Internet of Things Based Smart Community Design and Planning Using Hadoop-Based Big Data Analytics

Muhammad Babar[1,2(✉)], Waseem Iqbal[2], and Sarah Kaleem[3]

[1] Iqra University, Islamabad, Pakistan
muhammad.babar@iqraisb.edu.pk
[2] National University of Sciences and Technology, Islamabad, Pakistan
waseem.iqbal@mcs.edu.pk
[3] Iqra National University, Peshawar, Pakistan
sarah.kaleem.inu@gmail.com

Abstract. The current spreading out in big data is offering a hefty invention potential in itinerary of the fresh epoch of smart community. The foremost endeavor of smart community is to competently employ the asset of Big Data to manage and determine the issues face by recent smart cities for enhanced decision making. The applications of smart city fabricate a gigantic number of data that compose Big Data. This research proposes Big Data analytics architecture to address the challenges in Big Data analytics using Hadoop framework. The proposed framework is dealing particularly with data loading and processing. The proposal is consist of two parts that are Big Data loading (storage) in Hadoop file system and Big Data computation. The first part is liable for transferring Big Data from outer world and storing in Hadoop. The second part of the research deals with the data processing. YARN-based cluster management solution is provided to manage the cluster resource and process the data using Map-Reduce algorithm separately unlike traditional MapReduce architecture. The proposed architecture is tested with a variety of reliable datasets using Hadoop framework to verify and expose that the architecture offers precious imminent into the society organizations for development to improve the existing smart city architecture.

Keywords: IoT · Big data · Hadoop · Smart community

1 Introduction

In the present day computing devices and the internet are completely relying on humans for data and information. Approximately 50+ petabytes of data accessible on the fingertip via internet were initially captured, acquired and created by human beings [1]. It is hard to process and store using conventional data processing techniques or database management tools [2]. In current era, there is enormous data to be processed than the data in the past when traditional techniques were introduced; therefore, traditional and conventional techniques or tools are not capable to pact with Big Data and

K. Arai and R. Bhatia (Eds.): FICC 2019, LNNS 69, pp. 1046–1057, 2020.
https://doi.org/10.1007/978-3-030-12388-8_72

its dimensions. The traditional and classical techniques and process are not adequate to process the data (to specific Big Data) due to its qualities and characteristics [3].

The researchers describe the characteristics or dimensions of Big Data in terms of V have to consider and classify the data as Big Data is hard to process and cannot be processed by traditional mechanisms. There could be different range of V's but in this section five different Vs of Big Data are described. These five V's include Volume that refers to mass of data that is tremendously large and increase exponentially at a very fast speed day by day. The amount of data produced by humans, devices, and their communications on social sites itself is immense. It has been predicted by researchers that 40,000 Exabytes (40 Zettabytes) will be produced by 2020, which is a raise of almost 300 times from past [4, 5]. The huge and humongous data come from multiple sources contributing to Big Data is diverse in nature with different shapes and that is another V which is variety. The data that can be stored and processed in a pre-defined fixed format is known as Structured Data. The structured data is in having proper schema or in tabular form. The semi-structured data may not properly define in a schema-oriented form while the un-structured data is not in a tabular form at all. Unstructured data have unidentified structure and cannot be utilized using RDBMS.

Velocity is the pace and rate at which the information is generated by different multiple sources. As computer evolved and we came up with client/server approach, web applications, and internet, so everyone started using internet not only with computers but from mobile devices as well. Since then more users, more applications, and speed of data is increased. One of the bigger problems is how to extract the useful data from such huge, un-structured, and speedy data [6]. This scenario is known as Value which is another V of Big Data. Data with huge amount of size, speed, and different variety is bound to lose or miss some data packages. Getting or mining value out of such data (Big Data) is a difficult as such data has uncertainty. This doubt or uncertainty in data is termed as Veracity which is due to the data inconsistency.

