VARIOUS APPROACHES PROPOSED FOR ELIMINATING DUPLICATE DATA IN A SYSTEM

Roman Čerešňák[1,*], Karol Matiaško[1], Adam Dudáš[2]

[1]Department of Informatics, Faculty of Management Science and Informatics, University of Zilina, Zilina, Slovakia
[2]Department of Computer Science, Faculty of Natural Siences, Matej Bel University, Banska Bystrica, Slovakia

*E-mail of corresponding author: roman.ceresnak@fri.uniza.sk

Resume

The growth of big data processing market led to an increase in the overload of computation data centers, change of methods used in storing the data, communication between the computing units and computational time needed to process or edit the data. Methods of distributed or parallel data processing brought new problems related to computations with data which need to be examined. Unlike the conventional cloud services, a tight connection between the data and the computations is one of the main characteristics of the big data services. The computational tasks can be done only if relevant data are available. Three factors, which influence the speed and efficiency of data processing are - data duplicity, data integrity and data security. We are motivated to study the problems related to the growing time needed for data processing by optimizing these three factors in geographically distributed data centers.

Article info

Received 27 November 2020
Accepted 26 January 2021
Online 23 June 2021

Keywords:

duplication data,
distributed data processing,
software architecture

Available online: https://doi.org/10.26552/com.C.2021.4.A223-A232

ISSN 1335-4205 (print version)
ISSN 2585-7878 (online version)

1 Introduction

Severe data change led not only to a change of database types, specifically to a data transfer from the relational databases to non-relational ones, but it caused a rapid development of technology known as distributed data processing. Google belongs to the first pioneers of this technology, which spreads its network in 13 data centers in 8 countries on four continents [1]. The potential of big data indicates great benefit in various industries since 2007 - whether in economical statistics, transport system management or behavior of transport unit analysis, in the medical field with the diagnosis of illnesses or in the computer science. The costs, related to the performance of computing operations, were growing with the popularity growth in relevant industries.

Hand in hand with the increasing use of distributed processing systems, the first problems were occurring related to distributed data processing. A considerable effort was made to reduce the issues related to the processes working with distributed data. A method of maximal duplicated segment and choice of storage nodes to distributed file fragments was created. System of duplication detection and fragmentation (DDFP) and method for deduplication - which virtually eliminates the duplicated data - were designed. Other than that, there were significant methods developed for the storing of data fragments and allocation of unique occurrences of the data file on nodes [2]. Suppression of duplicities in the systems was also examined in study [3], which deals with data processing by technologies like MapReduce, Spark and SQL queries.

Secondly, the network security became crucial component in the data processing and various fields and sectors of informatics. Regarding distributed data processing, the security is divided into variety of elements, such as Tool security, environment Hadoop, Flink, Samza, Spark [4] or many others. Critical parts of cloud security consist of the safety of the data storing in the cloud [5], monitoring, anomaly and threat detection [6] and last but not the least data audit [7]. The largest part of security is management of accesses and keys, sharing the access groups and last, but not the least, there is a problem of security of data itself.

Thirdly, the service quality (QoS) of big data tasks was not taken into account in already existing work. Similar to conventional cloud services, the big data applications exhibit a service level agreement (SLA) between a service provider and the applicants. The computational tasks are foremost determined according to where they are located and number of computation resources assigned to the task. The portable speed is another significant influence factor because the big data tasks are devoted to the data. The computational task cannot continue until the available data do not match each other. Existing studies regarding the computing clouds are generally, for example [8], mainly focused on restriction of computational capacity while ignoring the limits of the system's portable speed.

Optimization of solutions, studied here, is related to data duplicity, data integrity and data security in the geographically distributed centers, to overcome the weak points related to the distributed data processing in a cloud envirnment. The servers are equipped by the limited storing and computational tools. Each data block has a demand for storage space. The goal is to minimalize the number of duplicities of the big data, reduce the number of accesses and increase data integrity. Main benefits of presented research can be summarized as follows:

- Based on the growing amount of data, it is necessary to control the data in the system; Not only that the speed of the data processing is increased, but the amount of storage space needed for the data is effectively reduced.
- Here is dealt with the minimization of the data access - the data centers - and thus the higher data security is ensured and the amount of restrictions is reduced.

The rest of the paper is arranged as follows. Part 2 summarizes the related work. This part is the crucial aspect of research direction compared to other researchers' approaches, who are dealing with the same research topics. Part 3 presents the system model. it is necessary to find out which architecture is more effective in eliminating duplicate data which enter the system during our research. Two architectures are introduced and the proposed methodology was tested experimentally. These experiments are described in part 4 and serve as verification of theoretical findings. Finally, conclusion of the presented research and experiments is provided in the part 5.

2 Related work

Many factors influence the current state of the distribution data processing. It is well known that data that comes to the system contains in any way the same duplicate data. This factor influence leads to an increase in the transformation process and of the course distribution process. The same data for multiple users are stored at the same time. This factor leads, with the replication factor equal to 3 for two users, to store data 6 times. This wasting memory and increasing server costs lead many researchers to develop methods, which decrease the duplicated data and save the operating costs. During this research, many papers that worked with the issue were studied.

