Indian Journal of Science and Technology, Vol 9(30), DOI: 10.17485/ijst/2016/v9i30/99062, August 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Real Time Vehicular Data Analytics Utilising Bigdata Platforms and Cost Effective ECU Networks

Yedu C. Nair*, P. V. Neethu, Vijay Krishna Menon and K. P. Soman

Center for Excellence in Computational Engineering and Networking, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Amrita University,Coimbatore - 641112, Tamil Nadu, India; yeducnair777@gmail.com, neethupv107@gmail.com,vijaykrishnamenon@gmail.com,kp_soman@amrita.edu

Abstract

Background/Objectives:This paper is aimed at performing real time bigdata analytics on vehicular data collected from a network of ECUs (Electronic Control Unit) in cooperated into the different automobiles. **Methods/Statistical Analysis**: The analytics has been performed by building a software model that is capable of handling the vehicular data in real time. Bigdata platforms like hadoop, Apache Storm, Apache Spark(real time streaming), Kafka are utilised here. Automotive sensor data from different Electronic Control Units are collected into a central data server and this data is pushed to kafka, from which the real time streaming models pulls the data and perform analysis. **Findings:**Automotive industry has undergone a drastic revolutionised innovation in the past decade in all of its respective segments. The industry had started utilizing the computational and mathematical aspects from top to bottom in its design strategies to achieve greater reliability on its products out on roads. Latest advancements in this field is the fully autonomous car. Today an automotive is a collection of innumerable sensors and microcontrollers which are under the command of the master ECU. A network of ECUs connected across the globe is a source tap of bigdata. Leveraging the new sources of bigdata by automotive giants boost vehicle performance, enhance loco driver experience, accelerated product designs. Statistical Projections reveal that automotive industry is likely to be the 2nd largest generator of data by mid of 2016. The contribution of this paper to the automotive industry is the real time vehicle monitoring utilizing Bigdata platforms. This can contribute to better customer-industry relations. **Applications/Improvements:**The model developed in this paper can contribute a lot to the automobile industry as it facilitates real time monitoring of the vehicles. This can improve customer-industry relation.

Keywords: ECU,Hadoop, Kafka, Spark, Storm

1. Introduction

Conventional automobiles had engine as the heart but it lacked a brain. ECU formed the solution for this which is now the brain of all modern locomotives. An automobile out on road is under the command of this immaculate intelligent electronic system. Here the aim is to design and develop a suitable Software system that can monitor the vehicular parameters (sensor outputs) in real time using various bigdata platforms and also a proposal for cost effective ECU design that can communicate to a central data server. Here comes the utility of bigdata platform. The ECUs are connected to a central data warehouse through networks into which huge volume of data creeps in at high frequencies. Data can be structured or unstructured. The respective data is subjected to real time streaming (Spark, Storm, Kafka) or near real time through batch processing techniques (Hadoop).

Spark streaming provides a real time response to the automobile (Driver warning system) in case of any malfunctioning sensors. Such a system is useful to the automobile manufacturer as well as the consumer. As a proposal model an ECU design template for the Vintage Automobiles is also discussed.

2. Methodology

The vehicular data from a central server is pushed to the software model. Here we have performed the analytics part using hadoop ecosystem[1], Apache storm, Apache

*Author for correspondence

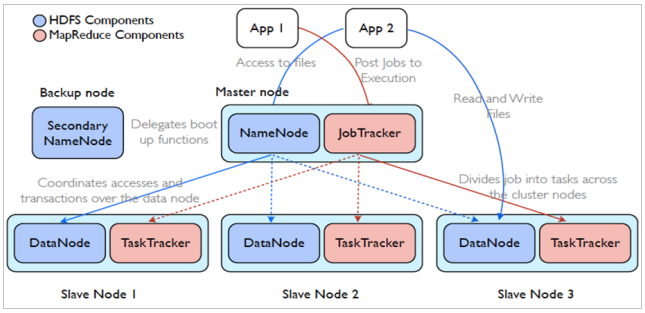

Figure 1. Hadoopt framework.

