matching algorithm is used to reduce to system load is studied. User can make a choice depending on the mining and load administrable is proposed. Moreover, four devoted load shedding schemes, which are transaction-based, attribute-based, priority-based and automatic are constructed and unified in the system based on the various concepts. An appropriate strategy is elected which are accepted to shed the excess amount of data every time when the system is too much loaded, based on the user requirement and system load. From the analyses result, it found that the load shedding strategies can indeed reduce the workload of a system as well as conserve the mining certainty at an adequate level. The pruned data are later send back to the buffer and are mined by the strategies, for processing of maximum incoming data. Hence, the proposed the load shedding strategies can process changing stream using mining algorithm accuracy.

References

1. Chao-Wei Li,Kuen-Fang Jea,Chih-Wei Hsu,Ru-Ping Lin,Ssu-Fan Yen,"A load shedding scheme for frequent pattern mining in transactional data streams",8thInternational Conference on Fuzzy System and knowledge Discover,Vol.2,pp. 1294- 1299, 2011.
2. Kuen-Fang Jea,Chao-Wei Li ; Chih-Wei Hsu ; Ru-Ping Lin,“A load-controllable mining system for frequent-pattern discovery in dynamic data streams”,Machine Learning and Cybernetics (ICMLC), 2010 International Conference on (Volume:5) , ISBN no. 978-1-4244-6526-2.
3. B.Babcock, M.Datar, R.Motwani,"Load shedding for aggregation queries over data streams", 20th International Conference on Data Engineering, pp.: 350-361, 2004.
4. PrashantLahane, R K Bedi, Prasad Halgaonkar,” Data Stream Mining”, International Journal of Advances in Computing and Information Researches, Vol 1(1), 2012.
5. 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.
6. YunyiZhang ,Deyun Zhang ; Chongzheng Huang .:A Novel Adaptive Load Shedding Scheme for Data Stream Processing. In: IEEE, Future Generation Communication and Networking sFGCN) Vol (1), pp. 378 - 384, 2007.
7. Garofalakis, M., Gehrke, J. &Rastogi, R. (2002), Querying and mining data streams: you only get one look a tutorial, in `ACM SIGMOD International Conference on Management of Data'.
8. Guha, S., Meyerson, A., Mishra, N., Motwani, R. & O'Callaghan, L. (2003), Clustering Data Streams: Theory and Practice, in `IEEE Transactions on Knowledge and Data Engineering', pp. 515-528.
9. Manku, G.S. &Motwani, R. (2002), Approximate Frequency Counts over Data Streams, in `VLDB International Conference on Very Large Databases', pp. 346-357.
10. Charu C. Aggarwal, 2001,’Data Stream: Models and Algorithms’, NY 10598.
11. Lin, C., Chiu, D., Wu, Y. & Chen, A. ,2005, Mining Frequent Item sets from Data Streams with a Time-Sensitive Sliding Window, in `SIAM International Conference on Data Mining', pp. 68-79.
12. Giannella, C., Han, J., Pei, J., Yan, X. & Yu, P.S.,2003 , Mining Frequent Patterns in Data Streams at Multiple Time Granularities, Next Generation Data Mining AAAI/MIT.
13. Chang, J.H. & Lee, W.S., 2003, finding recent frequent item sets adaptively over online data streams, in `ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 487-492.
14. Teng, W.G., Chen, M.S., & Yu, P.S. (2003), A Regression-Based Temporal Pattern Mining Scheme for Data Streams, in `VLDB International Conference on Very Large Data Bases', pp. 93-104.
15. Tatbul, N. &Zdonik, S.B., 2006, Window-Aware Load Shedding for Aggregation Queries over Data Streams, in `VLDB International Conference on Very Large Data Bases', pp. 799-810.
16. Hulten, G., Spencer, L. &Domingos, P. (2001),Mining Time-Changing Data Streams, *in* `ACMSIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 97-106.
17. ‘Data Stream Management Issues – A Survey' Lukasz Golab M. Tamer Ozsu ¨,Technical Report CS-2003-08April 2003
18. Garofalakis, M., Gehrke, J. &Rastogi, R. (2002),Querying and mining data streams, in `ACM SIGMOD International Conference on Managemen