Issue 6

https://doi.org/10.23939/sisn2019.02.001

УДК 004.9

МЕТОД ДЕДУБЛІКАЦІЇ ТА РОЗПОДІЛУ ДАНИХ У ХМАРНИХ СХОВИЩАХ ПІД ЧАС РЕЗЕРВНОГО КОПІЮВАННЯ ДАНИХ

Б. П. Русин[1], Л. В. Погрелюк[2], В. А. Висоцька[3], М. М. Осипов[4]

[1, 2] Фізико-механічний інститут імені Г. В. Карпенка НАН України,
[3, 4] Національний університет "Львівська політехніка", кафедра інформаційних систем та мереж,
[1] b.rusyn.prof@gmail.com, ORCID: 0000-0001-8654-2270
[2] liubomyr@inoxoft.com, ORCID: 0000-0003-1482-5532
[3] victoria.a.vysotska@lpnu.ua, ORCID: 0000-0001-6417-3689
[4] mykhailo.osypov@gmail.com, ORCID: 0000-0002-6611-724X

Розроблено інтелектуальну систему дедублікації та поширення даних у хмарних сховищах. Сформоване програмне забезпечення має зручний інтерфейс, який дає змогу створювати резервні копії та відновлювати дані. Здійснено аналітичний огляд методологічних засад дослідження, проаналізовано різні підходи до резервного копіювання даних із використанням дедублікації та розподілу даних у хмарному сховищі, висвітлено їхні переваги та недоліки. Детально розглянуто переваги та недоліки сучасних технологій дедублікації даних. Цей аналіз довів ефективність розроблення та впровадження інтелектуальної системи дедублікації та розподілу даних у хмарному сховищі. Виконано систематичний аналіз предметної області. Сформульовано мету функціонування та розвитку системи, мету та місце функціонування системи, визначено очікувані ефекти від впровадження програмного продукту. Розроблено та детально описано концептуальну модель системи. Наведено детальні діаграми прецедентів, стану переходів, послідовностей, компонентів та класів, що разом дають змогу визначити поведінку системи, встановити та сформулювати необхідні бізнес-процеси. Проаналізовано (наведено недоліки та переваги використання різних підходів) та вибрано ефективні методи розв'язання задач: гібридна дедублікація на рівні блока, розбиття даних на основі цифрового відбитка Рабіна, розподіл даних на основі хеш-значень одиниці дублювання та використання розподіленого індексу. Під час аналізу розв'язків задач вибрано мову програмування Rust для написання клієнтської частини, мову програмування Scala для серверної частини, Akka для управління розподіленими обчисленнями та Amazon S3 як хмарне зберігання. Розроблено інтелектуальну систему дедублікації та розподілу даних у хмарному сховищі, здійснено опис програмного забезпечення, розглянуто етапи роботи користувача. Проведено тестування роботи спроєктованої системи та створено кілька контрольних зразків, проаналізовано результати.

Ключові слова: дедублікація даних, розподіл даних, хмарне середовище, cloud computing, алгоритм Рабіна, хешування даних, гібридна дедублікація.

Вступ

Дедублікація є процесом усунення дублювальних копій повторюваних даних. Загалом є два види дедублікації: дедублікація на рівні екземплярів (англ. *single instance storage*) та дедублікація на рівні блоків (англ. *block-level deduplication*) [1–3]. Дедублікація на рівні екземплярів (або найчастіше файлів) доволі проста: якщо в системі є два або більше ідентичні файли, то фізично на диски чи у будь-якому сховищі зберігається лише один екземпляр, а інші отримують посилання на нього. Такий

підхід часто використовується в поштових сервісах, наприклад, у Microsoft Exchange для вкладень у електронних листах. Якщо один користувач надіслав лист із вкладеними зображеннями чи документами десяткові інших користувачів, то лише одна копія цих вкладень буде збережена на сервері. Такий підхід використовують також під час резервного копіювання даних, наприклад, робочих комп'ютерів працівників у компанії, де десятки резервних копій операційної системи Windows матимуть велику кількість ідентичних системних утиліт та dll-файлів. Більшість програм резервного копіювання збережуть лише один екземпляр із повторюваних файлів, а дублікати замінять посиланням на оригінал. Проте дедублікація на рівні екземплярів неефективна, якщо оригінальні файли були якось модифіковані, наприклад, додано новий рядок тексту в документ або змінено один піксель у великому за обсягом зображенні. Зміна пари байтів даних може призвести до повторного копіювання оригінальних даних, тим самим значно збільшуючи простір, зайнятий резервними копіями. В такому випадку необхідно виконувати другий тип дедублікації, а саме дедублікацію на рівні блоків.

