Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 176 (2020) 938–947

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

A real-time Decision Support System for Big Data Analytic: A case of Dynamic Vehicle Routing Problems

Ines Sbai[a*], Saoussen Krichen[a]

[a] *Université de Tunis, Institut Supérieur de Gestion de Tunis, LARODEC Laboratory, Tunisia*

Abstract

Recently, the explosion of large amounts of traffic data has guided data scientists to create models with big data for a better decision-making. Big Data applications process and analyze this huge amounts of data (collected from a variety of heterogeneous data sources) that cannot be processed with traditional technologies. In this paper, Big Data frameworks are used for solving an optimization problem known as Dynamic Vehicle Routing Problem (DVRP). Hence, due to the NP-Hardness of the problem and to deal with a large size of data, we develop a parallel Spark Genetic Algorithm named (S-GA). This parallelism aims to take the advantage of Spark's in-memory computing ability (as a master-slave distribution computing) and GA's iterations operations. Parallel operations were used for fitness evaluation and genetic operations. Based on the parallel S-GA a decision support system is developed for the DVRP in order to generate the best routes. The experiments show that our proposed architecture is improved due to its capacity when coping with Big Data optimization problems by interconnecting components and deploying on different nodes of a cluster.

Keywords: Big data analytic, Decision Support System, DVRP, S-GA, Spark

1. Introduction

Big data (BD) has a huge influence on logistics management due to the explosion of large amounts of traffic data. Big data analytics systems are required to have a modular architecture which makes them easy to develop and fix whenever an issue arises. In this paper we propose a real time big data architecture using a streaming engine composed of four modules, namely Data collection, processing, analytics, storage and visualisation. In the processing phase, to deal with a large size of data, we develop a parallel Spark Genetic Algorithm named (S-GA). This parallelism

* Corresponding author.
E-mail address: sbai_ines@yahoo.fr

1877-0509
10.1016/j.procs.2020.09.089

aims to take the advantage of Spark's in-memory computing ability (as a master-slave distribution computing) and GA's iterations operations. Parallel operations were used for fitness evaluation and genetic operations. To test this architecture, a case study has been presented based on a realistic online optimization of the well-known optimization problem of Dynamic Vehicle Routing Problem (DVRP). Based on the parallel S-GA a decision support system (DSS) is developed for the DVRP in order to generate the best routes. The experiments show that our proposed architecture is improved due to its capacity when coping with Big Data optimization problems by interconnecting components and deploying on different nodes of a cluster.

The rest of the paper is organized as follows. In Section 2, we introduce related work. Section 3 gives the description of the proposed big data analytic architecture. Section 4 describes our parallel S-GA for the processing phase. Section 5 presents our proposed DSS. Section 6 reports the evaluation of our algorithm and describes the experiments and analyses. Section 7 concludes the paper.

2. Related Work

In recent years, transportation including traffic control, is a challenges topic in Big Data analytic. In the literature, Jara et al. [18] presented the correlation of the traffic behaviour with respect to the temperature in the Santander City. Kitchen [19] detailed a number of projects that aim to produce a real-time overview and analysis of the city, and provided a critical reflection on big data and smart urbanism. Also, Naimi et al. [5] suggested a list of general requirements for big data smart city applications. They discussed the various challenges in this domain and identified several issues that may hinder big data applications development efforts. Rathore et al. [25] proposed a complete system, which consists of various types of sensors deployment including smart home sensors, vehicular networking, weather and water sensors, smart parking sensors, and surveillance objects. Sivarajah et al. [28] reviewed the past and current state of BD and BDA research published. Ahmad et al. ([2] , [3]) proposed a system of a complete four-tier architecture that selects features by using Artificial Bee Colony (ABC). The system were implemented using Hadoop and MapReduce with the ABC algorithm. ABC algorithm is used to select features, whereas, MapReduce is supported by a parallel algorithm that efficiently processes a huge volume of data sets. Gohar et al. [15] proposed a big data analytics architecture for intelligent transportation system composed of four modules, namely (1) Big Data Acquisition and Preprocessing Unit (2) Big Data Processing Unit (3) Big Data Analytics Unit and (4) Data Visualization Unit. Wang et al. [29] Wang et al. [30] studied the impact of big data on the logistics and its action mechanism based on the analysis of the characteristics of big data. Most of this above existing publication ignores the definition of the used BD architecture in more detail, including all the software components and how they are related.

Analysis of real time information can be used to support logistic value chains and processes. In this way, Barba-Gonzlez et al. ([12], [13]) presented a dynamic Big Data Optimization technique based on Spark clusters jMetalSP to minimize distance and travel time in transportation, which combines the multiobjective optimization features of the jMetal framework with the streaming facilities of the Apache Spark cluster-computing system. The authors generated a realistic use case based on the application of a multi-objective metaheuristic (NSGA-II) to solve the dynamic bi-objective Traveling Salesman Problem (TSP).

