Webology, Volume 16, Number 2, December, 2019

Development of Intellectual System for Data De-Duplication and Distribution in Cloud Storage

Vasyl Lytvyn

Professor and Head of Information Systems and Network Department, Lviv Polytechnic National University, Bandery St., 12, Lviv, Ukraine. ORCID: 0000-0002-9676-0180. E-mail: vasyl.v.lytvyn@lpnu.ua

Victoria Vysotska

Associate Professor and Deputy Head of Information Systems and Network Department, Lviv Polytechnic National University, Bandery Street, 12, Lviv, Ukraine. ORCID: 0000-0001-6417-3689. E-mail: victoria.a.vysotska@lpnu.ua

Mykhailo Osypov

Master of Information Systems and Network Department, Lviv Polytechnic National University, Bandery Street, 12, Lviv, Ukraine. ORCID: 0000-0002-6611-724X. E-mail: mykhailo.osypov@gmail.com

Olha Slyusarchuk

Associate Professor of Mathematics Department, Lviv Polytechnic National University, Bandery Street, 12, Lviv, Ukraine. ORCID: 0000-0003-3464-0252. E-mail: Olha.Z.Sliusarchuk@lpnu.ua

Yuriy Slyusarchuk

Associate Professor of Information Systems and Technologies Department, Deputy Head of Institute of Business and Innovative Technologies, Lviv Polytechnic National University, Bandery Street, 12, Lviv, Ukraine. ORCID: 0000-0003-2040-7898. E-mail: Yuriy.M.Slyusarchuk@lpnu.ua

Received May 24, 2019; Accepted December 22, 2019

Abstract

The system for backing up the data is designed. Client software works on the computer of user, takes all the necessary files for backup, and turns them into Stream of bytes. Then breaks it into blocks (from 32 KB to 64KB) using a Rabin algorithm. It is based on hash ring that is absorbing every incoming byte if the current hash mask or equal to a certain reach 64KB, there is division committed. This approach helps to avoid is coping all data if the content of a file has changed in separately. For each data block client software calculates the

hash. Then sends parts even 256 times or more of those hashes to the server and checks if they already know the system. The blocks are not known, refers to the server. As part of the de-duplication, data on distribution and cloud distribution occurs on hashes, hashes are SHA-1, so that 20 bytes are given in hexadecimal format. First number or letter and will serve key distribution. So essentially, you can evenly distribute the data among the workers from 2 to 16 pieces. You need to take the second number of hash, and more commit distribution.

Keywords

Stochastic game; Clustering, Ontology; Knowledge base; Intelligent agent; Data Sharing; Data de-duplication; Data Hashing; Cloud Environment; Cloud computing; Rabin algorithm; Hybrid De-duplication

Introduction

For the stable operation of companies that work with information is the process of data backup. Regardless of whether a bank or a web-site selling household goods, if unsystematic failure and damage to the main servers and databases is as soon as possible, a system error should be corrected and restored backup data. However, periodic backup is a problem with duplicate data, thus creating a problem for the finance departments of the major costs of cloud storage service providers or their data centers. To save disk space or funds to cloud storage services, many systems use de-duplication backup data (Crump, 2008). This approach can reduce the final size of backups by duplicating data cutoff or those that are already known to the system (by maintaining previously). However, these systems use the cloud only as endpoint storage (Antonyuk et al., 2020; Babichev et al., 2017; Basyuk et al., 2019; Rusyn et al., 2018; Rusyn et al., 2019). Picking of the right approach to the tasks and data distribution for de-duplication may use the full potential of cloud-based distributed systems to speed backups and its bandwidth. Given that urgent task is to research and further development of intelligent system de-duplication and distribution of data in cloud storage.

The aim is to create an intelligent system of de-duplication and distribution of data in cloud storage. To perform the task need to solve the following problems (Lytvyn et al., 2018):

- Modern technology to analyze the design and creation of systems in backup and recovery, comparative analysis of technical implementations and algorithms applied in implementation (Babichev et al., 2018; Vysotska et al., 2019);
- Build a conceptual model of intelligent information systems, de-duplication and distribution of data in the cloud storage for automation and effective ‘software in the backup and recovery (Kravets, 2007; Babichev et al., 2017; Basyuk, 2019);
- An analysis of methods and tools for implementing intelligent systems and to opt for the implementation of its own (Andrunyk et al., 2020; Vysotska et al., 2018);

- Design intelligent de-duplication and distribution of data in cloud storage.

The object of the research is the process of de-duplication and distribution of data in the cloud storage during data backup. The subject of the research methods and tools are design and implementation of de-duplication and distribution of data in cloud storage. Practical value of work is developed intellectual system of de-duplication and distribution of data in cloud storage. The result of the research is developed intelligent de-duplication and distribution of data in cloud storage that will be used as backup software to eliminate duplicate data and increase bandwidth input data streams by distribution problems in cloud computing cluster. The system can be used for integration with any other system, or as an independent product. The system is relevant, since it has broad functionality and can be used in practice.

Data de-duplication

De-duplication is the process of eliminating duplicate copies of duplicate data. Generally, there are two types of de-duplication, *Single instance storage* and *Block-level de-duplication* (Crump, 2008). De-duplication at copies (or most files) is quite simple: if the system there are two or more identical files, or physically to the disks in any store only one copy is kept, while others received a link to it. This approach is often used in postal services, such as Microsoft Exchange for attachments in email messages. If one has submitted a letter with embedded images or documents to dozens of others, only one copy of attachments will be saved on the server. Also, the approach used in the backup data, such as employees working computers in the company, where dozens of backup copies of Windows will have a large number of identical system utilities and dll files (Davydov et al., 2017; Lytvyn et al., 2018).

However, de-duplication at copies are not effective in cases where the original files are modified in any way, such as added a new line of text in a document or change a single pixel in a large volume image. Change a couple of bytes of data can lead to repeated copying of the original data, thereby significantly increasing the space occupied by backups. In this case, you must carry out a second type of de-duplication, such as de-duplication block level.

Block level de-duplication to ignore such things as file and copy or receives only the input data stream as plain bytes of information. In order to carry out de-duplication, this approach data stream divided into blocks of a certain size and for each hash, value calculated using hash functions (md5, sha). With hash values, the system determines whether a particular block of data already exists in the system by an index (or hash table). If the block is already known, then the backup copy is retained only link to it. Otherwise, the original data is stored, and its hash value is recorded in the index to further de-duplication. Based on where the data stream splitting blocks of computing hashes, construction index of unique and storage units are the following types of de-duplication: the original, target, and hybrid production.

Source de-duplication is de-duplication is performed on the device where the output data (Wendt, 2008; Kravets et al., 2009). Any data that is marked for backup will be divided into units, which counted hashes. Based on the hashes will be built to further de-duplication index data. However, this approach has two potential drawbacks. The first potential drawback is that the process of de-duplication device resources used, in which there are data. Thus, users should make sure they have enough free memory and processing power of the processor. Therefore, backup, such as a mail server or web site may lead at least to slow their work.

The second drawback relates to the location of the hash table index is de-duplication. In some cases, the hash table is limited to each individual computer that performs de-duplication. In this case, the index will be limited only local data. That is, the benefits of de-duplication will be limited only within the same computer. If other servers are identical pieces of data, the current server does not know that (Kravets, 2006; Lytvyn et al., 2018; Mirkin, 2005; Nazarkevych et al., 2018).

Target de-duplication is de-duplication is performed on the target central server, where the final stage of data will be stored (Wendt, 2008). Once the data arrive in the repository, the server will be able to build an index based on them and make the existing intersection, the central hash table to store only unique data. The advantage of this approach is the large amount of data. In addition, since the more data, the higher the index, which as a result makes it more likely to find identical blocks, is increasing the rate of de-duplication. In addition, significant advantage is the transfer of responsibility for the processing power of the client (device output data) to the target server that is only for de-duplication, making it practically impossible for a potential fight computing resources. However, this option does not solve all the problems, there are some points to consider and take into account. The target de-duplication requires that data be saved to disk before the process of breaking down and hashing not to overload the customer. So weak point of this approach is the speed of read / writes from disk. Another potential drawback is the possibility of collision hash functions, so that the possibility that the two sets of data are not identical will get the same hash value. Hash collisions can result in loss of data consistency that ultimately damages the original data without the possibility of recovery. To avoid conflicts must use the algorithms in which the chance of keeping smaller, but it leads to a greater burden on computing resources. However, the use of more powerful and sophisticated algorithms is not a problem, because de-duplication servers use specialized hardware equipment required by the task.

A third disadvantage is that the full amount of backup data to be transmitted over the network from clients to a central repository. This problem is particularly acute if the client is weak or highly available toll network coverage. With minor changes since the last backup, sending all

data is completely redundant (Kravets, 2003; Lytvyn et al., 2019). Also, the problem may be network congestion target server that will affect the ultimate bandwidth.

Inline de-duplication uses memory as a buffer zone for the process of partitioning data and calculation of hashes (Wesley, 2008). So data de-duplication before writing to disk, it eliminates the cost of speed write / read from disk. Streaming de-duplication should be viewed as an upgrade target de-duplication. The first has all the advantages of the latter, but no lack of work associated with a slow drive (Rusyn et al., 2020; Vysotska et al., 2015).

Hybrid de-duplication is an approach with a combination of source and destination (stream) de-duplication (Wesley, 2008; Neyman et al., 2012; Russell et al., 2009). Partition data into blocks and calculation of hashes occurs on the client (device from baseline). The resulting hashes are sent to the server for identification, after which the client sends the server only those data blocks that were previously not known to him. This approach eliminates the drawback of the target de-duplication and streaming online for using the network. Because the client will be sent to only those data blocks that were previously unknown, and if already known only sent their hashes that 2-3 thousand. Times are smaller than the original data (Kushner et al., 2013; Lytvyn et al., 2018; Rami et al., 2002; Solos et al., 2016).

Although the client in the case of hybrid de-duplication is still involved in the breakdown of data into blocks and calculating their hashes, as is the case with the initial de-duplication. However, the work index is completely in the target repository, significantly offloads the processing power of the client. If the moment is still a disadvantage for a particular case, the breakdown of the data into blocks and calculating hashes their possible transfer to another transit unit located within the network or the client completely delegate the job to the server.

De-duplication block level provides two types of partitioning data flow into blocks, this breakdown of fixed and variable length based algorithm (Wesley, 2008; Rzheuskyi et al., 2020). Breakdown of fixed length rather trivial and relatively fast algorithm complexity, but the disadvantage is offset data stream at the beginning because the blocks that will follow after the change will be considered as new. However, in case of breakdown blocks of variable length, actually breaking point defines the algorithm. This algorithm should work with endless streams of data using a hash function ring. Algorithm absorbs every byte input data stream, and once the ring hash function value corresponds to a given pattern before, it serves as a point of splitting the stream into blocks. Thus, the data changed or shifted a couple of bytes will be considered new only one data block that covers data. However, in order to track changes and correctly expose breaking point should be checked input to a digital template set - hash value. A common practice is to calculate a hash value every time the incoming byte in the data stream. The point of the partition will be the moment when the received hash values

match a given pattern. To make such a calculation effectively was coined ring hash algorithm (Wesley, 2008). One of the most common hash algorithms ring is a reference Rabin (Wesley, 2008). Fingerprint Rabin - a method of creation of digital prints (hashes) using polynomials over the field with a finite set of elements was invented M. Rabin (Rabin, 1981). The formal definition: having n-bit message m_0, ..., m_{n-1}, can be considered as a polynomial power of the n-1 over the field with a finite set of elements (Gaul field).

$$f(x) = m_0 + m_1 x + ... + m_{n-1} x^{n-1}, \qquad (1)$$

Choosing random irreducible polynomial $p(x)$ of degree k, we can determine the reference message m as the remainder of the division $f(x)$ to $p(x)$, which can be represented as a polynomial power of the k-1 or k-bit number. Rabin fingerprint algorithm used in the Rabin-Karp is string-searching algorithm. The idea is to avoid comparing all rows with all, thus eliminating the quadratic complexity of the algorithm. In the best-case complexity $O(n+m)$, at worst is $O(nm)$. String searching algorithm to search a similar point of breaking data stream into blocks as breaking point should be based on the content data and not on the basis of the length of the block, it is necessary to search for relevant information in an early time. The idea of the algorithm is as follows: for simplicity, assume that the alphabet consists of decimal digits $\Sigma=\{0,1,...,9\}$. After this string of k symbols can be regarded as the number of length k. That character string "12345" corresponds to the number 12345. For a given sample $P[1..m]$ p denote the corresponding decimal value. Similarly, Given a text $T[1..n]$ t_s denote the decimal value substring $T[s+1..s+m]$ length m at s=0,1,...,nm. Obviously, $t_s=p$ if and only if $T[s+1..s+m]$=P[1..m]; thus, s is allowable offset if and only if $t_s=p$. If the value of p can be calculated by $O(m)$ and the value t_s for total algorithmic time $O(n-m+1)$, all valid shifts could be found algorithmic time $O(m)+O(n-m+1)=O(n)$ by comparing p with every possible t_s. Using Horner's method p value can be calculated by time $\Theta(m)$ (2).

$$p = P[m] + 10(P[m-1] + 10(P[m-2] + ... + 10(P[2] + 10P[1]))...)), \qquad (2)$$

The value t_0 can be calculated from the array $T[1..n]$ the same way during $O(m)$. However, knowing the value of t_s, t_s+1 value can be calculated for a fixed time (1.3).

$$t_{s+1} = 10(t_s - 10^{m-1} T[s+1]) + T[s+m+1], \quad (3)$$

For example, if m=5 and t_s=31415, you need to remove a number of senior category $T[s+1]$=3, and add a number of junior level (albeit, $T[s+5+1]$=2). The result is t_s+1=10(31415-10000*3) +2=14,152. Thus, all t_s can be calculated by an algorithmic time $O(m)$. However, this algorithm has a disadvantage due to the fact that when p and t_s are too large and with them will not be easy to work with. However, the process of splitting the data stream into blocks of search strings makes no sense, because the input data stream is large enough, but the chance

to find *k*-bit line is small. However, this algorithm, namely part is ring hash can be used in another. Namely ring calculate the hash for each incoming data byte and consider splitting the time point when hash ring is equal to a given pattern. However, in order to obtain the desired length of the block data obtained hash of performing a bit-wise AND operation (AND) with the desired average length (4). If the resulting value is established pattern, the current position is the starting byte partition.

$$P = H \,\&\, A. \qquad (4)$$

The distribution of tasks and data

Distributed computing is a time-consuming way to solve computational problems by using two or more computers, networked (Tanenbaum et al., 2017). Distribution calculation can be considered a special case of parallel computing, where the division of tasks among multiple processors happens computer. A system's ability to increase the amount of work due to add new computing resources called scalability. Scalable system resources can be of two types, horizontal and vertical (Lytvyn et al., 2019; Neogy et al., 2018; Petrosjan et al., 2007):

- Horizontal scalability is to increase the volume of processing tasks by addition of additional nodes in the cluster computing, so that addition of additional computers on the network;
- Vertical scalability is to increase the amount of processing problems by increasing the resources of one node (computer), such as increasing the number of central processing units (CPU), memory and disk space.