Kafka and spark framework (spark streaming). Different possibilities of data analytics across various bigdata platforms is performed here. Initially a batch processing model based on mapreduce algorithm has been developed and then the data collected is fed in batches to the HDFS (Hadoop Distributed File System)[2] (Figure 1).

Hadoop ecosystem has two major components the computational part the mapreduce and the storage part called the HDFS[3]. Mapreduce job runs on top of this data and generates output files in case of any discrepancies in the sensor readings. This mapreduce implementation has been performed on java as well as in hive (an SQL like language)[4]. In this case the disadvantage is that the process is not real time and this mechanism is not reliable for critical systems in an automobile. Big Data[4] is known for its velocity, volume and variety. As per the previous methodology we couldn't handle the velocity of the Big Data. That is computation on the fly is exceptionally slow in hadoop. To overcome this limitation, we can rely on other open source technologies like Apache Storm, Apache Kafka and Spark Streaming.

2.1 Storm Streaming

Apache Storm operates on continuous stream of data or data in motion making it a real time processing system. In storm we use hadoop services like HDFS or Hbase or cassandra. Streaming is a means to keep the information you care about and throwing the rest away. Storm does not run on hadoop clusters but it can read and write data to HDFS. Storm uses independent topologies using the concept of 'Directed Acyclic Graphs'. Storm topologies can be written in any language.

Storm cluster has 3 nodes:

1. Nimbus node similar to hadoop job tracker which uploads computation, Distributes code, Launches workers, Monitors computation all across the cluster.

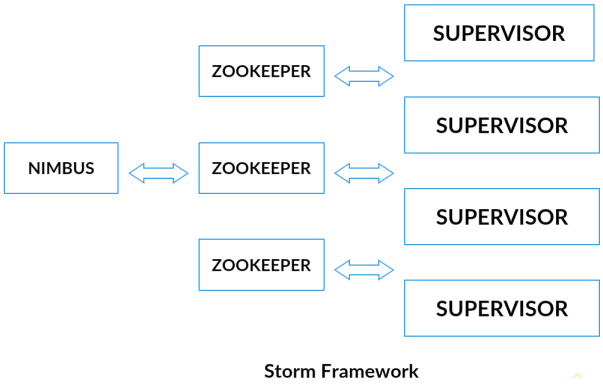

Figure 2. Storm framework.

2. Zookeeper nodes to coordinate the storm cluster.
3. Supervisor nodes communicate with nimbus via the Zookeeper nodes. Data Stream flow in storm comprises of Spout which is the source of streams, Bolt which has the computational logic and tuple which has the values (logs) (Figure 2).

As far as this paper is concerned the real time data is pushed on to Apache Kafka. Kafka consists of a message producer, message consumer and message broker. The producer forms the front end part. Here it is the data server. The Brokers passes on this to the consumer which is the storm framework that performs data processing. Zookeeper continuously monitors the working of the kafka and storm cluster. The data is input in a JSON format. For implementing the logic, we go for two classes Topology and Bolt. Inside Storm Topology the configuration code with details regarding storm, kafka, zookeeper is written (Similar to configuration settings in main method of mapreduce)[5]. Storm Bolt which implements IBasicBolt is used to implement the processing logic. When Storm Topology (main) is executed the Storm Bolt is called from within and the streamed sensor data is checked for any discrepancies. Here the uniqueid is the vehicleId. The data creeps in at a high throughput rate at various timestamps. If found faulty an entry is made into a cassandra/Hbase table using an update statement. Cassandra/Hbase is a column oriented database that has fast operating capabilities (Figure 3).

All daemons relating to kafka, storm and zookeeper must be started prior to the code execution. An alternative approach is that if any faulty readings are observed we may put on to another topic in kafka which is pulled by the data server periodically. As a part of future work a driver warning system can be developed in automobiles. Thus

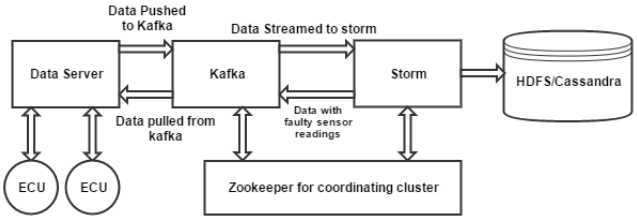

Figure 3. Storm streaming process flow.

real time vehicular monitoring is accomplished by using Apache Storm, Apache kafka and cassandra/Hbase.