One of the most studied variant of TSP is Vehicle Routing Problem(VRP). In the classical VRP, several dynamic features are defined to cause the dynamic aspect. The most frequently considered are uncertain inputs such as new requests or cancellations, modification of size or deadline of orders. Since the problem is NP-Hard, several types of metaheuristic methods have been applied to solve DVRP, Mandziuk and Zychowski [21] presented a Memetic Algorithm consisting of GA with a local search based on adaptive heuristic operators sequences for order requests. Okulewicz and Mandziuk [23] solved requests-to-vehicles assignment by the Particle Swarm Optimization algorithm, route optimization by a separate instance of the PSO algorithm. Yang et al. [32] proposed an ant colony based meta-heuristic for solving the DVRP with time windows and also, AbdAllah et al. [1] solved DVRP using GA. They proposed a weighted fitness evaluation approach as an alternative for the biased time-based approach. Recently, Xu et al. [31] solved the DVRP using an Ant Colony Optimization , which is the traditional ACO fusing improved K-means and crossover operation. In addition, Saber et al. [26] proposed a self-adaptive evolutionary algorithms (EA) for solving DVRP. The proposed EA evolves a set of configurations including parameter values, operator types, combination of operators and order of operator invocation.

Parallel population-based optimization techniques are nowadays widely used for solving difficult optimization problems. In recent years, to deal with large-scale optimization problems, researches are oriented to use a parallelism with big data frameworks such as Map reduce [4] Spark [16]. Spark is based on the master-slave distributed computing model and it is suitable for the global and distributed model of GA parallelization [24]. Their are a few work that deal with a parallel Spark and GA, Hu et al. [17] used Parallel SGA to investigate the sensor placement problem in large scale water supply networks with the objective of minimizing the impact of contamination events. In addition, Maqbool et al. [20] proposed a Scalable GA (S-GA) for large-scale optimization problems using Apache Spark. S-GA aims to reduce the communication overhead of Apache Spark by optimal resource utilization for large scale optimization tasks. In our knowledge, this paper is the first one using the parallel S-GA for solving optimum problem(DVRP).

3. The proposed real time big data analytic Architecture

In logistics and transportation the main benefit of Big Data is to share a real-time traffic information (Neilson et al. [22]). However, having relevant information about the traffic events allows the manager to make an informed decision. For Example the traffic congestion due to accidents or poor weather may be avoided if the drivers receive the information as soon as possible. So, they can change their pathway to commute. Therefore, this prediction helps to reduce the use of fuel and to save time. A decision maker can receive traffic information in real time with a predictive analysis of traffic conditions. In this section, we present the proposed architecture for big data analytic in

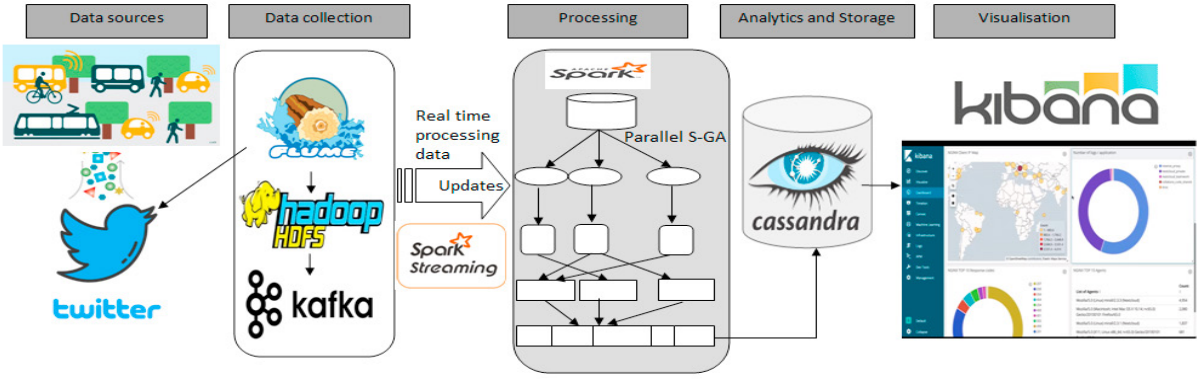

Figure 1: The proposed Big Data Analytic Architecture.

transportation area delineated in Figure 1. It consists of five functional layers: 1.Data collection, 2.Pre-processing, 3.Processing, 4.Analitics, 5.Visualisation detailed as follows:

1- Data sources:

A large and various size of data are collected from different source such as social Networks, sensors, etc. In the case of transportation, traffic data is collected from various sources for example, roadside sensors such as cameras and detectors used to detect speeds and time stamps of vehicles, radar, GPS-enabled smartphones and social media. The dynamic VRP consists in generating, in real-time, a new traffic information (example traffic congestion, roadworks, festival, strike, poor weather, etc). This encompasses a mix of structured, semi-structured and unstructured data.