However, the increase computing resources does not solve the problem of design. For maximum effect it is necessary to program, for which the scalability were able to allocate these resources correctly (Robinson, 1965). Typically, much cheaper to add a new node to the system to achieve improvements than invest time and financial resources in view of system architecture to achieve efficient operation in the multithreaded and distributed environment. However, this approach has reduced returns. For example, even, if only 70 percent of the program can speed through scalability. According to Amdahl's law (Amdahl, 1967), which determines the potential acceleration of the program by increasing the number of computing resources, part of a program that defies distribution, limit the overall effect of parallelization. Any problem usually consists of a plurality subtasks that can be done consistently and in parallel. This relationship is given by (5), where S is acceleration program (as a ratio to

$$S = \frac{1}{p + \frac{1-p}{n}}. \qquad (5)$$

initial time); *p* is part of tasks that can be performed sequentially; 1-*p* is part of tasks that can be performed in parallel; *n* is the number of processors. The new processors addition effect lost according to this equation, at some point (depending on the percentage of tasks that are performed sequentially). Figure 1 shows a graph of the acceleration program the number of processors, which clearly demonstrates the reduction.

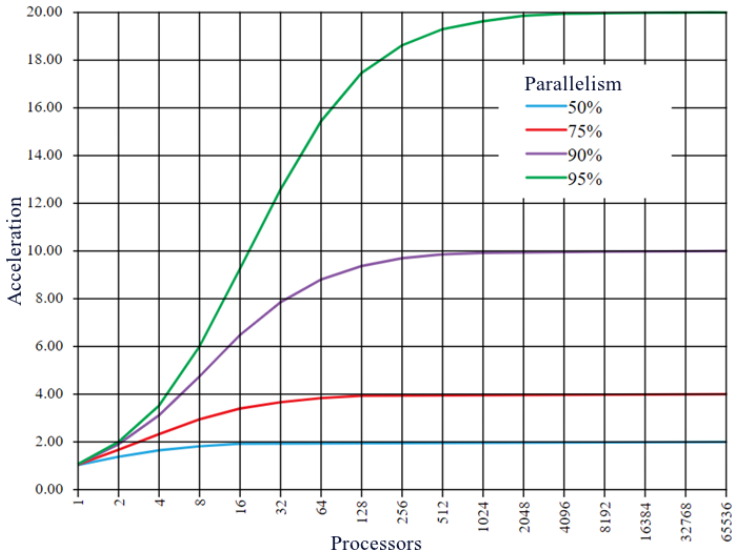

Figure 1. Graph of acceleration of the program the number of processors

So if to make parallel 90 percent of software problems (so that, 10% is a consistent problem), then speed up the program more than 10 times is not possible, regardless of the number of processors added. Therefore it is necessary to approach responsibly in the development of distributed systems architecture (Lytvyn et al., 2015).

Analysis of known systems

StorReduce is a specialized solution for de-duplication of data in the cloud (Wilson et al., 2019). StorReduce provides similar functionality to cloud storage providers, including storage facilities, user accounts, access keys, access control and policy-based management Web Toolbar StorReduce. StorReduce Server is running on a physical or virtual machine. The architecture of the solution allows it to run on cloud storage servers, thereby reducing network latency between the server and storage. StorReduce interface supports S3 storage facilities. This feature simplifies integration solutions for systems that are already using S3 storage for their data. However, according to the description of the decision, StorReduce server is running on a single server, thus scaling only vertically. So for more input data sets for backup, should be put in or buy cloud storage servers with more powerful processors and disk media, such as HDD. However, according to the data described in Section 1.3, the vertical scale has its limits.

OpenDedup is forum de-duplication file system open source (Silverberg, 2019). It has support for cloud storage, and it allows you to store data in storage with support for S3 interface. Still something to add! But one of the disadvantages of this decision is binding to the operating system Linux (making it impossible to use this system to Windows and MacOS), and was tested on distributions Ubuntu and CentOS, which indicates that users may have to adjust incompatibility may arise other distributions. Also, there is no user interface that allows the use of only highly-specialized users and integration with other solutions.

A systematic analysis of the research object and subject area

Designing intelligent system de-duplication and distribution of data in the cloud storage is a complex process that requires a detailed description and puts the researchers set issues. Since the object of study is the process of de-duplication and distribution of data in cloud storage. It is important at this stage to conduct a systematic analysis of the chosen subject area, in order to understand the full picture of the system processes the data flow in the system. Identify the main opportunities and the main business processes to the stage of implementation to avoid many mistakes in the architectural plan and model selection of problem solution of data de-duplication and distribution in cloud storage. An important goal is to analyze the system and understanding of how to achieve it, using available resources is hardware, specific methods and approaches Architecture for building information system. System analysis will divide the task set before us on the set of simple tasks, given the relationship simpler tasks among themselves (Chyrun et al., 2018; Fedushko, 2014; Lytvyn et al., 2018). So complex system is divided into components and with well-formed business processes between parts of the product (Rusyn et al., 2016; Tkachenko et al., 2018; Rusyn et al., 2020).

Specifying principles of system approach to the analysis of the intellectual system de-duplication and distribution of data in the cloud storage provides better insight and vision system overall understanding of its functioning, the necessary steps to improve and certain features of the system (Chyrun et al., 2018; Lytvyn et al., 2016).

The purpose of this study is to develop software that will be able to carry out de-duplication and distribution of data in cloud storage. First of all, the projected system should be viewed as a set of simple elements - its components, which are separate modules between which there is constant communication, so that elements are interconnected. We must also understand and that are within the system there are certain processes and outside the system influencing factors which also contribute their part in the operation of the system is with these factors also converts data streams. Therefore, we can define the basic principles of the system approach in constructing intelligent system de-duplication and distribution of data in the cloud storage (Davydov et al., 2017; Naum et al., 2017; Xue et al., 2015; Zhezhnych et al., 2018):

- the ultimate goal is the principle of de-duplication and distribution of data in the cloud storage;
- the principle of unity is that the system as a set of certain components, but on the other hand - the projected software is an integrated system as functional processes in which it has logically follow from elements that form the system, but no one element alone does not possess these functions;
- confirmed the principle connectivity system functioning in a certain environment - it is obvious that intelligent de-duplication and distribution of data in the cloud storage will be useful to use by many companies in the problem of data backup, respectively, should provide a flexible mechanism for setting different approaches;
- modular construction principle determined by the modular structure of the project is a complete operating system de-duplication and distribution of data in the cloud storage is divided into a number of modules, between which there is communication (modules can be connected or disconnected according to customer needs);
- the principle function determines that the design structure of the system is possible only after the determination and complete understanding of system functions according to exactly the start we formulate and examine the details of the system de-duplication and distribution of data in the cloud storage;
- the principle of development means that the system must be designed so as to later be able to grow through expansion, the connection of additional modules, replacement of implementing interfaces, storage of data in databases and scalable;
- the principle of decentralization is a right balance between centralization and decentralization, defined the aim and purpose of the system;
- uncertainty is probability of certain unpredictable factors that could affect the system, something that may happen or may not happen.

The main objective is to build systems analysis model (system), which will solve the problem. The final system must consist of interrelated elements that collectively lead to solving a particular problem. The base consists of system analysis system, the purpose element, function, environment and others. The term refers to a set of individual system elements between which there is some kind of interaction that leads to the achievement of a common goal and the plurality of elements form an integrated system that interacts with the environment as a whole (Gozhyj et al., 2018; Kanishcheva et al., 2017; Khomytska et al., 2016).

External entity intelligent system de-duplication and distribution of data in cloud storage functions as a data backup program administrator is conducting the data backup company, or the user conducting the backup personal data. Elements of the system being analyzed are:

- user interface, input point for users of the system allows to select files for backup, view existing copies, manage their life cycle, distribution and schedule tasks for periodic data backups and restore data from backups;
- client software, which receives data from the user interface, makes the distribution of data flow into blocks and calculating their hash values, testing hash values of the blocks with an index system and sending new blocks of data to cloud storage;
- index is the hash table that contains information about all known unique blocks of data systems responsible for eliminating duplicate at the time of saving them to the cloud storage (Lytvyn et al., 2018);
- application server is performs de-duplication algorithm operating units client application data and index, makes de-duplication division of tasks between workers (individual nodes in a clustered environment) and communication with the cloud data storage environments.

Our goal is to de-duplication and distribution of data in the cloud storage so, as to eventually eliminate backup data using repeating data capacity of distributed computing and cloud storage. Important to understand and define the environment of the system, which means the choice of a plurality of objects, which would have impact development system and which have changed under this influence, as well as those whose change would affect the projected system (Korobchinsky et al., 2017; Khomytska et al., 2017). Interaction of information system environment is due to the available input and output systems. The impact of the environment on the projected system is actually the entrance, and the result of this action is output. Generally accepted to consider only the most important communications systems with the environment, in this case it will be the end use (Vysotska et al., 2018).

It is known that the methodology of systems analysis systems is divided into two types: simple and complex. The systems consist of a small number of components and, accordingly, the ties between them have branched structure called simple systems. Difficult as are those built from a much larger number of elements respectively have more internal ties, perform complex functions and are heterogeneous (Korzh et al., 2014; Maksymiv et al., 2017; Poirriez et al., 2009; Ramirez-Ortegon et al., 2013). Obviously, the designed software product is a representative of complex systems. The designed intelligent de-duplication and distribution of data in the cloud storage consists of a large number of elements have strong communication channels between them; the effectiveness of each element separately will be much lower than the efficiency of their combination as a whole system (Vysotska et al., 2018).

Use Case Diagram is UML diagram, which depicts the relationship between actors (external nature) and the case (use cases) in the system (Rumbaugh et. al, 1999). The diagram is a graph consisting of a plurality of actors limited precedents border system (rectangle),

associations between actors and precedent relationships among precedents, and generalization relationships between actors. Figure 2 shows a diagram of ways to use the system. The diagram depicts the following actors (external reality):

- *User* is user system sends data to backup and restore the original data using the received backup (Lytvyn et al., 2018; Shu et al., 2019).
- *Index* is contains information about all known hash value system is used to eliminate redundant data blocks (Montes-y-Gómez et al., 2000; Vysotska et al., 2018).
- *Cloud storage* is serves as a repository for stored unique blocks of data and backups.

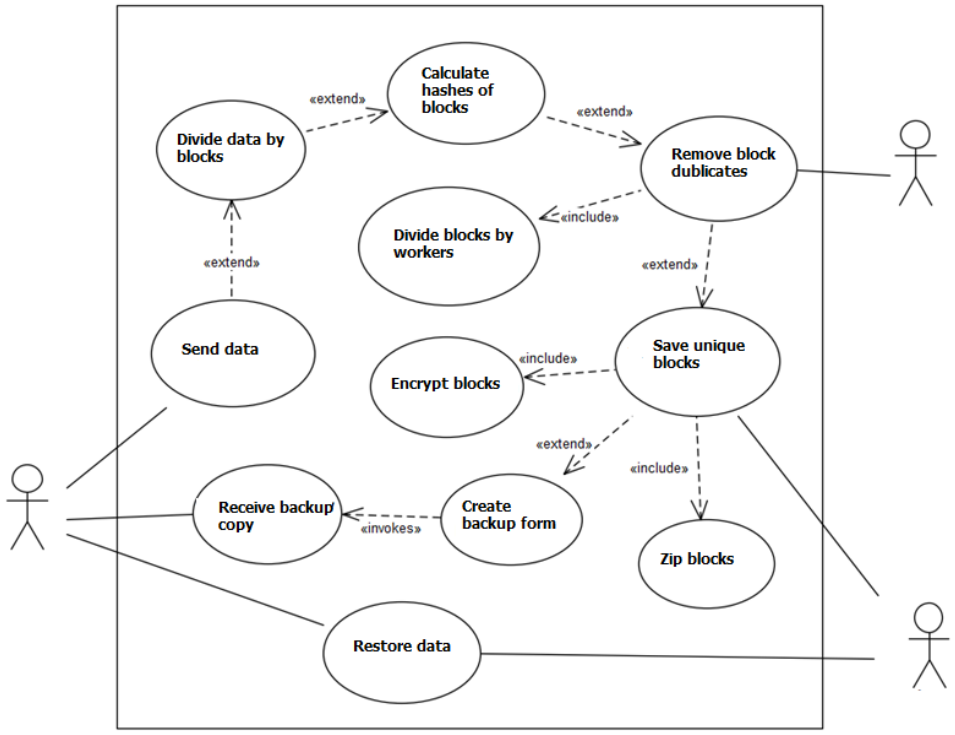

Figure 2. Use Case System

The diagram use case (Figure 2) is shown also following precedents (use cases):

- *Share data* is a user creates a list of files and sends them to the backup system.
- *Split data on the block* is system forms a stream of incoming data and breaks it into small blocks of data de-duplication comfortable.
- *Calculate the hashes blocks* are system for each incoming unit calculates a hash value using a hash function set.
- *Eliminate duplicate blocks* are using the index; the system checks existing units with the received hash values.
- *Distribute power between workers* are to increase the capacity and speed, the system distributes the work on the block between workers in the system (such as nodes in the cluster).

