The MapReduce-based approach to improve vehicle controls on big traffic events

Conference Paper · April 2017

DOI: 10.1109/LOGISTIQUA.2017.7962864

4 authors, including:

Hamilton Wilfried Adoni
Université Hassan II de Casablanca
4 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Tarik Nahhal
Faculty of Science, Hassan II University
9 PUBLICATIONS **8** CITATIONS

SEE PROFILE

Brahim Aghezzaf
Université Hassan II de Casablanca
35 PUBLICATIONS **398** CITATIONS

SEE PROFILE

The MapReduce-based approach to improve vehicle controls on big traffic events

Wilfried Yves Hamilton Adoni*, Tarik Nahhal*, Brahim Aghezzaf*, and Abdeltif Elbyed*
*Laboratoire Informatique, Modélisation des Systèmes et Aide à la Décision (LIMSAD)
Hassan II University of Casablanca, Faculty of sciences, Km 8 Route d'El Jadida B.P 5366 Maarif Casablanca 20100, Morocco
adoniwilfried@gmail.com, {t.nahhal, b.aghezzaf, a.elbyed}@fsac.ac.ma

***Abstract*—Recently, with the integration of Hadoop framework in the resolution of many related transportation problems, it has been show that the concept of parallel and distributed computing plays an important role in the management of large-scale traffic data. The advantages of Hadoop components allow intensive computing with huge volume of streaming data which cannot be processed by traditional Intelligent Transportation System (ITS). In this paper, we present a MapReduce-based approach to analyze big stream data in order to detect abnormal traffic events. Our approach consists in partitioning the big log file of traffic events into a set of sub-events then analyzing traffic events on each event block in parallel way. Experimental tests reveal that the proposed approach is very inspiring and achieves significant gain in term of calculation time.**

***Index Terms*—Hadoop, MapReduce, Big Traffic Data, Apache Flume, Vehicle fraud, Parallel and Distributed computing.**

I. Introduction

With explosion of connected devices, road sensors, embedded GPS system and electronic data interchange, transport generates today an exponential amount of data. Everything is recorded: the entries and exit of travelers, incidents, delays, congestions, traffic jams, vehicles tracking, etc. Combined and analyzed, these data represent a deposits of powerful knowledge to understand complex phenomena such as changes in traffic flow and traveler behavior. In this context, big data concept is well suited to understand the heterogeneous masses of traffic data, analyze the traveler behavior and help to effective decision for transport companies. The increasing mobility of people and introduction of new information technologies in the transport sector generate a significant amount of digital data. Big data can analyze real-time information related to traffic conditions and propose models to reduce transportation costs. It may be visual representations of maps, statistical analysis, real-time analysis, and traffic prediction with any kind of data.

Big Data has attracted great research interest for many city governments and research institutes in the field of urban transportation planning and management [1]–[3]. Kitchin and Rob [4] detail all related projects that use big data as technology to produce deep analysis and provide some efficient information for smart city management. They also focus on future problems related to the phenomenon of data explosion. In 2000, more than 800,000 petabytes (2^{50} bytes) of data have been produced [4]. Concerning road traffic, there are more than 1 billion cars running on all the roads around the world [5], [6], and all datasets produced by the travelers are increasingly supplemented by other data massively acquired on real-time by a growing number of traffic monitoring device. For example, in Morocco 10 million displacement are performed every day in the city of Casablanca and 1 million vehicles run daily. This provokes considerable traffic problems causing a significant cost for utilities and transportation companies: 1) at the social and ecological level: mental stress in driving [5], difficulties in accessing the work place and public services, deterioration of living conditions, significant increase of air pollution and 2) at the economic level: higher fuel/gasoline consumption because of the time lost in traffic congestion [5].

There are lot of suggestions to prevent bad events on urban road traffic: 1) reducing the number of trips; 2) switch to other travel modes or promote the utilization of public transportation modes and 3) improve ITS performance with big data approach to react quickly to all traffic events. It's not possible to reduce the demand of travelers and/or encourage travelers to change their travel mode because of user preferences and difficulties to access transport modes, finding quickly traffic congestions and vehicle collisions through Hadoop concept are considered most feasible and economic. In this sense, MapReduce-based approach is well suited to detect traffic events with big volume of data acquired on real-time. Our objective is to prove the efficiency of parallel processing in road traffic control.

