A Big Data Solution for Troubleshooting Mobile Network Performance Problems

K. Skračić, I. Bodrušić
Ericsson Nikola Tesla, Zagreb, Croatia
E-mail: kristian.skracic@ericsson.com

Abstract - Big Data has become a major competitive advantage for many organizations. The analytical capabilities made possible by Big Data analytics platforms are a key stepping stone for advancing the business of every organization. This paper illustrates the development of a big data analytics system for mobile telecommunication systems. The authors developed a solution for analyzing data produced by mobile network nodes which contain data relevant for predictive maintenance and troubleshooting purposes. The solution is built around the problem of working with small files in the Hadoop environment. The logs collected from mobile network nodes are small binary files between 5 and 15MB in size. These binary log files need to be decoded to a readable format, and then analyzed to extract useful information. In this paper, the authors provided a benchmark of various scenarios for collecting and decoding the binary log files in a Hadoop cluster. As a result, the scenario with the highest performance has been used in the implementation of our solution. The developed solution has been built and tested on a live Hadoop cluster using real-world data obtained from several telecom operators around the world.

I. INTRODUCTION

The telecommunications industry is one of the largest and fastest growing industries in the world. Over the past few decades we have witnessed the change from 2G, 3G, 4G and now 5G in the near future [1]. With the rising demand for constant connectivity, the availability of telecommunication systems and their various components has never been more important. This trend is particularly apparent in the case of mobile networks. As shown in [2], during 2016 there were 7.5 billion mobile subscriptions in the world. The report in [2] estimates that in 2022 there will be 8.9 billion mobile subscriptions, 8 billion mobile broadband subscriptions and 6.1 billion unique mobile subscribers in the world. Mobile networks allow a subscriber to consume various telecommunication services via mobile phone from any place of coverage. Thus, they have become one of the most important resources today. This paper proposes a solution for analyzing data produced by mobile network nodes which contain information relevant for predictive maintenance and troubleshooting purposes.

As telecommunication systems are becoming larger and more advanced, it is necessary to constantly monitor and measure the performance of their various subsystems. However, larger networks and higher Internet access speeds carry with them the need to analyze larger amounts of data in a short period of time, in order to prevent network outages as soon as possible. Such requirements can be addressed by leveraging the analytical capabilities made possible by Big Data analytics platforms [3]. Apart from improved performance, Big Data can enable deeper analytics by providing access to historical data. For example, by storing network performance data it becomes possible to compare the current results with those obtained in past measurements. By storing the insight obtained through troubleshooting, it becomes possible to predict and prevent the same failures from happening again, or at least to shorten the response time when the same or similar failures present themselves again [4].

In this paper, the authors developed a Big Data solution for analyzing data produced by mobile network nodes, which contain important information used for troubleshooting purposes. The solution ingests large amounts of logs which contain network event information. The logs are gathered through an event-based monitoring (EBM) system, which is an embedded recording tool in the Ericsson EPG, SGSN and MME nodes [5].

This paper is organized as follows. Section II provides an overview of existing research on small files in the Hadoop environment, as well as the research on the applicability of Big Data solutions in the telecommunications industry. Section III shows the architecture of the developed solution and how the small files problem in Hadoop impacts it. Section IV shows the benchmark results for the scenarios laid out in Section III.

II. RELATED WORK

This section provides an overview of the results of existing research in the field of applying Big Data in the telecommunications industry, as well as previous work done on the analysis of small files in Hadoop.

A. Big Data Analytics Platforms in the Telecommunications Industry

Big Data solutions have become an important part of today's industry for all types of businesses, such as finance [6], law enforcement [7], education [8] and others. To show the applicability of Big Data solutions in the telecommunications industry, we briefly summarized some existing use cases and their impact on the industry.

Telecom operators have access to large amounts of valuable data that can be used for various analytical use cases. The research in [9] shows a way to leverage Big

Data analytics for classifying subscribers based on their movement. Reusing existing data for new use cases such as this is the first step in data monetization, which is expected to become a major source of income for all types of businesses in the near future [10]. The work in [11] is able to predict customer churn for telecom operators, which has a direct impact on the operator's profitability. Similarly, the work in [12] predicts customer experience when using over-the-top (OTT) applications such as WhatsApp or Viber.