- *Save unique blocks* are after eliminating duplicate blocks system using cloud storage saves unique blocks in storage.
- *Compress blocks* are to save storage space, input power can be compressed one-compression algorithms.
- *Encrypt blocks* are for additional data protection, input units can also be encrypted.
- *Generate backup* is after processing of input units, output data stream is stored as a series of links to the hash value.
- *Get back up* is after the de-duplication process, the user receives a link to a backup for future recovery
- *Recover data* is using links to back up; the user can restore the original data.

State Diagram is a UML diagram object determines the change of state at the time (Martin et al., 2009). Activity Diagram is describing how processes perform certain work performed or system or external entities coordinate with each other. Performing certain operations may be dependent on certain preconditions to be met (Martinez-Gil et al., 2008). Go to another action occurs after the completion of the previous one. Usually the effect on the input received some information is a set of signals to be processed and converted to a plurality of output signals.

Figure 3 shows a diagram of states backing up user data de-duplication in system of intellectual and distribution of data in cloud storage. The object is a state diagram backup process, and continues to the point we understand this process is called object (Wooldridge, 2009).

It is state of user authentication for the initial backup process. If authentication data is not correct then the object remains in the same state and waits for new commands. If authentication data, such as login and password is correct then the object goes into the selection of files for backup. Next state is the creation of the data stream. From selected for de-duplication system creates a single data stream. After successfully creating data flow object goes into splitting the data stream into blocks. Block data is a unit of de-duplication process; duplicate it blocks the system tries to avoid. If the data stream is data to split into blocks, the object goes into the hash value calculation block. Each data block must have a digital signature, which is much smaller volume, and this is the hash value. Once received hash value, the object goes into a verification hash value to existing. To discard the duplicates should check whether the block exists in the system, but check the whole unit is quite expensive, so check on them existing hash value. If the hash value is already known to the system, the object goes back to the state of decomposition of data flow into blocks. If the hash value is unknown, the object goes into preserving data block. The new data block, previously known system, preserved in a special place in the cloud storage based on its hashes (to facilitate further search). After saving, the object enters back into the state of breaking data

stream into blocks. When an object is in this state and can no longer get new blocks of data flow facility goes into building a backup. After the de-duplication process, a list of data blocks presented as a list of its hash value, which is the backup. After receiving it, the object goes into a backup destination. This condition is the last front of the object to the final state.

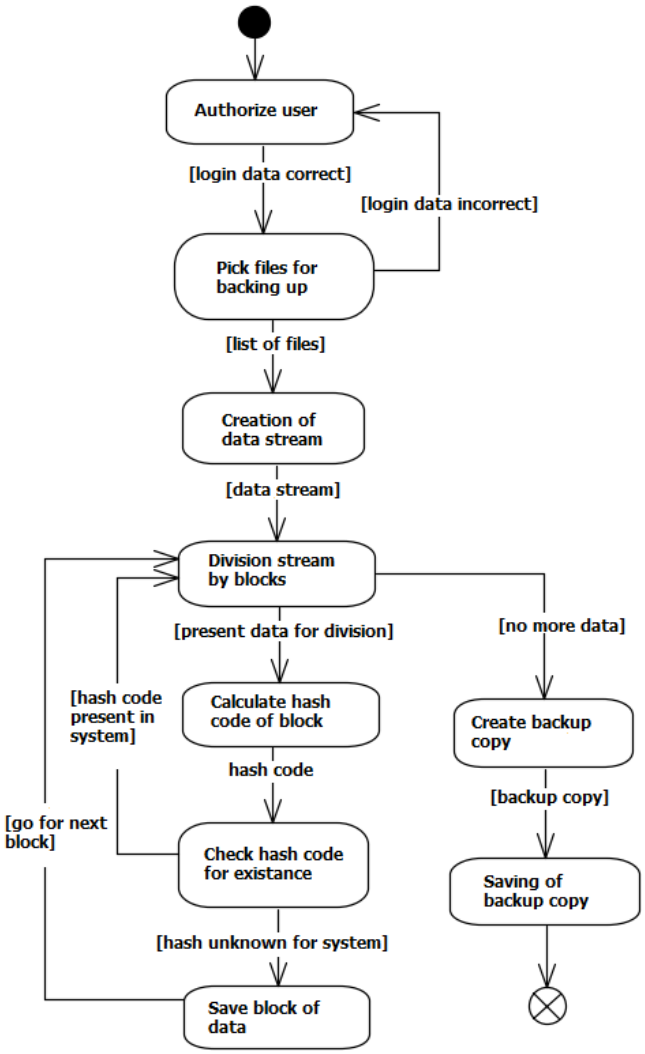

Figure 3. Diagram backup

Sequence diagrams are UML diagram that shows the interaction of objects arranged in time (Precup et al., 2013). In particular, the following charts show the sequence involved objects and sent messages. Sequence Diagrams shown as vertical lines of different processes or objects that exist simultaneously. It is sent represented as horizontal lines, in order of departure. Figure 4 shows a sequence of backup data using intelligent de-duplication and distribution of data in cloud storage. The diagram shows a line of life and User actor four objects: The client-server program, Index and cloud storage.

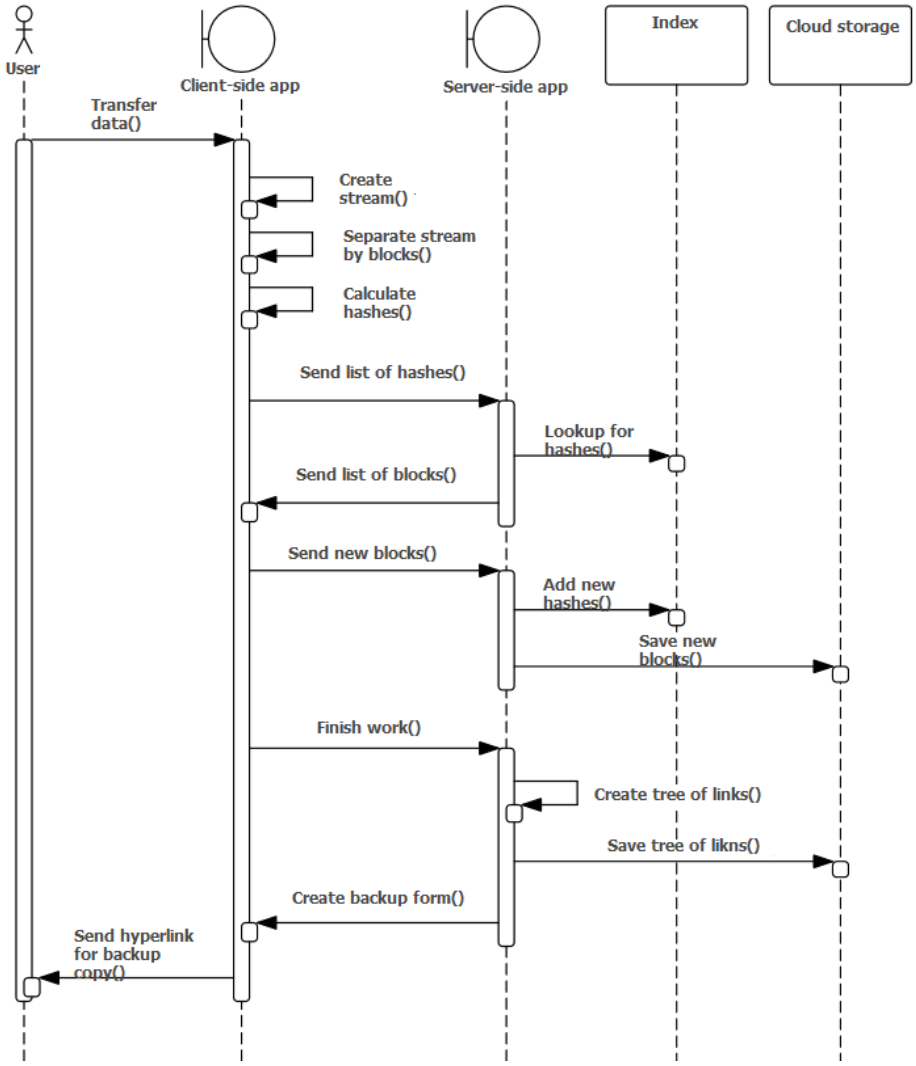

Figure 4. Sequence Diagrams

- *User* is the only actor in the system selects the data for backup and de-duplication gets results.
- *Client software* is the software on the user side, which is responsible for grouping user data bytes and forming flow from them. Manages the process of de-duplication on the part of the user.
- *The program server* is the software on the server side. The central coordinator of de-duplication process, performs communication with the client software for a list of hashes and blocks; generates a list of needed new units based on the index; takes account of new blocks of cloud storage; wood forms a link to restore the original data.
- *Index* is hash table contains all known system hashes and links to the appropriate block data in the cloud or other storage.

- *Cloud storage* is the storage location of data blocks and backup (trees references).

In general, workflow according to sequence diagrams, as follows. A user using a user interface in the form of a website or desktop application, choose the files / data de-duplication and to send them to the client software. This, in turn, divides them into blocks using a preset algorithm. For each block *Client* software computes a hash value and sends a list to the Program server. Last check hashes from the resulting list with data from the index and cut off those hashes are already known. As a result back to the client program are sent only hashes that were not previously known to the system, which must therefore add to the system. Based on this list, the client program sends selectively blocks the Program server. Each unit get stored in cloud storage with a unique identifier (referring to the block). The program also adds a server hash table index new key-value pair: hash block and link to it. This cycle, checking existing hash values and send new units performed as long as there is data in the stream. When the flow is completed, Client software signals the completion of de-duplication. The program is server-based processing of data blocks and their hashes building back up in a tree references to hashes. The current copy is returned to the user. Client software signals the completion of de-duplication. The program is server-based processing of data blocks and their hashes building back up in a tree references to hashes. The current copy is returned to the user. Client software signals the completion of de-duplication. The program is server-based processing of data blocks and their hashes building back up in a tree references to hashes. The current copy is returned to the user.

In addition to behavioral UML diagrams, should depict a system using structural diagrams (Vysotska et al., 2018). The first structural UML diagram is this chart component. It depicting are components and connections between them. The components are interconnected by means of structural links as Assembly connector. These connections are established between the interfaces of two components, whereby the components typically have at least one interface. In this connection the two components, one of which provides a different need for client-server template. Also, components can have a hierarchy that is one component may contain another. Figure 5 shows a diagram of the components of the intellectual system de-duplication and distribution of data in cloud storage that contains two main components are *Client* and *Server*. Client component contains software components that are on the user side:

- *UserApp* is user program component that allows the user to authenticate the system, select the files to back up, start the process of backup and restore data from a backup. Grouped data to transmit to the backup interface components Chunker FilesApi, the rest of the operations associated with the interaction with the server - because of UserInterfaceApi of antibacterial HttpClient.