The rest of this paper is organized as follows. Section 2 and 3 present respectively related works and Hadoop framework. Section 4 introduces our proposed approach. In section 5, we validate our approach with experimental tests and finally conclude the paper with outlook on future work.

II. Related works

Most systems offer multimodal services for transport companies however, difficulties arise when it's an individual demand triggered by a traveler. In this sense, some conducted works have estimated the individual request of a traveler from phone calls. The system analyzes the call detail records (CDR) from mobile phone to estimate variations of travel demand [7]. By combining data from mobile with traffic data, they [7] created an origin-destination matrices and applied traditional approach to estimate traveler demand. This approach is limited to small road networks. In case of large road networks, it's possible to create an image of traffic zone division based on big data from mobile phone base stations [8]. According to [9], each zone presents different characteristics that affect the complexity of traffic time series and making reliable at a reasonable cost. They introduced multiple-step planning to process the dataset based on time series. This is better suited for causality and regression analysis. The first step consists to use Granger causality and decomposition algorithm to separate trend and unsteady bursts. The experimental results showed that the statistical model is most efficiency with the concept of big data and presents a better forecast quality when we treat large volumes of traffic data.

Dodge and Kitchin [10] have implemented a transportation system based on big data solution which analyzes on real-time the movement of vehicles on road network. By combining data from cameras and data from road sensors, they estimated vehicle speeds and automatically administering penalties for violation of speed limit. The police services used related information to react immediately to incident detection and to generate a log file containing all record of vehicle tracking.

Lécué Freddy et al. [11] have implemented ITS to analyze and predict the traffic events of the city of Dublin in Ireland. The methodology employed is based on semantic web technology which integrates historical amount of sensor data. The system ensures handling large data generated on real-time. This allows to react instantly to traffic events and to offer an efficient urban planning. The predictive method implemented enables an estimate of future observations based on historical data. This also allows to determine the future states of road segment in order to anticipate events. The system has been experienced in Dublin and tests performed in Miami USA, Rio Brazil and Bologna Italia.

Other works [1], [2] have led to implement a monitoring system that is able to process traffic data flow of Stockholm city. By combining data from GPS sensors and road network, their system provides real-time mobility aid information. The research team used IBM Big Data solution: InfoSphere Streams for processing large volume of data from 120,000 GPS sensors per second combined with 600,000 routes to generate different kinds of statistical models and respond instantly to customer queries.

Some studies showed that it is possible to create an image of traffic from sensor data. Bouillet and Ranganathan [3] have implemented an ITS that provides a picture of traffic conditions based on data from GPS sensors. The estimation of travel time between two points and the spatio-temporal representation of traffic conditions has been feasible by collecting many GPS data such as data from taxis, public transport, private vehicles, utility vehicles, etc. The traffic flow generated on real-time are managed by stream process provides by InfoSphere Streams. The results showed that the system can handle GPS data from 1 million GPS access points per second and provides a cartographic representation with 1 trillion links.

In china metropolitan areas, several urban mobility projects were conducted to ensure the regulation of traffic flow. In this sense, all authors carried out a study on the rise of big data on urban studies and planning practices in China [12]. By using big data, they analyzed the traffic flow, behavior of travelers and traffic jams. After having carried out the analytical study, they set up a decision-support plan and have implemented new methodology for traffic data management by using big data tools. This study allowed to identify new issues such as the challenges related to Chinese population growth, congestion, traffic jams and air pollution.

Big data application in real-time is most efficient to collect traffic data and manage congestions. Other work [13] conducted in road network control has led to the adoption of a surveillance strategy based on the real-time data. By combining data from 275 detectors deployed on the roads of Orlando, they have improved the system performance by reducing congestion and crash risks.

III. Hadoop framework

Hadoop is an Apache open source framework designed to facilitate the creation of scalable and distributed computing of large amount of data [14]. Hadoop consists of two main components: MapReduce [15] and Hadoop Distributed File System (HDFS) [16]. It supports several related projects such as Apache Flume [17], HBase, Hive, Pig and more others.

A. Hadoop Distributed File System

HDFS [16] is a distributed file system designed to support very large file of data on commodity hardwares. Each big file is split into little blocks of file and replicated across Hadoop nodes (computers) for fault tolerance. HDFS works as master-slave architecture, the master node (NameNode) manages the file system, namespace and metadata while the slave nodes (DataNode) manages the storage of block files.

