MAIDS: Mining Alarming Incidents from Data Streams
(Demonstration Proposal) *

Y. Dora Cai[§] David Clutter[§] Greg Pape[§] Jiawei Han[†] Michael Welge[§] Loretta Auvil[§]

[§] Automated Learning Group, NCSA, University of Illinois at Urbana-Champaign, U.S.A.
[†] Department of Computer Science, University of Illinois at Urbana-Champaign, U.S.A.

ABSTRACT

Real-time surveillance systems, network and telecommunication systems, and other dynamic processes often generate tremendous (potentially infinite) volume of stream data. Effective analysis of such stream data poses great challenges to database and data mining researchers, due to its unique features, such as single-scan algorithm, multi-dimensional online analysis, fast response time, etc.

In this paper we propose to demonstrate our stream data mining system, MAIDS—Mining Alarming Incidents from Data Streams, which is a project supported by U.S. Office of Naval Research and National Science Foundation, jointly developed by Automated Learning Group, NCSA and Department of Computer Science, the University of Illinois at Urbana-Champaign. By integration of our most recent research results on stream data analysis, we have successfully developed the MAIDS system within the D2K data mining framework [19] with the following distinct features: (1) a tilted time window framework and multi-resolution model, (2) a stream "data cube" for multi-dimensional analysis, (3) online stream classification, (4) online frequent pattern mining, (5) online clustering of data streams, and (6) stream mining visualization. These stream data mining functions, working together, can successfully mine alarming incidents from data streams on the fly. The system will be demonstrated using network intrusion and sensor stream data sets.

1. INTRODUCTION

Owing to the rapid progress of computer and electronic technologies, vast amount of stream data have been collected and made available for data analysis. There are many applications that require handling data in the form of data streams, such as sensor data, network traffic flow, time-series data, stock exchange, telecommunication, Web clicking streams, weather or environment monitoring, and so on. As indicated by many researchers [2, 3, 9, 11, 10, 5], data streams are different from the finite, static data sets stored in flat files or in database systems; they are in high volume, potentially infinite, dynamically changing, and requires fast response. Moreover, a lot of stream data resides at the primitive abstraction level, and it is necessary to perform *multi-dimensional analysis* on such data to find interesting patterns at appropriate levels of abstraction and with appropriate dimension combinations. These unique characteristics have posted a great challenge for data analysis on data steams.

With years of research into this area, many efficient algorithms have been developed [2, 4, 6, 7] and many prototype systems for stream query processing have been built, such as STREAM, Cougar, Aurora, Hancock, Niagara, OpenCQ, Telegraph, Tradebot, Tribeca, etc. (see an overview in [2]). However, based on the best of our knowledge, there is no existing stream data mining system or system prototype that can integrate multi-dimensional OLAP stream data analysis with long-term multi-resolution stream book-keeping and perform multiple stream data mining functions, including stream classification, clustering, frequent pattern analysis, and so on.

In this demo proposal, we will describe our recent research and development of the MAIDS system, which integrates multi-dimensional OLAP stream analysis with data mining and provides the following major stream analysis functions: (1) a tilted time window framework and multi-resolution model, (2) a stream "data cube" for multi-dimensional analysis, (3) online stream classification, (4) online frequent pattern mining, (5) online clustering of data streams, and (6) stream mining visualization. The functionality, efficiency and effectiveness of the MAIDS system will be demonstrated using network intrusion and sensor stream data sets to show that the system can successfully mine alarming incidents from data streams.

The remaining of the paper is organized as follows. Section 2 will present the MAIDS system architecture. Section 3 will describe the major functional components of the system. And Section 4 will outline our demonstration plan and

* The work was supported in part by U.S. Office of Naval Research, U.S. National Science Foundation NSF IIS-02-09199 and NSF-IIS-0308215, the University of Illinois, and an IBM Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

conclude our proposal.

2. SYSTEM ARCHITECTURE

The architecture of MAIDS is shown in Figure 1. On the top are the incoming data streams from various applications that produce data streams indefinitely. After data preprocessing, such as data formatting, normalization, and transformation, the data streams are simultaneously sent to the following four modules:(1) *Stream Query Engine*, (2) *Stream Data Classifier*, (3) *Stream Pattern Finder*, and (4) *Stream Cluster Analyzer*, which will generate query results, classification models, frequent patterns, and data clusters, respectively, and these results will be output to users in various forms including graphs and charts accomplished by *Stream Mining Visualizer*.