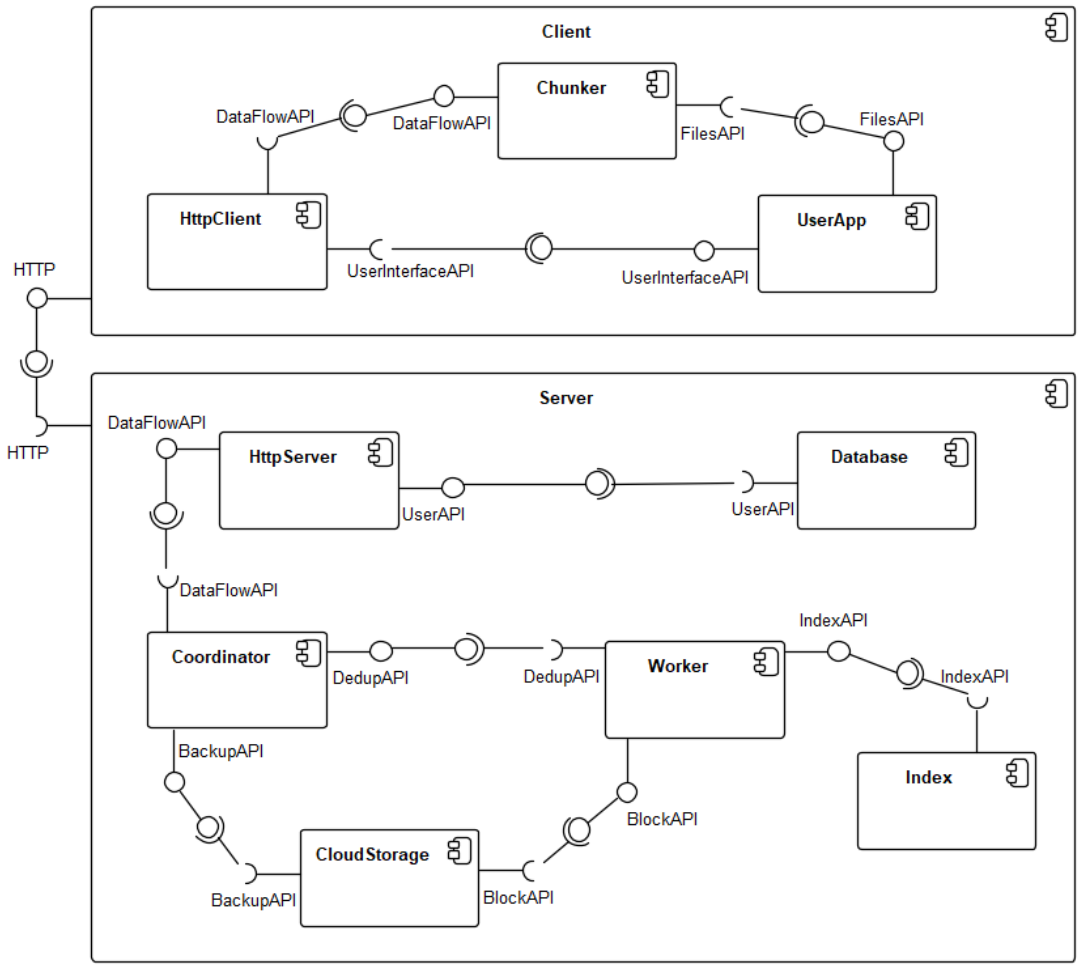

Figure 5. Chart Component

- *Chunker* is component forming and splitting the data stream, which is responsible for converting input custom files (obtained from UserApp via FilesApi) into a single byte stream data, partitioning it into blocks of data and send these units to HttpClient interface DataFlowApi.
- *HttpClient* is component HTTP-client receives data and commands from the user interface components UserApp (through UserInterfaceApi), as well as receiving block de-duplication of components Chunker (through DataFlowApi). All received requests sent to the server via HTTP.
- *Component Server* includes software components that are on the server side:
- *HttpServer* is HTTP-server component that handles HTTP requests from components Client. Incoming requests associated with the backup components are transferred to the Coordinator (through DataFlowApi), and those that processes user (user authentication) to component Database (via UserApi).

- *Database* is component database responsible for storing and reading user data and its backup; and is responsible for authentication, checking ID and password obtained from records in the database.
- *Coordinator* is Component Coordinator, manages the backup and de-duplication of data. Distributes incoming power between workers de-duplication systems (components Worker) and communicates between them via DedupApi. After the de-duplication process is responsible for the formation of backups and save them in the cloud storage (component CloudStorage), using BackupApi.
- *Worker* is Workers component of the system that is responsible for data de-duplication process, namely the cutoff overlapping blocks and save new. Communicates with the component Index (via IndexApi) to verify the hash values for blocks existing ones system and the preservation of the hash values of new units that are actually stored in the system in the cloud storage using component CouldStorage through their common interface BlockApi.
- *CloudStorage* is cloud storage component, responsible for maintaining the existing files with blocks of data links and trees backups in cloud storage.
- Index -component responsible for forming the index and de-duplication checking hash values for blocks existing ones.

Class diagrams are a structural UML diagram that describes the structure of the system showing the class system, their attributes, methods and relationship with other classes. Class diagram modeling the cornerstone of object-oriented programs. It is used for general conceptual structure modeling software. Classes in the class diagram representing the main elements of the program, their relationship in the program and actually most classes to be programmed. In most class diagrams depicted as rectangles with three compartments:

- Class name with capital letters, bold type in the middle compartment;
- Class attributes, with small letters, aligned to the left;
- Class methods with small letters, aligned to the left.

If classes are linked between them conducts communications association. If two classes defined association, you can move objects from one class to another object. It is acceptable when both ends associations belong to the same class. This means that the object of a class is allowed to associate with other objects of the same class. The association, which connects two classes, called binary. It is possible, although it is rarely necessary to create associations linking several classes. Graphically association depicted as a line connecting a class to himself or to other classes. Figure 6 shows a graph classes studied intellectual system de-duplication and distribution of data in cloud storage.

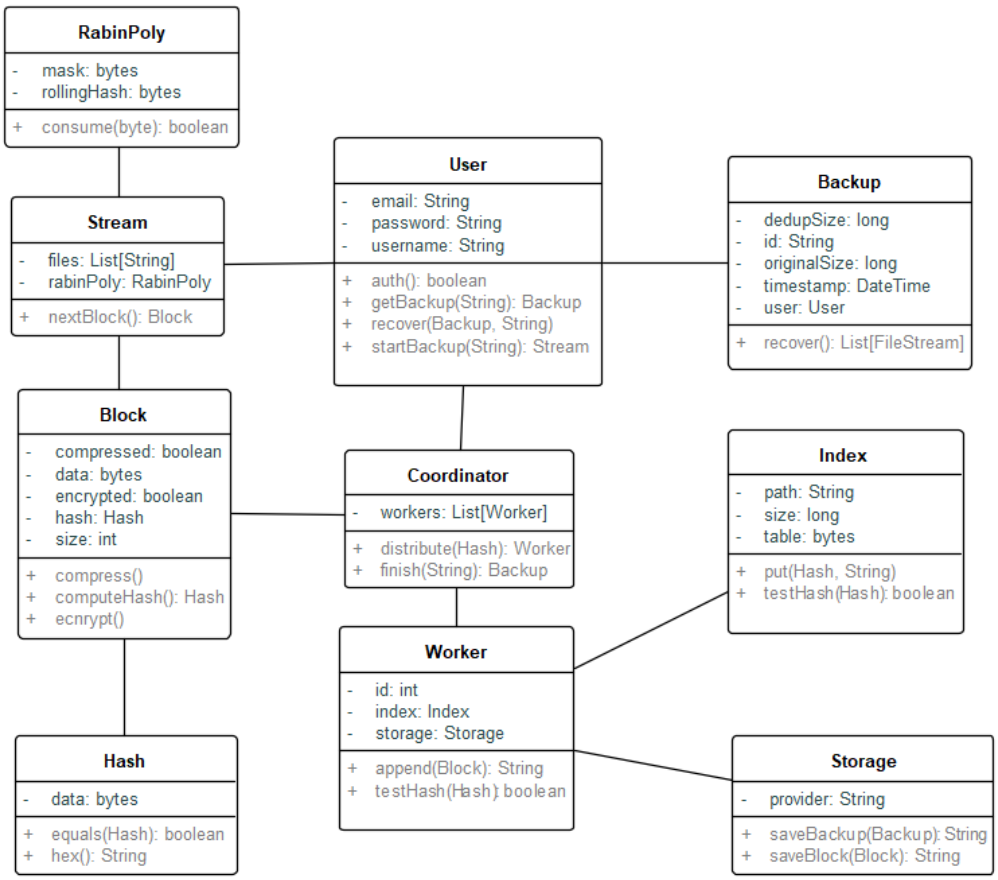

Figure 6. Class diagram

User class represents a user's system. Contains information about the user and is responsible for backing up and restoring data. Attributes and methods:

- *username* (attribute type: String) is unique user name in intelligent systems used to identify it.
- *email* (attribute type: String) is email user is used during authentication.
- *password* (attribute type: String) is password used during authentication.
- *auth* (method of resulting type: Boolean) is carries out user authentication based on his e-mail and password.
- *startBackup* (method of input parameter file paths for backup) is begins the process of de-duplication. The result is an object of type Stream, which represents the flow of data de-duplication.
- *getBackup* (method with the same input parameters as in startBackup) is checks whether the backup was completed, and if so, returns an object of type Backup.

- *recover* (method of input parameters object backup type Backup and towards the restored files of type String) is restores the original data from the backup in the specified path on the PC.

The class represents *Backup* that contains information about the creation of original size, size after de-duplication and owner. It is responsible for the restoration of the original data. Attributes and methods:

- *id* (attribute type: String) is unique identifier backup.
- *Timestamp* (DateTime type attribute) is creating a backup.
- *OriginalSize* (attribute such long) is the size of the original data to back up.
- *DedupSize* (attribute such long) is the size of backup data after de-duplication.
- *User* (type attribute User) is the owner of the backup.
- *Recover* (method is recovers files from backup). The result is a list of objects of type FileStream is class that contains metadata about the file and stream of bytes of the file.

RabinPoly class represents a component splitting the data stream into blocks using an algorithm Rabin. Attributes and methods:

- *Mask* (attribute type bytes) is mask set used for the logical product of the ring to get a point hash partitioning data.
- *RollingHash* (attribute type bytes) is ring a hash varies based on each input byte of the data stream.
- *Consume* (Method with the input parameter of type byte). Absorbs incoming bytes and renews ring hash-based algorithm Rabin. The result of true or false, tells whether to break the flow of the block at the moment whether to continue acquisitions.

Class Stream represents the flow of data obtained from the file list provided by the user. It is responsible for the formation of blocks of data de-duplication. Attributes and methods:

- *Files* (attribute type List [String]) is list of files on which to build a data stream.
- *rabinPoly* (attribute type RabinPoly) is contains the object class RabinPoly.
- *nextBlock* (Method from the resulting type Block) is gradually chastises data file method is to consume RabinPoly attribute for each input byte. When consume notifies breaking point, forms a Block object class that is the result of the method.

Block class to represent the data de-duplication derived from splitting the original data stream. Responsible for compression, encryption and hash value calculation data. Attributes and methods:

- *Data* (type attribute bytes) is the array of bytes of data.
- *Size* (attribute type integer) is length array of bytes of original data.
- *Hash* (attribute type Hash) is hash value of the block.

- *Compressed* (attribute type Boolean) is indicates whether the data byte array is compressed.
- *Encrypted* (attribute type Boolean) is indicates whether the data encrypted byte array.
- *computeHash* (method resulting type Hash) is calculates the hash value of the block of data using a hash function, the result is stored in the attribute hash.
- *Compress* (method) is provides data compression, the result is stored in the attribute data, if you change the size of the block is written to a new attribute size.
- *Encrypt* (method) is provides data encryption, the result is stored in the attribute data, if you change the size of the block is written to a new attribute size.

Hash class represents a hash value derived from data block. It is responsible for presenting hash 16-Covey notation and comparison with other hash. Attributes and methods:

- *Data* (type attribute bytes) is byte array that includes a hash value.
- *Equals* (method of input parameters that are other types of hash Hash) is compares the input hash value of its own.
- *Hex* (method resulting type String) is reflects the hash value in the 16-Covey system.

Class Coordinator appointed to coordinate and distribution units between workers in the cluster. Attributes and methods:

- *Workers* (attribute type List [Worker]) is working list of known objects in a class Worker.
- *Distribute* (method of input parameters which are object of type Hash) is based on the number 16 NGO received from the input hash object (calling method hex) decides which worker delegate tasks. The result is an object method call type Worker.
- *Finish* (method of input parameter which is the identifier backup) is de-duplication process completes and generates an object class backup Backup.

Worker working class represents a system, which is responsible for processing incoming data blocks and hashes to verify existing system. Attributes and methods:

- *Id* (attribute type integer) is unique identifier worker in the cluster.
- *Index* (type attribute Index) is Containing class Index.
- *Storage* (attribute type Storage) is contains the object class Storage.
- *testHash* (method of input parameters which are object class Hash) is hash checks if the input exists in the system using an attribute index and calling the appropriate method.
- *Append* (method of input parameters, which are object class Block) is block stores data in cloud storage using storage attribute and calling it an appropriate method.

Class Index represents the index system in a hash table that stores all the known hashes system as a key and a link to a block of data in the cloud in the form of value.

Attributes and methods:

- *Path* (Attribute type String) is contains the path to the index on the disc.
- *Size* (attribute such long) is the size of the index bytes.
- *Table* (type attribute bytes) is byte array that follows the structure of the hash table.
- *Put* (input parameters method, which is the hash, is Hash type path to block cloud type String - adds a new hash table key-value pair, where hash keys are input and sets the path to the block of data in the cloud.
- *testHash* (method input parameter type which is the hash) is Hash checks if the input is stored in a hash table.

Storage class represents cloud storage. It is responsible for maintaining the blocks of data and backups. Attributes and methods:

- *Provider* (Attribute type String) contains information about cloud storage providers.
- *saveBlock* (method input parameter which is a block of data type Block) is stores input data unit in cloud storage.
- *saveBackup* (method input parameter is a backup type Backup) is backs up data in the cloud storage.

Production problems and justification

The overall objective of development is to create an intelligent system of de-duplication and distribution of data in cloud storage that will be used as backup software to eliminate duplicate data and increase bandwidth input data stream and the overall speed of the process.

The study intelligent de-duplication and distribution of data in the cloud storage is designed to complete the existing de-duplication processes and their integration with the cloud. The place of application development system is a company or personal users aimed at backing up data.

According to their description, these systems show high rates de-duplication and the ability to integrate into existing backup applications and connect different cloud storage providers warehousing and even private domestic production (if using interface S3). However using cloud storage system examined only as part of final storage, while intelligent studied aimed at full utilization of cloud by distributing data and de-duplication of tasks. Thus, similar systems are inspected only on single-server monolithic machines. The effect of the introduction of intelligent system de-duplication and distribution of data in the cloud storage are:

- reduce the cost of back up is the process of de-duplication involves eliminating redundant data is critical for periodic backups, where the percentage change of data between them is small;

- increase performance, using algorithms distribution of data between cluster nodes working system;
- possibility of the system on remote servers and cloud storage to cloud storage;
- Easy integration into existing backup solution data.