B. MapReduce

MapReduce [15] is a framework allowing intensive calculation of large among of datasets in distributed environment within a cluster of Hadoop nodes and it needs HDFS for good parallelism with data I/O. MapReduce also works as master-slave architecture, the master (JobTracker) assigns and controls all tasks on multiple slave nodes (TaskTracker). As show in Fig. 1, MapReduce runs by dividing the job into two main phases: Map and Reduce phase. In the map phase, the mappers take pieces of files into the HDFS and process each piece line by line to produce a set of intermediate <key, value> pairs. When all map tasks are completed, the results are sent to the reduce phase. The reducers take the results of the mapper phase as input and merge all intermediate values with same keys to produce a set of smaller <key, {value, ..., value}> pairs. The final results are stored into the HDFS when all tasks are completed.

C. Apache Flume

Apache Flume [17] is another related project based on Hadoop kernel, it's distributed, scalable and reliable service for efficiently collecting, aggregating and transporting massive

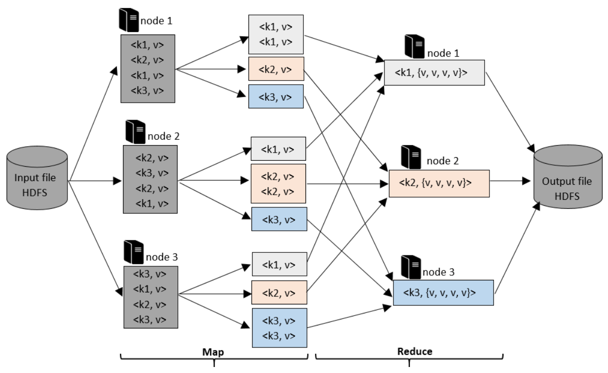

Fig. 1. MapReduce framework overview

quantities of log events to centralized them into HDFS. The Flume graph flow is composed of a set of agents connected. As show in Fig. 2, the Flume agent consists of three components: Source, Sink and Channel. A Flume event contains a byte array that is to be moved into the HDFS accompanied by the event header.

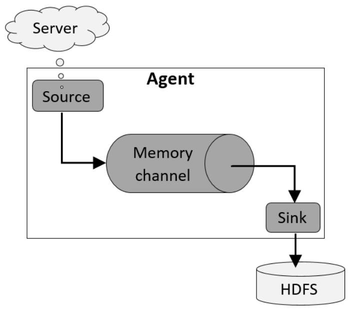

Fig. 2. Flume agent component

IV. Proposed Framework

We use the advantage of Apache Flume to collect and move big volume of traffic stream from various sources (GPS system, road sensor, radar system, controller cabinet, video surveillance, etc...) into the HDFS. Thereafter, we use the concept of parallelism provided by MapReduce framework to search in parallel way all traffic events. The proposed framework (see Fig. 3) consists of five main phases: Flume phase, Input phase, Map phase, Reduce phase and Output phase.

A. Flume phase

The Flume phase consists to collect big volume of traffic events via Flume agents. Each agent is assigned to each slave node, and works in parallel way according to the configuration number of tier agents in the Flume. The first tier agents (Agent Tier-1) use a source component to receive traffic events from road sensors and transport them into the channel component. The channel buffers the incoming events until they are consumed by Avro Sink. The Sink removes the traffic events from the channel and sends them to the second tier agents (Agent Tier-2). The second tier agents merge, group the traffic events based on information contained into the event headers and buffer them in the channel until they are consumed by the HDFS Sink. Finally, the HDFS Sink removes the events from channel and stores them as key-value pair into the HDFS.