2- Data collection

Data collection has an important role in the Big Data analytic cycle.

- Apache Flume: Flume [8] is a distributed service for collecting a large amounts of streaming data.
- Hadoop Distributed File (HDFS): HDFS [9] is a highly fault-tolerant designed to run on commodity hardware.
- Apache Kafka [6] is an open source big data processing framework. It is defined as a distributed streaming messaging system. In our case, data collection block is used for collecting traffic information data from various

sources including roadside sensors. Then, producer Kafka streams real-time data from roadside sensors monitoring devices. After that, Kafka cluster server stores publishers message for some period of time and publish them to a stream of data called topic. Topics are the core abstraction which Kafka provides for a stream of records. Each of these topics is divided and stored into a set partitions. A Consumer can join to one or more topics and consume the published messages by extracting data from the Kafka servers.

3- Spark Streaming: Data processing
The Apache Spark is the most suitable platform for dynamic data/stream-data handling, and for real-time data analytics (Chaudhari and Mulay [14]). The collected data streaming are transmitted, in real-time, in order to be proceeded. In our case of DVRP, a change of the environment can be detected in each time, so, Spark update automatically the streaming data. A parallel S-GA metaheuristic is presented to optimize the DVRP, which reacts when changes in the problem data are detected. Our parallel S-GA is detailed in the next section.
4- Apache Cassandra: Data and results storage
Apache Cassandra [7] is an open source, distributed and decentralized/distributed storage system (database). In our case traffic data obtained from Spark in real time are transmitted to Apache cassandra to be stored using datastaxs spark Cassandra connector library. This library provides API to store RDD to Cassandra by a saveToCassandra() method.
5-visualization: Kibana
Kibana [11] is an open source data visualization dashboard for Elastic search. It provides visualization capabilities on top of the content indexed on an Elastic search cluster The visualization techniques help in easy understandability of the results and help in decision-making of the city traffic.

4. The case of Dynamic vehicle routing problem using S-GA

4.1. The dynamic VRP

In DVRP, vertices change as the system and time(t) progress. DVRP is defined on a complete undirected graph $G = (V, E)$, where $V = \{0,\ldots,n\}$ is the vertex set and $E = \{(i, j)|i, j \in V, i \neq j\}$ is the edge set. Vertices {1,...,n} represent customers, each customer i is associated with a nonnegative demand q_i at time (t) to be delivered in the dynamic case by a vehicle k. Each customer has spatial coordinates (x_i , y_i), Vertex 0 represent the central depot at which a fleet of m homogeneous vehicles of capacity Q is based. Each vehicle starts its tour from the depot, visit all designated nodes, and then returns to the depot. $x_{ijk}(t)$, $y_{ik}(t)$ are a decision variables x_{ijk} takes 1 if vehicle k travels from customer i to j at t and 0 otherwise and y_{ik} takes 1 if vehicle k visits customer i at t and 0 otherwise.
A nonnegative cost, $c_{ij}(t)$, is associated with each edge $\{i, j\} \in E$ and represents the travel cost spent to go from vertex i to vertex j at t.
The mathematical formulation of the DVRP is expressed as follow:

$$Min \sum_{k=1}^{m} \sum_{i=0}^{n} \sum_{j=0, j\neq i}^{n} c_{ij}(t) x_{ij}^{k}(t) \tag{1}$$

S.t.

$$\sum_{j=1}^{n} x_{0j}^{k}(t) = \sum_{i=1}^{n} x_{0i}^{k}(t) = 1, \quad k \in \{1,\ldots,m\} \tag{2}$$

$$\sum_{j=1}^{n} x_{ij}^{k}(t) = 1, \quad 1 \leq i \leq n, \quad k \in \{1,\ldots,m\} \tag{3}$$

$$\sum_{i=1}^{n} \sum_{j=0, j\neq i}^{n} y_{ik}(t) q_i(t) \leq Q, \quad k \in \{1,\ldots,m\} \tag{4}$$

$$\sum_{i\in S_t}\sum_{j\in S_t} x_{ij}^k(t) \le |S_t| - 1, \quad k \in \{1,\dots,m\}, \quad S_t \subseteq V, \quad |S_t| \in \{2,\dots,n\} \tag{5}$$

$$x_{ij}^k(t), y_i^k(t) \in \{0,1\}, \quad i \in \{1,\dots,n\}, \quad k \in \{1,\dots,m\} \tag{6}$$

The objective function (1) consists of minimizing the total cost of all vehicles. Constraints (2) express that each travel should begin and end at the central depot. Constraints (3) provide that a single vehicle leaves each node i at t. Constraints (4) guarantee that the vehicle capacity is not exceeded. Constraints (5) eliminate the sub tour. Constraints (6) state the binary nature of the variables.