The Hadoop framework [7–9] is freely available accomplishment of the MapReduce skeleton that is a programming paradigm Hadoop has turned out to be a major technology for big data because of constant raise of data volumes and varieties. In addition, its model of distributed computing provides a flexible way to processes Big Data fast. An added benefit is that Hadoop is free and uses reasonable and inexpensive technology (commodity hardware) to store and process huge data [10]. Hadoop is currently having two different descriptions. This paper presents Big Data analytics architecture using Hadoop framework to address the challenges in Big Data. The planned architecture is a Hadoop-based architecture dealing particularly with Big Data loading and computation. The proposed architecture is responsible for transferring and storing the Big Data in Hadoop as it processes only that data which is available in the HDFS (Hadoop Distributed File System) and performs data computation and processing using Yet Another Resource Negotiator (YARN) based cluster management with Map-Reduce algorithm.

2 Literature Review

The typical smart community design can present a variety of returns. In current times, research groups are functional to develop different solutions [11]. To conquer the issues regarding the analysis of Big Data generated in the IoT based smart community environment, many proposals have been advised. [12–14]. Moreover, there are a number of urban developmental, technological, and service-oriented promoters are promoting the smart urban. These smart urban proponents are from both public and private sectors. These promoters in private sector include, but not limited to Google, IBM, Siemens, Cisco, Microsoft, Honeywell, Schneider Electric, Panasonic, and so forth. The public sector promoters include, but not limited American Planning Association (APA) [15], Rocky Mountain Institute (RMI) [16], World Resources Institute (WRI) [17], Smart City Council (SCC) [18], and so forth. The CICSO, IBM, and Google provide a number of solutions related to the smart city elements. The trend leading to smart urban is from different perspectives such as environmental trend, social trend, political trend, and technological trend. One of the most important and vital technological trend of the smart community is the big data technology in the current era.

The Hadoop employs MapReduce that is accountable for the widespread diversity of everyday jobs [8, 19]. MapReduce partitions file into self-governing lumps that are handled in parallel. This architecture categorizes the maps outcomes and transmits as input to the reduce job. Yet Another Resource Negotiator (YARN) is the brain of Hadoop which is responsible for the core activities [20]. It is responsible for the cluster management in Hadoop later description. It performs all the processing actions by scheduling tasks and allocating the resources. It is comprised of two major units which are Resource Manager and Node Manager. YARN is introduced in latest description of Hadoop [21–23]. It detaches the main functionalities or operations of the resource management, job tracker, and job scheduling to a particular separate daemon.

Those applications, which require write once and read many times will get the most utilization out of this programming paradigm [24]. To create physical implementation of extensive IoT infrastructure, various test bed mechanisms have been projected [13]. It is stated that the Things can be linked and communicated via internet to be utilized for different applications [25]. The internet vision can also be assumed as ‘Ubiquitous IoT’ [25] which is close to the idea of social association model. One of the complementary approaches for smart unbans to conquer the mobility issues in mobility is to spearhead the scientific bound with the Big Data [26] and also for smart grid management the Big Data management is the key factor to be handled [27].

In addition, different solutions are proposed to tackle Big Data technologies. Big Data coming from linked things can be analyzed with assistance of various storage services [28]. These storage techniques improve data scalability, accessibility, flexibility, and compliance. Connecting IoT with social network, the concept of Big Data is kept sidewise. Since IoT and Big Data has an extremely influential relationship to work together as these are the main sources of smart urban. Moreover, there are architectures based on pre-processing are also proposed, but data loading efficiency is overlooked [29, 30].

3 Proposed Architecture

The proposed work is basically composed of two layers that are data loading and storage in Hadoop and data processing that is depicted in Fig. 1.