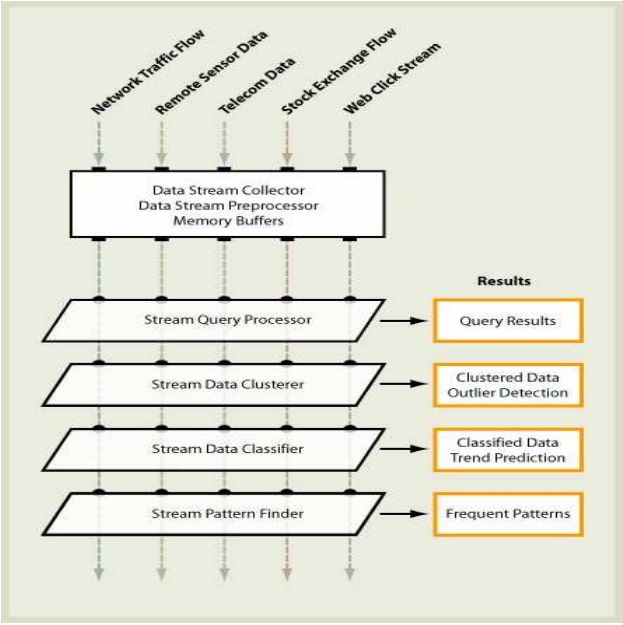

Figure 1: MAIDS Architecture

There are several unique features of the MAIDS system.

First, the system adopts a flexible *tilted time window* framework throughout all of the functional modules. The idea of *tilted time window* has been seen in quite a few studies in stream data analysis [2, 4]. It is proposed based on the following philosophy. In stream data analysis, both of the recent data and historical data are required, but the recent data are usually more important than the historical data. People are often interested in recent changes at a fine scale, but long term changes at a coarse scale. Naturally, time can be registered at different levels of granularity. The most recent time can be registered at the finest granularity; the more distant time can be registered at the coarser granularity; and the level of coarseness depends on the application requirements. By aggregating data to a coarser granularity for the distant time period, we can substantially save storage space and computation time, but still preserve important information.

The titled time window, first designed in our research in [4], and later further developed in [1], proposed a flexible framework for a multi-resolution model. Based on the applications, it may adopt a few models, including (1) *natural titled time window*, (2) *logarithmic titled time window*, and (3) *pyramidal titled time window* [4, 1]. The current implementation of MAIDS adopts the *natural titled time window*, that uses the natural time to configure the time granularity, such as second, minute, quarter, hour, day, etc. However, being implemented in the object-oriented programming methodology, this tilted time window module can be changed easily into other models based on the application requirements.

The *natural titled time window* is implemented using circular queues. Each queue is for a specific time granularity. For example, we may use six time granularities: minute, quarter, hour, day, month, and year to store aggregates from the finest level to the coarsest level. The *titled time window* is self-maintained automatically. Whenever reaching the boundary of a time granularity, the aggregates stored in a finer granularity level are summarized and propagated to a coarser granularity level. For example, when time reaches 9:15AM, all counts accumulated during 9:01AM–9:15AM would be summarized and propagated to the first quarter slot in the quarter granularity level. This technique has substantially compressed the data without loosing important information, and thus has made possible the long-run analysis on the data stream, because it dramatically reduces the processing requirement on memory and CPU.

Second, it facilitates multi-dimensional analysis using a stream cube architecture [4] which will be detailed in the next section. This architecture ensures that online analytical processing can be performed on stream data in a similar way as OLAP in data cubes.

Third, the architecture facilitates the integration of multiple stream mining functions into one platform so that multiple mining functions can cooperate to discover patterns and alarming incidents in real time.

MAIDS is a general-purpose tool for data stream analysis and is designed to process high rate and multi-dimensional stream data. MAIDS can interface to data streams generated by various types of devices. It can be used in many applications, such as network intrusion detection, telecommunication data flow analysis, credit card flaw prevention, Web click streams analysis, financial data trend prediction, and so on. MAIDS is built on the D2K (Data to Knowledge) framework [19] and has become a new component of D2K. D2K is developed by Automated Learning Group (http://alg.ncsa.uiuc.edu), NCSA, in the University of Illinois at Urbana-Champaign. D2K is a rapid, flexible data mining and machine learning system that integrates analytical data mining methods for prediction, discovery, and deviation detection, with data and information visualization tools. It offers a visual programming environment that allows users to connect programming modules together to build data mining applications and supplies a core set of modules, application templates, and a standard API for software component development. All D2K components are written in Java for maximum flexibility and portability.