Choice and justification of methods of solving the problem

With de-duplication species considered in Section 1.1 was selected de-duplication block level. Despite the fact that this type of de-duplication is difficult to implement and design, block level de-duplication to reduce the input volume provides data up to 95 percent (Crump, 2008). While de-duplication at instances depends on the type of data backups, and the slightest change in the source files require re-record them whole. Also, block level de-duplication to ensure uniform distribution of tasks and their volume between workers in a clustered environment that promotes the efficient use of computing resources, and distributes data between different cloud storage nodes.

In the first section considered and detailed four types of de-duplication: the original, target, and hybrid production. Among them need to select the type of de-duplication is best suited for distributed computing and data storage in the cloud storage to satisfactory research objectives.

The initial de-duplication is performed on the device, where the original data. Since the initial de-duplication works on the user's computer, the only way to increase its speed is vertical scalability by increasing the resources of the computer user (buying a new CPU, RAM and disk space). However, the user's computer containing the program it differently everyday use and drive used to store the original data. Therefore, to provide additional space for backup will be financially costly. Given these factors, the initial de-duplication is not suitable for research purposes and objectives developed intellectual system.

The target de-duplication is performed on the target central server, where the final stage of data will be stored. This approach is better suited than the previous one in that work connects the central server, thus accumulating data de-duplication from different sources. Also, this approach is better suited for the scalability of the system, as computing resources are installed on dedicated server's typically greater productivity; and can count on maximum use of resources because the server will only include de-duplication software that applies only de-duplication and data storage. Thus, this de-duplication type is advantageous in terms of scalability. However, given the fact that the target de-duplication requires a complete transfer of data storage on disk and only spend cutoff duplicates optimize limited to horizontal scalability. Vertical scalability is no longer, because regardless of the number of processors, all data passing through the drive which inherently single threaded and slower than RAM.

Streaming de-duplication to its nature, designed to solve the problem with a disk in the target de-duplication. The process of splitting the data stream occurs at the time of receiving data on the server and cutoff duplicates before burning to disc. This approach increases the load on memory, keeping in memory specific window of data, where the breakdown and cutoff that minimizes the load on the drive. It also allows the system to scale vertically. However, there are still only drawback of this approach, namely the load on the network and Internet user traffic, since all user data backup must be sent to the server. This becomes a big problem when performing periodic backups and percentage changes between copies is very small.

Hybrid de-duplication is a mixture of the source, target and streaming type of de-duplication. Namely partitioning data into blocks and duplicate cutoff occurs on the user side and the server is sent only new data blocks. It should be stressed that the computer user does not directly inserted into the process of eliminating duplicate directly client software sends the user a list of the hash values of the blocks, and the server indicating those hash values that he does not know. That work index lies entirely on the server. This prevents additional loading on the user's computer on the network and Internet traffic. Also, given the natural randomness of hash values, work with blocks easily shared among cluster nodes, thereby scaling the system to achieve maximum performance server hardware. Given these factors,

Selection algorithm partitioning the data stream

Breakdown of the input data stream into blocks for de-duplication block level includes two algorithms is fixed length and change. Under subsection fixed-length approaches involves splitting the flow at the level of pre-selected length, whereas the approach with a variable length, breaking point selects an algorithm based on the same thread. The algorithm with variable-length partition does not guarantee that the units will have the same length, but may set the minimum and maximum size blocks. For example, take the data stream where the unit is the letter of the Ukrainian alphabet, and the average length of the block 5 letters. After selecting letters from 'А' to 'Ф', we get 5 blocks of data. However, if the output stream is changing, such as the number 1 will be put between the letters 'Г' and 'Д' can and algorithms behave differently. Figure 7a shows a sample set using the approach of the breakdown of fixed length. The first block has not changed, and therefore considered to be existing and unnecessary repeated copying. However, all subsequent blocks will be considered for the new system because their content will be different from the original. Figure 7b shows a sample set using the approach of the breakdown of variable length. Unlike fixed-length partitioning algorithm with variable length, leaving the letters in the second block, although it increased output block size, but all subsequent blocks remain identical to the original. Thus, only the changed block, which occurred in the region offset. The algorithm partitioning variable-length works not only shift, as well as any changes using a hash algorithm ring. As an algorithm,

partitioning data stream block selected digital signature Rabin described in Section 1, which is based on calculating the hash ring. Optimal size of the window array is 16 bits long, because the size larger values show no significant improvement in the algorithm (Negruseri, 2012).

a)

АБВГҐ	ДЕЄЖЗ	ИІЇЙК	ЛМНОП	РСТУФ
АБВГҐ	1ДЕЄЖ	ЗИІЇЙ	КЛМНО	ПРСТУ

b)

АБВГҐ	ДЕЄЖЗ	**ИІЇЙК**	**ЛМНОП**	**РСТУФ**
АБВГҐ	1ДЕЄЖЗ	**ИІЇЙК**	**ЛМНОП**	**РСТУФ**

Figure 7. Breakdown of a) fixed length and b) variable length

So for each incoming data byte is calculated ring hash value, which is applied to a bit-wise operations AND (AND) with the value of the average length of the block. For this study, the average length was chosen block 4096, minimum 2048, and 8192. According to most studies mentioned above, these are the parameters yield maximum accuracy when splitting the data window length of 16 bits. So, each time performing input byte bit-wise AND operation on the ring hash value and the average length of the block - A, we get the number in the limit of zero to A. If the result is equal to a given pattern, the current byte is considered breaking point flow on the next block. Given the limits of the length of the block if the point found earlier than the minimum length, it is ignored by the time the minimum length. If breaking point is found, and the length of the block reaches its maximum length, the breakdown occurs at the moment of maximum length of the block. Such a case is possible when filling out repetitive bytes of data flow, such as a data area is filled with zeros, and therefore ring hash will be different from the specified template in any part of the troubled region. Given the limits of the length of the block if the point found earlier than the minimum length, it is ignored by the time the minimum length. If breaking point is found, and the length of the block reaches its maximum length, the breakdown occurs at the moment of maximum length of the block. Such a case is possible when filling out repetitive bytes of data flow, such as a data area is filled with zeros, and therefore ring hash will be different from the specified template in any part of the troubled region. Given the limits of the length of the block if the point found earlier than the minimum length, it is ignored by the time the minimum length. If breaking point is found, and the length of the block reaches its maximum length, the breakdown occurs at the moment of maximum length of the block. Such a case is possible when filling out repetitive bytes of data flow, such as a data area is filled with zeros, and therefore ring hash will be different from the specified template in any part of the troubled region. The most difficult in terms of computing system is the intellectual content index and search a identical data blocks based on their hash values. So this problem should be scaled horizontally between the nodes of the cluster system and horizontally between processors working machines. For horizontal scaling must select key distribution. Given the random nature of the hash values of the blocks depicting a hash value as numbers of N-number system, you can get almost nothing-weighted

key distribution. For example, take 16 systems (as currently the most common type of representation hash values). Taking the first number of hash values, we get a number from 1 to 16, with a probability of 1/16. Therefore, we get an even distribution of data blocks between nodes in 16 systems. If the system available fewer nodes the choice node number is modulus of the first of the hash value of the number of system units. If the units are more than 16, the second number is taken, thereby obtaining 256 combinations (162). Having defined the horizontal distribution of data blocks between workers, can achieve the parallel recording blocks in each storage node. Also, this way you can achieve and implement a distributed index. Find duplicate using the index is only CPU-dependent task, as the index is stored in memory thus avoiding the disc. A CPU-dependent task may be scaled vertically paralleling work between computer processors, in this case - search hash values in the index.

The database structure

The study intelligent de-duplication and distribution of data in the cloud storage includes work with blocks of bytes of data and controls their location-based index (hash table) in cloud storage. So the structure of a traditional relational database should be built only for purpose of information such as a list of users and their backups. Figure 8 is a diagram of a relational database, namely two tables: Backup and User, which are connected to bond many other. Overall these data sufficient for the operation of the system, since information on other system stores data in cloud storage, and they are not suitable for relational relations.

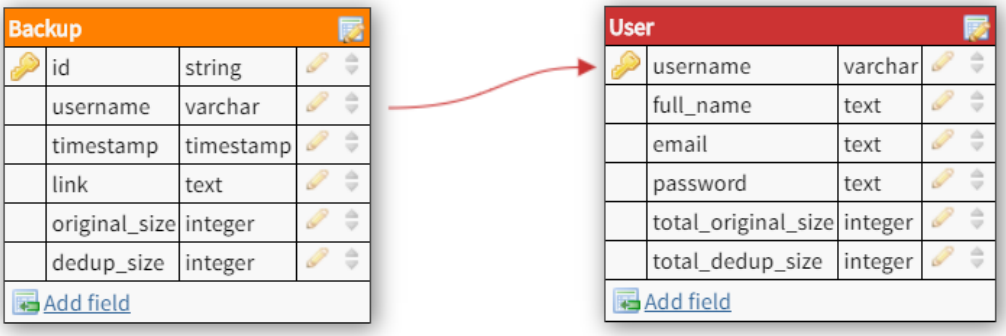

Figure 8. Relational database

User Table contains data about users of its columns:

- *username* is type varchar, unique key table contains a unique username and is its identifier can be used for user authentication;
- *full_name* is such as text, contains the full user name used for a formal appeal to the user;
- *email* is such as text, email user, can be used for user authentication;
- *password* is type text, password hash value is used to verify the real password during authentication;

- *total_original_size* is type integer, the overall size of the original data saved by the user varies with the addition of new backup or delete them;
- *total_dedup_size* is type integer, the total amount of data stored by the user after de-duplication, varies depending on the addition of new backup or delete them.

With data from columns total_original_size and total_dedup_size, can determine the rate of de-duplication: 1 is total_dedup_size / total_original_size, and the amount of disk space savings through de-duplication: total_original_size - total_dedup_size. Table Backup provides data backup system, its columns:

- *id* is type of text, the name of the backup, mainly consisting of user ID and time stamp of the backup;
- *username* is type varchar, unique user ID used to connect to the table User, realizing many relationship to each other;
- *timestamp* is type timestamp, timestamp backup is used to determine the age of the backup;
- *link* is such as text, links to backup to cloud storage;
- *original_size* is type integer, the original size of backup;
- *dedup_size* is type integer, the size of the backup after de-duplication.

In order for data in columns and total_original_size total_dedup_size User plates were always relevant, you must create a database trigger on the label on the actions addition of Backup and delete rows. Figure 9 shows a SQL trigger user_total_change change the values of columns and total_original_size total_dedup_size.

```
CREATE OR REPLACE TRIGGER user_total_change
    AFTER INSERT OR DELETE ON backup FOR EACH ROW
BEGIN
    IF TG_OP = 'INSERT' THEN
        UPDATE user set
            total_original_size = total_original_size + :new.original_size,
            total_dedup_size = total_dedup_size + :new.dedup_size
        WHERE username = :new.username;
    ELSIF TG_OP = 'DELETE' THEN
        UPDATE user set
            total_original_size = total_original_size - :old.original_size,
            total_dedup_size = total_dedup_size - :old.dedup_size;
        WHERE username = :new.username;
    END IF;
END;
```

Figure 9. SQL trigger changes total_original_size speakers and total_dedup_size

This trigger fires after the successful addition of either delete records from the table Backup, which works as follows:

- If it added a new line, the operation will be executed update, which will increase the value in columns and total_original_size total_dedup_size on original_size and dedup_size bound according to the user;
- If removed the old line, the operation will be executed update, which respectively reduce the value total_original_size and total_dedup_size.

The structure of the file system

For data storage either on disk space, or in cloud storage, you must use a custom file format as the storage blocks of data in relational databases is overspending read, write, and their index. Because relational database designed to store different types of data, which are uncertain scheme generated during program execution. However, in the case of de-duplication, you can avoid many of these restrictions, since the whole structure and layout data is known in advance. Consequently, the storage data block on the disk, you must have the following metadata - the serial number, the actual size of the block, the original block size, the hash value compression algorithm, encryption algorithm. It should be added that the actual block size may vary from the original because the compression algorithms and encryption can change it. Serial number of the unit needed for its quick access when restoring backups. Figure 10 shows a file structure that will store data blocks.

Figure 10. Structure list file data blocks

On the user side must have a Windows operating system with preinstalled Microsoft C ++ Runtime Library 2015 and above operating system or Linux. Also, regardless of the operating system must be installed Chrome browser or similar Chromium.

On the server side, must be installed JVM (Java Virtual Machine). Also open ports for access via HTTP protocol. Also requires installation on the primary node cluster with each other relational databases, such as Postgres or MySQL.

To connect cloud storage is requires paid subscription and opened on Amazon S3. Using your own cloud storage environment must have a server with a Linux operating system and installed one of S3 of free media are LeoFS, RiakCS, Minio, Ceph, SwiftStack and others.

The user interface is written in JavaScript, Html and CSS. Client software is written in Rust. The program server on Scala.

Intelligent data de-duplication and distribution of cloud storage software is used for data backup. This system can be used as a standalone product and as a module or the system.

Intelligent data de-duplication and distribution of cloud storage is multimodal and includes the following modules: user interface, applications, client applications and the index server.