Постановка проблеми

Дедублікація на рівні блоків ігнорує такі поняття, як екземпляр або файл, і отримує на вхід лише потік даних у вигляді звичайних байтів інформації. Для того щоб здійснити дедублікацію, в цьому підході потік даних розділяють на блоки певного розміру та для кожного обчислюють хеш-значення, використовуючи хеш-функції (md5, sha). За допомогою хеш-значень система визначає, чи певний блок даних уже існує в ній, за допомогою індексу (або хеш-таблиці). Якщо цей блок уже є, то в резервній копії зберігається лише посилання на нього. В іншому випадку зберігається оригінал даних, а його хеш-значення записується в індекс для подальшої дедублікації.

Аналіз останніх досліджень та публікацій

Один із найпоширеніших алгоритмів кільцевого хешу – цифровий відбиток Рабіна [4]. Це метод утворення цифрових відбитків (хешів) за допомогою поліномів над полем зі скінченної множини елементів, який розробив М. О. Рабін [5]. Формальне визначення: маючи n-бітове повідомлення $m_0,\ldots, m_{n-1}$, його можна розглянути як поліном степеня $n-1$ над полем зі скінченної множини елементів (поле Галуа):

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

Вибравши випадковий незвідний поліном $p(x)$ степеня k, можна визначити цифровий відбиток повідомлення m як залишок від ділення $f(x)$ на $p(x)$, який можна подати як поліном степеня $k-1$ або k-бітового числа. Цифровий відбиток Рабіна використано в алгоритмі Рабіна–Карпа – алгоритмі пошуку рядка. Ідея полягає в уникненні порівняння всіх рядків зі всіма, тим самим усуваючи квадратичну складність алгоритму. В кращому випадку складність $O(n + m)$, в гіршому – $O(nm)$. Алгоритм пошуку рядків подібний до пошуку точки розбиття потоку даних на блоки, оскільки точка розбиття повинна ґрунтуватись на змісті даних, а не на довжині блока, тобто необхідно відшукати потрібні дані якнайшвидше. Ідея алгоритму така: для простоти припустимо, що алфавіт складається із десяткових цифр $\Sigma = \{0, 1,\ldots,9\}$. Після цього рядок із k символів можна розглядати як число довжини k. Тобто символьний рядок "12345" відповідає числу 12345. Для заданого зразка $P[1..m]$ позначимо через p відповідне йому десяткове значення. Аналогічно, для заданого тексту $T[1..n]$ позначимо через t_s десяткове значення підрядка $T[s+1..s+m]$ довжини m, якщо $s = 0, 1,\ldots, n-m$. Очевидно, що $t_s=p$ тоді й тільки тоді, коли $T[s+1..s+m]=P[1..m]$; отже, s – допустимий зсув тоді й лише тоді, коли $t_s=p$. Якщо значення p можна обчислити за $O(m)$, а значення t_s за сумарний алгоритмічний час $O(n-m+1)$, то усі допустимі зсуви можна було б знайти за алгоритмічний час $O(m) + O(n-m+1) = O(n)$, порівнюючи p із кожним з можливих t_s. За допомогою схеми Горнера значення p можна обчислити за час $\Theta(m)$ (2).

$$p = P[m] + 10(P[m-1] + 10\big(P[m-2]+\ldots+10(P[2] + 10P[1])\big)\ldots)), \quad (2)$$

Значення t_0 можна обчислити з масиву $T[1…n]$ аналогічним способом за час $O(m)$. Водночас, знаючи величину t_s, значення t_{s+1} можна обчислити за фіксований час (3):