3. MAJOR FUNCTIONAL COMPONENTS

The MAIDS system consists of the following systems components: (1) *Stream Query Engine*, (2) *Stream Data Classifier*, (3) *Stream Pattern Finder*, (4) *Stream Cluster Analyzer*, and (5) *Stream Mining Visualizer*. Each of these functional modules will be outlined in this section.

3.1 Stream Query Engine

Stream Query Engine serves as a powerful stream query processor that supports many query options, including *single-dimensional* vs. *multi-dimensional* queries, *ad-hoc* vs. *continuous* queries, *drill-down* vs. *roll-up* OLAP queries, *point* vs. *duration time* queries, and *exact* vs. *approximate* queries. The query result can be either presented in a report format or visualized in a chart or a graph. Figure 2 shows the visualization of a three-hour continuous query that requests the dimensions *Protocol Type* and *Time* for the network traffic stream data. The *Time* dimension has been presented in the *tilted time window*. Three time granularities (minute, quarter and hour) can be easily identified by three spectrums of colors.

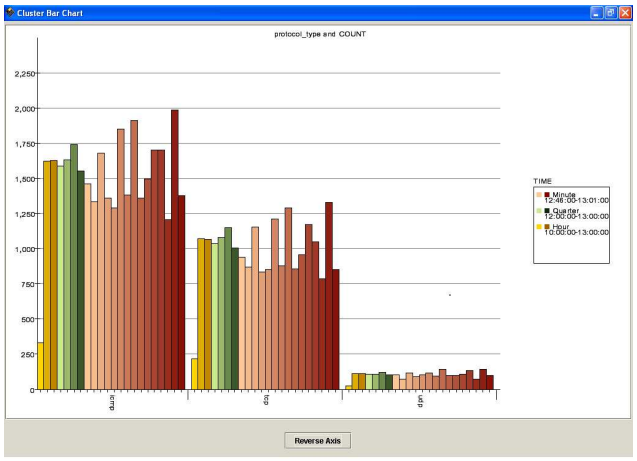

Figure 2: Query With Time Dimension

There are several novel features in the design and implementation of *Stream Query Engine*: (1) It implements a largely virtual, dynamically constructed and incrementally updated *stream data cube* using the H-tree data structure described in [13, 4], and the various kinds of stream queries are processed using this cube structure; (2) due to the nature of huge volume in data streams, it is unrealistic and unnecessary to construct and materialize a full cube. Thus only two layers of the cube: (i) m-layer *(Minimum Interest Layer)*, and (ii) o-layer *(Observation Layer)*, and the internal nodes of the tree along the popular querying path between these two layers are materialized; (3) queries that fall outside of the popular querying path are answered by minimal computation on target cells from those reachable at the run-time; and (4) each tree node in the H-Tree stores a *tilted time window* that tracks the aggregates (count, sum, max, min) in different time slots for the node.

3.2 Stream Data Classifier

Stream Data Classifier constructs classification models dynamically based on the current as well as historical stream data collected in the *tilted time window*. The model construction follows the philosophy of the data classification. In particular, we have integrated Naive Bayes algorithm [17, 20] with the new features of stream data analysis. A user-friendly interface has been built in this module that can display the list of models and the predicted alarming events. Figure 3 is a screen capture of *Stream Data Classifier*. The list of models generated at various time is shown on the left. The variable statistics for the selected model is displayed as pie charts in the center. And the conclusion probability for the selected variables is presented on the right.

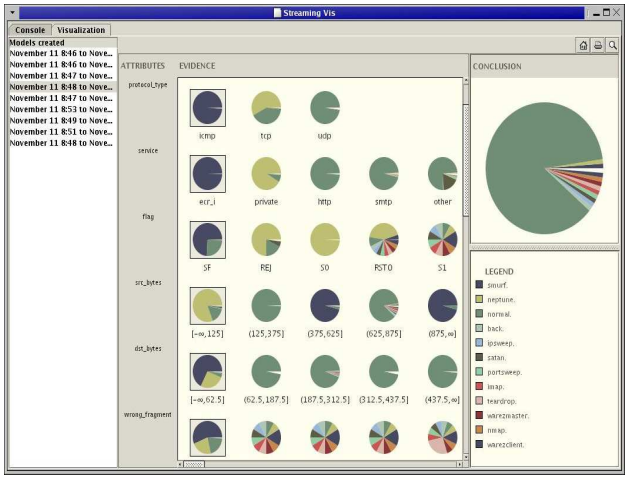

Figure 3: Stream Classification Models

Several distinct features can be identified in our design and implementation of *Stream Data Classifier*, as illustrated below: (1) An efficient data structure, called *AVC-list* [12] (i.e., Attribute-Value-Class_Label list), has been constructed dynamically and maintained incrementally to accumulate single variable statistics for the Naive Bayesian classifier, and a *tilted time window* is associated with the each node in the AVC-list, (2) the classification models can be built automatically on the requested time horizons and at the specified time intervals, (3) we have applied many techniques in the model building to promote model accuracy, such as multi-model evaluation and boosting, and (4) the model constructed can be immediately applied to predict events for the incoming data streams.

