Smart Design Pattern and Applicable Model for Fine Tuning of Sensor Data Analysis with Local Data Historian

U. Surya Kameswari [#1], Prof. I. Ramesh Babu [*2]
[#1 *2] Department of CSE, Acharya Nagarjuna University, India
[1]u.suryakameswari@gmail.com
[2] rinampudi@outlook.com

Abstract –Every process industry is highly equipped with wireless sensors for process monitoring in those locations human intervention has to be limited. Sensor data analysis and anomaly detection using predictive analytics for process industries where average performance done and only applicable for standalone installation. It can also require sufficient memory and processing speed. We are proposing two solutions in current work to significantly improve the applicability and performance for both standalone and distributed environments. Standalone model is implemented through file level partition based data analysis and Distributed analysis is implemented through Intra-Node Cluster with Local Historian. Users of these models are freedom to choose any model based on their requirement. Prescriptive analysis is used here because output or old values are fed back as input to proposed elastic clustering algorithm. Our simulation result shows the performance of the both the models in terms of time and data size.

Keywords - IIoT, Sensor Data analysis, Prescriptive Analytics, File Partition, intra-node cluster, data historian

I. INTRODUCTION

In general Predictive Analytics [4][5][6] focus only on some sought of regression analysis. It doesn't care about feedback of the impact of the outcome. It is more statistical. But some time there is need to consider the output and fed that output as input to again more accurate data to predict the result. Predictive analytics mostly works on structured data. But prescriptive analytics is independent of type, format, data source and size. Predictive analytics is just a probable condition. It doesn't focus on the impact of the prediction, if that prediction is used in a decision making process.

But Prescriptive analytics [7][8][9] gives optimal solutions, alternatives and impact of each alternative in decision making. Based on these decisions operational or business managers can take timely actions to reduce the production or maintenance cost. So that organizations can withstand or get stabilized economically, and able to provide valuable services to their customers. In case of energy field it is more important to predict and give proper guidelines or notifications based on prescriptive process rules in case of equipment or component or interface failures. It greatly reduces operational costs and increase industry safety also.

This paper focuses on how to design a background model for such type of prescriptive analytics tool. This model is designed in such a way so that no single point failure occurs and scalability and availability are increased. It can work both in standalone and distributed modes.

II. LITERATURE SURVEY

This topic as a basis, we are extending our previous work published in IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions conference (IEEE WCI 2015). Limitation of our previous work includes average performance and only applicable for standalone installation. It requires sufficient memory and processing speed. We are proposing two solutions in current work to significantly improve the applicability and performance for both standalone and distributed environment The Industrial Internet of Things (IIoT) is the next brandish of innovation, touching the way that the world connects and optimizes machines. With the use of sensors, advanced analytics and intelligent decisions, the IIoT will unfathomed transform the way, field assets connect and communicate with the enterprises

A.Prescriptive analytics:

The relatively new field of prescriptive analytics allows users to prescribe a number of different possible actions to guide them towards a solution. Prescriptive analytics attempt to quantify the effect of future decisions in order to advise on possible outcomes before the decisions are actually made. Prescriptive analytics uses what you know and what you can predict to make the right decision for the desired outcome.

B. Apache Cassandra:

Apache Cassandra is a highly scalable, high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. It is a type of NoSQL row partitioned database.

III. Proposed Approach

We are proposing two solutions in current work to significantly improve the applicability and performance for both standalone and distributed environment. Standalone model is implemented through file level partition based data analysis and Distributed analysis is implemented through Intra-Node Cluster with Local Historian.

A. Standalone Model

This model can work on single system with minimum primary memory and moderate hardware environment. It is a pure desktop application. Following modules of applications are required to explain standalone model of sensor data analysis.

1) Extracting of Data from Remote Process Data Historian Server:

A plug-in is required to extract the data according to the historian model and provider. This plug-in is installed and first export the data from historian to a excel file locally at remote server. That file is downloaded using any remote desktop application or email using cloud drive. But capacity of the file must not be larger than fixed size. This limitation is not problem with plug-in, it is the limitation of spreadsheet application or text editors. To overcome this limitation, partition and export is the only solution.