The researchers in [9] created an approach based on the method known as Data deduplication. This data deduplication process is widely used in cloud storage to decrease storage space and upload bandwidth. However, only one copy of each file is stored, even if many users own such a file. By using, deduplication system, the progress of storage utilization and reliability is increased. Besides that, the dare of privacy for sensitive data also occurs when outsourced by users to the cloud. In this approach authors used new distributed deduplication systems with upper dependability. The data chunks are distributed from the corner to cornering multiple cloud servers. The well-being needs of information security and label soundness are refined by presenting a deterministic mystery sharing plan in circulated capacity frameworks. On the other hand, a deduplication technique can reduce the server-side storage cost and save the upload bandwidth at the user side.

In further research, authors came across a method of data mining and fuzzy logic, the aim of which is to eliminate duplication. In the paper [10], a practical and novel duplicate elimination system is presented, which exploits a fuzzy inference engine for handling the uncertainty involved in detecting fuzzy duplicates. The system's innovation is capturing an expert's knowledge in the form of natural language fuzzy rules and using these simple rules to clean the data efficiently. This, in turn, reduces the time required for the repetitive and time-consuming task of hard-coding for deduplication based on a scheme for each database.

The presented examination issue's objective was to dispense with duplication of information, which is the essential objective. Notwithstanding, it is understood that replication in numerous regions also guarantees the framework's unwavering quality during the exploration. To look after dependability, authors were compelled to address consistency with steadfast quality in lessening duplications. The specialists [11] considered the referenced viewpoint and they thought of the case that the deduplication framework improves capacity usage while reducing dependability. Besides that, the test of protection for touchy information likewise emerges when clients redistribute them to the cloud. Meaning to address the above security challenges, this paper makes the primary endeavor to formalize the idea of dispersed, dependable deduplication frameworks. The new disseminated deduplication frameworks were proposed with higher unwavering quality in which the information pieces are distributed over various cloud workers. The security prerequisites of information classification and label consistency are likewise accomplished by presenting a deterministic mystery sharing plan in appropriated capacity frameworks, rather than utilizing united encryption, as in past deduplication frameworks. Security examination shows that the proposed deduplication frameworks are secure, as far as the proposed security model's definitions. As confirmation of the idea, the proposed frameworks were actualized and it is indicated that the acquired overhead is restricted in practical conditions. Cloud companies from Amazon, Microsoft or Google, are among the areas that are gaining more and more popularity, which results in exploring duplicates even in a quiet environment. In their work [11], the researchers presented certain pitfalls

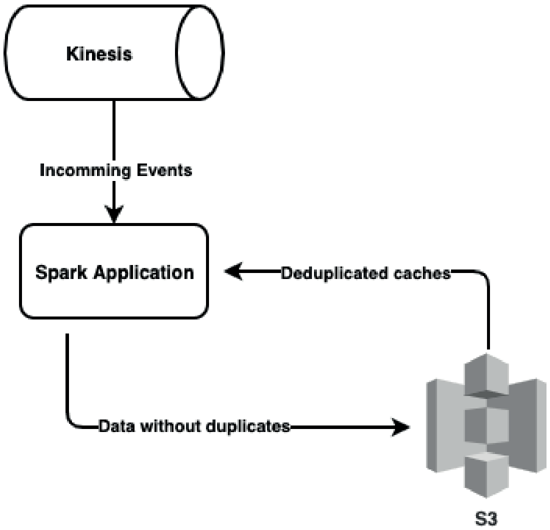


***Figure 1** Stream data processing*

that occur in cloud services. They also provided a solution at work that has developed a solution that includes data security and space efficiency in server storage and distributed content checksum storage systems. Here is adopted a method called interactive Message-Locked Encryption with Convergent Encryption (iMLEwCE). In this iMLEwCE, the data is encrypted firstly and then the ciphertext is again encrypted. Block-level deduplication is used to reduce the storage space. Encryption keys are generated in a consistent configuration of data dependency from the chunk data. The identical chunks will always encrypt to the same ciphertext. The hacker cannot deduce the keys configuration from the encrypted chunk data. Thus, the information is protected from the cloud server. This paper focuses on reducing the storage space and providing security in online cloud deduplication.

In paper [12], researchers defined their own proposed temporal architecture concerning element registration in the system. Whereas the data models depicted attributes and temporality definition, evolve, it is necessary to create a complex environment and possibilities for dealing with these changes and reflections to the temporal management layer. The temporal registration concept does it. Researchers developed a method that significantly decreases the duplication of data in a database, which led to reduced performance of used servers.