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

Наприклад, якщо m=5 і t_s=31415, то потрібно видалити цифру в старшому розряді $T[s+1]$=3 і додати цифру в молодший розряд (нехай $T[s+5+1]$=2). В результаті отримаємо t_{s+1}=10(31415–10000×3)+2=14152. Отже, всі t_s можна обчислити за алгоритмічний час $O(m)$. Проте цей алгоритм має недолік, пов'язаний з тим, що коли p і t_s занадто великі, з ними незручно працювати. Проте для розподілу потоку даних на блоки пошук рядків не має сенсу, оскільки вхідний потік даних доволі великий, а шанс знайти k-бітовий рядок досить малий. Проте цей алгоритм, а саме його частину – кільцевий хеш, можна використати по-іншому, обчислюючи кільцевий хеш на кожен вхідний байт даних і вважаючи точкою розбиття той момент, коли кільцевий хеш дорівнюватиме заданому шаблону. Проте для того, щоб отримати блок даних необхідної довжини, над одержаним хешем виконують побітову операцію І (AND) із бажаною середньою довжиною (4). Якщо отримане значення дорівнює встановленому шаблону, то поточна позиція байта є точкою розбиття.

$$P = H \,\&\, A, \quad (4)$$

Розподілені обчислення – це спосіб виконання трудомістких обчислювальних завдань з використанням двох або більше обчислювальних машин, об'єднаних у мережу [6–9]. Розподілені обчислення можна вважати окремим видом паралельних обчислень, коли розподіл завдань відбувається між декількома процесорами обчислювальної машини. А здатність системи збільшувати обсяг роботи за рахунок додавання нових обчислювальних ресурсів називають масштабованістю. Масштабованість ресурсів системи може бути двох типів: горизонтальна і вертикальна [10–16]:

- горизонтальна масштабованість – це збільшення обсягу опрацьованих задач додаванням додаткових вузлів у кластер обчислення, тобто додаванням додаткових комп'ютерів у мережу;
- вертикальна масштабованість – це розширення обсягу опрацьованих задач збільшенням ресурсів одного вузла (комп'ютера), наприклад, збільшенням кількості центральних процесорів (CPU), оперативної пам'яті та дискового простору.

Проте збільшення обчислювальних ресурсів не вирішує проблем проєктування. Для досягнення максимального ефекту необхідно, щоб програми, для яких забезпечується масштабованість, мали змогу розподілити ці ресурси коректно. Як правило, набагато дешевше додати новий вузол до системи, щоб досягти покращень, ніж інвестувати час та фінансові ресурси у перегляд архітектури системи, щоб забезпечити ефективну роботу в багатопоточному та розподіленому середовищі. Проте віддача за такого підходу зменшена. Для прикладу, нехай лише 70 % програми можна пришвидшити завдяки масштабованості. Згідно із законом Амдала [7], який визначає потенційне прискорення роботи програми у разі збільшення кількості обчислювальних ресурсів, частина програми, яку вже неможливо розподілити, обмежить загальний ефект від розпаралелення. Будь-яка задача зазвичай складається із множини підзавдань, які можуть виконуватись послідовно та паралельно. Цей зв'язок задають за допомогою формули (5), де S – прискорення роботи програми (як відношення до початкового часу) ;

$$S = \frac{1}{p + \frac{1-p}{n}}, \quad (5)$$

p – частина завдань, які можна виконати послідовно; $1–p$ – частина завдань, які можна виконати паралельно; n – кількість процесорів. Згідно з цим рівнянням, у певний момент (залежно від відсотка задач, які виконують послідовно), ефект від додавання нових процесорів втрачається. На рис. 1 подано графік залежності прискорення програми від кількості процесорів, що яскраво демонструє зменшення ефекту.

