Analyzing massive machine data maintaining in a cloud computing

Balamanigandan R[1]

[1]*Research scholar, Karpagam University, Coimbatore/Assistant professor, Dept of CSE, Gopal Ramalingam Memorial Engineering College, Chennai,*

ABSTRACT:

We present a novel framework, Cloud View, for storage, processing and analysis of massive machine maintenance data, collected from a large number of sensors embedded in industrial machines, in a cloud computing environment. This paper describes the architecture, design, and implementation of CloudView, and how the proposed framework leverages the parallel computing capability of a computing cloud based on a large-scale distributed batch processing infrastructure that is built of commodity hardware. A case-based reasoning (CBR) approach is adopted for machine fault prediction, where the past cases of failure from a large number of machines are collected in a cloud. A case-base of past cases of failure is created using the global information obtained from a large number of machines. Case-base creation jobs are formulated using the MapReduce parallel data processing model. CloudView captures the failure cases across a large number of machines and shares the failure information with a number of local nodes in the form of case-base updates that occur in a time scale of every few hours. The case-base is updated regularly (in the time scale of a few hours) on the cloud to include new cases of failure, and these case-base updates are pushed from CloudView to the local nodes. Our approach, in addition to being the first reported use of the cloud architecture for maintenance data storage, processing and analysis, also evaluates several possible cloud-based architectures that leverage the advantages of the parallel computing capabilities of the cloud to make local decisions with global information efficiently, while avoiding potential data bottlenecks that can occur in getting the maintenance data in and out of the cloud.

Keywords: *Fault prediction, machine data analysis, case-based reasoning, cloud computing,*

I. INTRODUCTION:

Modern complex industrial machines and systems have thousands of sensors that gather maintenance data continuously for condition monitoring and failure prediction purposes. In systems such as power grids, real-time information is collected using specialized electrical sensors called Phasor Measurement Units (PMU) at the substations. The information received from PMUs must be monitored in real time for estimating the state of the system and for predicting failures.

CBR is a method that finds solutions to new problems based on past experience. This past experience is organized and represented as cases in a case base. The processes involved in CBR are

1. Retrieving similar cases from case base,
2. Reusing the information in the retrieved cases,
3. Revising the solution, and
4. Retaining a new experience into the case base.

CBR systems utilize incremental learning in which new cases are added into the case base with time. The major contributions of this paper are: 1) We propose for the first time, “CloudView,” a framework for machine data organization and analysis in a computing cloud, that allows efficient collection of machine sensor and fault data and creation of case libraries that capture a global knowledge of machine failures, 2) A hybrid approach for machine data analysis and fault prediction which includes a cloud for massive data organization and analysis and local nodes for real-time fault prediction, thus avoiding data communication bottlenecks present in typical cloud architectures. Instead of proposing new algorithms for case-based data analysis (which is an established area of research within computer sciences and engineering), our effort behind the proposed hybrid approach and the CloudView framework is aimed at supporting a wide variety of data analysis algorithms within cloud architecture and providing support through experimental results.

II. PROPOSED FRAMEWORK

CloudView is based on Hadoop which is a framework for running applications on large clusters built of commodity hardware. Hadoop comprises of two major components:

1. HDFS: HDFS stores files across a collection of nodes in a cluster. Large files are split into blocks and each block is written to multiple nodes (default is three) for fault tolerance.
2. MapReduce: MapReduce is a parallel data processing model which has two phases: Map and Reduce. In the Map phase, data are read from a distributed file system (such as HDFS), partitioned among a set of computing nodes in the cluster, and sent to the nodes as a set of key-value pairs. The Map tasks process the input records independent of each other and produce intermediate results as key-value pairs. The intermediate results are stored on the local disk of the node running the Map task. When all the Map tasks are completed, the Reduce phase begins in which the intermediate data with the same key is aggregated. An optional Combine task can be used to perform data aggregation on the intermediate data of the same key for the output of the mapper before transferring the output to the Reduce task.

III. MODULES OF CLOUD VIEW

A. Data collectors

It collects the streaming time-series data from the sensors embedded in industrial machines. Each incoming data stream is mapped to one of the Data Collector nodes. Each Data collector node has a Data Aggregator, Data Filter, and Data Archiver module.The Data Collectors buffer preprocess and filter the streaming data into larger chunks and stores it in HDFS Data Collectors use Hadoop's SequenceFile class which provides a persistent data structure and serves as a container for multiple records.