Another interesting idea was presented in [13], which deals with the granularity management of the temporal system proposes a data sharing model based on the reliability, sensitivity and precision of data providers. A new concept of the temporal benefit is introduced, which is consecutively evaluated in the experiment section. Optimization of the data flow by historical data aggregation and limitation of the data amount is a core part for the system decision making, whereas the time for data transferring is strictly limited.

A similar method, as one presented in [9], was shown in [2]. The process they took as the primary method and from which they relied on is called data deduplication. In their paper, the researchers saw the main challenge of identifying the maximum duplicate segment and selecting the storage nodes to distribute files. In this paper, researchers proposed Duplicate Detection and Fragment Placement (DDFP). This deduplication system virtually eliminates duplicate data and fragments placement that allocates individual data files on storage nodes. For repeated data, a reference point is used and unique data is stored on the storage node. The created approach increases the percentage of duplicate data detection. A fragment placement algorithm is used for placing fragments on different storage nodes. The T-coloring is used to select nodes, set T is used, which restricts the nodes at distance T from one another. The DDFP considerably achieves duplicate elimination and obtain a high level of security on data fragments by storing fragments of the data file using the T-coloring. This selects the nodes that are not adjacent, which prevents unauthorized access to data from other users.

As can be seen from previous research, the problem being solved in the current paper increasingly impacts efficiency of the algorithm and other factors, such as reliability, reducing server utilization and reducing the cost of storing data in the system.

3 Design and implementation of the real-time data catching method

In the rapidly developing world, the duplicated data are a big problem, which was examined in various studies. This paper's main objective was designing and implementing a method capable of catching the data in real-time, thus reducing the data duplicity to a minimum. The architecture, presented in Figure 1, is designed to fulfill this objective.

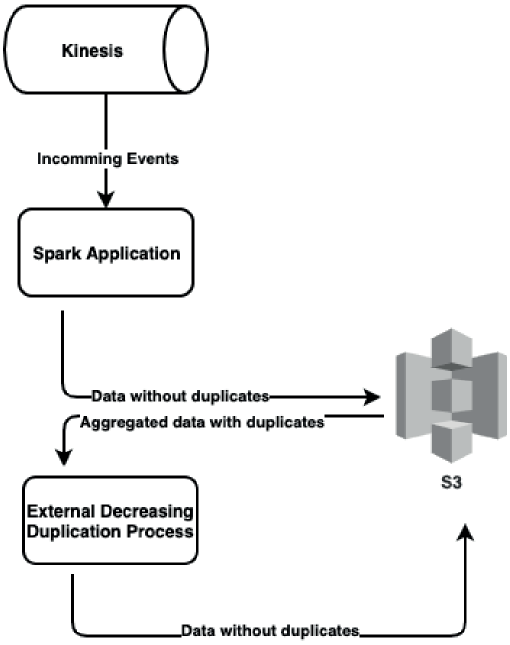


Figure 2 *Improved stream data processing*

3.1 One data storage

As presented in Figure 1, this architecture consists of three essential parts: Amazon Kinesis, Apache Spark and S3.

Naturally, while burning-through information from Kinesis, Spark provides data catching in any time period. This is accomplished with Kinesis checkpointing, which keeps up the putting away and balances time frame Kinesis stream in DynamoDB. On the off chance that Spark crashes in the middle of the Kinesis checkpointing, it always begins with calling of all the functions from the earliest starting point of the span. This implies that each operation can be processed or entered into system more than once, which can lead to creation of duplicates.

Apache Spark provides a component to avoid duplicates, which recoups information after the crashes and keeps the streams from handling documents called Spark checkpointing. The implicit Spark Streaming element should not be mistaken for the Kinesis checkpointing (which was referenced above). At the point when the Spark checkpointing is empowered, Spark stores metadata and handles RDDs to dependable, persevering capacity, e.g., HDFS. Another component of the Spark Streaming are the Write Ahead Logs (WALs). The WAL contains information obtained from Kinesis (or some other information stream). It is utilized for the state recuperation after the disappointments of the driver and the beneficiaries. Once implemented, the blend of checkpointing and WAL should be useful as a way of precision boosting of the system. As it may be, these highlights in Spark don't work effectively and cause execution corruption, mainly if more seasoned renditions are utilized. In this implementation, Spark 1.5.2 was used.

To maintain a necessary distance from each of these issues, it is chosen to utilize Kinesis checkpointing to update it at any rate, ensure and plan a different lightweight answer to limit the probability of duplicates. Two unique methodologies were designed; the second worked fundamentally better than the first one. Both methodologies are next described in some detail and it is clarified how authors came to that conclusion.

The system portrayed in Figure 2 did not make the data catching effective even though the duplicity recognition was possible. The architecture presented in the Figure 2 solves this problem with a different approach. It catches not only the data, which do not contain duplicity but exactly the data, which include deception. it was decided to use the Amazon S3 file storage, since it was necessary to effectively control parallel processes, parallel approach and useful incoming data expanding the in-memory database such as Redis or Memcached.