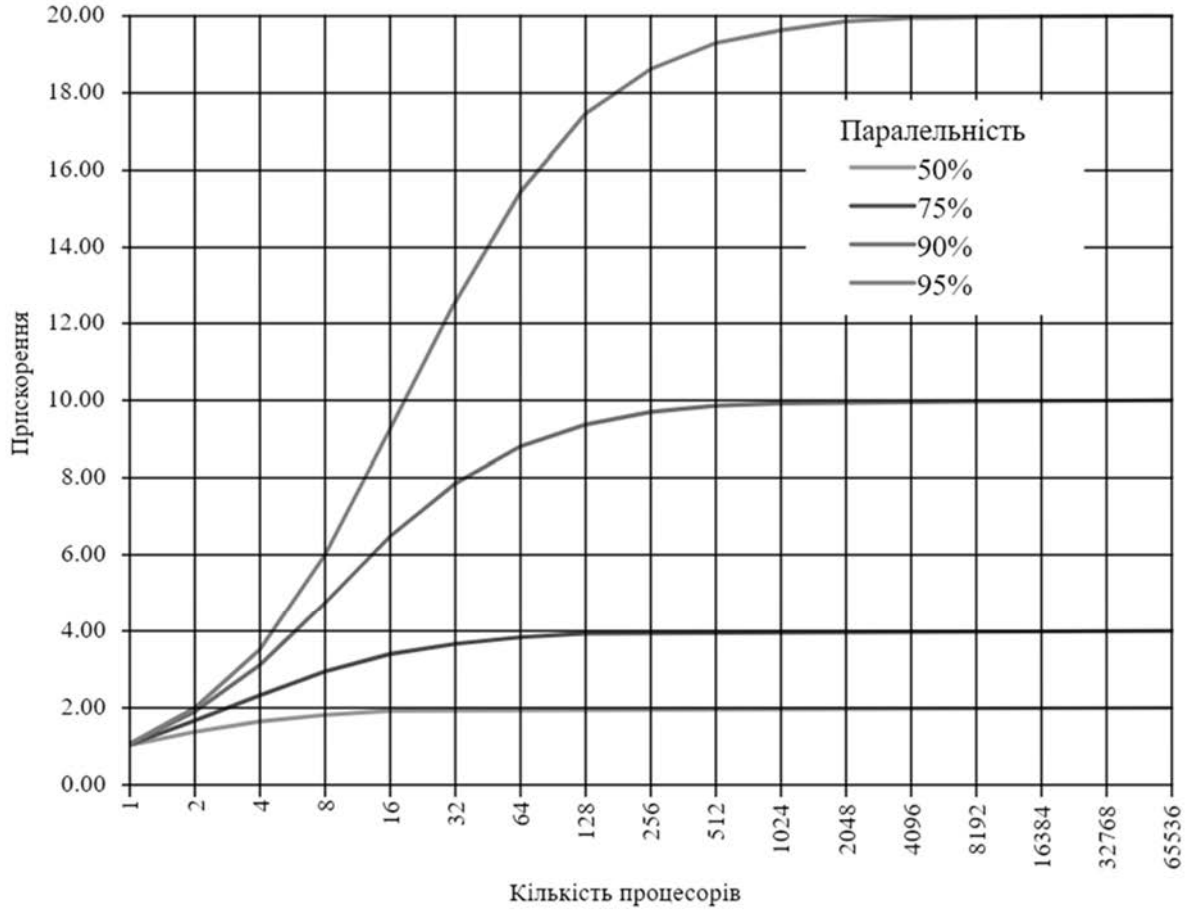

Рис. 1. Графік залежності прискорення програми від кількості процесорів

Отже, якщо розпаралелити 90 % програмних задач (тобто 10 % – це послідовні задачі), то прискорити виконання програми більш ніж у десять разів неможливо, незалежно від кількості доданих процесорів [17–25]. А отже, необхідно відповідальніше ставитись до розроблення архітектури розподілених систем.

Аналіз відомих систем

1. StorReduce – це спеціалізоване рішення для дедублікації даних у хмарі [8]. StorReduce забезпечує функціональність, подібну до постачальників хмарних сховищ, зокрема сховище об'єктів, облікові записи користувачів, ключі доступу, політики контролю доступу та вебінтерфейс управління, панель інструментів StorReduce. Сервер StorReduce працює на фізичній або віртуальній машині. Архітектура цього рішення дає йому змогу працювати на серверах хмарних сховищ, тим самим зменшуючи мережеву затримку між сервером та сховищем. StorReduce підтримує S3-інтерфейс для зберігання об'єктів. Така можливість спрощує інтеграцію рішення для систем, які уже використовують *S*3-сховища для своїх даних. Проте, згідно з описом цього рішення, сервер StorReduce працює на одиничному сервері, тим самим масштабуючись лише вертикально. А отже, для резервного копіювання більших вхідних масивів даних необхідно ставити або купувати у постачальників хмарних сховищ сервери із потужнішими процесорами та дисковими носіями типу HDD. Проте вертикальне масштабування має свою межу.