User input is a set of files for backup. Client software generates based on the data stream of bytes of data and breaks it into blocks using partitioning algorithm based on hash ring. For each received block of data hash value is calculated and checked for the presence of the hash in the system. If there is already known hash using a link to this hash, is instead of the entire block. If not, then the block is stored in the system and writes a hash index referring to the block of data. This process is repeated for each block until the end of the stream. As a result, the input stream is stored as a list of hashes for future recovery from a backup. The distribution of data between workers in the cluster is based on hash values. The hash value is represented as a set of numbers with hexadecimal notation. To select the worker uses the first number of hash, depending on the number of workers (this should always be the number of two degrees). For example, if the number of workers 16, the hash value of the number equal to the number of the worker, if less than 16, the number is scaled to the required base. If the number of workers over 16 is used next number of workers, etc. It should also add that code is distributed, that each worker has its own index. If the number of workers over 16 is used next number of workers, etc. It should also add that code is distributed, that each worker has its own index. If the number of workers over 16 is used next number of workers, etc. It should also add that code is distributed, that each worker has its own index.

Module user interface written in the programming language JavaScript. To implement the user interface as a desktop application used ElectronJS library, this uses the power of Chrome web browser to display web pages outside the browser. To build the structure of the user interface used architectural pattern Model-View-Controller and its implementation AngularJS. Structural module consists of:

- package.json is configuration file contains a list of connected libraries and Problems of construction project.
- main.js is file business logic of the main window contains configuration discovery programs and checks the local cache. Depending on whether the user opens it identified the main window or the login window.
- main.css is file containing the style sheet describes the appearance of the user interfeyu.
- main.html is file describing the structure of the window and its layout.
- login.html is file form greeter.

- dashboard.html is file view general information about the backup user can create new and renew existing data.
- bridge.js is file business logic for communication with the client.

Module client software is written in the programming language Rust. The program is implemented as an architectural pattern command line interface. That is, this program can be used without a user interface by calling the first of the transfer of input arguments. However, this approach is used only for debugging. To process the command line is using the library Clap-rs. Uses Reqwest is as an HTTP client. For stream splitting into blocks is using library Rabinpoly, and to calculate their hashes is SHA-1. Structural module consists of:

- cargo.toml is configuration file contains a list of connected libraries and Problems of construction project.
- main.rs is entry point into the application process includes input arguments and call the necessary functions.
- rabinpoly.rs is splitting the data stream into blocks of variable length using an algorithm Rabin.
- hash.rs is hash value calculation block de-duplication.
- http.rs is interface to invoke HTTP requests.
- test_hashes.rs is check hashes on existing ones.
- send_data.rs is send new units to the server.
- backup.rs os composition of available functions and interfaces for the backup process using de-duplication.
- recover.rs is recover data from backups.

Module application server written in Scala programming and implementing actor's model library via Akka. As an HTTP server is using Akka-http. For use of cloud storage library Aws-sdk, this provides an interface to communicate with Amazon S3. Structural module consists of:

- build.sbt is configuration file contains a list of connected libraries and Problems of construction project.
- MasterMain.scala is point login main node cluster coordinator.
- WorkerMain.scala is entry point in the worker node cluster.
- HttpServer.scala is HTTP server configuration file and routing Web requests.
- MasterActor.scala is actor Coordinator, coordinates incoming and hashes based on their values employee chooses to delegate tasks.
- WorkerActor.scala is actor-worker communicates with the index and performs sending block de-duplication to cloud storage.

The program server and its employees use cloud storage to store data blocks and wood objects as backup. As cloud storage services must use S3 compatible interface. IP client part of the system intended for use on personal computers that run on Linux, MacOS and Windows. System requirements:

- RAM 1 GB.
- Disk space is at least 256 MB.
- Video memory is at least 256 MB.

The server component of intellectual systems intended for use in a distributed cluster environment, working with at least one node that runs on Linux, MacOS and Windows. System requirements for each node separately:

- Memory 4-16 GB.
- Disk space 1 TB.

User interface and client software must first be installed on your computer. The user interface is invoked by running the newly created icon on your desktop or running with a script located in the directory of the installed system. Client software no needs to call, this happens automatically, depending on the manipulation of the user interface. For successful startup and index server must first set up a working environment in a distributed cluster. You must know all the IP-addresses of each node of the cluster and point them at startup server. The index is built automatically at the start of every worker program client. The point of entry on the part of the user is the user interface in a desktop application. Entry point to server-side script is run clustered environment.

The input data for the user interface is user manipulation, the list of files to link to the de-duplication and backup data recovery. The input data for the program is the client file for de-duplication, block de-duplication list to send. The input data for the application server is the name of a backup list of hashes input block de-duplication and unique blocks of de-duplication. The input data for the index and links are hashes on the blocks in the repository.

Initial data for the user interface is the status of the backup process, references to backup files backup. Initial data for the client software blocks are all de-duplication, a list of hashes block de-duplication. Initial data for the application server is backed up. Initial data for the index is the answer for the hash block in the system.

Analysis of the results

So, consider the possibility of intelligent system de-duplication and distribution of data in the cloud storage and review its performance on the test case. To use the IP system must deploy backend, you can do both locally (for the test run) and cloud storage in Amazon AWS. Also is installed on the test PC client software.

After deploying the server, the user can connect to it using the client. However, you must first authenticate to the system (Figure 12). If the authentication fails (incorrectly entered mailbox or password), the program asks the user to enter data and check them again. After entering the correct password and pressing Enter button, the system successfully authenticates the user and the user interface main window (Figure 13).

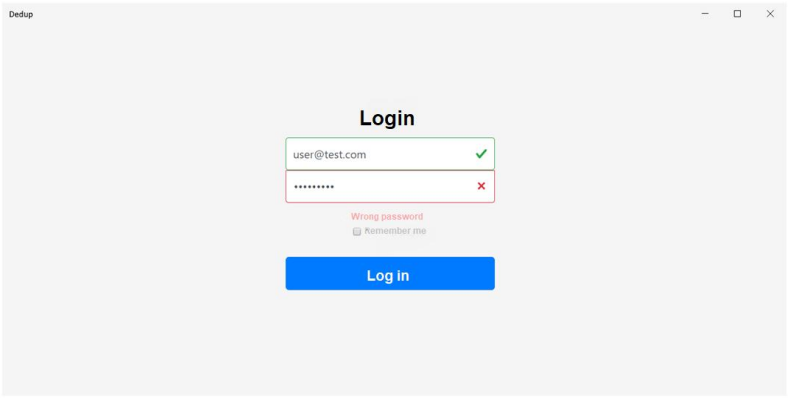

Figure 12. Notification window in the wrong password

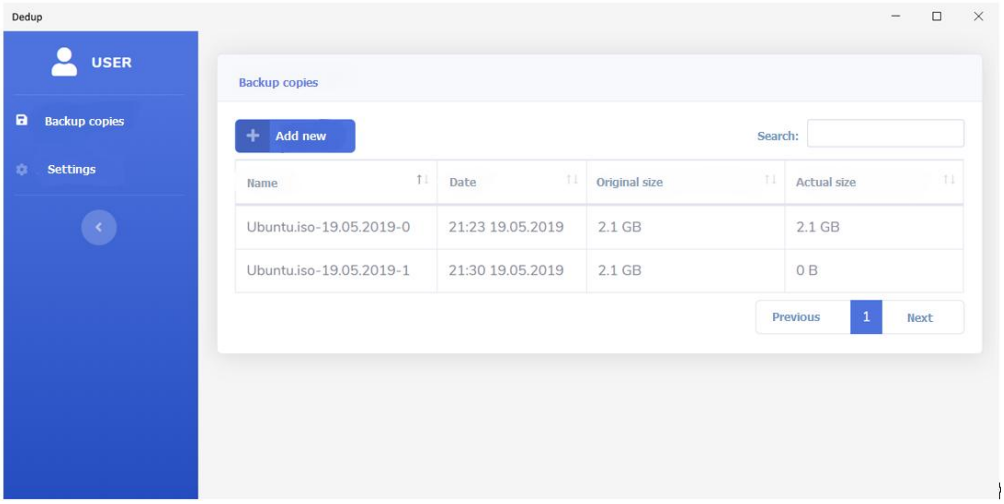

Figure 13. Main window

The main window consists of two parts is the sidebar and the main field. In the sidebar, you can choose the content of active field - back up or setting. By default, when a user logs in, he opened the box with backup, which shows a list of previously created backups. The interface allows sorting and searching for content backup lines. Also present Add button, which allows you to create a new backup, when clicked, opens a file selection window for backup (Figure 14).

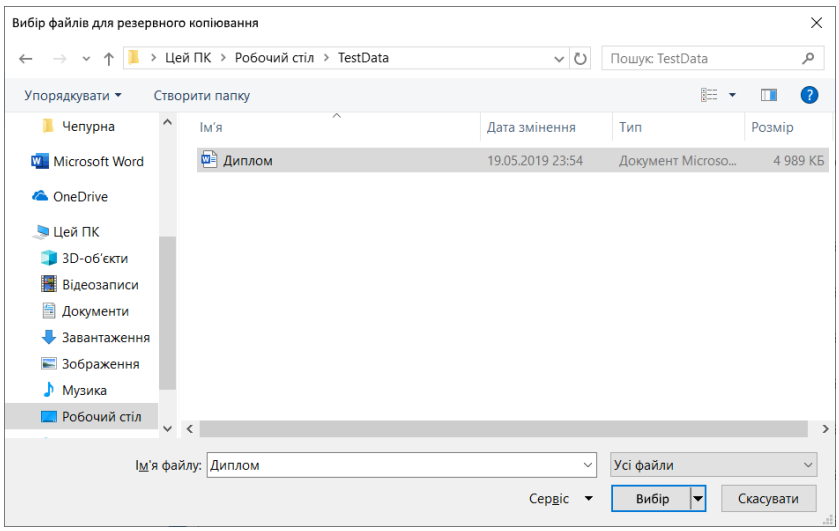

Figure 14. File selection window for backup

For the test run, "Диплом.docx" selected paper size 5MB. After selecting a file and pressing start Selection de-duplication process, namely the creation of data flow through this file and partition it into blocks; calculating the hash value of each unit and checking existing ones it already; sending new blocks of data to the server; saving the backup to cloud storage and display it to the user program. The result of the backup is shown in Figure 15, where the list of backups is added document file called Dokument.docx-19.05.2019-0 size 4.99 MB.