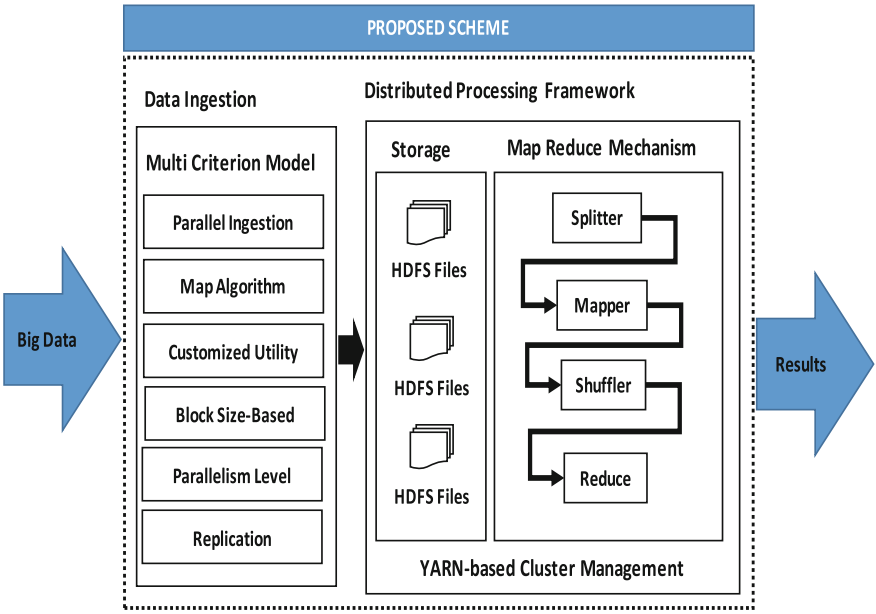

Fig. 1. Proposed architecture

Given that a number of devices are used to produce and generate the very huge, massive and gigantic data with diverse arrangements, diverse origin, and diverse timing. Therefore, efficient data loading techniques are applied in this phase which reduce the data loading time and improves the execution time of the overall architecture. Slow data loading will lead to slow processing. Applying data loading techniques before processing can considerably enhance the speed and performance of the actual processing and analysis. Similarly, the second part of the architecture deals with the data computation and processing. Unlike traditional MapReduce architecture, customized YARN-based cluster resource management solution is provided to manage the cluster resource separately and process the data using Map-Reduce algorithm separately.

3.1 Data Loading

Hadoop platform can analyze and process enormous sum of information in parallel. Data Analysis is the half part of the analytics and processing of Big Data, loading the data into Hadoop server would be other half which is a very challenging task. Loading the Big Data into HDFS (Hadoop) is the primary activity and one of the key factors in

any architecture. Moreover, data are required to organize for efficient map-reduce utilization; therefore, the proposed architecture provides a solution based on map-only algorithms using Sqoop utility. Typically the HDFS commands are preferred to load data into Hadoop ecosystem if one time ingestion is required. These commands and scripts can always be written to ingest the data into Hadoop but this procedure would be very difficult, time consuming and ineffective if the data is being generated constantly. The proposed data loading is based on multiple attributes which are customized block size of HDFS, replication in HDFS, data loading utility/tool using parallel data loading and map-only algorithms, and level of parallelism.

The HDFS splits and stores the large files into small pieces called blocks which are the smallest part of data in a file system. The master part of HDFS includes Name Node which control these blocks and users do not have control. The default size of is 128 MB, but we preferred 256 MB of block size in proposed architecture which is large. This huge size selection is because of the volume of the input datasets i.e. hundreds of GBs, terabytes and petabytes of different sources of smart societies department. If we would have preferred lesser size then there would be too many data blocks in HDFS which will be creating too much of metadata eventually. So, controlling enormous number of blocks and metadata will be creating huge overhead and traffic. The file size of the blocks can be larger than the proposed, but in contrast, the size of block cannot be so huge that the Hadoop system requires waiting for a long time.

In addition to block size, the number of replicas are also taken into consideration while data loading and storing the data. The replica mechanism makes the actual dataset size several times larger which causes more time to load the dataset. The default size of the number of replicas in Hadoop is 3, but we preferred and configured the HDFS number of replicas to 2. The replication process is used to copy the actual data blocks several times that is time consuming; therefore, we proposed customized replication factor. The configured and customized replication improves the performance of data loading in the context of time consuming.