2. OpenDedup – це рішення дедублікованої файлової системи із відкритим вихідним кодом [9]. Має підтримку хмарних сховищ, а саме дає змогу зберігати дані у сховищах із підтримкою S3-інтерфейсу. Проте одним із недоліків цього рішення є прив'язка до операційної системи Linux (що унеможливлює використання цієї системи на Windows та MacOS), а тестування відбувалося на дистрибутивах Ubuntu та CentOS, тому користувачам доведеться самостійно налаштовувати несумісності у разі їх виникнення на інших дистрибутивах. Відсутній також

користувацький інтерфейс, тобто користуватись системою зможуть лише вузькоспеціалізовані користувачі або з інтеграцією до інших рішень [26–32].

Формулювання цілі статті

Мета – розробити метод дедублікації та розподілу даних у хмарних сховищах. Для виконання поставленого завдання необхідно виконати такі завдання:

· здійснити аналіз сучасних технологій проєктування та створення систем у сфері резервного копіювання та відновлення даних, порівняльний аналіз технічних реалізацій та алгоритмів, застосованих у реалізаціях;

· побудувати концептуальну модель інтелектуальної інформаційної системи дедублікації та розподілу даних у хмарних сховищах, задля автоматизації та ефективної роботи програмного забезпечення у сфері резервного копіювання та відновлення даних;

· проаналізувати методи та засоби реалізації інтелектуальної системи та вибрати доцільні для реалізації власної;

· спроєктувати інтелектуальну систему дедублікації та розподілу даних у хмарних сховищах.

Виклад основного матеріалу

На діаграмі (рис. 2) наведені такі актори (зовнішні сутності):

· *Користувач* – користувач системи, надсилає дані для резервного копіювання та відновлює оригінальні дані за допомогою отриманих резервних копій.

· *Індекс* – містить інформацію про всі відомі хеш-значення системи, використовується для вилучення блоків даних, які дублюються.

· *Хмарні сховища* – слугують сховищами для збережених унікальних блоків даних, а також резервних копій.