4.2. The proposed Spark-GA

Algorithm 1 The proposed S-GA

Input: :
popsize, t:number of generation, Partition, maxgeneration; # GA parameters
Begin
Phase 1: Parallel Fitness Evaluation
Generate the initial population : Randomly
Population ← *initializePop*(*popsize*) # Create an initial population
populationRDD ← *parallelize*(*Population*) # Parallelize the population into a population RDD
for *eachIndividual* ∈ *populationRDD* **do**
    fitnessRDD ← *Individual.map*(*assessFitness*()) # Evaluate each individual using the fitness function
end for
collect() ← *fitnessRDD.collect*() # Collect the individual with fitness value to the driver
Phase 2: Parallel Genetic Operators
while (*t* < *maxgeneration*) # Stopping criterion: maximum number of generation is not reached **do**
    if Environmental change **then**
        for *eachPartition*(*i*) ∈ *RDD* **do**
            Population ← *sortbyvalue*(*result*) # Sort the individual by fitness value
            populationRDD ← *parallelize*(*popwithfitnessvalue*) # Parallelize the population with fitness value into a subpopulation RDD
            for *eachsubpopulation* ∈ *populationRDD* **do**
                evaluation ← *map* − *Partitions*(*evolution*()) # Performs Genetic operators: Tournament selection as a selection, a PMX as a crossover and a replacement as a mutation operator
                bestIndividualRDD ← *evaluationRDD*(*getBest*()) # Get the best individual of each subpopulation
            end for
        end for
        collect() ← *bestIndividualRDD.collect*() # Collect the best individual on every worker to obtain the best solution
        t ← *t* + 1
    end if
end while
Output: Best found solution # routes with shortest total travelled distance
End

GA is a metaheuristic used often for solving optimization problems (Sbai et al. [27]). It is proved that GA produces good results for solving dynamic VRP problems (AbdAllah et al. [1]) but only for a small scale of instances. GA is an iterative evaluation algorithm which can be performed in parallel. So, in order to improve the performance and effectiveness of GA, we use Spark to parallelize it. Algorithm 1 describes the different steps of the proposed parallel S-GA. We start by randomly creating an initial population and store the whole population as RDD in memory. GA involves a population of individuals encoded as chromosomes; each chromosome is encoded as a series of genes that denotes the **sequence of vertices visited by the vehicle**. The start and the end of each vehicle route is presented by a depot 0.

Then, we apply the fitness first phase of the parallelism where the initial population is divided into a population RDD by a *parallize()* spark method. Each individual is evaluated by its fitness value using the objective function in order to choose the best one. After that, we apply a map*(assessFitness())* to produce the fitness from the population RDD. This function is done in the driver program to run in the cluster. Finally, the pair of individuals are collect to the driver with the *collect()* step.

In the case of the second phase which is the parallel genetic operator, the population of individuals with fitness value divided into a population RDD by a *parallize()* spark method. After that, a *mapPartitions(evolution())* conversion

is apply to split the population into a subpopulation stored into a set of partitions of the populationRDD. Genetic operators (selection, crossover and mutation) are executed using the Tournament selection as a selection, a Partial-Mapped Crossover (PMX) as a crossover and a Replacement as a mutation operator. The last operators is applied in the driver program and run in the cluster using *evolution()* function. Finally, the best individual getting by (getBest()) are collected from each worker to obtain the best solution. This process is repeated until reached a maximum number of 100 generation. Figure 3 describes the two-phase parallelization of our proposed S-GA. Apache Spark [10] was

Figure 3: The S-GA two-phase parallelization.

developed in 2009 in AMPLab in UC Berkeley University, it is an open source framework for Big data processing, designed to be able to solve the Hadoops shortcomings. Spark is a distributed in-memory data processing engine commonly used for batch and stream processing of large datasets to achieve high performance. Apache Spark is a classic master/workers mode, the master node is responsible for scheduling, whereas workers are responsible for parallel executing. Figure 2 gives a simple illustration on Apache Spark model. Spark is based on resilient distributed dataset (RDD), which is considered as a database table that is distributed among the different nodes of the cluster. The RDDs could be created by reading an external data source or by parallelizing a collection of data. RDD is divided and stored in the memory, that division is called partition (P).

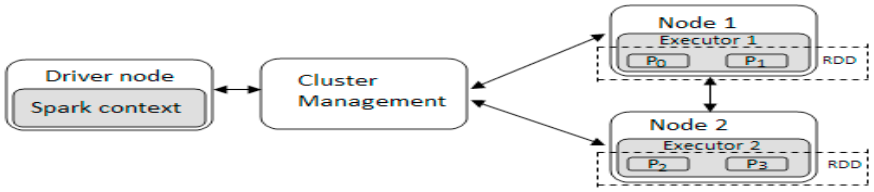

Figure 2: Apache Spark Architecture.