As the Internet of Things (IoT) is evolving, it is expected to have an impact in the way telecommunications providers analyze the large amounts of sensor data such systems bring with them [13]. We would also like to note that the move to Big Data has a big impact on network infrastructure evolution, as such systems require higher link speeds to transfer the data form one node to another [14].

The solution developed in this paper is focused on mobile network performance troubleshooting. Thus, it is used to calculate various key performance indicators (KPIs) relevant for this domain. KPI measurement is frequently used by mobile operators and mobile network infrastructure vendors as a means to systematically search and identify network bottlenecks and anomalies [15].

B. Analyzing Small Files in the Hadoop Environment

Hadoop is an open-source software framework used for distributed storage and processing of very large data sets [16]. The Hadoop distributed file system (HDFS) has been widely adopted as the standard for storing data in Hadoop based clusters [17]. In the Hadoop ecosystem, access to stored data is handled by a system called Namenode, which manages the file system namespace and regulates client access. First the client asks the Namenode for instructions on where to find the files it needs to read, as well as the location of a free block it can write to [18]. Figure 1 illustrates this process. DataNodes provide block storage and serve I/O requests from clients.

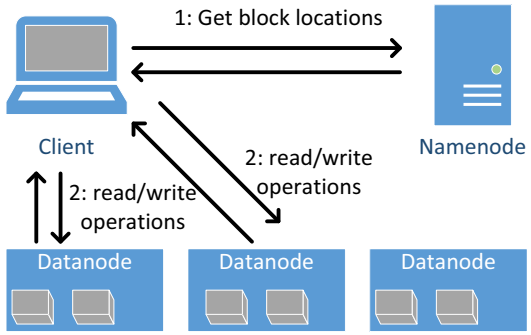

Figure 1. HDFS system overview

A major drawback of HDFS is its poor performance with large numbers of small files, which has attracted significant attention [19]. According to the research in [19], the main reasons for such lower performance are:

- large numbers of small files impose a heavy burden on NameNode memory;
- correlations between small files are not considered for data placement;
- no optimization mechanism, such as prefetching, is provided to improve I/O performance

We would like to note that when small files are stored on HDFS, disk utilization is not a bottleneck. The research in [20] shows that a small file stored on HDFS does not take up any more disk space than is required to store its contents. More precisely, a 6 MB file stored with an HDFS block size of 128 MB uses 6 MB of disk space, not 128 MB.

HDFS is designed to read/write large files, and provides no optimization for handling small files. In cases where large amounts of small files are accessed directly in HDFS, a mismatch of accessing patterns will emerge [21]. HDFS will ignore the optimization offered by the native storage resource, which will lead to local disk access becoming a bottleneck [22]. Additionally, in such a scenario data prefetching is not employed to improve access performance for HDFS [22].

The research in [21] considers all files smaller than 16MB as small files, although no justification or proof were provided as to why this size was chosen as the cut-off point between large and small files in the context of HDFS. The research in [19] has quantified this cut-off point through experimentation. The study indicates that access efficiency starts to drop significantly with files smaller than 4.35 MB.

The small file processing problem in Hadoop has seen many different solutions with various levels of success, depending on the nature of the data. One of these is the merging of multiple small files into a single bigger file, which has shown some significant performance improvements [21], [23]. This paper explores different scenarios in which this solution can be applied to mobile network data. The scenarios are explained in more detail in the following sections.

III. System Overview

This section provides an overview of the developed Big Data solution for mobile network performance troubleshooting.

A. Data Collection

Event data has been used for various troubleshooting purposes [5]. The event data is collected from EPG, SGSN and MME nodes within the core network. The environment is based on the Evolved Packet Core (EPC) [24], as sown on Figure 2. This study uses only event data generated by these nodes. The authors used an event-based monitoring (EBM) system, which is an embedded recording tool in the Ericsson EPG, SGSN and MME. We collected events on 2G, 3G and 4G networks.