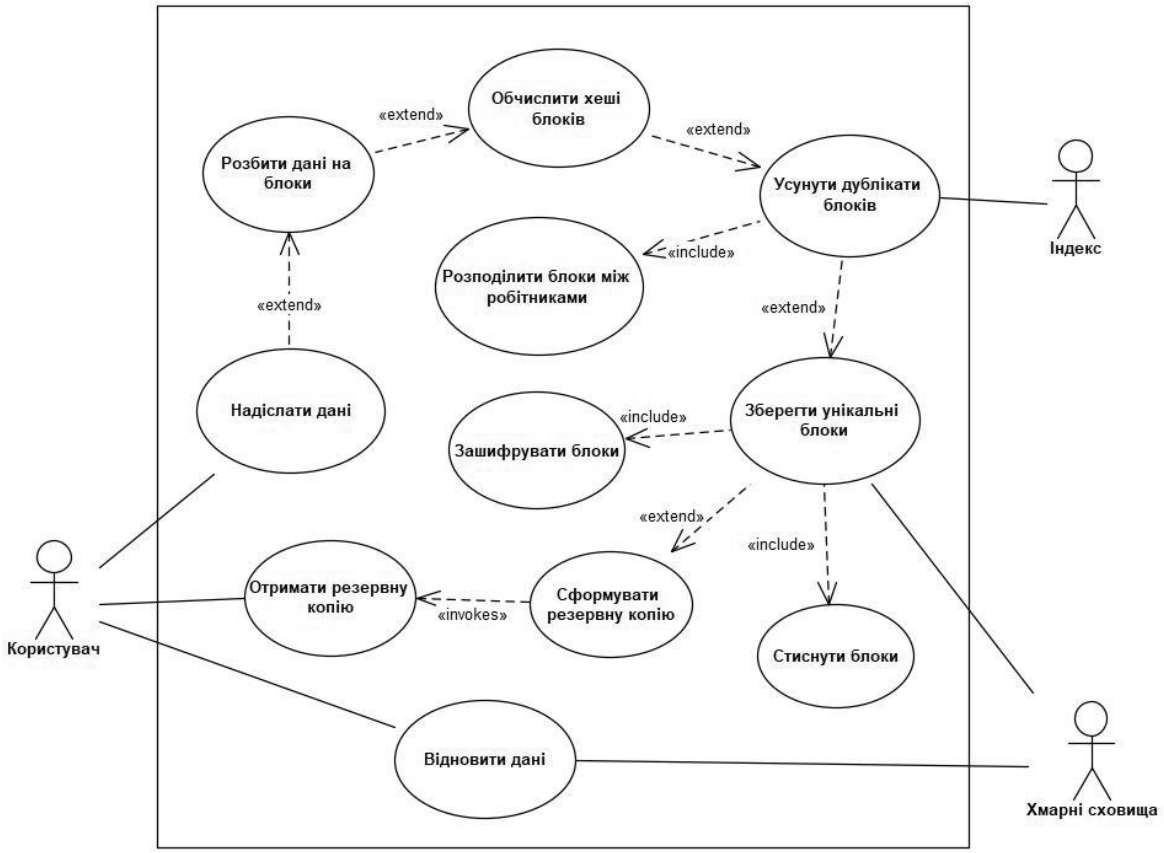

Рис. 2. Діаграма варіантів використання системи

На рис. 3 подано діаграму станів резервного копіювання даних користувачем інтелектуальної системи дедублікації та розподілу даних у хмарних сховищах. Об'єктом діаграми станів є процес резервного копіювання (надалі в цьому підпункті називатимемо цей процес *об'єктом*). Стан *автентифікації користувача* є початковим для резервного копіювання. Якщо дані автентифікації введено неправильно, то об'єкт залишається в тому самому стані й очікує на нові команди. Якщо дані автентифікації, а саме логін та пароль, введено правильно, об'єкт переходить до стану *вибору файлів для резервного копіювання*. Наступний стан – *створення потоку даних*. З вибраних файлів для дедублікації система створює єдиний потік даних. Після успішного створення потоку даних об'єкт переходить у стан *розподілу потоку даних на блоки*. Блок даних – це одиниця процесу дедублікації, саме дублікатів блоків система намагається уникати. Якщо в потоці даних є дані для розподілу на блоки, то об'єкт переходить у стан *обчислення хеш-значення блока*. Кожен блок даних повинен мати свій цифровий підпис, набагато менший за обсягом, яким і є хеш-значення. Після того як отримано хеш-значення, об'єкт переходить у стан *перевірки хеш-значення на існування*. Для того, щоб відкинути дублікати, треба перевірити, чи такий блок існує у системі, проте перевіряти весь блок цілком витратно, тому перевіряють, чи існують їх хеш-значення. Якщо хеш-значення уже відоме системі, то об'єкт переходить назад до стану *розбиття потоку даних* на блоки. А якщо хеш-значення невідоме, то об'єкт переходить до стану *збереження блока даних*. Новий блок даних, раніше не відомий системі, вона зберігає у спеціально відведеному місці в хмарному сховищі, на основі його хешу (для спрощення його подальшого пошуку). Після збереження об'єкт повертається до стану *розподілу потоку даних на блоки*. Коли об'єкт у цьому стані й більше не може отримати нових блоків з потоку даних, він переходить до стану *побудови резервної копії*. Після дедублікації список із блоків даних сформовано у вигляді списку із його хеш-значень, що і є резервною копією. Отримавши її, об'єкт переходить у стан *збереження резервної копії*, що є останнім етапом перед досягненням об'єктом кінцевого стану.

Результат резервного копіювання подано на рис. 4, де до списку резервних копій вже додано файл *Документ* під назвою *Документ.docx-19.05.2019-0* розміром 4,99 мегабайта.

На рис. 5 подано останні повідомлення кластера для резервної копії *Документ.docx-19.05.2019-0*. Як видно із повідомлень, усі вхідні блоки даних збережено як нові, оскільки ці дані раніше були не відомі системі. Можна також звернути увагу на коректність роботи алгоритму розподілення завдань між вузлами кластера (наприклад, якщо хеш починається на *f*, то він завжди буде розподілений до працівника з індексом 0, таку аналогію можна знайти на зображенні й для інших хеш-значень та працівників). Останнє повідомлення – інформація про завершення, з вказанням кількості загальних та нових байтів даних, а також кількості блоків даних.