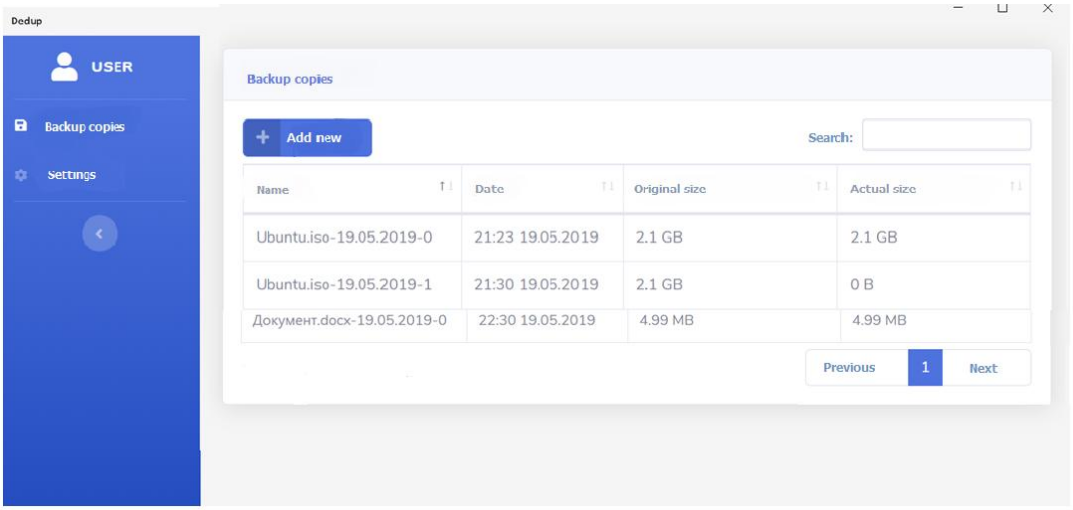

Figure 15. Backup window with the attached document

Figure 16 shows a recent message to the cluster backup «Документ.docx-19.05.2019-0». As is evident from reports all incoming data blocks saved as new as the data previously not known to the system.

```
22:29:570 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 0cacd9ccc3158f2e13c43d881d323289c2d61e94 : 4348
22:29:577 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 496a96d3bd7a0f7ffccfff98bb095d7d65617741 : 7379
22:29:580 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 62c63821f4ad60cd9ff39d2f902160183e651f7b : 5256
22:29:581 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 273094160eb75ab9e6641269a2b632297e606cc3 : 4269
22:29:589 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : c015ee1685fb0e18bbaa67573046a6029a251a24 : 7275
22:29:597 [worker-2] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : db7c7cbfbc805990074866e19d49d969565ee8f5 : 3679
22:29:598 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 6abb04c8a488900fefe4e0290b676e482aa319ce : 5352
22:29:607 [worker-0] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 726b41f26b99659422d191ac17239c50465a42e3 : 6479
22:29:615 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 655bf7aabc4efeb02111d58b6bcb79a895e98f7a : 3359
22:29:620 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 6b6bc85ccadb6512801a29d4dfe27d35cabcce3d : 2680
22:29:627 [worker-0] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : fe2507a3c018a6222597eb9bb382360af9dbeeb9 : 3742
22:29:633 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : ad3bae80b2634facd42c415022ed86995e68d7dc : 4516
22:29:639 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 0d167488d55a8352e2e3cb5f657190d2d1ba2e56 : 7184
22:29:643 [worker-2] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : dfb7c7920d70fdd1220f7e911d504812212f53cc : 2451
22:29:646 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 8142e05b10d6ae3e6a380de4c0e4d4deb508e596 : 3095
22:29:650 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 40d9cd5f2f181e2c5730bbc6e0798b6bad254d3e : 5281
22:29:654 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 0650cf5d3e9011dfa0f05045b8215cee4bd0d1f7 : 2078
22:29:659 [worker-2] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 15f30fba6da4780a91b976f6bf6d8695ea2634d9 : 3223
22:29:667 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : eb135e66faf6c9b13334b5cb3a65d77aff5538f2 : 6251
22:29:670 [worker-0] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 3b08598a0aa641844b4f0dd640ef8c79cfc5dc9e : 2154
22:29:671 [worker-2] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 98638d137342f1b12299c272f06a76bf608e73b3 : 3965
22:29:680 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 8d6aece9d8dc0a2ae56c45b89ef41134463111c6 : 7649
22:29:685 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 276cbfe9b2a705e80039c5a4099da0442ed6781f : 7759
22:29:690 [worker-2] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 1a76827c2321b5774429dc4d47b50713aa03b0f0 : 6946
22:29:694 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 04d4c76e47733da3b0aa9eecb2da299987733c1f : 4532
22:29:696 [worker-2] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 587983b6b8f6e98f79a4341fd960dfa99eea7c34 : 3422
22:29:697 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : a8e4ac9aa00c8d615748e5543b1cdff1a9e72455 : 7856
22:29:702 [worker-2] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : dc88178f2db20baab4ed6ec4860941a33255f4e4 : 7641
22:29:708 [worker-0] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : bb1c612876f77288afb55af6297315d54d1f2b5a : 2120
22:29:710 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 4168d41cb2435eeff3ad09bdf220d771873599e3 : 7627
22:29:719 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 4b63f065137cf98601cd790896c99e5983abf636 : 5349
22:29:723 [worker-1] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 45ab2cdd0f5b69741c9fd591da207b157930ac8d : 5016
22:29:733 [worker-2] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : dcb24f1d6880050e7981b68db7a72f00926fecfb : 5122
22:29:738 [worker-0] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 3b093efda2be455a94a0cc282e293844f3a044b1 : 6418
22:29:745 [worker-0] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 72078081259bd09985f3e13760f7a3cd56286bca : 3721
22:29:746 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : 6f62f16678eb3b8a6f613de66493df08fb6dd329 : 2704
22:29:755 [worker-0] INFO - NEW BLOCK : Документ.docx-19.05.2019-0 : f1b49172485754ff80780cc5da8cf58104f0907e : 2622
22:30:012 [coordinator] INFO - FINISHED : Документ.docx-19.05.2019-0 : ALL_BYTES=52232394 : NEW_BYTES=52232394 : BLOCKS=10435(new=10435)
```

Figure 16. Message log backup first

Similarly, you can draw attention to the correctness of the algorithm distribution of tasks between the nodes of the cluster (for example, if the hash starts at f, it always wills distributed to the employee with an index of 0, this analogy can be found in the picture for other hash values and workers). The last message is about completion, which indicates the number of new and total bytes of data and the number of data blocks. To test the de-duplication process during backup, you need to add document files again and look at the actual size of the field changes. Figure 17 shows a window where the backup was again added document file. As shown in the figure, the name of the backup is «Документ.docx-19.05.2019-1» and its actual size - 0 bytes on the server that is not saved any new blocks.

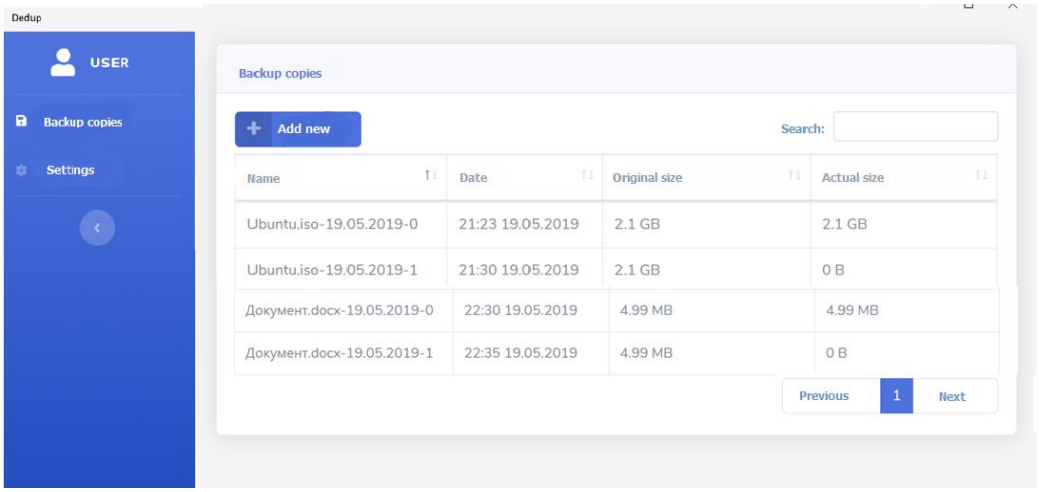

Figure 17. Backup Window with a second attached document

Figure 18 shows a recent message to the cluster backup «Документ.docx-19.05.2019-1». Since all data is already known to the system, the magazine no reports of new units, and thus there are only two notifications about the beginning and the end. Last post also indicates that the system received new byte 0 and 0 new units.

```
22:33:703 [coordinator] INFO - STARTED : Документ.docx-19.05.2019-1 : USER=user
22:35:214 [coordinator] INFO - FINISHED : Документ.docx-19.05.2019-1 : ALL_BYTES=52232394 : NEW_BYTES=0 : BLOCKS=10435(new=0)
```

Figure 18. Log messages for the second backup

To test the algorithm or splitting the data stream into blocks working properly, and not any significant changes to the document will not involve the replacement of all the blocks after the point of change. To do this, open the original file and delete the document from which some information and repeat steps as in Figure 12-13 in order to make a new backup.

Figure 19 shows a list of backups, with newly-«Документ.docx 19.05.2019-2» containing document files with slightly modified data in the middle of the file. As seen from the results, the original size of 4.87 megabytes (because of the data has been removed), and in fact is 13 kilobytes. That means that the algorithm partitioning blocks its purpose, and only changed blocks, where there were changes. Figure 20 shows a recent message to the cluster backup «Документ.docx-19.05.2019-2». According to these reports, was added only two new blocks (with 13 kilobytes), covering the place where the data has been removed from the original document. There was also reduced the total number of bytes and blocks, which corresponds to the data in Figure 20. So, during the test run, the file document was sent three times to backup that through de-duplication, data warehouse takes up only 4.99 MB + 131 KB = 5.1 MB.

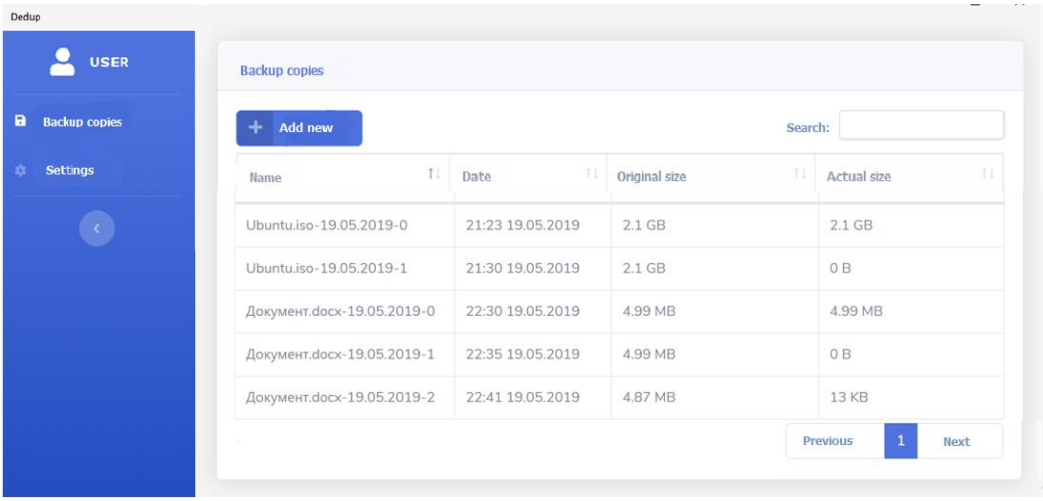

Figure 19. Window backups with third document attached

```
22:39:201 [coordinator] INFO - STARTED : Документ.docx-19.05.2019-2 : USER=user
22:40:365 [worker-3] INFO - NEW BLOCK : Документ.docx-19.05.2019-2 : 6019aef800db07e4c1f7b7a2a8236b235c0ff344 : 7651
22:40:387 [worker-0] INFO - NEW BLOCK : Документ.docx-19.05.2019-2 : f5eb3712d8ead6a4637dc82989acde77ffd64abf : 5246
22:41:856 [coordinator] INFO - FINISHED : Документ.docx-19.05.2019-2 : ALL_BYTES=5106565 : NEW_BYTES=12897 : BLOCKS=10415(new=2)
```

Figure 20. Log messages for the second backup

Then, if the copy would happen without de-duplication, the actual amount would be 4.99 MB + 4.99 MB + 4.87 MB = 14.85 MB.

Conclusion

During the conducting of the work is developed intelligent de-duplication and distribution of data in cloud storage. The resulting software has a user-friendly interface that allows you to backup and restore data. Analytical review of methodological principles of research, analyzes existing approaches to data backup using de-duplication and distribution of data in the cloud storage allocated to their advantages and disadvantages. It is considered in detail the advantages and disadvantages of modern technology de-duplication of data. The analysis proved the effectiveness of the design and implementation of intellectual system de-duplication and distribution of data in cloud storage. There is made a systematic analysis of the study of the subject area. A purpose of functioning and development of the system, purpose and location of the system, the expected effects of the introduction of the product is formulated. A conceptual model of the system is developed and detailed. These detailed diagrams precedents transitions states, sequences of components and classes, all of which helps to determine the behavior of a system to identify and formulate the necessary business processes. Analyzed (are advantages and disadvantages of different approaches) and selected effective methods for solving problems: hybrid de-duplication block level, splitting the data stream from digital print Rabin, distribution data based on hash values block de-duplication and use of the distributed index. The analysis means solving is chosen Rust programming language for writing client side, Scala programming language for server side, Akka tools for the management of distributed computing and Amazon S3 as cloud storage. Developed intellectual de-duplication and distribution system data to cloud storage, describes the software-discussed steps for the user. Testing of the system are designed several test cases, analyze the results. Adds up the above, we can conclude that the purpose of development achieved. Powered smart de-duplication and distribution of data in the cloud storage, which at completion of verification of the integrity of backups and optimize disk space after deleting old backups can be used in practice.

Acknowledgements

The article describes the results or research performed in Information systems and networks department in Lviv Polytechnic National University. We are grateful to our colleagues for

their support and understanding. All authors have completed the Unified Competing Interest form (available on request from the corresponding author) and declare: no support from any organization for the submitted work; no financial relationships with any organizations that might have an interest in the submitted work in the previous 3 years; no other relationships or activities that could appear to have influenced the submitted work.

References

Amdahl, G. (1967). The validity of the single processor approach to achieving large-scale computing capabilities. *Proceedings of AFIPS*, Atlantic City.

Andrunyk, V., Pasichnyk, V., Antonyuk, N., Shestakevych, T. (2020). A Complex System for Teaching Students with Autism: The Concept of Analysis. Formation of IT Teaching Complex. *Advances in Intelligent Systems and Computing IV*, 1080, 721-733.

Antonyuk, N., Medykovskyy, M., Chyrun, L., Dverii, M., Oborska, O., Krylyshyn, M., Vysotsky, A., Tsiura, N., & Naum, O. (2020). Online Tourism System Development for Searching and Planning Trips with User's Requirements. *Advances in Intelligent Systems and Computing IV*, 1080, 831-863.

Babichev, S., Taif, M.A., Lytvynenko, V., & Osypenko, V. (2017). Criterial analysis of gene expression sequences to create the objective clustering inductive technology. *37th International Conference on Electronics and Nanotechnology*, 244-248.

Babichev, S., Korobchynskyi, M., Lahodynskyi, O., Korchomnyi, O., Basanets, V., & Borynskyi, V. (2018). Development of a technique for the reconstruction and validation of gene network models based on gene expression profiles. *Eastern-European Journal of Enterprise Technologies*, 1 (4-91), 19-32.

Babichev, S., Lytvynenko, V., & Osypenko, V. (2017). Implementation of the objective clustering inductive technology based on DBSCAN clustering algorithm. *12th International Scientific and Technical Conference on Computer Sciences and Information Technologies*, 479-484.

Basyuk, T., Vasyliuk, A., & Lytvyn, V. (2019). Mathematical Model of Semantic Search and Search Optimization. *CEUR Workshop Proceedings*, Vol-2362, 96-105.

Chyrun, L., Kis, I., Vysotska, V., & Chyrun, L. (2018). Content monitoring method for cut formation of person psychological state in social scoring, *International Scientific and Technical Conference on Computer Sciences and Information Technologies*, 106-112.

Chyrun, L., Vysotska, V., Kis, I., & Chyrun, L. (2018). Content Analysis Method for Cut Formation of Human Psychological State, *International Conference on Data Stream Mining and Processing*, 139-144.

Crump, G. (2008). *Explaining deduplication rates and single-instance storage to clients*. Retrieved July 15, 2019, from https://searchitchannel.techtarget.com/tip/Explaining-deduplication-rates-and-single-instance-storage-to-clients.

Davydov, M., & Lozynska, O. (2017). Information System for Translation into Ukrainian Sign Language on Mobile Devices. *International Conference on Computer Science and Information Technologies*, 48-51.

Davydov, M., & Lozynska, O. (2017). Linguistic Models of Assistive Computer Technologies for Cognition and Communication. *International Conference* on *Computer Science and Information Technologies*, 171-175.