During the handling, the information was divided into the Spark stream. Each function got a particular ID dependent on the current timestamp and was set in a basin determined as a hash code of ID function. Here is an example of the information partitioning:

```
def createPair(fields: Map[String, String]):
(DateTimePartitionKey, Map[String, String]) = {
    val partitionKey = DateTimePartitionKey(
        fields(FlowField.EventId)
        fields(FlowField.EventHour),
        fields(FlowField.EventDate),
    )
    partitionKey -> fields
    }
```

To accomplish the non-problematic execution, it was expected to run the deduplication prepared concurently.

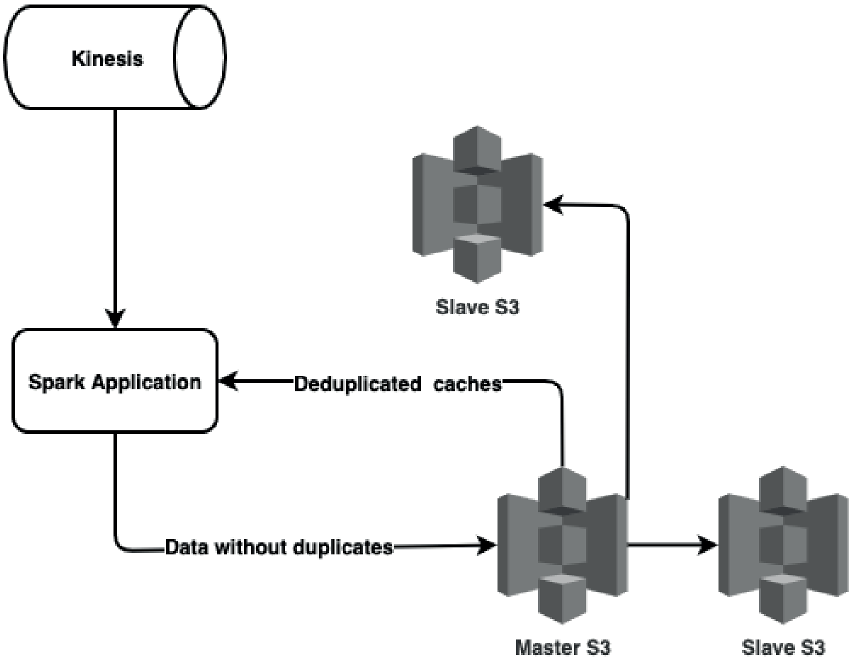


***Figure 3** Multiple data storage architecture*

Since the information, corresponding to the parceling plan depicted above was previously handled, the similar parallelism was used for deduplication. Consequently, it was expected to parcel the reserve put away in the S3 file system in a similar way the information in the stream was divided.

3.2 Multiple data storage

During the research, the second method, capable of working with the data, was also examined. As opposite to the first proposed method, the data is stored in several data storages. A new type of architecture was developed due to the multiple data storage spaces, which is portrayed in Figure 3.

To achieve the satisfying performance, it was necessary to start the deduplication processing in parallel. Since the data were already processed in parallel with the above-described division scheme, the same type of parallelism was used for the deduplication process. That is why it was necessary to divide the cache. The date in the file system S3 was stored by use of the method for splitting the data in the stream.

The division into sections was implemented with the hierarchical directory structure:

- The first level - batch creation data
- The second level - time (hour) of the batch creation
- The third level - the storing of cache files to a bucket

Having multiple cache files for each segment could become a problem caused by the high latency of reading operations in S3. Operation efficiency of S3, however, was sufficiently compelling and because of the achievement of remaining performance, the files were read in several threads. The same approach was also valid for writing the data to S3. The S3 File System was used since the parallel processing of the data was needed to secure the testing and real purposes for several Amazon instances. Amazon S3 File system allows several sources to access the data and read the information in them, which is one factor that played an essential role in the seletion of data storage.

For the consistency sake, both data and storages were endured in one exchange. This was conceivable, since both knowledge and reserves were put away in the S3 filesystem. During the group preparation, information and stores were kept in touch with the transitory catalogs. When a clump was handled, its substance was moved to the last area as a nuclear activity. In the event, when one of the move tasks fizzled, the entire stream was halted until the predictable state was reestablished physically.

The Cache itself is a straightforward information structure comprised of two sections. The first is an unchanging set, which containes information stacked from S3. The other one is a variable set, which was utilized for including new function IDs.

Thus, it was necessary to perform various intermediate steps, as they are portrayed in Figure 4 and to create a code capable of suppressing the duplicity. The source code is provided at the following address: https://github.com/romanceresnak/deduplication/blob/main/deduplication.java.