2.2 Spark Streaming

In the third methodology spark streaming application is developed. Though storm stream processing and hadoop batch processing offers low latency we lacked a single system which combined the effect of both. While performing statefull computations using external databases storm is much slower compared to spark. Spark framework has comeup with the solution for this. Spark has been optimised to process batches in milliseconds thus achieving low latency. Spark streaming is built on top of spark combining both batch jobs and streams[5]. Advantage is that we need to design algorithm once and run it on spark standalone mode as well as on spark streaming. Spark streaming connectors includes kafka, hdfs, flume etc. Spark streaming[6] allows arbitrary RDD computations under the abstraction of Dstreams. Dstreams are sequence of RDDs[7]. The system is fault tolerant that is it keeps track of parent RDDs. Master or Driver saves the state of Dstreams to a check point file. In case the master fails it can be restarted using the check point file. The framework periodically saves the DAG (Directed Acyclic Graphs) of Dstreams to fault tolerant storage, probably a check point directory that should be configured initially. Automatic restart is possible. Spark along with shark and spark streaming forms a unistack that can solve all your data analytics (Figure 4).

Spark framework can be configured as standalone/MESOS/YARN[8]. The user code runs on the driver and tasks are send to the executors. Executor which is the receiver collects the data stream and divides into blocks and keep in memory. Blocks are synchronously replicated to another executor for fault tolerance. In every batch interval the driver launches tasks for processing the blocks. Here every data is processed exactly once unlike the storm where data is processed at least once. The same

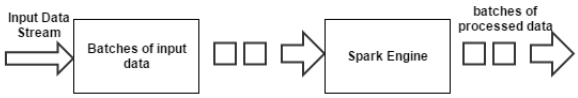

Figure 4. Spark streaming framework.

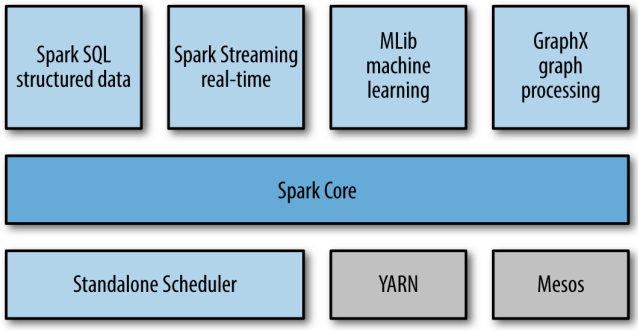

Figure 5. Spark framework.

logic implemented in the hadoop and storm can be easily implemented here also (Figure 5).

Create streaming context as an entry point of streaming functionality. Then create Dstreams from kafka data using the KafkaUtils functionality. The transformations are applied on this. That is the streaming data with faulty sensor readings are immediately written to HDFS/Cassandra/Hbase table or an arbitrary computation to update a UI. More logic if needed can be applied. Dstreams[9] after processing may be also stored to an HDFS location using saveAsHadoopFiles() function. Writing to kafka may be accomplished by using create Kafka Producer Pool functionality. After writing the logic in scala[10] 'context.start' function commands the system to start data streaming[11]. The data server periodically pulls the files with faulty sensor readings which can be send back to the ECU equipped with a GSM module (Figure 6).

2.3 Data Set Description and Performance Statistics

Automotive data has been analysed here on a time critical basis. A record in a file is comprised of a Vehicle Id(unique), sensor1 reading, sensor2 reading, sensor3 reading followed by timestamp. The data server pushes the data as JSON by means of kafka to storm/spark. The performance characteristics has been projected on the bargraph.

The data has been subjected to analysis on a 5 Node cluster where one node acts as the master and the others

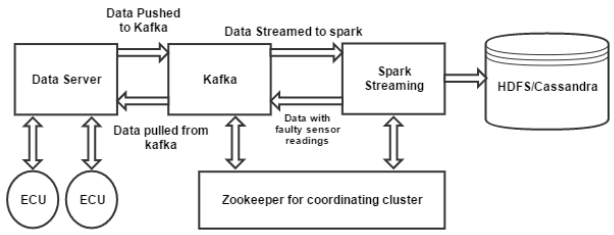

Figure 6. Spark streaming process flow.