The event data is collected in small log files which are between 5 and 15 MB in size, depending on the configuration. The overall size and and velocity of the logs depends on the size of the network (e.g. number of base stations/eNodeB, number of EPG, SGSN and MME

nodes, overall network throughput). An average operator will generate around 200 GB of logs per day.

The log files are stored in a binary format which needs to be decoded to text (usually CSV) in order to be processed. After the decoding process, the files are up to 10 times larger than in their binary format. EBM logs contain information that documents successful and unsuccessful events for completed mobility and session management procedures.

As shown in the following section, the developed solution was tested on several small and large networks around the world.

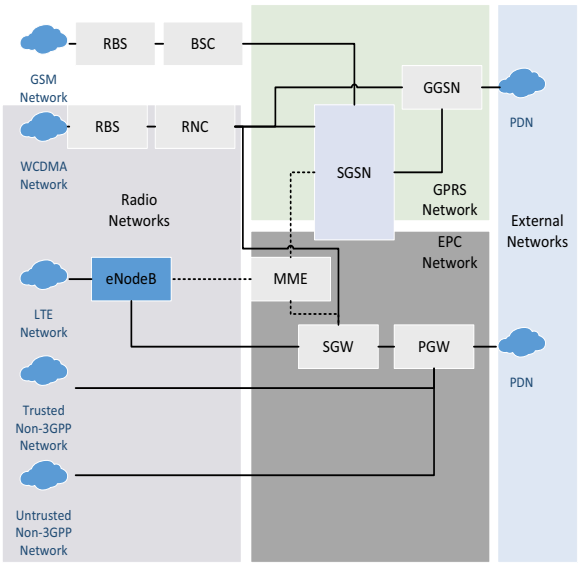

Figure 2 - Evolved Packet Core (EPC) schema

B. *Architecture*

Figure 3 shows the architecture of the developed solution. Apache Flume is used to transfer the data from a network location to HDFS. Flume was chosen because it is a widely used distributed, reliable and available service for efficiently collecting, aggregating and moving large amounts of streaming event data [25]. The binary logs are usually dumped to a server, which is usually somewhere in the operator's network. Using Flume, the binary logs are transferred to the cluster that hosts the proposed solution. Although the proposed solution can be deployed within the operator's network, we argue that a centralized off-site deployment is more appropriate. A centralized approach enables data aggregation from various networks into a single cluster, and thus enriches the data with variety that comes from different network configurations and environments.

HDFS is used to store the raw binary log files until they are decoded. A MapReduce job is used to decode the binary files into CSV. The decoding process is explained in more detail in the following section.

The decoded CSV files are imported into an Apache Hive database [26]. Hive offers an SQL-like query language which enables data access. The developed solution has a number of Hive queries that calculate various KPI's, which provide insight about mobile network bottlenecks. This is a quick method of finding out which parts of the network are worth looking into during troubleshooting. The results, or KPI's calculated from such queries, are stored in a separate relation database, which is based on PostgreSQL. External applications can also connect to the Hive database and query the data to calculate KPI's relevant for mobile network troubleshooting. One of the goals for this solution is to enable data mining. Mobile network experts can connect to the Hive database either though the Hive shell or by using a visualization tool like Tableau. Using the original measurement data stored in Hive, mobile network experts can extract new insight and get to the root cause of a problem. This is often not possible with aggregated KPI data because it hides much of the information it is derived from.

Hive provides several mechanisms for optimizing the storage of the data and query performance. The developed solution makes use of partitioning and bucketing functionalities offered by Hive. Partitioning in Hive is the process of horizontally dividing the data into a number of smaller and more manageable slices. Every partition is stored as a directory within a data warehouse table in Hive. The developed solution partitions the decoded CSV data based on the event identifier (or event name) attribute.