3.3 Stream Pattern Finder

Data streams may contain many hidden patterns, such as *frequent patterns* and *sequential patterns*, which can be used to discover unusual incidents by comparing the current patterns with certain previous durations based on the patterns stored in the *tilted time window*. A pattern mining module, *Stream Pattern Finder*, has been constructed to discover frequent patterns (implemented and tested) and sequential patterns (under the research and development stage). The underlying algorithm essentially adopts the extended frequent pattern growth approach [14, 8] which discovers frequent patterns for the interested sets of items specified by users. This philosophy is somewhat different from counting approximate frequent pairs in data streams [16, 8], however, it provides an alternative angle to examine the problem.

Since a user may have knowledge about the set of interested items to be traced, this approach will derive precise patterns of user's interest. Further extension of this framework may handle additional frequent patterns beyond user-specified items with good approximation bound. This module dynamically constructs a FP-tree while scanning data streams. Each node in the tree contains a *tilted time window* that accumulates counts of the frequent patterns for each time slot. The frequent patterns and association (or correlation) rules for a requested time horizon can be extracted and visualized using the FP-tree structure [14]. The extension of the method for mining sequential patterns is under active investigation and experiment. Figure 4 shows the association rules discovered for a 5-minute period. In this 3-D bar chart, the height of bars represents the rule's *Support*, and the color of bars represents the rule's *Confidence*.

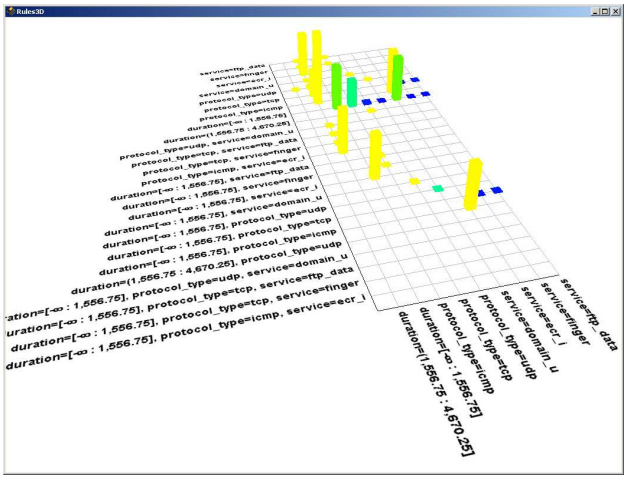

Figure 4: Stream Association Rules

3.4 Stream Cluster Analyzer

Stream Cluster Analyzer dynamically performs cluster analysis based on the current data set as well as those stored in different slots in the *tilted time window*. This framework is different from steam clustering method proposed in [11, 18], but is essentially based on the CluStream framework investigated in our research [1]. The method constructs clusters in two steps: (1) micro-clustering, and (2) macro-clustering. The first step dynamically builds a hierarchical CF-Tree (Clustering Feature Tree) similar to the BIRCH algorithm [21]. However, in our CluStream algorithm, each entry of a leaf node in the CF Tree stores a *tilted time window*, each time slot of the *tilted time window* holds a *Clustering Feature Vector*, and the entries in the leaf nodes form micro-clusters. The integration of CF-Tree with *tilted time window* has made it possible that the hierarchical CF-Tree is maintained in a stream data environment and the time-series information is stored along with clustering features. The second step of the clustering process uses a modified K-Means algorithm [15]. The macro-clusters are computed for a requested time horizon. The computation applies several techniques, such as seed sampling, distance-based partition, and weighted centroid adjustment. A major advantage of this model is that clustering can be performed based on different weighting factors of different time durations, and one can compare the clusters at different time durations and discover the evolving behaviors and contrasts of the clusters at different times. *Stream Cluster Analyzer* is currently at the refinement and testing stage, based on the work done in [1].