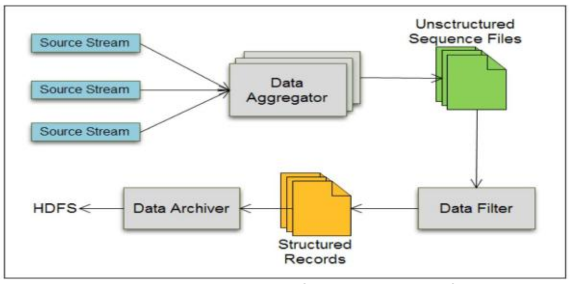

Fig. 1 Data flow in a Data Collector node which forms a part of the CloudView framework.

As shown in fig 1.The streaming data from the machines are in the form of records where each record is a single line of text. The Data Aggregator aggregates streams of sensor data into Unstructured-SequenceFiles on the local disk of the Data Collector node. The Data Aggregator writes the records of input data streams as key-value pairs in SequenceFiles, where key is just a sequence number. The Data Filter converts the Unstructured-SequenceFiles into structured records by parsing the records (lines) in Unstructured-SequenceFiles and extracting the sensor readings. The Data Filter also filters out bad records in which some sensor readings are missing. The Data Archiver moves the Structured Records to HDFS.

B. Case-base creation module

A CBR is adopted for machine fault prediction, where the past cases of failure from a large number of machines are collected in a cloud.Create case bases from the legacy data in databases at the initial stage and to automatically collect new cases at the online runtime stage in CBR systems.Legacy databases are maintained by Diagnostics Centers for machines. It has data on the past faults in the machines, and the Machine Sensor Database has the time stamped sensor data. Case-Base creation proceeds by retrieval of information on past faults in a machine from the Maintenance Database, and then the sensor data in a small time window before the fault from the Machine Sensor Database.

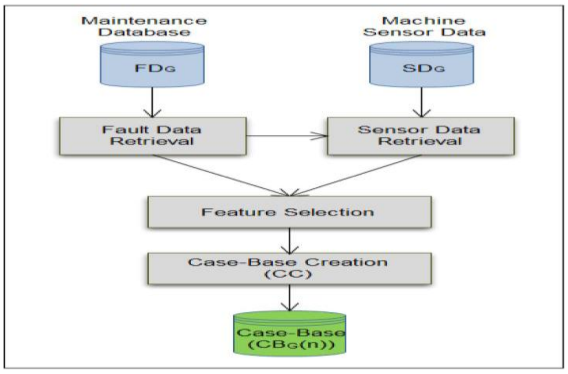

Fig. 2. Creation of case base from the machine data collected in the cloud.

Fig. 2 shows the steps involved in Case-Base creation (CC task). For creating the Case Base we propose using the machine data in the legacy databases as well as the machine sensor (SDG) and faults data (FDG) which is collected in HDFS. Such legacy databases are typically maintained by the Maintenance and Diagnostics Centers for machines. The Machine Maintenance Database has data on the past faults in the machines, and the Machine Sensor Database has the time stamped sensor data. To leverage the parallel computing capability of Hadoop for Case-Base creation, we propose automated migration of the data in the legacy Machine Maintenance and Machine Sensor databases to HDFS. This allows formulation of fault data and sensor data retrieval tasks as MapReduce jobs.

C. Map reduce job

Cloud View is based on Hadoop which is a framework for running applications on large clusters built of commodity hardware. Hadoop comprises of two major components:
1. HDFS: HDFS stores files across a collection of nodes in a cluster. Large files are split into blocks and each block is written to multiple nodes (default is three) for fault tolerance.
2. MapReduce: MapReduce is a parallel data processing model which has two phases: Map and Reduce. In the Map phase, data are read from a distributed file system (such as HDFS), partitioned among a set of computing nodes in the cluster, and sent to the nodes as a set of key-value pairs. The Map tasks process the input records independent of each other and produce intermediate results as key-value pairs. The intermediate results are stored on the local disk of the node running the Map task. When all the Map tasks are completed, the Reduce phase begins in which the intermediate data with the same key is aggregated. An optional Combine task can be used to perform data aggregation on the intermediate data of the same key for the output of the mapper before transferring the output to the Reduce task.

D. Case-Base Updating Module

The case-base updating module sends updates of case base to the local nodes when new cases are added to the casebase. In push-based updation approach, the local nodes can also pull the recent case base from CloudView.CloudView captures the failure cases across a large number of machines and shares the failure information with a number of local nodes in the form of case-base updates that occur in a time scale of every few hours.The case-base is updated regularly on the cloud to include new cases of failure, and these case-base updates are pushed from CloudView to the local nodes.