The function *getCache* needed to be called just once for each parcel key. In the other case, one can use an obsolete reserve since some new function IDs may be continued in an impermanent area. The simplest method to do this is to sort information by key. This should be possible during the repartitioning utilizing worked in the *repartitionAndSortWithinPartitions* Spark

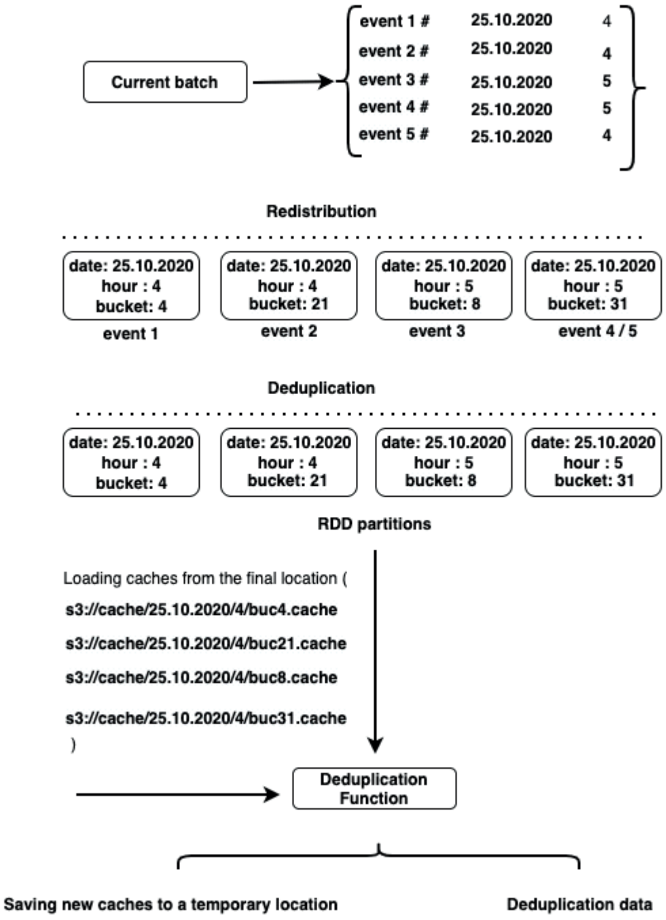

Current batch
event 1 # 25.10.2020 4
event 2 # 25.10.2020 4
event 3 # 25.10.2020 5
event 4 # 25.10.2020 5
event 5 # 25.10.2020 4
Redistribution
date: 25.10.2020 hour : 4 bucket: 4
event 1
date: 25.10.2020 hour : 4 bucket: 21
event 2
date: 25.10.2020 hour : 5 bucket: 8
event 3
date: 25.10.2020 hour : 5 bucket: 31
event 4 / 5
Deduplication
date: 25.10.2020 hour : 4 bucket: 4
date: 25.10.2020 hour : 4 bucket: 21
date: 25.10.2020 hour : 5 bucket: 8
date: 25.10.2020 hour : 5 bucket: 31
RDD partitions
Loading caches from the final location (
s3://cache/25.10.2020/4/buc4.cache
s3://cache/25.10.2020/4/buc21.cache
s3://cache/25.10.2020/4/buc8.cache
s3://cache/25.10.2020/4/buc31.cache
)
Deduplication Function
Saving new caches to a temporary location
Deduplication data