4.3. Streaming Data Sources

In our work, streaming data are collected from directory, Kafka and social media (Twitter). New Paris's traffic data information is done in real time and stored in a directory file. Consequently, Spark updates automatically problem information such as time or distance when it detect a change. In the Kafka way, the latter produces randomly a set of messages with data to update the problem with reference to an uniform and a normal distributions. In the case of Twitter, tweets are collected from Twitter with the subject of Paris traffic information and also information well be updated when a new change is detected.

5. Decision support system

The overall architecture of the decision support system is shown in Figure. 4.

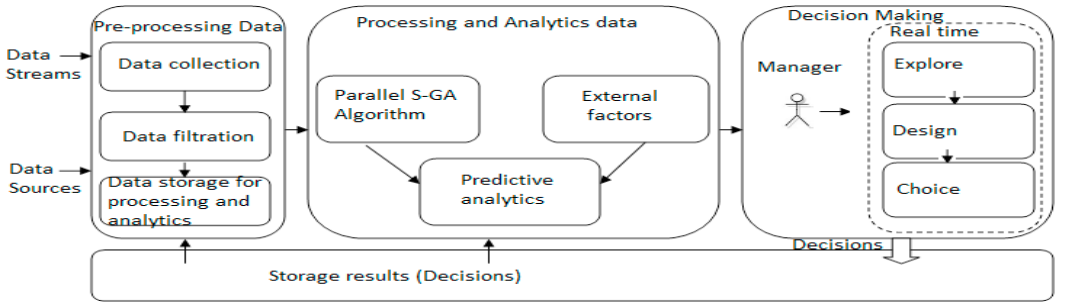

Figure 4: A DSS Architecture for Big Data Analytic.

Our proposed DSS architecture consists of four major components:

1. Pre-processing data: In the pre-processing step, the heterogenous data collected from various sources such as roadside sensors, cameras and detectors used to detect speeds and time stamps of vehicles, radar, GPS-enabled smartphones and social media is processed in order to produce as output a list of suitable features used as input for the processing and analytics steps.
   The data process includes two sub-steps. The first one is the data cleaning and filtration consists in the removal or substitution of the erroneous sensor data by removing data redundancy, noise and other error from data. The main goal of this sub-step is to raise the underlying data quality used for analysis. The second one consists in the data storage prepared for the Processing and analytics steps.
2. Processing and analytics data: the received data are then performed the required predictive analytic approach using the proposed parallel S-GA to solve the DVRP taking into account the external factor such as distance matrix (vertices coordinations) and vehicles.
3. Decision making: Based on the obtained recommendations, the decision maker (manager) selects the most appropriate elements (possible scenarios of routes) for him as shown in Figure. 5. and takes the best decision.
4. Storage decisions: best decision are stored for a future cycles of Big data analytics.

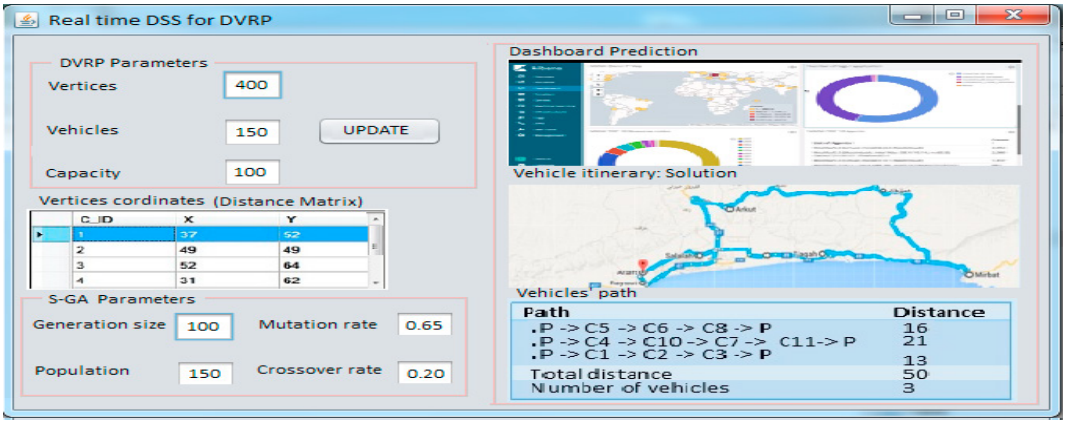

Figure 5: The decision support system interface

6. Experiments

In this section, we present the experimental studies we have conducted to evaluate the proposed parallel S-GA. We have designed two types of experiments. In the first one, we present our experimental environment and our experimental protocol. In the second one, we test the working of parallel S-GA when solving the DVRP.

6.1. Experimental environment

All the experiments were performed on a cluster with eight servers, each one is equipped with Quad-Core CPU, 16GB of main memory and 500 GB of local storage. This servers are used as a Slave nodes with aimed to generate the fitness evaluations of the algorithm using TaskTracker (Spark) and DataNode (HDFS).