B. Input phase

When the set of traffic events E with specific size: E_{size} is stored into the HDFS, Hadoop splits the file event into little piece of sub-events based on the defined block size: B_{size} and replicates them across the set of slave nodes for good parallelism and fault tolerances. The total number of event block is calculated as:

$$N_{block} = E\left[\frac{E_{size}}{B_{size}}\right] + \theta \tag{1}$$

where $\theta = \begin{cases} 0, \text{ if } E_{size} \equiv B_{size} \\ 1, \; else \end{cases}$

C. Map phase

This phase consists to compute and detect abnormal events such as speeding on the road and abnormal vehicle behavior. In map phase, each map task (mapper) is assigned to each block of traffic event. The MapReduce framework allocates the number of mapper: N_{map} taking account of the total block of event: N_{block} consumed by the large-scale event file E. To ensure the running on all event blocks, the total number of mapper is calculated as follow:

$$N_{map} = N_{block}, \forall B_{size} \tag{2}$$

The mappers running across the Hadoop cluster nodes use **Event_Mapper** procedure (see Algorithm 1) to search and locate abnormal traffic events on the road. Each map task transforms line by line the input of traffic event blocks as a set of key-value pairs to produce another key-value pairs of intermediate traffic events. Next, the results will be sorted and stored locally until all slave nodes complete these tasks before sent them to the reducer phase.

D. Reduce phase

Each reduce task (reducer) takes intermediate events from the mapper node that it's able to access. Next, it aggregates and concatenates the set of intermediate traffic events which share the same key headers and generates a smaller key-value pair of final traffic events. This is achieved by the procedure **Event_Reducer** (see Algorithm 2). Moreover to ensure the good parallelism, the right number of reduce task is calculated as follow [14]:

$$N_{red} = 0.95 \times N_{node} \tag{3}$$

We suppose that all slave nodes have the same hardware configurations (ram memory and CPU) and assuming that the reducer nodes are able to access the results of the map phase.