Fedushko, S. (2014). Development of a software for computer-linguistic verification of socio-demographic profile of web-community member. *Webology*, 11(2), Article 126. Retrieved July 15, 2018, from http://www.webology.org/2014/v11n2/a126.pdf

Gozhyj, A., Chyrun, L., Kowalska-Styczen, A., & Lozynska, O. (2018). Uniform Method of Operative Content Management in Web Systems. *CEUR Workshop Proceedings*, 2136, 62-77.

Kanishcheva O., Vysotska, V., Chyrun, L., & Gozhyj, A. (2017). Method of Integration and Content Management of the Information Resources Network, *Advances in Intelligent Systems and Computing*, 689, 204-216.

Khomytska, I., & Teslyuk, V. (2016). Specifics of phonostatistical structure of the scientific style in English style system. *Computer Science and Information Technologies*, 129-131.

Khomytska, I., & Teslyuk, V. (2017). The Method of Statistical Analysis of the Scientific, Colloquial, Belles-Lettres and Newspaper Styles on the Phonological Level. *Advances in Intelligent Systems and Computing*, 512, 149-163.

Korobchinsky, M., Vysotska, V., Chyrun, L., & Chyrun, L. (2017). Peculiarities of content forming and analysis in internet newspaper covering music news. *International Conference on Computer Science and Information Technologies*, 52-57.

Korzh, R., Peleschyshyn, A., Syerov, Y., & Fedushko, S. (2014). The cataloging of virtual communities of educational thematic. *Webology*, 11(1), Article 117. Retrieved July 15, 2019, from http://www.webology.org/2014/v11n1/a117.pdf

Kravets, P., & Prodanyuk, O. (2009). Game task of resource allocation. *Experience of Designing and Application of CAD Systems in Microelectronics*, 437-438.

Kravets, P. (2007). Game methods of the stochastic boundary problem solution. *Perspective Technologies and Methods in MEMS Design*, 71-74.

Kravets, P. (2006). Adaptive method of pursuit game problem solution. *Modern Problems of Radio Engineering, Telecommunications and Computer Science Proceedings of International Conference*, 62-65.

Kravets, P. (2003). Game methods of construction of adaptive grid areas. *The Experience of Designing and Application of CAD Systems in Microelectronics*, 513-516.

Kushner, H., & Yin, G.G. (2013). *Stochastic Approximation and Recursive Algorithms and Applications*. Springer Science & Business Media.

Lytvyn, V., & Vysotska, V. (2015). Designing architecture of electronic content commerce system. In: Computer Science and Information Technologies. *International Conference on Computer Science and Information Technologies*, 115-119.

Lytvyn, V., Sharonova, N., Hamon, T., Cherednichenko, O., Grabar, N., Kowalska-Styczen, A., & Vysotska, V. (2019). Preface: Computational Linguistics and Intelligent Systems (COLINS-2019). *CEUR Workshop Proceedings*, Vol-2362.

Lytvyn, V., Vysotska, V., Burov, Y., & Demchuk, A. (2018). Architectural ontology designed for intellectual analysis of e-tourism resources, *International Scientific and Technical Conference on Computer Sciences and Information Technologies*, 1, 335-338.

Lytvyn, V., Vysotska, V., Burov, Y., Bobyk, I., & Ohirko, O. (2018). The linguometric approach for co-authoring author's style definition, *International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems*, 29-34.

Lytvyn, V., Vysotska, V., Dosyn, D., Lozynska, O., & Oborska, O. (2018). Methods of Building Intelligent Decision Support Systems Based on Adaptive Ontology, *International Conference on Data Stream Mining and Processing*, 145-150.

Lytvyn, V., Vysotska, V., Peleshchak, I., Rishnyak, I., & Peleshchak, R. (2018). Time dependence of the output signal morphology for nonlinear oscillator neuron based on Van der Pol Model. *International Journal of Intelligent Systems and Applications*, 10, 8-17.

Lytvyn, V., Peleshchak, I., Peleshchak, R., Vysotska, V. (2019). Information Encryption Based on the Synthesis of a Neural Network and AES Algorithm. *3rd International Conference on Advanced Information and Communications Technologies*, 447-450.

Lytvyn, V., Vysotska, V., Shatskykh, V., Kohut, I., Petruchenko, O., Dzyubyk, L., Bobrivetc, V., Panasyuk, V., Sachenko, S., Komar, M. (2019). Design of a recommendation system based on Collaborative Filtering and machine learning considering personal needs of the user. *Eastern-European Journal of Enterprise Technologies*, 4(2-100), 6-28.

Lytvyn, V., Gozhyj, A., Kalinina, I., Vysotska V., Shatskykh, V., Chyrun, L., & Borzov, Y. (2019). An intelligent system of the content relevance at the example of films according to user needs. *CEUR Workshop Proceedings*, 2516, 1-23.

Lytvyn, V., Vysotska, V., Dosyn, D., & Burov, Y. (2018). Method for ontology content and structure optimization, provided by a weighted conceptual graph, *Webology*, 15(2), 66-85.

Maksymiv, O., Rak, T., & Peleshko, D. (2017). Video-based Flame Detection using LBP-based Descriptor: Influences of Classifiers Variety on Detection Efficiency. *International Journal of Intelligent Systems and Applications*, 9(2), 42-48.

Martin, D., Toro, del R., Haber, R., & Dorronsoro, J. (2009). Optimal tuning of a networked linear controller using a multi-objective genetic algorithm and its application to one complex electromechanical process. *International Journal of Innovative Computing, Information and Control*, 5/10(B), 3405-3414.

Martinez-Gil, J., Alba, E., & Aldana-Montes, J.F. (2008). Optimizing ontology alignments by using genetic algorithms. *The workshop on nature based reasoning for the semantic Web*. Karlsruhe, Germany.

Mirkin, B.G. (2005). *Clustering for Data Mining. A Data Recovery Approach*. London: CRC Press.

Montes-y-Gómez, M., Gelbukh, A., & López-López, A. (2000). Comparison of Conceptual Graphs. *Artificial Intelligence*, 1793. Retrieved July 15, 2018, from https://pdfs.semanticscholar.org/572a/8355404a7e6e909a9e923f5d469a cb9ec347.pdf.

Naum, O., Chyrun, L., Kanishcheva, O., & Vysotska, V. (2017). Intellectual system design for content formation. *Computer Science and Information Technologies*, 131-138.

Nazarkevych, M., Klyujnyk, I., & Nazarkevych, H. (2018). Investigation the Ateb-Gabor Filter in Biometric Security Systems. *International Conference on Data Stream Mining & Processing*, 580-583.

Nazarkevych, M., Klyujnyk, I., Maslanych, I., Havrysh, B., & Nazarkevych, H. (2018). Image filtration using the Ateb-Gabor filter in the biometric security systems. *International Conference on Perspective Technologies and Methods in MEMS Design*, 276-279.

Negruseri, C. (2012). Rolling hash, Rabin Karp, palindromes, rsync and others. Retrieved July 15, 2019, from https://www.infoarena.ro/blog/rolling-hash

Neogy, S.K., Bapat, R.B., & Dubey, D. (2018). *Mathematical Programming and Game Theory*. London: Springer.

Neyman, A., & Sorin, S. (2012). *Stochastic Games and Applications*. London: Springer Science & Business Media.

Petrosjan, L.A., & Mazalov, V.V. (2007). *Game Theory and Application*. New York: Nova Science Publishers.

Poirriez, V., Yanev, N., & Andonov, R. (2009). A hybrid algorithm for the unbounded knapsack problem. *Discrete Optimization*, 6(1),110-124.

Precup, R.-E., David, R.-C., Petriu, E.M., Preitl, S., & Rădac, M.-B. (2013). Fuzzy logic-based adaptive gravitational search algorithm for optimal tuning of fuzzy controlled servo systems. *IET Control Theory & Applications*, 7(1), 99-107.

Rabin, M. (1981). Fingerprinting by random polynomials. *Center for Research in Computing Technology*, Harvard University Report, Harvard.

Rami, J., & Vlach, M. (2002). Pareto-optimality of compromise decisions Fuzzy Sets and Systems, 129(1), 119-127.

Ramirez-Ortegon, M. A., Margner, V., Cuevas, E., & Rojas, R. (2013). An optimization for binarization methods by removing binary artifacts. *Pattern Recognition Letters*, 34(11), 1299-1306.

Robinson, J. Alan. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM* (JACM), 12(1), 23-41.

Rumbaugh, J., & Booch, G. (1999). *The unified modeling language reference manual*. Addison Wesley Longman Inc.

Russell, S.J., & Norvig, P. (2009). *Artificial intelligence: A modern approach*. (3rd ed.). Prentice Hall.

Rusyn, B., Lutsyk, O., Lysak, O., Lukeniuk, A., & Pohreliuk, L. (2016). Lossless Image Compression in the Remote Sensing Applications. *International Conference on Data Stream Mining & Processing* (DSMP), pp. 195-198.

Rusyn B., Pohreliuk L., Rzheuskyi A., Kubik R., Ryshkovets Y., Chyrun L., Chyrun S., Vysotskyi A., & Basto Fernandes V. (2020). The Mobile Application Development Based on Online Music Library for Socializing in the World of Bard Songs and Scouts' Bonfires. *Advances in Intelligent Systems and Computing IV*, 1080, 734-756.

Rusyn, I. B., & Valko, B. T. (2019). Container landscaping with Festuca arundinaceae as a mini bioelectrical systems in a modern buildings. *International Journal of Energy for a Clean Environment*, 20 (3), 211-219.

Rusyn, I. B., & Hamkalo, Kh. R. (2018). Bioelectricity production in an indoor plant-microbial biotechnological system with Alisma plantago-aquatica. *Acta Biologica Szegediensis*, 62 (2), 170-179.

Rzheuskyi, A., Kutyuk, O., Voloshyn, O., Kowalska-Styczen, A., Voloshyn, V., Chyrun, L., Chyrun, S., Peleshko, D., & Rak, T. (2020). The Intellectual System Development of Distant Competencies Analyzing for IT Recruitment. A*dvances in Intelligent Systems and Computing IV*, 1080, 696-720.

Silverberg, S. (2019). *OpenDedup Overview*. Retrieved July 15, 2019, from https://opendedup.org/odd/overview/

Solos, I. P., Tassopoulos, I. X., & Beligiannis, G. N. (2016). Optimizing shift scheduling for tank trucks using an effective stochastic variable neighbourhood approach. *International Journal of Artificial Intelligence*, 14(1), 1-26.

Shu, C., Dosyn, D., Lytvyn, V., Vysotska V., Sachenko, A., & Jun, S. (2019). Building of the Predicate Recognition System for the NLP Ontology Learning Module. *International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 802-808.

Tanenbaum A.S., & van Steen, M. (2017). Distributed Systems. *Upper Saddle River*, Pearson Prentice Hall.

Tkachenko, R., Duriagina, Z., Lemishka, I., Izonin, I., & Trostianchyn, A. (2018). Development of machine learning method of titanium alloys properties identification in additive technologies. *EasternEuropean Journal of Enterprise Technologies*, 3(12), 23-31.

Vysotska, V., Burov, Y., Lytvyn, V., & Demchuk, A. (2018). Defining Author's Style for Plagiarism Detection in Academic Environment, *International Conference on Data Stream Mining and Processing*, 128-133.

Vysotska, V., Fernandes, V.B., & Emmerich, M. (2018). Web content support method in electronic business systems. *CEUR Workshop Proceedings*, 2136, 20-41.

Vysotska, V., Fernandes, V.B., Lytvyn, V., Emmerich, M., & Hrendus, M. (2019). Method for Determining Linguometric Coefficient Dynamics of Ukrainian Text Content Authorship, *Advances in Intelligent Systems and Computing*, 871, 132-151.

Vysotska, V., Hasko, R., & Kuchkovskiy, V. (2015). Process analysis in electronic content commerce system. *International Conference Computer Sciences and Information Technologies*, 120-123.

Vysotska, V., & Chyrun, L. (2015). Methods of information resources processing in electronic content commerce systems. *The Experience of Designing and Application of CAD Systems in Microelectronics*, CADSM 2015-February.

Wendt, J. (2008). *Inline vs. post-processing deduplication appliances*. Retrieved July 15, 2019, from https://searchdatabackup.techtarget.com/tip/Inline-vs-post-processing-deduplication-appliances

Wesley, D. (2008). *Introduction to Data Deduplication*. Retrieved July 15, 2019, from https://www.petri.com/data-deduplication-introduction

Wilson, V., & Cox, M. (2019). *Using StorReduce for cloud-based data deduplication*. Retrieved July 15, 2019, from https://cloud.google.com/solutions/partners/storreduce-cloud-deduplication

Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. London: John Wiley & Sons.

Xue, X., Wang, Y., & Hao, W. (2015). Optimizing ontology alignments by using NSGA-II. *The International Arab Journal of Information Technology*, 12(2), 176-182.

Zhezhnych P., & Markiv O. (2018) Linguistic Comparison Quality Evaluation of Web-Site Content with Tourism Documentation Objects. In: Shakhovska N., Stepashko V. (eds), *Advances in Intelligent Systems and Computing II. CSIT 2017. Advances in Intelligent Systems and Computing*, 689. Springer, Cham, 656-667.

Bibliographic information of this paper for citing:

Lytvyn, Vasyl, Vysotska, Victoria, Osypov, Mykhailo, Slyusarchuk, Olha, & Slyusarchuk, Yuriy (2019). "Development of intellectual system for data de-duplication and distribution in cloud storage." *Webology*, 16(2), Article 188. Available at: http://www.webology.org/2019/v16n2/a188.pdf