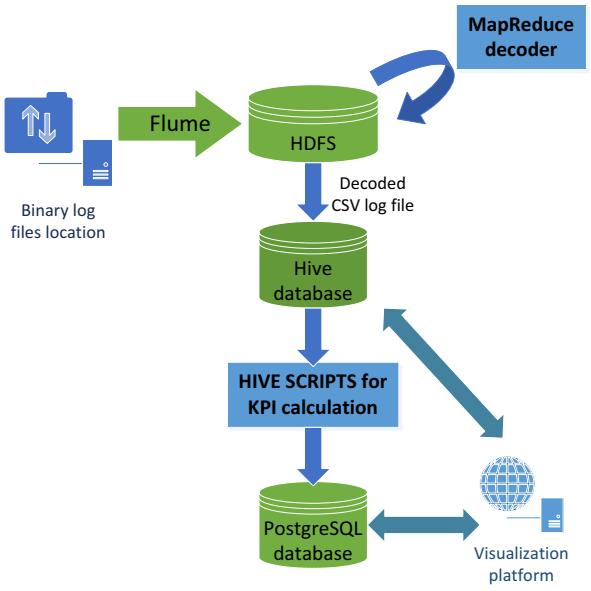

Figure 3 – Architecture of the developed solution

Bucketing is another technique of decomposing data into more manageable parts. This optimization method distributes the data evenly across multiple files. It is used to distribute and organize the table or partition data into multiple files so that similar records are present in the same file. The value of this column will be hashed into buckets by a user-defined number. Bucketing has many performance benefits, most notably faster Map side joins, more efficient grouping using "Group By" statements and more efficient sampling. As the developed solution is used by mobile network experts, such statements are used very often when accessing the Hive database directly.

C. Log Decoder and the Impact of Small Files in Hadoop

The developed solution implements a MapReduce job to decode the binary files into the CSV format. The MapReduce job first reads the binary files in memory and then decodes them in parallel. The number of parallel decoding jobs is defined by the number of input splits in Hadoop. This is where the small files problem influences the developed solution. If each raw log file is decoded separately, it will have a negative performance impact. In the developed solution, we tried to influence the number of input splits by combining multiple raw log files into one larger file. More precisely, we tested the performance impact of using small files with the following scenarios:

- Scenario 1: the raw logs are stored as small files in HDFS and they are directly used as input splits in the MapReduce job. A custom Hadoop input reader is used to read the binary log files, which is based on the native RecordReader class in Hadoop (*org.apache.hadoop.mapreduce.RecordReader*)
- Scenario 2: the raw logs are combined into larger files, stored in HDFS and then decoded using MapReduce jobs. Each line in the combined file is a hexadecimal representation of the binary log file. In this scenario, Flume is used to combine the raw log files. Thus, a native Hadoop input reader is used (located in *org.apache.hadoop.mapreduce.lib.input.TextInputFormat*)
- Scenario 3: the raw logs are combined into larger files, stored in HDFS and decoded using MapReduce jobs with only mappers and no reducers. Like in Scenario 2, each line in the input file contains a single file, and Flume is used to combine the raw log files

IV. RESULTS

For the purposes of this study, a Hadoop-based cluster was used to evaluate the developed solution. The cluster is based on the Hortonworks Data Platform (HDP) [27] and is composed of 10 servers (2 masters and 8 slaves). The master nodes have 2 model E5-2630 CPU-s, 128GB of RAM and 6 hard disk drives (HDD), each with 3TB of space. The slave nodes have 1 model E5-2623v3 CPU, 64GB of RAM, and 8 HDDs, each with 2TB of space. Each node runs on top of CentOS v7, which is installed on a separate SSD disk which is not part of the HDFS.

The input data was collected from several small and large networks around the world, including operators from Europe, North and South America, and Southeast Asia.

A. Small Files Decoding Benchmark

The combined files in Scenario 2 and 3 were grouped into larger files. Several performance tests were carried out on batches of 2, 9 and 33 GB of raw log files.

Table 1 shows how the decoder performs in Scenario 1, when the small log files are used directly. It can be seen that the MapReduce decoder is having difficulties processing even the smaller batches of 2 and 9 GB of raw logs. As stated in previous sections, the reason for this is the large number of small files that is imposing a heavy burden on NameNode memory. In contrast, Scenario 2 (Table 2) shows a significant performance improvement of the MapReduce decoder. Also, we used 18 reducers in Scenario 2, one for each event type available through the logging system.