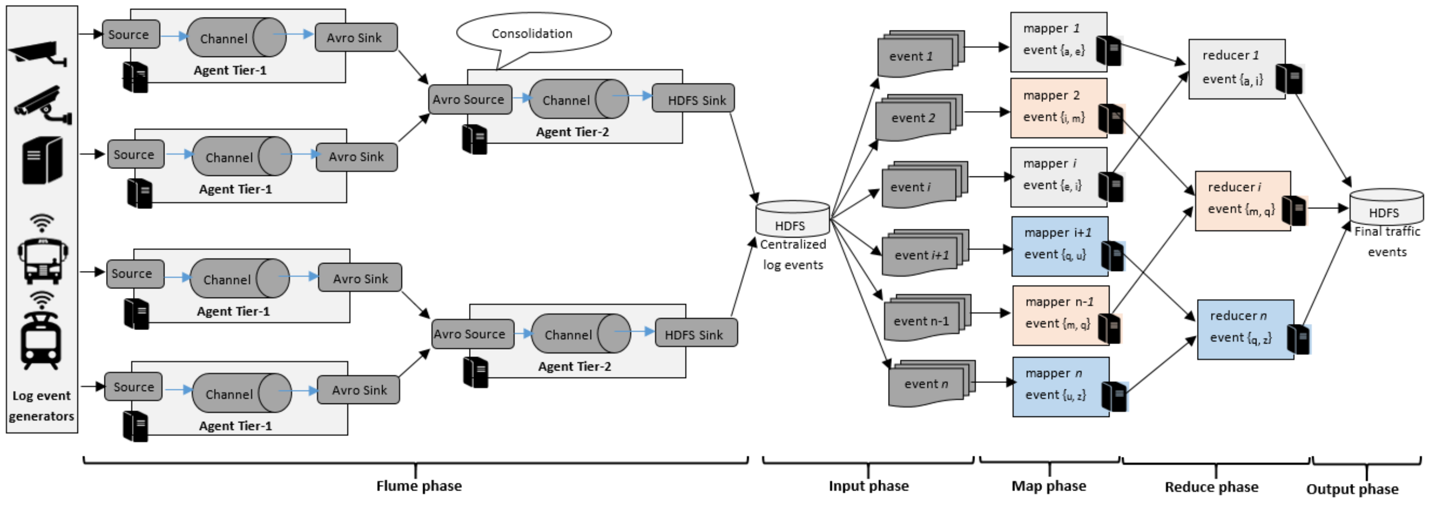

Fig. 3. Proposed framework in practice

Algorithm 1 Event_Mapper

```
Input: block of log events from HDFS .
    // $h_i$ : event i header
    // $v_i$ : vehicle i id
    // $x_i, y_i$ : vehicle i coordinates
    // $s_i$ : vehicle i speed
    $< key \leftarrow h_i$ , $value \leftarrow \{v_i, x_i, y_i, s_i\} >$
            ...        ....        ....
    $< key \leftarrow h_n$ , $value \leftarrow \{v_n, x_n, y_n, s_n\} >$
 1: for $line \in$ input do
 2:     $h \leftarrow line.key$
 3:     $byte \leftarrow line.value$
 4:     $v \leftarrow byte.token[0]$
 5:     $x \leftarrow byte.token[1]$
 6:     $y \leftarrow byte.token[2]$
 7:     $s \leftarrow byte.token[3]$
 8:     if ($s > maxspeed$) then
 9:         $e \leftarrow Speeding(v, gps(x, y), s)$ // vehicle speeding
10:         $StoreLocally(< key \leftarrow h, value \leftarrow e >)$
11:     else if ($s = 0$) then
12:         $e \leftarrow Abnormal(v, gps(x, y))$ // abnormal behavior
13:         $StoreLocally(< key \leftarrow h, value \leftarrow e >)$
14:     end if
15: end for
16: <<map another event block if possible>>
17: <<wait until all mappers completed>>
18: <<send results to reduce phase>>
```

Algorithm 2 Event_Reducer

```
Input: set of intermediate traffic events from map phase .
    $< key \leftarrow h_i$ , $value \leftarrow \{e_1, e_2, .., e_i, e_n\} >$
1: $h \leftarrow line.key$
2: $byte[] \leftarrow line.value$
3: $q \leftarrow \oslash$ // event queue
4: for ($e \in$ byte[]) do
5:     $q \leftarrow q \cup e$
6: end for
7: $SendToMasterNode(< key \leftarrow h, value \leftarrow q >)$
8: <<reduce another event set if possible>>
9: <<wait until all reduce tasks completed>>
```

E. Output phase

Finally, the master node uploads the final events from each reducer into the HDFS. Each reducer output is written into separate files, the number of file is equal to the number of reducer: N_{red}. To ensure fault tolerances, each file is replicated across the cluster nodes.

V. Experimental Results

To prove the efficiency of our framework, we perform experiment tests on a 4-node Hadoop cluster (see Fig. 4) consisting of 1 master node and 3 slave nodes. The hardware configuration of the cluster is 8 GB RAM for the master node and 2 GB RAM for the slave nodes based on Intel Core i5-2410M CPU 2.30 GHz running under Linux Red Hat Enterprise 6 64 bit. The number of core used in the map and reduce phase is set to 1. The tests have been achieved on Hadoop 2.2.0

We test the performance of our framework by generating big log of artificial traffic events. We evaluate the system capability to collect and support large volume of events before applying intensive calculation to detect abnormal traffic events. Fig. 5 shows the number of traffic events collected per second during the Flume phase. We remark that more than 2 million events have been collected and stored into the distributed file system in 1 minute. This can be improved by adding additional nodes in the Hadoop cluster.

Fig. 6 shows the influence of the increasing number of Hadoop node on the calculation time. We remark that, the addition of new nodes in the Hadoop cluster impacts on the