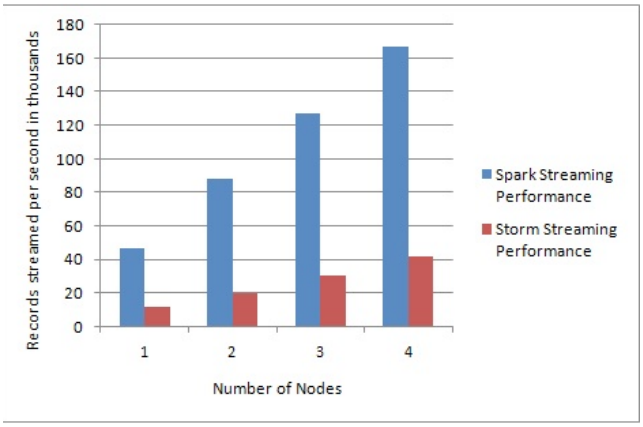

Figure 7. Output performance statistics.

as slaves(executors). Each node has 32 GB RAM, 3 TB Hard Disk, 8 core i7 extreme processors (Figure 7).

2.4 Proposal for a Low Cost ECU

ECU stands for electronic control unit. It can be also referred to as Engine Management System. This paper details the development of an ECU capable of performing or controlling fuel injection, ignition, fuel pump, idle air valve, engine coolant temperature, Air pressure sensor etc. This low cost ECU is designed using an STM32F4 Discovery dev board. Along with a Real Time Operating system (RTOS) such as the chibios the ARM board creates a splendid development environment for the Automotive ECU developers. Using an RTOS can reduce the development time, code implementation with minimal bugs and making production costs economical. The low cost ECU here is tested on a 1999 Matiz engine with limited functionalities.

Let us see how fuel injection system is controlled by this ECU. The fuel injection system consists of a fuel pump, fuel injectors, fuel pressure regulators, ECU, wiring harness and engine sensors. For an EFI (Electronic Fuel Injection) the inputs from following sensors are required: oxygen sensor, Manifold Air Pressure sensor (MAP), air temperature sensor, coolant sensor and throttle position sensor. The fuel pump, pumps the fuel from a fuel tank through a fuel filter to the injectors via a supply line based on the throttle position sensor input. The fuel pressure regulator senses the pressure and ensures that the injectors maintain the desired pressure level[12]. A pathway is created for the unused fuel to return back to the fuel tank after providing proper lubrication to the injector components. Such a closed loop fuel supply system guarantees consistent fuel spray and fuel quantity from each of the injectors. The wiring harness connects the injectors to the ECU and to a power source.

The sensors mentioned above periodically sends their readings to the ECU (here ARM board)[13]. As an example if high acceleration or torque is required, the throttle/gas pedal is pushed fully and the throttle valve opens to its maximum, pulling in large volume of air. The current throttle position sensor reading stimulates the arm board via the interrupt level subroutines. Multithreading is utilised to achieve parallelism and for utilising the available ADC's effectively. Pulse width modulation is used for regulating amount of fuel. Higher duty cycle implies a large ON time pulse that is more quantity of fuel. A series of commands are generated by the ECU[14]. ECU(ARM Processor) stimulates the fuel pump to pump in more quantity of fuel. The pressure regulator sensor is instructed so that the fuel is pumped at the adequate pressure corresponding to the respective throttle position after cross validation from a look up table (already mapped engine parameters). Air temperature sensor reading helps to calculate the rate of combustion occurring in the chamber through a mapping. Similarly, the data from manifold air pressure sensor is used to calculate density and determine the engine's air mass flow rate that aids the ECU to determine the quantity of fuel for optimum combustion.

The oxygen sensors regulate the fuel injection by calculating the oxygen concentration difference between the inside of the exhaust and outside atmospheric content. These sensors are used to check the Air/Fuel ratio (14.7 ideal). The difference in concentration is converted to a voltage spike. A rich mixture generates a high voltage spike (converted by ADC in the STM32 board). The so called interrupt computation sends an instruction to pump less fuel to the fuel pump motor. As of now these functionalities have been implemented in this ECU.