2) Building of database schema:

This module is used to build separate database for each process industry to store information about sensors and values of selected sensors for further analysis. But limitation is number of columns allowed in each table of any selected database are limited. So according to that number, number of databases has to be divided. For this reason vertical partition based database structure is developed with proposed standalone approach. These databases are created by considering number of sensors present in the input file header. Each database table contains "max-allowed-columns" predefined constant. Again these columns are divided into two types, one for actual data and other column to store respective status of actual data.

3) Local Partitioning of Download file:

We are considering "max-file-size" property to partition the file into small files. File must be partitioned horizontally. Now this file is loaded into a Matrix and extracts the header part to find out respective database table of all the sensors. Again these files must be vertically partitioned in the same order as mentioned in previous module. This is very important step for bulk export data from file to database.

4) Machine State Assignment:

Based on past sensor data compare the current sensor data, mark the state of the machine with available state behaviors. This state identification can be achieved through various data mining and machine learning techniques such as regression analysis, for example. But proposed work is depended on non-parametric unsupervised learning technique called "elastic clustering". This novel method is proposed for sensor state analysis to handle concept-drift in sensor data streams. This algorithm is proposed as follows.

```
Algorithm  : Elastic Clustering
Input      : Dataset, Steady State Range
Output     : Clusters
step 1:  Initialize  sim-th = random number between (0-1)
step 2:  Load existing similarity thresholds if any from database for all sensors
step 3:  Apply incremental Adaptive Micor-clustering proposed on our previous paper[1] with
current sim-th
step 4:  auto tune the sim-th based on the quality of the cluster with existing similarity thrsholds and extra random numbers.
step 5: repeat steps 3 and 4 till best quality clusters found i.e., by calculating the error variance among previous iteration errors.
step 6: extract the best threshold and store it in database for respective sensor.
step 7: now mark cluster states and accordingly mark the state of points those fall in this clusters.
step 8:  return Final Clusters.
```

According to proposed algorithm, clusters don't store data points for long time. Data points are stored to calculate the quality of the cluster. Once best quality is found, data points are cleared from respective clusters. Because they are micro clusters,i.e they are used to store statistics of cluster such as mean, variance, number of points and timestamps. Next point in the algorithm includes states. We are considered Four states Shut-down, Transient, Steady State, No Change. Each one is described as follows.

Shut-down means machine is not running. Transient means no proper value is maintained within a range for long time. Steady state means the values generated from these machines or respective sensors monitoring these machine are within supplied range. This state is very important for any other analysis because we can predict the machine behavior only when it is in steady state. No Change state is used to identify interface failures or sensor failures. This can be calculated if data generated from sensor is unchanged. Finally cluster quality is calculated using Davies-Boulding Index proposed in [2].

5) Exporting Data to Database:

After assignment of state, update the matrix with state codes for all the sensor values. Now write back this matrix to data file. Now apply bulk loader to update the database respective database table in which current set of sensors are placed. This data is used for any data analysis techniques.

In this way standalone model can work with any number of records and any number of columns. But it is little bit time consuming. To reduce the time complexity, concurrent execution on multi-core system is possible. We applied the same if selected system supports multiple cores by get the information of system configuration.

*B.*Distributed Analysis

This is mostly a hybrid approach. It is implemented through Intra-Node Cluster with Local Data Historian and Cloud Interface. We are try to simulate this model locally. This model is having following modules.

1) Extracting of Data:

Sensor data accessed from OPC (OLE for Process Control) protocol is not having any data storing capacity. And all the data of each sensor are replaced with new value periodically. So the previous data has to be stored each time. The data generated from sensors each time called as sensor data stream and data reading through OPC is called data in highway or snapshot data. For analysis this data must be stored.

All types of process industries may or may not have data historians. They might ignore the data or maintain for some period (last one month or 6 months for example) or use traditional relational databases. Limitation of relational database itself is not scalable and doesn't support high dimensionality. Query response type is also high. So it leads to degradation of performance.