The proposed architecture is integrated with Sqoop to insert the data into Hadoop. The Sqoop is an open source utility by Apache that is configurable. The Sqoop transfers data between data warehouses, relational databases (e.g. Oracle, MySQL, and Teradata), other Hadoop storage mechanism like HBase and Hive. The Sqoop can be utilized to import either complete databases or some part of the databases to HDFS. Sqoop produces the MapReduce code internally to move the data. One of the most important of features is to export the data from Hadoop to external sources and allow very simple import and export. In addition, the Sqoop can also be utilized to automate and schedule the imports and exports being integrated with other external tools. In contrast the computation is carried out using Map Reduce algorithm along with YARN-based cluster resource management.

3.2 Data Processing Using MapReduce Algorithms

Map Reduce algorithm is proposed to process the traffic dataset. MapReduce is accountable for the widespread diversity of everyday jobs [31]. The proposed algorithm is used to collect the information regarding vehicles on roads. The graphical

representation of the proposed MapReduce algorithm for traffic dataset is given as Fig. 2. The Map function of the proposed algorithm takes the line offset as key and the values of entire row as value. The timeStamp as key and the required associate values are emitted as value by the map function. The Reduce function groups the required associate values against each timeStamp and compares with the TLV.

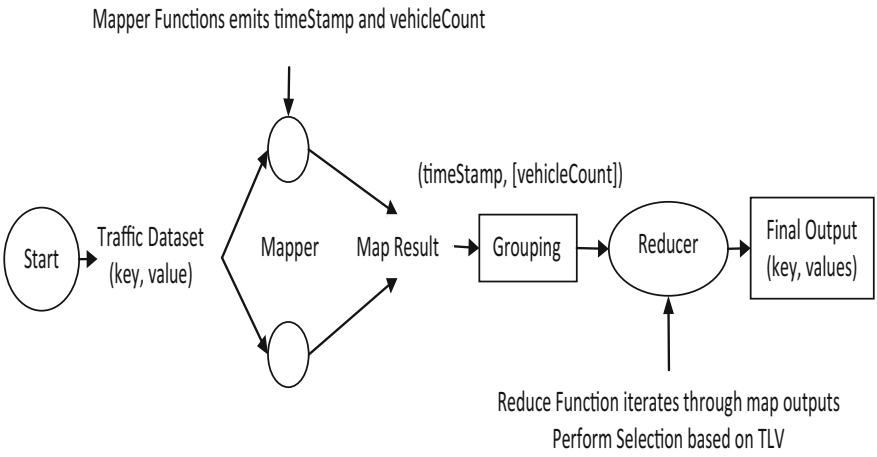

Fig. 2. MepReduce algorithm for traffic dataset

4 Implementation and Results Discussion

The implementation is perfumed using Hadoop server on Ubuntu 16.04 LTS along with MapReduce programming in Java. Additionally, a corei7 CPU with RAM size 8 GB is utilized. The datasets are acquired from diverse sources that are accessible. In this research the traffic dataset is used. The traffic data is information regarding cars in lanes in Aarhus city, Denmark [32] is used. Map Reduce algorithm is implemented in Java programming language. This is used to collect the information regarding vehicles on roads. The Map function of the proposed algorithm takes the line offset as key and the values of entire row as value. The timeStamp as key and the required associate values are emitted as value by the map function. The Reduce function groups the required associate values against each timeStamp and compares with the TLV.