2.5 ECU Network Communication using GSM/GPRS

A GSM (Global System for Mobile Communication, originally from Group Special Mobile) modem is a wireless modem that works with a GSM wireless network. A GSM modem interfaced with the ECU[15] makes the communication possible with a central data server. The ECU's equipped with this GSM module from different automobiles are wirelessly connected to the central data server. This ECU data collected in the central data server is taken for real time data analytics in spark, storm, hadoop etc. The GSM data may be send in the form of a string to the data server where it may be converted to JSON or other convenient forms.

3. Conclusion

We have presented a software model for performing real time big data analytics on vehicular data. The analytics has been performed on bigdata frameworks like hadoop and spark. Batch processing model is developed using hadoop mapreduce and hive. Real time stream processing has been achieved through spark streaming and storm streaming. A comparison of the same has been done. A proposal model for designing a cost effective ECU and its wireless networking with a central data server is discussed. In the future scope a fully functional ECU with immense capabilities that can communicate with the central data server will be developed. A highly reliable real time driver warning system development is also there in the future scope. Real time networked automobiles can contribute to safety, enhance engine health and performance, optimised research and development and much more.

4. Acknowledgement

The authors would like to express their gratitude towards Ajith Peter and Achyuth Manalithazha for their valuable comments and reviews on this topic.

5. References

1. Gopalani S, Arora R. Comparing apache spark and map reduce with performance analysis using K-means. International Journal of Computer Applications. 2015 Mar; 113(1):8–11.
2. Spark Streaming [Internet]. [Cited 2016 Jan 19]. Available from: http://spark.apache.org/streaming/.
3. Purcell B. The emergence of big data technology and analytics. Journal of Technology Research. 2013 Jul; 4:1.
4. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: Cluster computing with working sets. University of California, Berkeley; 2010 Jun. p.1–7.
5. ZahariaM, BorthakurD, SarmaJS, ElmeleegyK, Shenker S, Stoica I. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling.EuroSys; 2010 Apr. p. 1–14.
6. Zaharia M, Das T, Li H, Shenker S, Stoica I. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. University of California, Berkeley; 2012 Jun.
7. ZahariaM, ChowdhuryM, DasT, DaveA, MaJ, McCauleyM, Franklin M, ShenkerS, StoicaI. Resilient distributed datasets:A faulttolerant abstraction for in-memory cluster computing. NSDI; 2012 Apr. p. 2-2.
8. Spark Kafka Integration[Internet]. [Cited 2016 Jan 15]. Available from:http://spark.apache.org/docs/latest/streaming/kafka/integration.html.
9. Spark reference databricks[Internet]. [Cited 2016 Jan 18]. Available from: https://databricks.gitbooks.io/databricks spark reference applications/.
10. Scala programming language[Internet]. [Cited 2016 Feb08]. Available from: http://www.scala-lang.org.
11. LogothetisD, OlstonC, ReedB, WebbKC, Yocum K.Stateful bulk processing for incremental analytics. SoCC; 2010 Jun. p. 51–62.
12. Ajudia MK, Kolte MK, Sarkar P. Validation process and development of control strategy of electronic control unit for injector and ignition coil drivers. International Journal of Scientific and Research Publications. 2014May;4(5):1–5.
13. Zeng J, Zhang L, Kong F, Song X. Development of 32-bit universal electronic control unit UECU32 for automotive application. 2006 9th International Conference on Control, Automation, Robotics and Vision; 2006 Dec. p. 1–6.
14. Cebi A, Guvenc L, Demirci M, Karadeniz CK, Kanar K, Guraslan E. A low cost portable engine electronic control unit hardware in-the-loop test system. Proceedings of the IEEE International Symposium on Industrial Electronics, Dubrovnik: Croatia. 2005 Jun; 1:293–8.
15. Huizong F, Ming C, Yu Z, Jianchun J, Huasheng D. A weak coupled calibration system architecture for electronic control unit. IEEE Vehicle Power and Propulsion Conference (VPPC),China; 2008 Sep. p. 1–4.