To overcome these limitations proposed approach uses columnar database instead of traditional relational databases. Columnar databases have many prominent features when compared to RDBMS. Some of them are, High Speed Query Processing, Significant compression and highly scalable and support high dimensionality. Among the available columnar databases Apache Cassandra [3] is one of the best options. Proposed approach uses Apache Cassandra as local data historian at data processing centre. But to store the data in this local historian, periodical interaction with cloud drive is required. Because data from highway is maintained in flat files for some period and they flushed also periodically. So before flushing those files are store in the cloud drive for remote access or to make those files available from remote data processing centers.

2) Cloud Plug-in

Next stage of work is developing a cloud-plug in to download data from cloud to local data historian. This plug-in uses third party API based on the chosen cloud service provider. For example, If Microsoft Azure is used then respective Cloud API has to be used to develop plug-in. This plug-in is used as an interface between industry sensor network and data processing centre. In our case, a common cloud gateway is developed so that any cloud api can be accessed from single window based on supplied credentials.

3) Local Data Historian

As discussed in the previous module Apache Cassandra is used as local data historian. It is a hybrid NoSQL Data store and row partitioned and partially columnar database and its stable version recently released this year (Jan 18th,2016). Sensor data streams are treated as temporal data or time series data. So each point is associated with a timestamp. So according to this model on record is a snapshot retrieved from the DCS (Distributed Control System) at process industry and it is uniquely identified based on timestamp. These snapshot are read periodically with one minute or 'n' minute interval based on the user parameter. Each snapshot contains required number of values with respective to the sensors and a timestamp. Database structure is required to build data historian which includes Sensor Master, DataArchive, ErrorCodes and Interfaces column families.

For storing the data downloaded from cloud drive, bulk data loader program is used. It reduces individual record insertion time complexity. After this step cloud drive has been cleared. Proposed local data historian has a capability to work in standalone and distributed mode. So we can store the data in a single system or more than one system. But proposed system uses single system for data historian and other systems are processing systems such that these systems can access the data from local data historian.

4) Distributed Processing

As discussed in earlier modules volume of sensor data is high and it is difficult to apply analysis on single system. For that purpose proposed system introduces a hybrid model with RMI and Apache Cassandra. Sensor analysis is running as a service at each system. So RMI (Remote Method Invocation) is used to set or get the state of service. It acts like a bridge between actual client process (sensor analytics) and data historian. The systems participating in the sensor analysis is treated as cluster. Each system in the cluster gets the data from local data historian and processes the data and push back results to historian.

Same Machine State Assignment and Elastic Clustering discussed earlier is converted into parallel and distributed algorithm. Here parallel means efficient utilization of latest development in hardware with core processors at each system and distributed means processing of unique subset of sensor data is processed at each node and finally all the results are merged at server. Partially this process is like Map-Reduce framework in Hadoop.

B. Simulation Environment

Under java domain, Matrikon OPC Simulation Server, OpenScada as OPC Server interface, Apache Cassandra as Data Historian is used to design and develop our simulation environment. We setup the simulation environment for both standalone and distributed models. By tuning the some of the parameters in the configuration file, Cassandra can be applicable for both of the models. Real time data is used in the simulation to test the work load. Due to privacy agreement with data providers, we can not disclose the source of the data.

IV. RESULTS

TABLE I Memory Occupation for Proposed Data Historian

No of Sensors	Total Readings/Month	Total Memory (in GB)/Month	Total Memory (in GB) /Year
1000	44640000	2.227	26.724
2000	89280000	4.453	53.436
3000	133920000	6.68	80.16
4000	178560000	8.906	106.872
5000	223200000	11.133	133.596
6000	267840000	13.359	160.308
7000	312480000	15.586	187.032
8000	357120000	17.813	213.756
9000	401760000	20.039	240.468
10000	446400000	22.266	267.192

Above table is generated by considering each sensor reading for one minute interval for one month. So, total of 44640 readings for 31 days. Memory occupied by each sensor with timestamp, sensor value and status of value is 2.28 MB.