3.5 Stream Mining Visualizer

We have developed an effective stream query and mining interface with a set of visualization tools (some of them have been displayed in previous subsections). These interfaces and visualization modules will be further refined or redesigned based on user's feedbacks and application scenarios. The distinct features of the MAIDS *Stream Mining Visualizer* are that they can be served as continuous queries and continuous mining displays. The models, rules, clusters and OLAP results will change from time to time for continuous updates. They are the watchdogs of the dynamic streaming system and they will trigger alarms and giving messages when some alarming incidents are being detected based on the ongoing stream data.

4. CONCLUSIONS AND DEMONSTRATION PLAN

The MAIDS system is a joint research and development effort by Automated Learning Group, NCSA and Department of Computer Science in the University of Illinois at Urbana-Champaign. It is based on the integration of our fruitful research results on data mining, data warehousing, and especially on recent stream data mining for finding dynamics and alarming incidents in multi-dimensional stream data. It is a special component of D2K for stream data analysis. Please visit http://maids.ncsa.uiuc.edu for detailed information about MAIDS.

The MAIDS project has achieved great progress. Based on our current testing, the MAIDS system has shown excellent real-time performance on several major functions. We expect to have an impressive demo ready for SIGMOD'04 conference. We plan to demo five modules of MAIDS in the coming SIGMOD conference: *Stream Query Engine*, *Stream Data Classifier*, *Stream Pattern Finder*, *Stream Cluster Analyzer*, and *Stream Mining Visualizer*.

We are going to use KDDCup99 data to simulate network data streams and show how MAIDS can monitor the network flow and detect the network intrusions. We will also bring a data set from a real application, with minor preprocess to remove sensitive private identities, to show how MAIDS can be used in the real world, and how business companies can get benefit from it.

In the demonstration, visitors are encouraged to play with MAIDS and explore the features by themselves. We will also present our experience and lessons from our research and implementation using posters.

5. REFERENCES

[1] C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proc. 2003 Int. Conf. Very Large Data Bases (VLDB'03)*, Berlin, Germany, Sept. 2003.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems.

In *Proc. 2002 ACM Symp. Principles of Database Systems (PODS'02)*, pages 1–16, Madison, WI, June 2002.

[3] S. Babu and J. Widom. Continuous queries over data streams. *SIGMOD Record*, 30:109–120, 2001.

[4] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *Proc. 2002 Int. Conf. Very Large Data Bases (VLDB'02)*, pages 323–334, Hong Kong, China, Aug. 2002.

[5] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proc. 2002 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'02)*, pages 61–72, Madison, Wisconsin, June 2002.

[6] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. 2000 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'00)*, pages 71–80, Boston, MA, Aug. 2000.

[7] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: You only get one look (a tutorial). In *Proc. 2002 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'02)*, page 635, Madison, WI, June 2002.

[8] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining frequent patterns in data streams at multiple time granularities. In H. Kargupta, A. Joshi, K. Sivakumar, , and Y. Yesha, editors, *Data Mining: Next Generation Challenges and Future Directions*. AAAI/MIT Press, 2003.

[9] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proc. 2001 Int. Conf. on Very Large Data Bases (VLDB'01)*, pages 79–88, Rome, Italy, Sept. 2001.

[10] S. Guha, N. Koudas, and L. Shim. Data streams and historgrams. In *Proc. ACM Symposium on Theory of Computing (STOC'00)*, pages 471–475, Crete, Greece, July 2001.

[11] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS'00)*, pages 359–366, Redondo Beach, CA, 2000.

[12] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 73–84, Seattle, WA, June 1998.

[13] J. Han, J. Pei, G. Dong, and K. Wang. Efficient computation of iceberg cubes with complex measures. In *Proc. 2001 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'01)*, pages 1–12, Santa Barbara, CA, May 2001.

[14] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pages 1–12, Dallas, TX, May 2000.

[15] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York, 2001.

[16] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. 2002 Int. Conf. Very Large Data Bases (VLDB'02)*, pages 346–357, Hong Kong, China, Aug. 2002.

[17] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[18] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. High-performance clustering of streams and large data sets. In *Proc. 2002 Int. Conf. Data Engineering (ICDE'02)*, San Fransisco, CA, April 2002.

[19] M. Welge, L. Auvil, A. Shirk, C. Bushell, P. Bajcsy, D. Cai, T. Redman, D. Clutter, R. Aydt, and D. Tcheng. Data to knowledge (D2K), A rapid application development environment for knowledge discovery in databases. In *Technical Report*, 2003.

[20] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

[21] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pages 103–114, Montreal, Canada, June 1996.