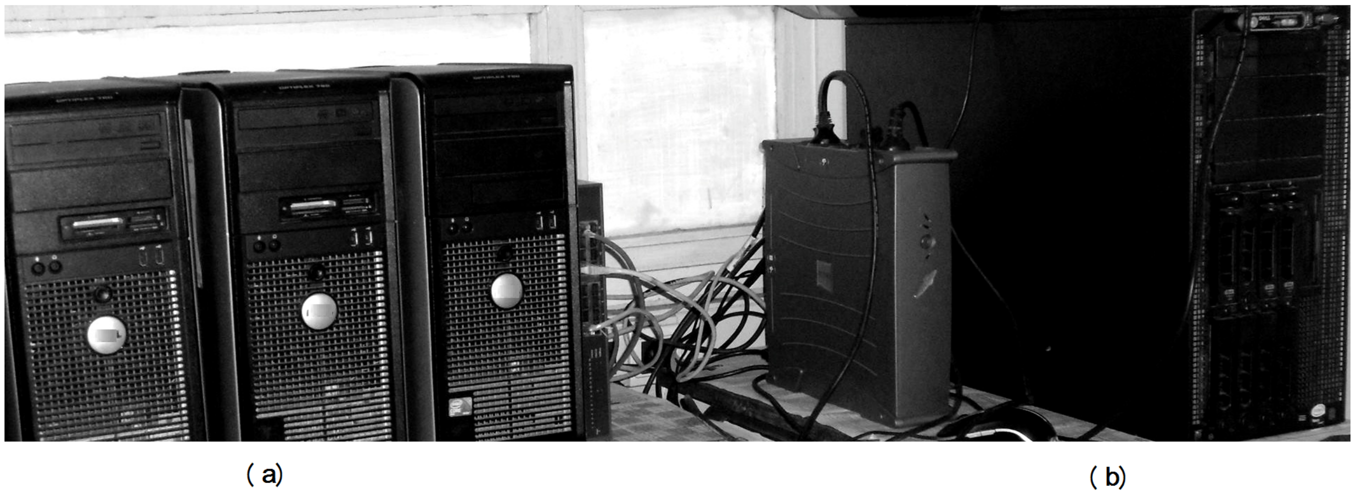

Fig. 4. Hadoop cluster of 4 nodes: (a) the 3 slave nodes and (b) the master node

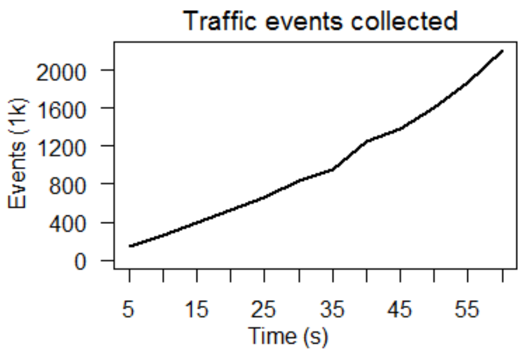

Fig. 5. Number of traffic events collected per second

computational time. For example with log event file of 7 GB (E_{size} = 7 GB), the time decreases from 223 s to 151 s with 3 nodes and from 151 s to 85 s with 4 nodes.

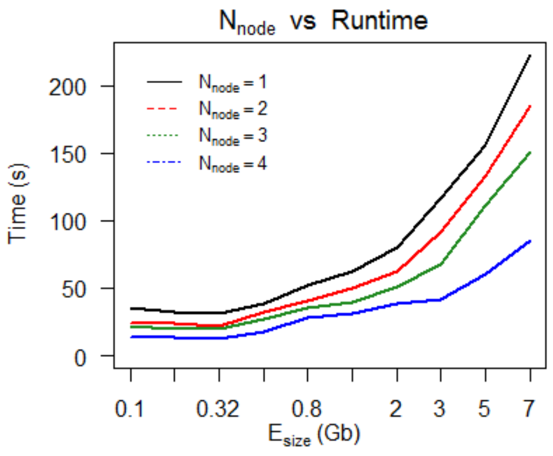

Fig. 6. Influence of increasing nodes on traffic events processing

Table I gives the experimental results of event detections from streaming data. We report all captured events in the table with the vehicles whose speed exceeds the speed limit (column 2 and 3) and vehicles with abnormal behavior (column 4 and 5). In term of veracity, the relative gap between counted and detected events remains constant value (= 1 for all), this means that the proposed approach offers the best performance and captured all abnormal events from vehicle stream.

TABLE I
EXPERIMENTAL RESULTS OF EVENT DETECTIONS.

Traffic stream	**Speeding**		**Abnormal event**	
No. of events	Counted	Detected	Counted	Detected
10x10^3	63	63	12	12
50x10^3	96	96	29	29
10x10^4	188	188	34	34
50x10^4	277	277	48	48
1x10^6	427	427	58	58
5x10^6	524	524	80	80
10x10^6	748	748	92	92
25x10^6	1478	1478	136	136
50x10^6	3472	3472	253	253
75x10^6	4128	4128	312	312
100x10^6	6174	6174	448	448

Table II presents an evaluation between our framework and proposed framework in [1], [2] based on criteria defined in column 1. At first glance, we remark that our framework requires less ram memory, CPU speed, nodes and core used per slave nodes. Compared to [1], [2], our ITS presents best performance. We can handle a maximum throughput of 142000 GPS points per second versus 100000 GPS points per second with a cluster of 6 nodes.