TABLE II Time Complexity for Dump Data to Historian using Proposed Model and Traditional RDBMS Model

No of Sensors	Total Memory (in GB) /Year	Data Dump Time To Local Historian (in hours)	
		Proposed Model	Traditional RDBMS
1000	26.724	0.5	22
2000	53.436	0.675	37.4
3000	80.16	0.911	63.58
4000	106.872	1.23	108.086
5000	133.596	1.661	183.746
6000	160.308	2.242	312.368
7000	187.032	3.027	531.026
8000	213.756	4.086	902.744
9000	240.468	5.516	1534.665
10000	267.192	7.447	2608.931

Traditional RDBMS Model actually not suitable for time series data. But if it is designed by considering each tag is a column in the table then time complexity may look like above table. But most unfortunate thing about commercial RDBMS software currently available cannot support more than 2000 columns/Table limited to 256MB/row. So, above table is showing assumption values for RDBMS. In contrast, proposed system using the hybrid model of both SQL and NoSQL. Apache Cassandra is a row partition based columnar database. Proposed model is simulated with single node cluster only. Its performance and availability is more increased if it is deployed in multi-node environment.

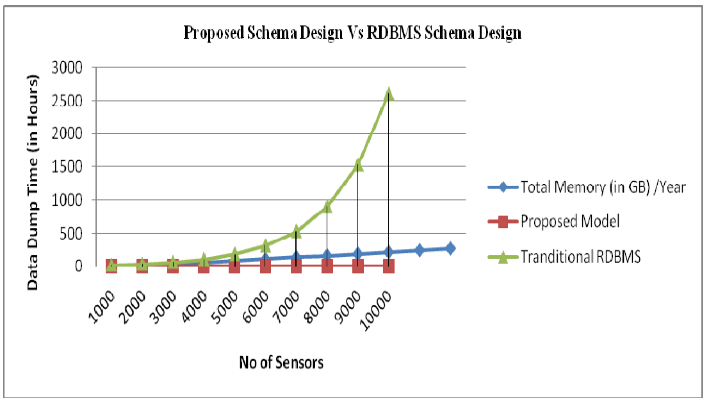


Fig. 1. Time complexity for Data Dump into Local Historian using Proposed Model Vs RDBMS

TABLE III Query Time Complexity for Any Sensor Data for Various Number of Sensors

No of Sensors	Total Readings/Month	Total Readings/Year	Query Time (in sec)*
1000	44640000	535680000	0.003
2000	89280000	1071360000	0.0065
3000	133920000	1607040000	0.01
4000	178560000	2142720000	0.0135
5000	223200000	2678400000	0.017
6000	267840000	3214080000	0.0205
7000	312480000	3749760000	0.024
8000	357120000	4285440000	0.0275
9000	401760000	4821120000	0.031
10000	446400000	5356800000	0.0345

It is time complexity to query any one sensor data at a time

Above table is only for proposed system as already told that RDBM cannot handle that number of columns. If so, it will take greater than any 15 minutes of time for single tag data.

TABLE IV Time complexity to Download Raw and Interpolated Data from Local Historian

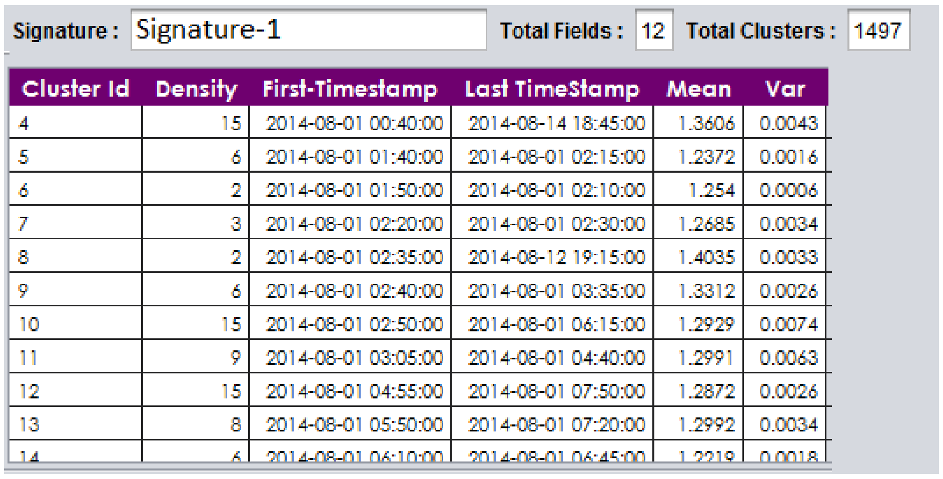
No of Sensors	Total Readings/Month	Raw Data Retrieve Time (in minutes)	Interpolated Data Retrieve Time (in minutes)
1000	44640000	0.35	0.3605
2000	89280000	0.66	0.6798
3000	133920000	0.8	0.824
4000	178560000	0.87	0.8961
5000	223200000	0.94	0.9682
6000	267840000	1.23	1.2669
7000	312480000	1.28	1.3184
8000	357120000	1.5	1.545
9000	401760000	1.8	1.854
10000	446400000	2.25	2.3175