6.2. Experimental protocol

In our tests, we performs Hadoop 2.6.3 and Spark 1.6.0.
We evaluate real-time data processing capabilities of Spark in the Streaming mode. As shown in section 2, scenario is divided into seven main steps as follows:

1. Data sources: data can be collected from various sources such as social networks (Twitter), sensors, radar, GPS-enabled smartphones. In this work we use Twitter as the main source of data collection.
2. Realtime Twitter Data collection using Apache Flume version 1.9.0: Data collection from Twitter (1 billion tweets) are collected using Apache Flume do to its facility integration with Hadoop HDFS and its flexible architecture based on streaming data flows.
3. Stream Data storage on HDFS: data collected by flume are loaded as a files with name FlumeData.* storage inside Tweets directory in our home directory in HDFS.
4. HDFS to Kafka ingestion: data storage on HDFS are then sent to Kafka do to its fault-tolerant and highly scalable. Kafka is able to storage data streaming as topic messages that will be read easy by the streaming engines. In our case, the problem is updated when a new information arrived. So, Kafka generates randomly a set of message (3KBs) of information as a DVRP Matrix of Data (DVRPMatrixData).
5. Data processing: Spark extract data from Kafka using an Extract, Transform and Load (ETL) routine. After that a parallel S-GA are applied to generate the optimum of the problem.
6. Cassandra for real-time storage and query: Results generates by Spark are stored in Cassandra Big data base in the form of Cassandra CQL table.
7. KIBANA visualization dashboard:

To check the performance of SGA, we implemented parallel S-GA on a cluster with eight servers. Each server is equipped with a 2.0 GHZ dual-core processor and 16 GB memory. Table 1 summarized the parameters of the cluster. Table 2 listed the parameters setting of our S-GA. For our tests we used Spark 1.6.0.

Cluster parameters

Parameters	**Values**
Number of servers	8
Processor	2.0 GHZ
Memory	16 GB
Operation System	Ubuntu 12.04
Hadoop	Hadoop-2.6.3
Spark	Spark-1.6.0

S-GA parameters

Parameters	**Values**
Population Size (N)	150
Selection	The tournament selection
Crossover rate	0,65
Mutation rate	0,20
Maximum number of generations	100

6.3. Experimental results

In this section, we evaluate the performance of the studied frameworks (Spark, Kafka stream, Flume). Therefore, we adapt the method used by ([12] and [13]) using two different experiments (test cases) under different conditions of message and registered the execution traces during half-hour. It consists in injecting 10,000 messages every 1 second and 0.1 second respectively from the data sources to the workflow. In order to facilitate the reading of message from streaming engines, messages are written in Kafka topic. In our case we divided the time period of 20 min for each messages treat by each streaming processing engine. For this purpose, in order to control the CPU load, we calculate the number of thread at kernel level for each cluster.

1. First experiments: 10,000 messages each 1 s: Fig 6(a) illustrates the processing time through half-hour CPU time of the number of cluster for each cluster. Spark collects data in the first 20 minutes from Kafka (using the message topics) and performs processing task after that. This explain the rise up of the processing time at the level of cluster9 which is the master in the begin of execution. After that, in the case of Kafka, we observe an increase of the processing time for two peaks and then it become stable. As a conclusion, this explain that Spark and Kafka are able to load messages.
2. Second experiments: 10,000 messages each 0.1 s: Fig 6(b) expresses also the processing time through half-hour CPU time of the number of cluster for each cluster in the faster demanding environment of 0.1s. We also notice that their is an increase of the processing time in the first minutes of execution of the master node (cluster9) when Spark read data streaming from Kafka. Finally, the system read data from Kafka which leads to the destabilisation of the processing time.

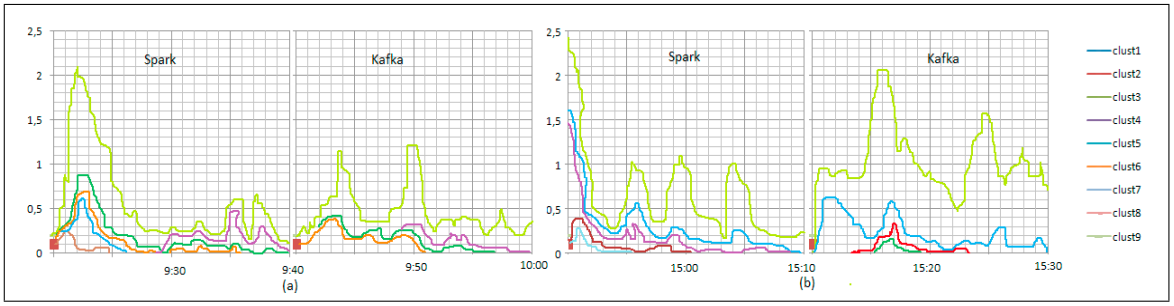

Figure 6: Big Data Analytic Architecture.