Figure 4 *Individual data processing steps*

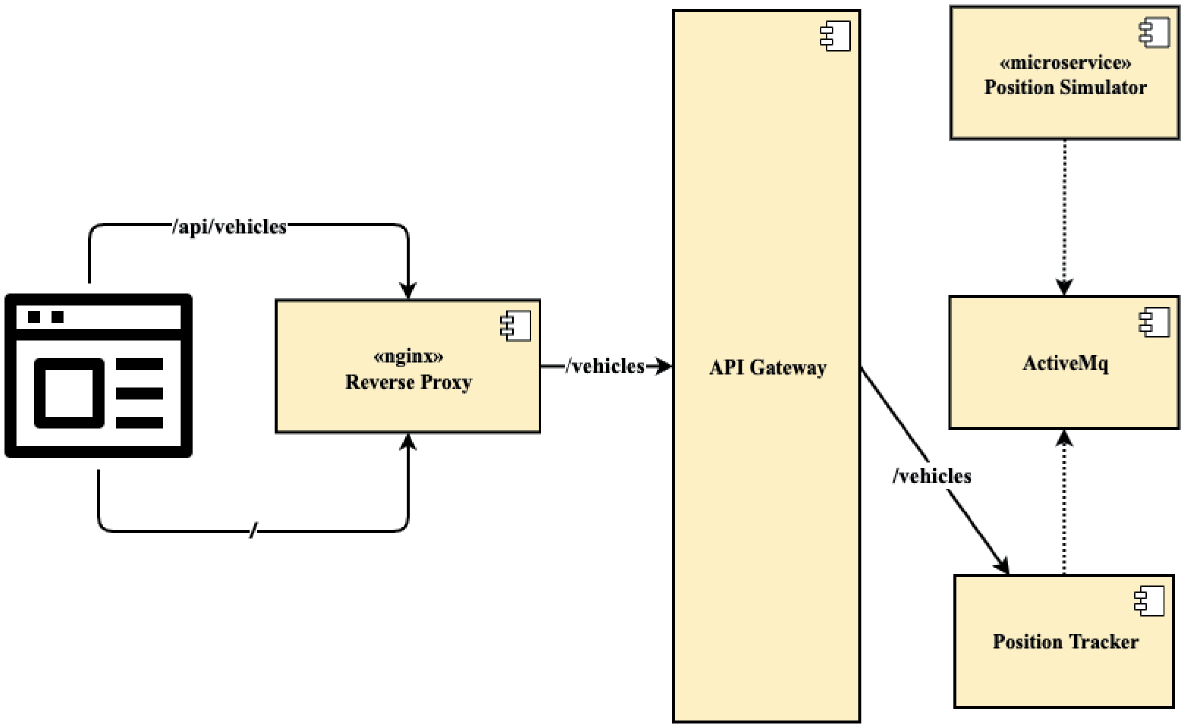
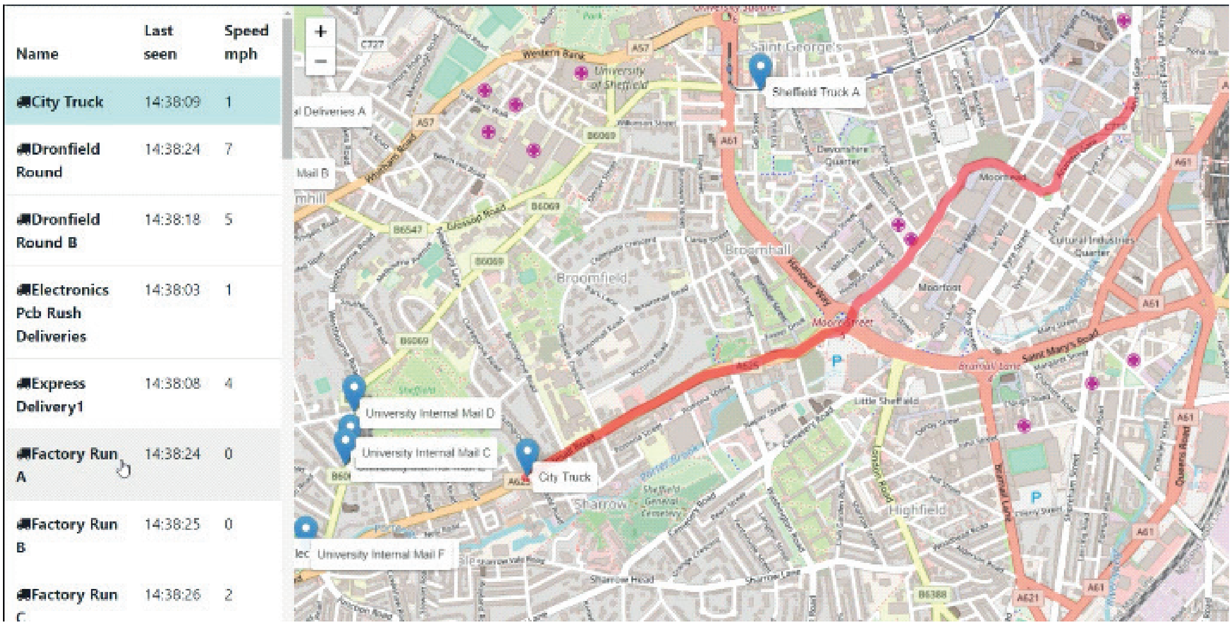


Figure 5 *Architecture of application*



Name	Last seen	Speed mph
City Truck	14:38:09	1
Dronfield Round	14:38:24	7
Dronfield Round B	14:38:18	5
Electronics Pcb Rush Deliveries	14:38:03	1
Express Delivery1	14:38:08	4
Factory Run A	14:38:24	0
Factory Run B	14:38:25	0
Factory Run C	14:38:26	2

Figure 6 Tracking car

Table 1 Server configuration

technology	parameters
• EC2 instance	• cloud
	• a1.medium • vCPU: 1 • MeM(GiB): 2
• EMR cluster	• master: 1x • m3.xlarge • core: 2x • m4.4xlarge
• spark	• 1 r5.12xlarge master node • 19 r5.12xlarge core nodes • 8 TB total RAM • 960 total virtual CPUs • 170 executor instances • 5 virtual CPUs/executor • 37 GB memory/executor • parallelism equals 1,700

technique. At last, the deduplication part of work can be described as follows.

In this capacity, the data stream is mapped to a flood of sets where the principal component is a composite parcel key while the subsequent one contains the information. From that point forward, a changing work is applied to this stream, which permits to work with the RDDs and bunch time. At last, the *repartitionAndSortWithinPartitions* was considered on the RDD and the copies were channeled utilizing the pass-through storing system.

4 Experiments for the real-time data catching method

For experimental testing of proposed methods, an application, which monitors the GPS position of the delivery car, was created. That application is based on microservices and is shown in Figure 5.