The mapper of the parking dataset is given as Algorithm 1. This task emits the timeStamp as key and vehicleCount on road as value. The Algorithm 1 is implemented using Mapper class of Java programming language. Similarly, the reducer of the parking dataset is given as Algorithm 2. The Reduce function groups the required associate values against each timeStamp and compares with the TLV. The Reducer algorithm is implemented using Reducer class of Java programming language.

```
BEGIN
        I/P
                k: line-offset     ||      v:= row
        O/P
               k: timestamp      ||      v: vehicleCount
                        // line splitting
         timestamp, vehicleCoun:=  line.split ('\t')
         k:= timeStamp
         v:= vehicleCount
         emit (k, v)
END
```

Algorithm. 1. Mapper for Traffic Dataset

```
BEGIN
        I/P
               k: timestamp          ||      v: vehicleCount
        O/P
               k: timestamp          ||      v: vehicleCount
          initialize threshold
    final []
    FOR each (vehicleCount) at timeStamp DO
    IF (vehicleCount > threshold)
    Begin
         final.append (vehicleCount)
         key:= timeStamp
         value:=final
         emit (key, value)
     End IF
END
```

Algorithm. 2. Reducer for Traffic Dataset

4.1 Results Discussion

The results are discussed in this section. The manual data loading commands do not support parallel data loading, which causes the loading process more time consuming which is shown in Fig. 3. The time of loading data increases almost linearly with the size using both methods. It is clearly viewed from Fig. 3 that the tool used for data loading is faster than the manual Hadoop commands. The faster data loading is because of the parallelism being provided by the tool utility embedded with proposed solution.

Like previous experiments results, the same point is also observed in the results of this experiment that less dataset size has almost no impact on the data loading time while bigger file size has a significant impact when file size is greater than 1 GB.

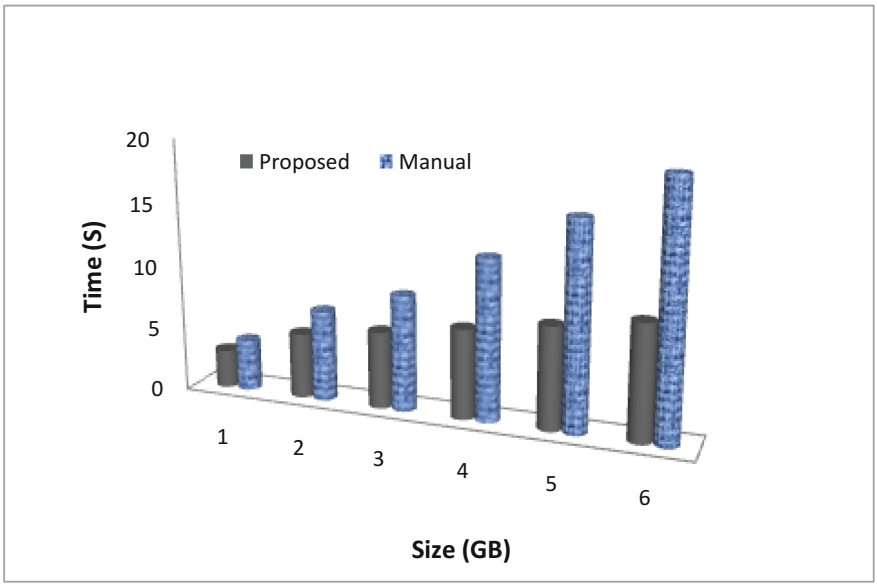

Fig. 3. Data loading efficiency of proposed method

As discussed earlier, data loading time difference is not noticeable when the data size is small. Due to replica mechanism, the data loading time is quite noticeable when dataset size is large. The point of consideration is what is the specific dataset size (threshold)? Dataset size threshold is a value of dataset size that is the equivalent size of dataset from where the data loading time difference is noticed. To find the threshold (size), we measure the performance of data loading using test datasets of different size. The threshold of dataset size is the value where time variation tends to be greater than 0. When the variation is greater than zero (0) significant changes occur.

As Hadoop might be occupied by other jobs running by some other users, we may get dissimilar time to load the same size of dataset twice. Therefore, the time variation equivalent to threshold value can be described as a specific range in order to overcome the said issue such as (0–6) seconds to discover the threshold. The thresholds for different parameters are established using results of the same experiments. Taking data loading utility experiments into consideration, the threshold is up to 900 MB (dataset file size) where the impact of data loading time starts shown in Fig. 4. Figure 4 shows that up to 1 GB of dataset file is not generating any difference even if automated data loading technique is used. The efficiency is achieved when the dataset size is greater than at least 900 MB.