Undoubtedly, the best performance was achieved in Scenario 3 by combining the input files into larger files and using map-only jobs (Table 3). The reason for such an improvement is that both the shuffle-sort and the reduce phases of the MapReduce job are skipped, thus drastically reducing the amount of processing power and memory needed to decode a large input file. For the largest batch of 33 GB, we can see that there is a 37% improvement compared to Scenario 2. We would like to note that this improvement increases with batch size (Figure 4). Figure 5 shows the performance gain for each scenario. The performance improvements rise with the batch size, which is traditionally not the case for solutions that are not based on Big Data.

The drawback of using the approach in Scenario 3 is that the output of the decoder is split into several smaller files which need to be imported into Hive. This is due to the way MapReduce jobs work. The intermediate results of the mappers are shuffled and sorted before being delivered to the reducers. By having only map jobs in our decoder, the unsorted intermediate results become the output. In contrast, when the reducers are used we can influence the number of files that will be generated. For example, each event type could be stored into a separate file. However, the files can easily be merged within Hive, as the output is textual (CSV). Also, the partitioning and bucketing in Hive restructures the physical layout of the data, so that the output of the map-only decoder does not influence the performance of the Hive queries.

TABLE 1. DECODER BENCHMARK FOR SCENARIO 1 – USING SMALL LOG FILES AS INPUT

Raw log size (GB)	*Seconds*	*Minutes*
2.00	843	14.05
9.00	3206.00	53.43
33.00	13740	229.00

TABLE 2. DECODER BENCHMARK FOR SCENARIO 2 – USING 18 REDUCERS AND SEVERAL LARGER COMBINED INPUT FILES

Raw log size (GB)	*Seconds*	*Minutes*
2.00	428.00	7.13
9.00	1405.00	23.41
33.00	7282.00	121.36

TABLE 3. DECODER BENCHMARK FOR SCENARIO 3 – USING ONLY MAPPERS AND SEVERAL LARGER COMBINED INPUT FILES

Raw log size (GB)	*Seconds*	*Minutes*
2.00	348.00	5.80
9.00	883.00	14.71
33.00	3194.00	53.23