Above table is describing time complexity to download data from local historian for data analysis. In general data coming from sensors or OPC is called as Raw data. Timestamps or scan frequency is different for different sensors. So, there is no need of equal number of readings for all the sensors in the given time period. i.e one sensor reading is noted for each one minute interval and other sensor reading is noted for each five minutes interval in an hour. According to that first sensor is having 60 readings and second sensor is having only 12 readings in an hour. But to do data analytics all sensors must maintain equal number of readings. To achieve that goal, interpolation is used. Here liner interpolation is enough. So it takes little higher time to generate readings when compared to raw data download.

TABLE V Steady State Percentage of Selected Sensors

Sensor	Steady State %
Sensor1	99.9215
Sensor2	99.9215
Sensor3	99.8766
Sensor4	98.4747
Sensor5	99.8878
Sensor6	0.0000
Sensor7	99.9215
Sensor8	0.0449
Sensor9	93.7865
Sensor10	88.8403

Above table is showing the results of steady state percentage of selected sensors for example. First column is sensor name and second column is how many number of values maintained steady state in the given time period. Except Sensor 8, all other sensors ran in steady state at most.



Signature : Signature-1 Total Fields : 12 Total Clusters : 1497

Cluster Id	Density	First-Timestamp	Last TimeStamp	Mean	Var
4	15	2014-08-01 00:40:00	2014-08-14 18:45:00	1.3606	0.0043
5	6	2014-08-01 01:40:00	2014-08-01 02:15:00	1.2372	0.0016
6	2	2014-08-01 01:50:00	2014-08-01 02:10:00	1.254	0.0006
7	3	2014-08-01 02:20:00	2014-08-01 02:30:00	1.2685	0.0034
8	2	2014-08-01 02:35:00	2014-08-12 19:15:00	1.4035	0.0033
9	6	2014-08-01 02:40:00	2014-08-01 03:35:00	1.3312	0.0026
10	15	2014-08-01 02:50:00	2014-08-01 06:15:00	1.2929	0.0074
11	9	2014-08-01 03:05:00	2014-08-01 04:40:00	1.2991	0.0063
12	15	2014-08-01 04:55:00	2014-08-01 07:50:00	1.2872	0.0026
13	8	2014-08-01 05:50:00	2014-08-01 07:20:00	1.2992	0.0034
14	6	2014-08-01 06:10:00	2014-08-01 06:45:00	1.2219	0.0018

Fig. 2. Result of Elastic Clustering

In the above figure the result of elastic clustering is described. In that figure, Signature represents the group of sensors which are technically monitoring one section or module or a machine itself in the real plant. So, the data of all those sensors influences one single sensor called as dependent sensor and it is the value to be predicted by proposed algorithm. Total fields are total number for independent sensors participated for a section or machine. Total clusters are number of clustering formed due to elastic clustering. This number can vary based on the threshold and sensor data stream values.

The table in the figure contains information about micro clusters. Cluster id is serial number of the cluster, Density is the number of points or values fell in that cluster, First timestamp is when that cluster is created and last timestamp is when that cluster is recently updated, mean is average of all the values in that cluster and var is variance of the cluster values.