Delivery routes can be the same at specific points, but based on this proposal, the duplicate data were captured and were not store it in the database. In the end, this approach allowed not only to reduce the amount of data in the system, which is shown in Figure 6, but also to lower the load of the server.

```
Host namenode
    HostName ec2-18-216-40-160.us-east 2.compute.amazonaws.com
    User ubuntu
    IdentityFile ~/.ssh/MyLab_Machine.pem

Host datanode1
    HostName ec2-18-220-65-115.us-east-2.compute.amazonaws.com
    User ubuntu
```

__Table 2__ Results of GPS data recording in the interval of 1 hour

configuration	one data storage (architecture A)	multiple data storage (architecture B)
RAM	1.2 TB	2 TB
CPU	5 GB	2 GB
FILES	1	50
transform time	85 seconds	90 seconds

__Table 3__ Results of GPS data recording in the interval of 2 hours

configuration	one data storage (architecture A)	multiple data storage (architecture B)
RAM	3.6 TB	4.8 TB
CPU	10 GB	4 GB
FILES	1	103
transform time	2 min and 32 s	2 min and 32 s

__Table 4__ Results of GPS data recording in the interval of 3 hours

configuration	one data storage (architecture A)	multiple data storage (architecture B)
RAM	5 TB	8 TB
CPU	8 GB	4 GB
FILES	1	158
transform time	7 min and 13	5 min and 25 s

```
    IdentityFile ~/.ssh/MyLab_Machine.pem
Host datanode2
    HostName ec2-52-15-229-142.us-east-2.compute.amazonaws.com
    User ubuntu
    IdentityFile ~/.ssh/MyLab_Machine.pem

Host datanode3
    HostName ec2-18-220-72-56.us-east-2.compute.amazonaws.com
    User ubuntu
    IdentityFile ~/.ssh/MyLab_Machine.pem
```

4.1 Results of experiment for the proposed methodology

Three types of data were used for experimental work. For the purpose of the experiments, the configuration shown in Table 1 was used. The application records the GPS positions at intervals of one, two and three hours. The recorded values are presented in Tables 2, 3 and 4, respectively.

The values that were measured during the data recording for the interval of one hour are presented in e Table 2. This data set contains 34 similar records. Three aspects were evaluated - RAM, CPU and number of files - which point to the fact, that with the same configuration, architecture A uses more CPU performance to watch the duplicated components, but it uses less RAM. It also clearly states that all the data are stored in a single file, as oppose to architecture B, which needs up to 50 files in order to work correctly and keep watch over the duplicated data.

Time is crucial. Based on measured values, finding of the duplicated values is faster while using architecture A. A large problem was recognized during the run of the process - it was not the inefficiency of architecture B, but the necessity to merge and manage the duplicated data in various files in parallel.

The values that were measured during the data recording for the interval of two hours are presented in Table 3. it was chosen to limit values, where the efficiency of duplicated data's processing is the same. The CPU load has doubled, which does not cause any significant problems in actual processing. During the experiments, the number of files increased, as well, which allowed effective searching, as oppose to architecture A, processing all the data in a single file – approach, which degrades with higher ammount of data.

Lastly, the values that were measured during the data recording for the interval of three hours are presented in Table 4. In this step, the efficiency of parallel processing with various files can be seen clearly. Even if the operations of file division and subsequently their connecting are time-consuming with the vast amount of the data, the parallel processing, clearly surpasses the efficiency of the processing of data in one file.

The results at the beginning hinted at the superiority of the architecture A. However, regarding the interval of 2 hours, the efficiency was equal for both proposed architectures and architecture B proved its efficiency with the increasing ammount of data.

5 Conclusion

In the current, quickly developing world, the duplicity suppression has a significant influence on the velocity of system performance and the storing of the data to the data storages. The primary objective of many researchers in the area is examining the problem related to the duplicity suppression - user data searching in various data storages and the reduction of the systems needed for the processing of data by system. Finally, the duplicated data increases ammount of space needed in the data storages, which is, in the end, mirrored even in need of buying new hardware and expanding the data centers.

To make the process more effective and reduce computational costs, authors created the method dealing with the catching of the same data coming from several sources to the system. Specifically, the duplicated data are caught in the first phase of our proposed algorithm and by creating a reference to a particular source creating a link. The second phase of proposed algorithm is based on distributed data processing. After finishing the processing, the system always found a particular value, which means the algorithm must create a correct reference for the given value.

The experiments, performed for a test of the designed method confirmed this. The conventional opposite process of distributed data processing, which missed the duplicated value, this system effectively reduced the server overload. It reduced the time needed for the whole distribution of the data on several nodes. The second significant result, which is the data decrease in the nodes and the cost reduction, is related to the data node reduction.

The future research, dealing with the duplicity suppression, will be related to improvement of the designed method, not only the in cloud service Amazon, but in creating a general module in programming language golang, as well, which will help with implementation of parallelism. Authors also want to focus on more effective reference creation in the primary step in the control of duplicities entering the system. The research presented in this paper also points to the selected topics in the area of signal theory similar to research presented in [14].