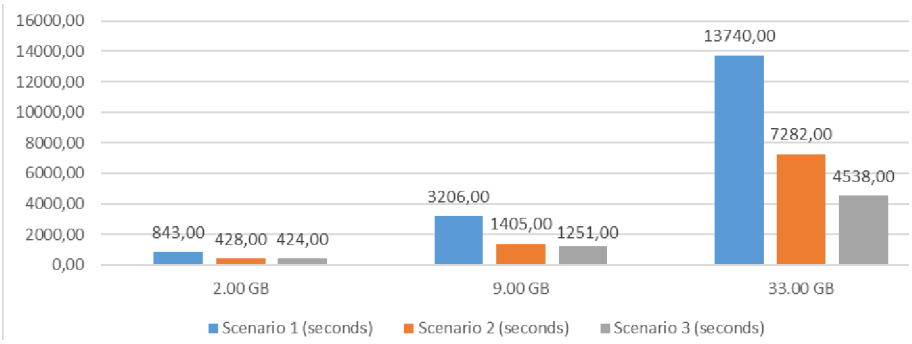

Figure 4 – Batch processing benchmark for 3 considered Scenarios

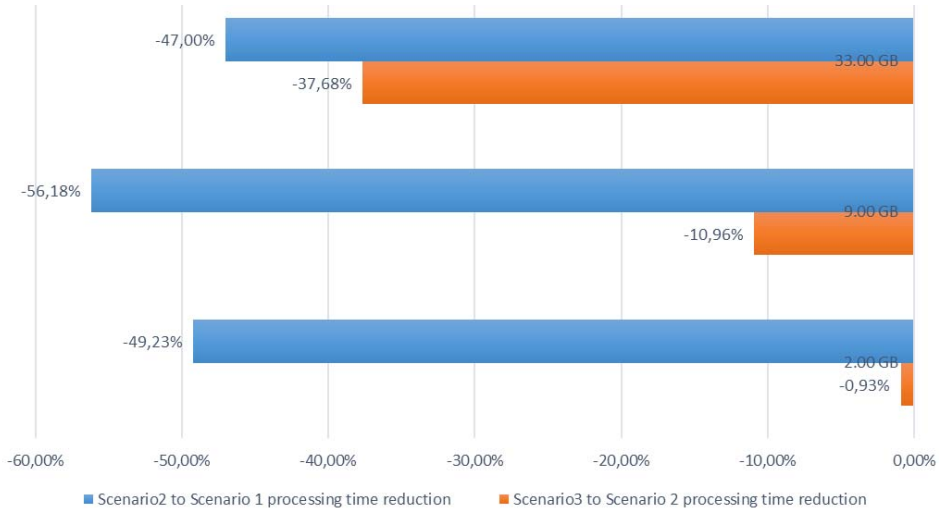

Figure 5 – Scenario to scenario processing time reduction comparison

B. *Comparison with existing solutions*

In order to clearly show the performance benefits when using big data solutions and technologies in this scenario, the study also shows the benchmark for decoding EBM logs without the proposed solution. Table 4 shows the performance benchmark on a single server, with the same hardware as the master node in the HDP cluster.

As shown in table Table 4, the proposed solution is up to 6 times faster than existing solutions. The largest set of logs was unable to be measured since it was not possible with the existing solutions.

We note that the performance gains represent only one benefit of the proposed solution. The main advantage of the proposed solution is the ability to process a much larger set of logs than was possible with legacy solutions. Also, the proposed solution provides a way to continuously gather and store logs for deeper analytics, which was no the case with legacy solutions.

TABLE 4 EXISTING SOLUTIONS BENCHMARK

Raw log size (GB)	***Seconds***	***Minutes***
2.00	1264.00	21.07
9.00	5538.00	92.30
33.00	N/A	N/A

V. CONCLUSION

Big Data analytics can provide insight into the data available within the telecommunications industry. This paper demonstrates one use case in which analytics can be leveraged to improve the efficiency and value of troubleshooting in mobile networks. The main advantage of the developed solution is its ability to adapt to any new analytical requests, as well as the ability to adapt to changing input file sizes. More precisely, the results of this study show that the developed solution is capable of processing small files in an efficient manner within the Hadoop environment, which was not built for processing large amounts of small files. Additionally, the study shows that by skipping the reduce phase we can decrease the execution time of the MapReduce job used for decoding. This was shown to be the most time-consuming process. The developed solution was tested using log data collected in various small and large networks from around the world, which demonstrates its applicability for mobile network troubleshooting. The developed solution has proven that Big Data platforms are suitable for processing large batches of mobile network data, and that they bring significant performance and scalability improvements compared to traditional solutions. Future research may include the use of other Big Data tool that run on the Hadoop platform. Most notably, the use of Apache Cassandra instead of the Hive, and the use of Apache Spark as a replacement to the MapReduce decoder job.

ACKNOWLEDGMENT

This study was fully funded by Ericsson Nikola Tesla. The authors thank their leadership team for providing an environment in which it was possible to combine research and industry into one.

REFERENCES

[1] G. Fettweis and S. Alamouti, "5G: Personal mobile internet beyond what cellular did to telephony," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 140–145, Feb. 2014.

[2] C. Patrik, L. Anette, and J. Peter, "Ericsson Mobility Report," EAB-16:018498 Uen, Revision A, Nov. 2016.

[3] D. Šipuš, "Big data analytics for communication service providers," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 513–517.

[4] L. Yang, G. Kang, W. Cai, and Q. Zhou, "An Effective Process Mining Approach against Diverse Logs Based on Case Classification," in *2015 IEEE International Congress on Big Data*, 2015, pp. 351–358.

[5] I. da Silva, Y. Wang, F. Mismar, and W. Su, "Event-based performance monitoring for inter-system cell reselection: A SON enabler," in *2012 International Symposium on Wireless Communication Systems (ISWCS)*, 2012, pp. 6–10.

[6] A. Munar, E. Chiner, and I. Sales, "A Big Data Financial Information Management Architecture for Global Banking," in *2014 International Conference on Future Internet of Things and Cloud*, 2014, pp. 385–388.

[7] A. Jain and V. Bhatnagar, "Crime Data Analysis Using Pig with Hadoop," *Procedia Computer Science*, vol. 78, pp. 571–578, Jan. 2016.