6.4. Optimization Results

In this section, we study the performance of our parallel S-GA algorithm which can be justified by the evaluation the following criteria:

1. Execution time: the effectiveness of our S-GA is evaluated in terms of execution time. To do this, we changed the generation number according to the number of CPU cores(i. e., cluster size) 2, 4, 8 and 16. Figure 7(a) presents the effect of variation of number of 100 generations compared to execution time. We can conclude that the increase in the number of generations implies a long execution time (longer evolution time). In addition, we can also notice that the more the cluster size increase, the longer the execution time is. This remarks indicate the reliability and efficiency of our S-GA to explore the resources.
2. Average detection time: the Average detection time is used to check the effect of the number of generation within the objective function (minimize the total distance). Figure 7(b) describes the average detection time, we can notice that the more the number of generation increases the average detection time decreases in the four case number of core of 2, 4, 8 and 8 respectively. Moreover, the increase of the number of generation is favorable to the fitness value.
3. Speed Up: The speed Up consists in varying number of computer nodes. In our case, we use the speed up to explore the impact of the cluster size within the execution time. The speed up is measured as follows:

$$Speedup = T_1/T_m \tag{7}$$

where T_1 denotes the execution time with one computer node and T_m with m computer nodes respectively. Figure 7(c) outlines the measure of speed up with the number of Cpu cores. We can see that the more the speedup is close to linear, the best the algorithm is. Results show a good speedup for the proposed S-GA method.

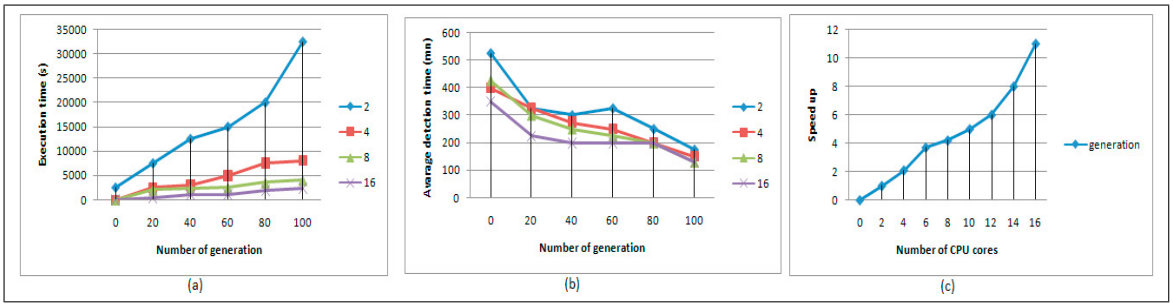

Figure 7: Optimization results: (a) execution time, (b) Average detection time, (c) Speed up.

7. Conclusion

In this paper, we have presented a big data analytic architecture to deal with dynamic Big Data Optimization problems known as DVRP where the objective is to minimize the total distance. In the processing phase of our architecture, we combining optimization framework with the Spark cluster computing system using the parallel S-GA. Based on the parallel S-GA a decision support system is developed for the DVRP in order to generate the best routes. Our data source are Twitter and Kafka. A real instance collected from Open Data of the city from France has been generated to update the problem data in streaming. To check the performance of our proposed architecture, we evaluated first the effectiveness of the studied frameworks by injecting 10.000 messages every 1s and 0.1s and then we studied the performance of our S-GA by evaluating the execution time, the average detection time and the speed up criteria. As a result, we can notice from the experience that our architecture incorporate different streaming engines and switch between them without changing any component from its core.