Acknowledgement

This work was supported by Grant System of University of Zilina No. 1/2020 (8056).

References

[1] KAMAL, J., MURSHED, M., BUYYA, R. Workload-aware incremental repartitioning of shared-nothing distributed databases for scalable OLTP applications. *Future Generation Computer Systems* [online]. 2016, **56**(C), p. 421-435. ISSN 0167-739X. Available from: https://doi.org/10.1016/j.future.2015.09.024

[2] PATIL, J., BARVE, S. S. DDFP: Duplicate detection and fragment placement in deduplication system for security and storage space. In: 1st International Conference on Intelligent Systems and Information Management ICISIM 2017: proceedings [online]. 2017. Available from: https://doi.org/10.1109/ICISIM.2017.8122177

[3] DOLEV, S., FLORISSI, P., GUDES, E., SHARMA, S., SINGER, I. A survey on geographically distributed big-data processing using MapReduce. *IEEE Transactions on Big Data* [online]. 2017, **5**(1), p. 60-80. eISSN 2332-7790. Available from: https://doi.org/10.1109/TBDATA.2017.2723473

[4] GAI, K., QIU, M., ZHAO, H. Security-aware efficient mass distributed storage approach for cloud systems in big data. In 2nd IEEE International Conference on Big Data Security on Cloud IEEE BigDataSecurity 2016, 2nd IEEE International Conference on High Performance and Smart Computing IEEE HPSC 2016 and IEEE International Conference on Intelligent Data and Security IDS: proceedings [online]. Vol. 1. 2016. Available from: https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.68

[5] CAO, N., WANG, C., LI, M., REN, K., LOU, W. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems* [online]. 2014, **25**(1), p. 222-233. ISSN 1045-9219, eISSN 1558-2183. Available from: https://doi.org/10.1109/TPDS.2013.45

[6] KOTENKO, I., SAENKO, I., BRANITSKIY, A. Framework for mobile internet of things security monitoring based on big data processing and machine learning. *IEEE Access* [online]. 2018, **6**, p. 72714-72723. ISSN 2169-3536. Available from: https://doi.org/10.1109/ACCESS.2018.2881998

[7] DIAO, Z., WANG, Q., SU, N., ZHANG, Y. Study on data security policy based on cloud storage. In: 3rd IEEE International Conference on Big Data Security on Cloud BigDataSecurity 2017, 3rd IEEE International Conference on High Performance and Smart Computing HPSC 2017 and 2nd IEEE International Conference on Intelligent Data and Security: proceedings [online]. 2017. ISBN 978-1-5090-6296-6, p. 145-149. Available from: https://doi.org/10.1109/BigDataSecurity.2017.12

[8] RAO, L., LIU, X., XIE, L., LIU, W. Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment. In: IEEE INFOCOM 2010: proceedings [online]. 2010. Available from: https://doi.org/10.1109/INFCOM.2010.5461933

[9] JUNGHARE, S. A., MAHALLE, V. S. Overview of secure distributed de-duplication system with improved reliability. In: International Conference on Inventive Systems and Control ICISC 2017: proceedings [online]. 2017. Available from: https://doi.org/10.1109/ICISC.2017.8068613

[10] SHAHRI, H. H., BARFORUSH, A. A. Z. Data mining for removing fuzzy duplicates using fuzzy inference. In: Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS 2004: proceedings [online]. 2004. ISBN 0-7803-8376-1. Available from: https://doi.org/10.1109/NAFIPS.2004.1336229

[11] LI, J., CHEN, X., HUANG, X., TANG, S., XIANG, Y., HASSAN, M. M., ALELAIWI, A. Secure distributed deduplication systems with improved reliability. *IEEE Transactions on Computers* [online]. 2015, **64**(12), p. 3569-3579. ISSN 0018-9340, eISSN 1557-9956. Available from: https://doi.org/10.1109/TC.2015.2401017

[12] KVET, M. KVET, M. Temporal database management: temporal registration. In: International Conference on Information and Digital Technologies IDT 2017: proceedings [online]. 2017. ISBN 978-1-5090-5688-0, p. 227-233. Available from: https://doi.org/10.1109/DT.2017.8024301

[13] KVET, M. Data distribution in ad-hoc transport network. In: International Conference on Information and Digital Technologies IDT 2019: proceedings [online]. 2019. ISBN 978-1-7281-1401-9, p. 275-282. Available from: https://doi.org/10.1109/DT.2019.8813437

[14] DOBRUCKY, B., HARGAS, L., MARCOKOVA, M., KONARIK, R., KONIAR, D. A LabVIEW implementation of digital waveform quadrature oscilator by virtual instrumentation. In: International Siberian Conference on Control and Communications SIBCON 2017: proceedings [online]. 2017. eISSN 2380-6516. Available from: https://doi.org/10.1109/SIBCON.2017.7998547