[8] F. Xhafa, D. Garcia, D. Ramirez, and S. Caballé, "Performance Evaluation of a MapReduce Hadoop-Based Implementation for Processing Large Virtual Campus Log Files," in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 2015, pp. 200–206.

[9] B. Furletti, L. Gabrielli, C. Renso, and S. Rinzivillo, "Analysis of GSM calls data for understanding user mobility behavior," in *2013 IEEE International Conference on Big Data*, 2013, pp. 550–555.

[10] H. Cao *et al.*, "SoLoMo analytics for telco Big Data monetization," *IBM Journal of Research and Development*, vol. 58, no. 5/6, p. 9:1-9:13, Sep. 2014.

[11] H. Li, D. Yang, L. Yang, YaoLu, and X. Lin, "Supervised Massive Data Analysis for Telecommunication Customer Churn Prediction," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, 2016, pp. 163–169.

[12] E. Diaz-Aviles *et al.*, "Towards real-time customer experience prediction for telecommunication operators," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 1063–1072.

[13] S. Din, H. Ghayvat, A. Paul, A. Ahmad, M. M. Rathore, and I. Shafi, "An architecture to analyze big data in the Internet of Things," in *2015 9th International Conference on Sensing Technology (ICST)*, 2015, pp. 677–682.

[14] I. Tomkos, C. Kachris, P. S. Khodashenas, and J. K. Soldatos, "Optical networking solutions and technologies in the big data era," in *2015 17th International Conference on Transparent Optical Networks (ICTON)*, 2015, pp. 1–1.

[15] S. Singh, Y. Liu, W. Ding, and Z. Li, "Evaluation of Data Mining Tools for Telecommunication Monitoring Data Using Design of Experiment," in *2016 IEEE International Congress on Big Data (BigData Congress)*, 2016, pp. 283–290.

[16] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–10, May 2010.

[17] W. Tantisiriroj, S. Patil, and G. Gibson, "Data-intensive File Systems for Internet Services: A Rose by Any Other Name... (CMU-PDL-08-114)," *Parallel Data Laboratory*, Oct. 2008.

[18] S. Bende and R. Shedge, "Dealing with Small Files Problem in Hadoop Distributed File System," *Procedia Computer Science*, vol. 79, pp. 1001–1012, Jan. 2016.

[19] B. Dong, Q. Zheng, F. Tian, K.-M. Chao, R. Ma, and R. Anane, "An optimized approach for storing and accessing small files on cloud storage," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1847–1862, Nov. 2012.

[20] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2009.

[21] X. Liu, J. Han, Y. Zhong, C. Han, and X. He, "Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS," in *2009 IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1–8.

[22] J. Shafer, S. Rixner, and A. L. Cox, "The Hadoop distributed filesystem: Balancing portability and performance," in *2010 IEEE International Symposium on Performance Analysis of Systems Software (ISPASS)*, 2010, pp. 122–133.

[23] P. Gohil, B. Panchal, and J. S. Dhobi, "A novel approach to improve the performance of Hadoop in handling of small files," in *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2015, pp. 1–5.

[24] G. Kuhn, J. Eisl, and H. Becker, "Co-operative handover in 3G System Architecture Evolution," in *32nd IEEE Conference on Local Computer Networks (LCN 2007)*, 2007, pp. 643–650.

[25] P. B. Makeshwar, A. Kalra, N. S. Rajput, and K. P. Singh, "Computational scalability with Apache Flume and Mahout for large scale round the clock analysis of sensor network data," in *2015 National Conference on Recent Advances in Electronics Computer Engineering (RAECE)*, 2015, pp. 306–311.

[26] G. P. Haryono and Y. Zhou, "Profiling apache HIVE query from run time logs," in *2016 International Conference on Big Data and Smart Computing (BigComp)*, 2016, pp. 61–68.

[27] K. K. Gadiraju, M. Verma, K. C. Davis, and P. G. Talaga, "Benchmarking performance for migrating a relational application to a parallel implementation," *Future Generation Computer Systems*, vol. 63, pp. 148–156, Oct. 2016.