References

[1] AbdAllah, A. M. F., Essam, D. L., and Sarker, R. A. (2017). On solving periodic re-optimization dynamic vehicle routing problems. Applied Soft Computing, 55, 1-12.
[2] Ahmad, A., Khan, M., Paul, A., Din, S., Rathore, M. M., Jeon, G., and Choi, G. S. (2018). Toward modeling and optimization of features selection in Big Data based social Internet of Things. Future Generation Computer Systems, 82, 715-726.
[3] Ahmad, A., Paul, A., Din, S., Rathore, M. M., Choi, G. S., and Jeon, G. (2018). Multilevel data processing using parallel algorithms for analyzing big data in high-performance computing. International Journal of Parallel Programming, 46(3), 508-527.
[4] Alanzi, E., and Bennaceur, H. (2019). Hadoop MapReduce for Parallel Genetic Algorithm to Solve Traveling Salesman Problem. (IJACSA) International Journal of Advanced Computer Science and Applications, 10(8), 97-107.
[5] Al Nuaimi, E., Al Neyadi, H., Mohamed, N., and Al-Jaroodi, J. (2015). Applications of big data to smart cities. Journal of Internet Services and Applications, 6(1), 25.
[6] Apache kafka. https ://kafka.apache.org.
[7] Apache cassandra. https://tutorialspoint.com/cassandra/cassandra introduction.htm.
[8] Apache Flume. https://flume.apache.org
[9] Apache Hadoop. https://hadoop.apache.org
[10] Apache Spark. https://spark.apache.org/
[11] kibana. https://en.wikipedia.org/wiki/Kibana
[12] Barba-Gonzlez, C., Garca-Nieto, J., Nebro, A. J., Cordero, J. A., Durillo, J. J., Navas-Delgado, I., and Aldana-Montes, J. F. (2018). jMetalSP: a framework for dynamic multi-objective big data optimization. Applied Soft Computing, 69, 737-748.
[13] Barba-Gonzlez, C., Nebro, A. J., Bentez-Hidalgo, A., Garca-Nieto, J., and Aldana-Montes, J. F. (2020). On the design of a framework integrating an optimization engine with streaming technologies. Future Generation Computer Systems, 107, 538-550.
[14] Chaudhari, A. A., and Mulay, P. (2019). SCSI: real-time data analysis with cassandra and spark. In Big Data Processing Using Spark in Cloud (pp. 237-264). Springer, Singapore.
[15] Gohar, M., Muzammal, M., and Rahman, A. U. (2018). SMART TSS: Defining transportation system behavior using big data analytics in smart cities. Sustainable cities and society, 41, 114-119.
[16] Gaifang, D., Xueliang, F., Honghui, L., and Pengfei, X. (2017). Cooperative ant colony-genetic algorithm based on spark. Computers & Electrical Engineering, 60, 66-75.
[17] Hu, C., Ren, G., Liu, C., Li, M., and Jie, W. (2017). A Spark-based genetic algorithm for sensor placement in large scale drinking water distribution systems. Cluster Computing, 20(2), 1089-1099.
[18] Jara, A. J., Genoud, D., and Bocchi, Y. (2014, May). Big data in smart cities: from poisson to human dynamics. In Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on (785-790). IEEE.
[19] Kitchin, R. (2014). The real-time city? Big data and smart urbanism. GeoJournal, 79(1), 1-14.
[20] Maqbool, F., Razzaq, S., Lehmann, J., and Jabeen, H. (2019). Scalable Distributed Genetic Algorithm Using Apache Spark (S-GA). In International Conference on Intelligent Computing (424-435). Springer, Cham.
[21] J.Mandziuk, A. Zychowski, (2016). A memetic approach to vehicle routing problem with dynamic requests, Applied Soft Computing. 48, 522-534.
[22] Neilson, A., Daniel, B., and Tjandra, S. (2019). Systematic Review of the Literature on Big Data in the Transportation Domain: Concepts and Applications. Big Data Research.
[23] Okulewicz, M., and Mandziuk, J. (2017). The impact of particular components of the PSO-based algorithm solving the Dynamic Vehicle Routing Problem. Applied Soft Computing, 58, 586-604.
[24] Qi, R. Z., Wang, Z. J., and Li, S. Y. (2016). A parallel genetic algorithm based on spark for pairwise test suite generation. Journal of Computer Science and Technology, 31(2), 417-427.
[25] Rathore, M. M., Ahmad, A., Paul, A., and Rho, S. (2016). Urban planning and building smart cities based on the internet of things using big data analytics. Computer Networks, 101, 63-80.
[26] Sabar, N. R., Bhaskar, A., Chung, E., Turky, A., and Song, A. (2019). A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion. Swarm and evolutionary computation, 44, 1018-1027.
[27] Sbai, I., Krichen, S. and Limam, O. (2020). Two meta-heuristics for solving the capacitated vehicle routing problem: the case of the Tunisian Post Office. Oper Res Int J. doi:10.1007/s12351-019-00543-8
[28] Sivarajah, U., Kamal, M. M., Irani, Z., and Weerakkody, V. (2017). Critical analysis of Big Data challenges and analytical methods. Journal of Business Research, 70, 263-286.
[29] Wang, G., Gunasekaran, A., Ngai, E. W., and Papadopoulos, T. (2016). Big data analytics in logistics and supply chain management: Certain investigations for research and applications. International Journal of Production Economics, 176, 98-110.
[30] Wang, Y., Feng, L., Chang, H., and Wu, M. (2017). Research on the impact of big data on logistics. In MATEC Web of Conferences (Vol. 100, p. 02015). EDP Sciences.
[31] Xu, H., Pu, P., and Duan, F. (2018). Dynamic vehicle routing problems with enhanced ant colony optimization. Discrete Dynamics in Nature and Society, 2018, 13 pages.
[32] Yang, Z., van Osta, J. P., van Veen, B., van Krevelen, R., van Klaveren, R., Stam, A., ... and Emmerich, M. (2017). Dynamic vehicle routing with time windows in theory and practice. Natural computing, 16(1), 119-134.