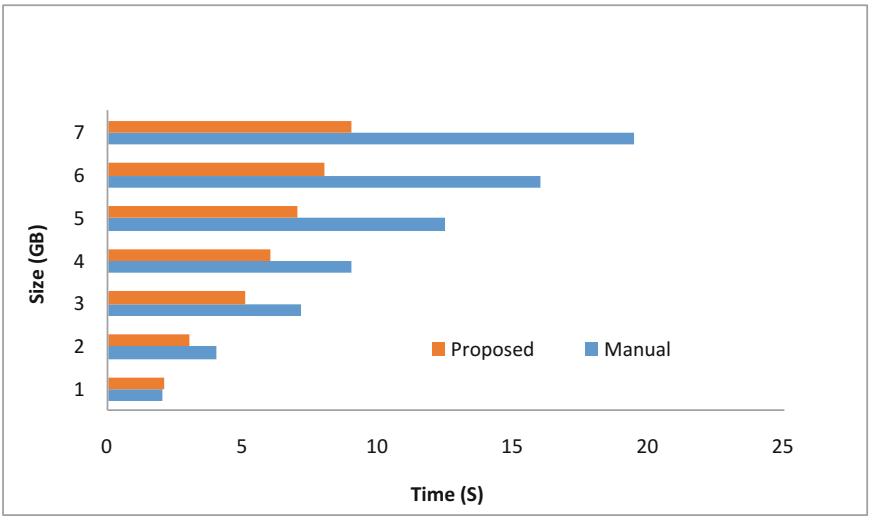

Fig. 4. Dataset size threshold

Taking into consideration the traffic data about road congestion traffic management, the data is processed to overcome traffic issues when the numbers of vehicle go beyond the threshold on a road or lane. Figure 5 shows the number of vehicles on the roads at different timestamp of the day. It is noticed that there are more vehicle on the road between 7:00 AM and 10:00 AM due to the school and office timings in the city.

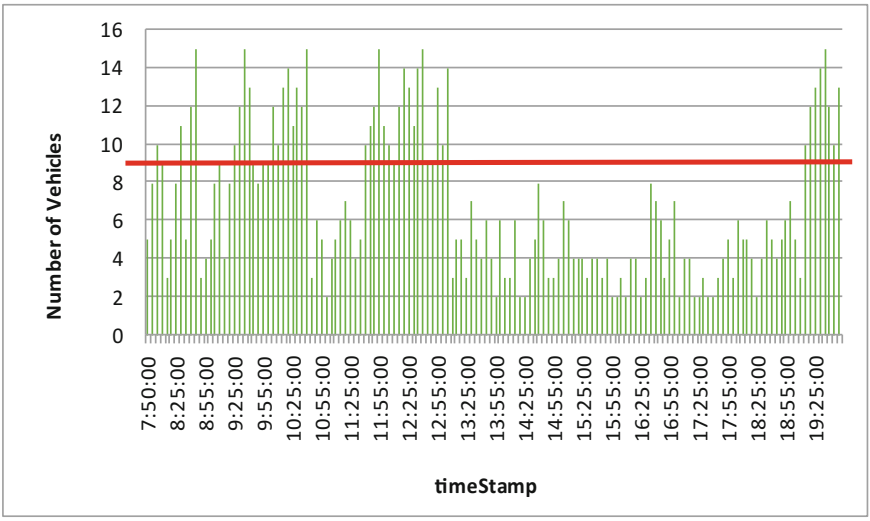

Fig. 5. Vehicles on road at different time

In addition, the average pace of the automobiles on the road is also demonstrated in Fig. 6. The horizontal axis of the Fig. 6 shows different timestamps of the day while the vertical axis highlights and demonstrates the average speed of vehicles on the road. It is observed and noticed that the average pace of the automobiles is pretty identical throughout the entire day except the time interval when there are lesser vehicles on road. This time interval is between 1:00 PM to 6:00 PM.