VI. CONCLUSION

In the present work, we prove the concept of parallel processing in road traffic monitoring. Our approach consists in partitioning the large-scale traffic events under sub-events and computes each block in parallel way to analyze driver behavior. The proposed framework is scalable and suited to collect, store and manage large volume of traffic events in

TABLE II
OUR CONTRIBUTION VERSUS WORK CONDUCTED IN [1], [2]

Criteria	**[1], [2]**	**Proposed framework**
No. of GPS point per second	[50000,100000]	142000
No. of node	[4,6]	4
No. of core used per node	4	1
Ram (GB)	16	2
CPU (GHz)	3	2.3
Approach	Stream computing	Flume processing + MapReduce paradigm
Platform	IBM InfoSphere Stream	Native Hadoop

optimal time with commodity hardware. Experimental results prove that our approach achieves significant gain of time. Big Data technology presents great interest in transportation research because of it's capability to face the challenges of velocity, variety, veracity and volume of traffic data. This work can arouse curiosity for urban transportation planning agencies and emergency services because it could improve the response time and enhance the quality of offered service. The future works consist to:

- Adapt advanced method of traffic analysis;
- Reuse the same approach to detect traffic congestions;
- Study deeply the time complexity of the proposed framework;
- Apply experiment tests on real case, especially with public transportation vehicles of Casablanca city.

ACKNOWLEDGMENT

The work is supported in part by the Laboratoire Informatique, Modélisation des Systèmes et Aide à la Décision (LIMSAD) of Hassan II University of Casablanca.

REFERENCES

[1] A. Biem, E. Bouillet, H. Feng, A. Ranganathan, A. Riabov, O. Verscheure, H. Koutsopoulos, and C. Moran, "IBM infosphere streams for scalable, real-time, intelligent transportation services," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, ser. SIGMOD '10. ACM, pp. 1093–1104.

[2] A. Biem, E. Bouillet, H. Feng, A. Ranganathan, A. Riabov, O. Verscheure, H. N. Koutsopoulos, M. Rahmani, and B. G, "Real-time traffic information management using stream computing." vol. 33, no. 2, pp. 64–68.

[3] E. Bouillet and A. Ranganathan, "Scalable, real-time map-matching using IBM's system s," in *2010 Eleventh International Conference on Mobile Data Management (MDM)*, pp. 249–257.

[4] R. Kitchin, "The real-time city? big data and smart urbanism," vol. 79, no. 1, pp. 1–14.

[5] D. Zhang and L. Xiong, "The research of dynamic shortest path based on cloud computing," in *Computational Intelligence and Security (CIS), 2016 12th International Conference on*. IEEE, pp. 452–455.

[6] IBM smart traffic (2010). [Online]. Available: http://www.ibm.com/smarterplanet/us/en/traffic_congestion/ideas

[7] J. L. Toole, S. Colak, B. Sturt, L. P. Alexander, A. Evsukoff, and M. C. Gonzlez, "The path most traveled: Travel demand estimation using big data resources," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 162–177.

[8] H. Dong, M. Wu, X. Ding, L. Chu, L. Jia, Y. Qin, and X. Zhou, "Traffic zone division based on big data from mobile phone base stations," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 278–291.

[9] L. Li, X. Su, Y. Wang, Y. Lin, Z. Li, and Y. Li, "Robust causal dependence mining in big data network and its application to traffic flow predictions," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 292–307.

[10] M. Dodge and R. Kitchin, "The automatic management of drivers and driving spaces," *Geoforum*, vol. 38, no. 2, pp. 264–275, 2007.

[11] F. Lcu, S. Tallevi-Diotallevi, J. Hayes, R. Tucker, V. Bicer, M. Sbodio, and P. Tommasi, "Smart traffic analytics in the semantic web with star-city: scenarios, system and lessons learned in dublin city," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 2728, pp. 26–33, 2014.

[12] J. Hao, J. Zhu, and R. Zhong, "The rise of big data on urban studies and planning practices in china: Review and open research issues," *Journal of Urban Management*, vol. 4, no. 2, pp. 92–124, 2015.

[13] Q. Shi and M. Abdel-Aty, "Big data applications in real-time traffic operation and safety monitoring and improvement on urban expressways," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 380–394.

[14] Apache hadoop. [Online]. Available: http://hadoop.apache.org/

[15] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," vol. 51, no. 1, pp. 107–113.

[16] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. New York, NY, USA: ACM, pp. 29–43.

[17] Apache flume. [Online]. Available: https://flume.apache.org/