E. Case-Base Maintenance Module

In Case-Base Maintenance module , adds new cases into the case base, detects and removes redundant and inconsistent cases in the case base. Addition of new cases to the case base is done by incorporating the new faults and sensor data which are collected from machines in HDFS, into the case base. Avoiding potential data bottlenecks that can occur in getting the maintenance data in and out of the cloud. In this approach each case is classified according to its competence into four categories,

3. Pivotal cases, that are the only case that can solve a target problem,
4. Auxiliary cases, that are the cases which are completely subsumed by other cases in the case base,
5. Spanning cases, that link together areas which are independently covered by other cases, and
6. Support cases, that exist in groups and solve similar target problems as other cases in a group.

IV. LOCAL NODES

Local nodes are present in each plant or turbine farm and are connected to CloudView. Each local node maintains a local copy of the case base which is created in CloudView from the faults and sensor data of a large number of machines.

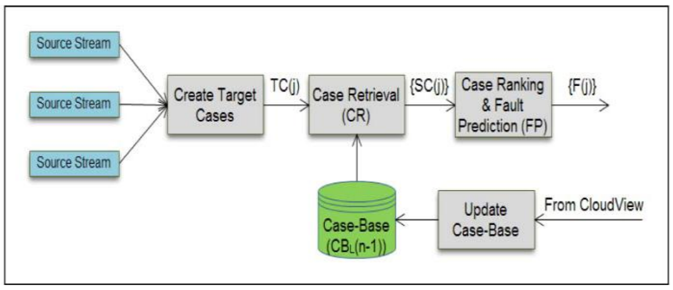

Fig. 3.Case-based reasoning at the local node in the proposed hybrid approach to predict incipient failure

The functions of the local node are as follows:

1. Target Case Creation. To match the real-time data to the cases in the case-base, target cases (TC) are created from the real-time data.Target case creation involves parsing the sensor data stream into different sensor readings. The information from the machine profiles is used to identify the different fields in the sensor data streams for parsing the streams.
2. Case Retrieval. Case Retrieval (CR) module retrieves all the cases from the local case base that are similar to the target case. The Case Retrieval module uses a similarity degree for matching the target case to the past cases.
3. Case Ranking and Fault Prediction. The Case Ranking & Fault Prediction (FP) Module ranks the cases retrieved by the Case Retrieval module according to a degree of similarity to the target case. Each retrieved case has a fault type associated with it.The output of this module is a set of cases (F) ranked according to the similarity to the target case from which the incipient faults are determined.

V. CONCLUSION

Case-based reasoning (CBR) puts forward a paradigmatic way to attack AI issues, namely problem solving, learning, usage of general and specific knowledge, combining different reasoning methods, etc. CBR emphasizes problem solving and learning as two sides of the same coin: problem solving uses the results of past learning episodes while problem solving provides the backbone of the experience from which learning advances. Particularly in its many activities related to integration of CBR and other approaches and by its movement toward the development of application-oriented CBR systems.

VI. VI.ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their constructive and detailed comments that helped in improving the paper.

REFERENCES

[1] R.R. Hill, J.A. Stinebaugh, D. Briand, A.S. Benjamin Dr., and J. Linsday, “Wind Turbine Reliability: A Database and Analysis Approach,” Sandia Report, Feb. 2008.

[2] S. Baumik, “Failure of Turbine Rotor Blisk of an Aircraft Engine,”Eng. Failure Analysis, vol. 9, pp. 287-301, 2002.

[3] F.J.G. Carazas and G.F.M. de Souza, “Availability Analysis of Gas Turbines Used in Power Plants,” Int’l J. Thermodynamics, vol. 12,no. 1, pp. 28-37, Mar. 2009.

[4] J. Kolodner, Case-Based Reasoning. Morgan Kaufmann, 1993.

[5] K.Kalpakis and S. Tang, “Collaborative Data Gathering in Wireless Sensor Networks Using Measurement Co-Occurrence,” Computer Comm., vol. 31, no. 10, pp. 1979-1992, June 2008.

[6] S.G. Lee and Y.C. Ng, “Hybrid Case-Based Reasoning for On-Line Product Fault Diagnosis,” Int’l J. Advance Manufacturing Technology, vol. 27, pp. 833-840, 2005.

[7] B. Smyth and P. Cunningham, “The Utility Problem Analysed—A Case-Based Reasoning Perspective,” Proc. Third European Workshop Case-Based Reasoning (EWCBR), 2000.

[8] J. Zhu and Q. Yang, “Remembering to Add: Competence Persevering Case-Addition Policy for Case-Base Maintenance,” Proc. 16th Int’l Joint Conf. AI, pp. 234-239, 2002.