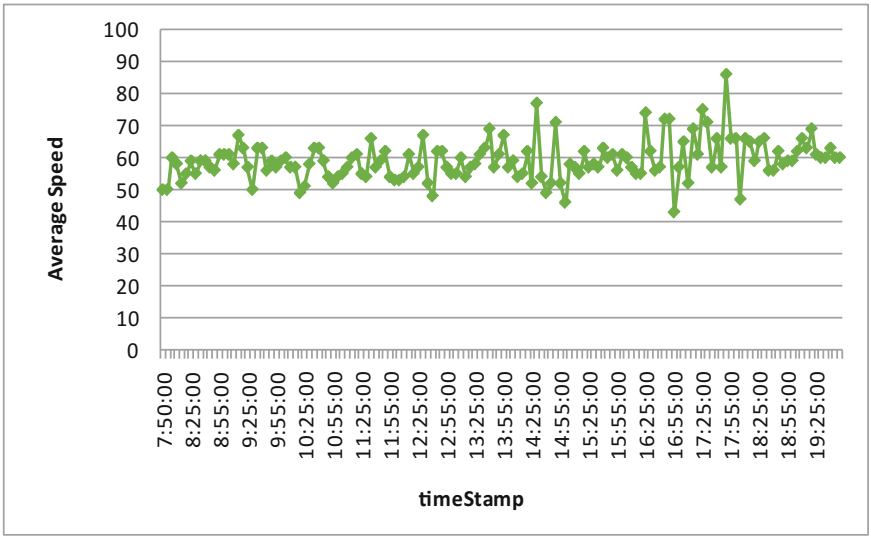

Fig. 6. Average speed of vehicles on road at different time

5 Conclusion

Hadoop-based solution is proposed to deal with the problems of data loading and processing huge and gigantic data in this research. The proposed framework is dealing particularly with Big Data loading and Big Data processing. The proposal has two different parts which are (1) Big Data loading and storage in Hadoop file system and (2) Big Data computation and processing. The data loading is performed and compared with different decisions repeatedly and the persuade aspects are examined. MapReduce algorithm is proposed to process the data in YARN-based cluster resource management environment. The proposed architecture is tested with a variety of reliable datasets using Hadoop framework to verify and expose that the architecture offers precious imminent into the society organizations for development to improve the existing smart city architecture.

References

1. Snijders, C., Matzat, U., Reips, U.D.: Big data: big gaps of knowledge in the field of internet science. Int. J. Internet Sci. **7**(1), 1–5 (2012)
2. Hurwitz, J., Nugent, A., Halper, F., Kaufman, M.: Big Data for Dummies. Wiley, Hoboken (2013)
3. Villars, R.L., Olofson, C.W., Eastwood, M:. Big data: what it is and why you should care. White Paper, IDC (2011)
4. Gantz, J., Reinsel, D.: The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East, sponsored by EMC Corporation, December, 2012 white paper Big Data Meets Big Data Analytic
5. Big Data: A New World of Opportunities, Networked European Software and Services Initiative (NESSI) White Paper, December 2012
6. Li, B.: Survey of Recent Research Progress and Issues in Big Data, December 2013
7. Gang, L.: Applications and development of Hadoop. Zhangtu Information Technology Inc., Beijing (2014)
8. Lublisnky, B., Smith, K.T., Yakubovich, A.: Professional Hadoop Solutions. Wros Press (2013)
9. White, T.: Hadoop: The Definitive Guide, 3rd edn. O'Reilly Press, Sebastopol (2012)
10. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), March 2010
11. Ahn, H.Y., Lee, K.H., Lee, S.H., Lee, Y.J., Lee, S.M., Kim, Y.K.: An efficient method for enhancing the storage efficiency in Hadoop DFS. J. KISS Comput. Pract. **19**(3), 144–148 (2013)
12. Cheng, B., Longo, S.,Cirillo, F., Bauer, M., Kovacs, E.: Building a big data platform for smart cities: experience and lessons from santander. In: Proceedings of the 4th IEEE International Congress on Big Data (BigData Congress 2015), New York, NY, USA, pp. 592–599, July 2015
13. Sanchez, L., Muñoz, L., Galache, J.A., et al.: SmartSantander: IoT experimentation over a smart city testbed. Comput. Netw. **61**, 217–238 (2014)
14. Rong, W., Xiong, Z., Cooper, D., Li, C., Sheng, H.: Smartcity architecture: a technology guide for implementation and design challenges. China Commun. **11**(3), 56–69 (2014)
15. American Planning Association, Making Great Communities Happen, United States of America, (USA). https://www.planning.org/
16. Rocky Mountain Institute, Colorado, United States. https://www.rmi.org/
17. World Resources Institute: Making Big Ideas Happen, Washington, D.C., United States, Founded: 1982. www.wri.org/
18. Smart Cities Council, Livability, Workability, and Sustainability, Smart Cities Council, Inc 1900 Campus Commons Drive, Suite 100 Reston, VA 20191
19. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation, vol. 6, p. 10 (2004)
20. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B., Baldeschwieler, E.: Apache hadoop YARN: yet another resource negotiator. In: Proceedings of 4th ACM Symposium on Cloud Computing (SoCC 2013). ACM (2013)