For given one month of data for 2000 sensors, the clustering process took less than 2 minutes to generate above clustering results. Why we considered 2000 sensors is that most of the medium and semi-large industries require analysis of those 2000 sensors data even though they are having greater than that number of sensors.

V. Conclusion

Proposed approach fine tunes the performance and improved scalability and availability of the data with two different approaches. Standalone model is implemented through file level partition based data analysis and Distributed analysis is implemented through Intra-Node Cluster with Local Historian. Users of these models are freedom to choose any model based on their requirement. Proposed approach is economically and technically feasible to be implemented even in small and medium level industries. It is also more reliable model also. In general 1 TB of hard disk is able to store nearly one decade of time series data with 2000 sensors and ready to query and analyze that data any time. Due to proposed model machine behavior and performance can be periodically evaluated and take necessary action in the production or work environment to reduce operational, calibration and machine and man power safety costs. Finally in simple words proposed work is a pilot to "Smart Big Time series analytics with local data historian".

VI. Future Work

We are simulated the current work. Current work can be extended by real time deployment and testing the performance and security issues.

References

[1] U. Surya Kameswari, I. Ramesh Babu, "Sensor Data Analysis and Anomaly Detection using Predictive Analytics for Process Industries", IEEE workshop on Computational Intelligence: Theories, Applications & Future Directions (IEEE WCI 2015). DOI: 10.1109/WCI.2015.7495528
[2] Davies, David L.; Bouldin, Donald W. (1979). "A Cluster Separation Measure". IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-1 (2): 224–227. doi:10.1109/TPAMI.1979.4766909.
[3] http://cassandra.apache.org/
[4] Stepan Ivanov, Kriti Bhargava, and William Donnelly, "Precision Farming: Sensor Analytics", IEEE Intelligent Systems, pp. 76-80 , 2015.
[5] Hadi Banaee and Amy Loutfi, "Data-Driven Rule Mining and Representation of Temporal Patterns in Physiological Sensor Data", IEEE Journal of Biomedical and Health Informatics, Vol. 19, No. 5, pp. 1557-1566, SEPTEMBER 2015.
[6] Girma Kejela, Rui Maximo Esteves, and Chunming Rong, "Predictive Analytics of Sensor Data Using Distributed Machine Learning Techniques", IEEE 6th International Conference on Cloud Computing Technology and Science, pp. 626-631, 2014.
[7] Bertolucci, Jeff, Prescriptive Analytics and Data: Next Big Thing?InformationWeek. (April 15, 2013).
[8] McCormick Northwestern Engineering Prescriptive analytics is about enabling smart decisions based on data.

AUTHOR PROFILE

Mrs. U. Surya Kameswari received her Degree of Master of Science in Computer Science from Andhra University, Visakhapatnam in 2007 and M.Tech from Karnataka State Open University, Mysore, Karnataka in 2012. She is currently serving as an Asst. Professor in the Department of Computer Science and Engineering, Acharya Nagarjuna University, Nagarjuna Nagar, Guntur, Andhra Pradesh. Her Research Interest lies in the area of Data Mining and Data Warehousing, Computer Networks, Network Security. She has nine years of experience in teaching in many subjects and in various academic projects. She is a member in International Associations like IAENG, CSTA. She has many publications in various journals and in conferences.

Prof. I Ramesh Babu received his B.E in Electronics and Communication Engineering from University of ysore, India in 1981, M.E in Computer Science and Engineering from Andhra University, India in 1984 and Doctorate degree in computer science and engineering from Acharya Nagarjuna University, Guntur in 1994. He has been working as a Professor in the department of Computer Science and Engineering of Nagarjuna University, Guntur till date. He held many positions in Acharya Nagarjuna University as Head, Director-Computer Center, Chairman-Board of Studies, member of Academic Senate and Member of Executive Council. His areas of interest include image processing, computer graphics, and Data Mining. He is a member of many professional societies like IEEE, CSI, ISTE, IETE, IGISS, and Amateur Ham Radio (VU2 IJZ). He has many publications in national and international journals and conferences.