21. He, B., Fang, W., Luo, Q., Govindaraju, N.K., Wang, T.: Mars: a MapReduce framework on graphics processors. In: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques - PACT 2008, p. 260 (2008)
22. Lin, J.C., et al.: ABS-YARN: a formal framework for modeling Hadoop YARN clusters. In: International Conference on Fundamental Approaches to Software Engineering. Springer, Heidelberg (2016)
23. Kulkarni, A.P., Khandewal, M.: Survey on Hadoop and introduction to YARN. Int. J. Emerg. Technol. Adv. Eng. **4**(5), 82–87 (2014)
24. Yang, G. (2011). The application of MapReduce in the cloud computing. In: 2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing (IPTC), Hubei, RPC, 22–23 October 2011. IEEE (2011)
25. Uppoor, S., Trullols-Cruces, O., Fiore, M., Barcelo-Ordinas, J.M.: Generation and analysis of a large-scale urban vehicular mobility dataset. IEEE Trans. Mobile Comput. **13**(5), 1061–1075 (2014)
26. Ning, Huansheng, Wang, Ziou: Future Internet of Things architecture: like mankind neural system or social organization framework? Commun. Lett. IEEE **15**(4), 461–463 (2011)
27. Schatzinger, S., Lim, C.Y.R.: Taxi of the future: big data analysis as a framework for future urban fleets in smart cities. In: Smart and Sustainable Planning for Cities and Regions, pp. 83–98. Springer International Publishing (2017)
28. Nguyen, T.H., Nunavath, V., Prinz, A.: Big data metadata management in smart grids. In: Studies in Computational Intelligence, pp. 189–214. Springer Verlag (2014)
29. Le, X.H., Lee, S., Truc, P.T., Khattak, A.M., Han, M., Hung, D.V., Hassan, M.M., et al.: Secured WSN-integrated cloud computing for u-life care. In: Proceedings of the 7th IEEE Conference on Consumer Communications and Networking Conference, pp. 702–703. IEEE Press (2010)
30. Babar, Muhammad, Arif, Fahim: Smart urban planning using big data analytics to contend with the interoperability in Internet of Things. Future Gener. Comput. Syst. **77**, 65–76 (2017)
31. Babar, M., Rahman, A., Arif, F., Jeon, G.: Energy-harvesting based on internet of things and big data analytics for smart health monitoring. Sustainable Comput. Inform. Syst. **20**, 155–164 (2017)
32. Dataset, Dataset Collection. http://iot.ee.surrey.ac.uk:8080/datasets.html#traffic. Accessed 12 Jan 2017