Для тестування процесу дедублікації під час резервного копіювання необхідно додати файл *Документ* ще раз і подивитись, як змінилось поле *Фактичний розмір*. На рис. 6 подано вікно з резервними копіями, де вдруге додано файл *Документ*. Як бачимо, назва резервної копії – *Документ.docx-19.05.2019-1*, а її фактичний розмір – 0 байтів, тобто на сервері не було збережено жодних нових блоків. На рис. 7 подано останні повідомлення кластера для резервної копії *Документ.docx-19.05.2019-1*. Оскільки усі дані уже відомі системі, в журналі немає повідомлень щодо нових блоків, а отже, є лише два сповіщення – про початок та кінець. Останнє повідомлення також вказує, що система отримала 0 нових байтів та 0 нових блоків.

Для того щоб протестувати, чи алгоритм розбиття потоку даних на блоки працює правильно та чи будь-які незначні зміни в документі не спричинять заміну всіх блоків після точки змін, відкриємо оригінальний файл *Документ* та видалимо звідти частину інформації й повторимо кроки, створивши нову резервну копію. На рис. 8 подано список резервних копій з новоствореною *Документ.docx-19.05.2019-2*, яка містить файл *Документ* з дещо зміненими даними всередині файла. Як видно з отриманих результатів, оригінальний розмір 4,87 мегабайта (оскільки видалено частину даних), а фактичний – 13 кілобайт. Що означає, що алгоритм розбиття блоків виконав своє завдання і змінив лише ті блоки, у яких відбувалися зміни.

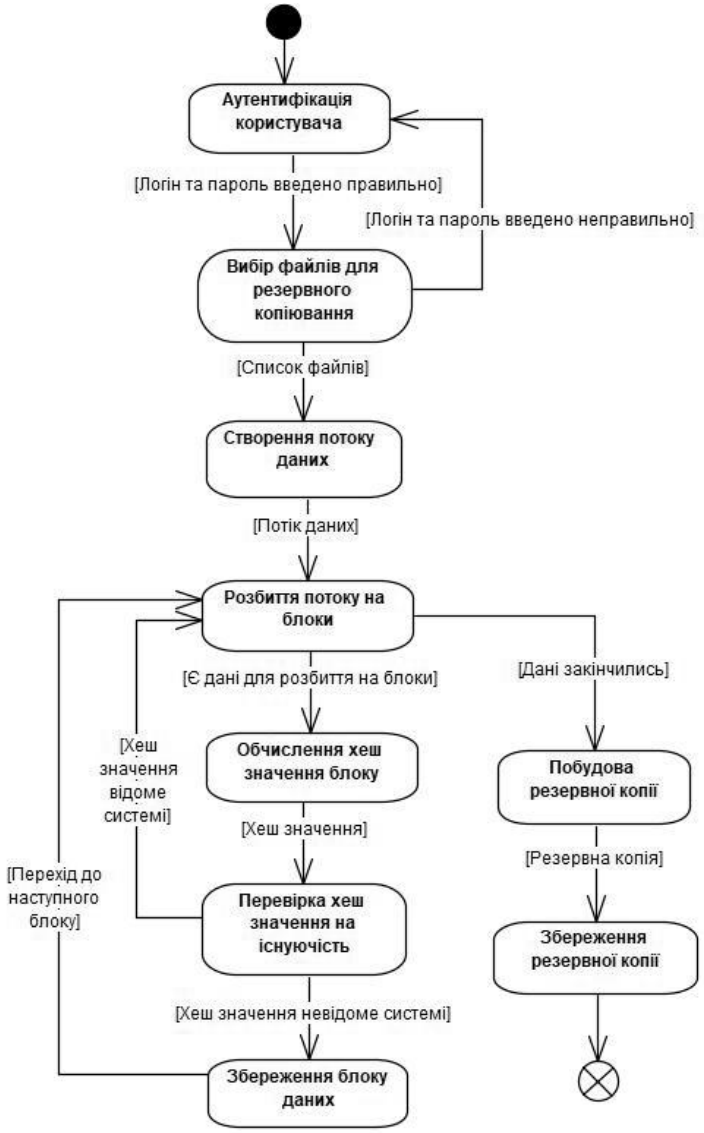

Рис. 3. Діаграма станів резервного копіювання

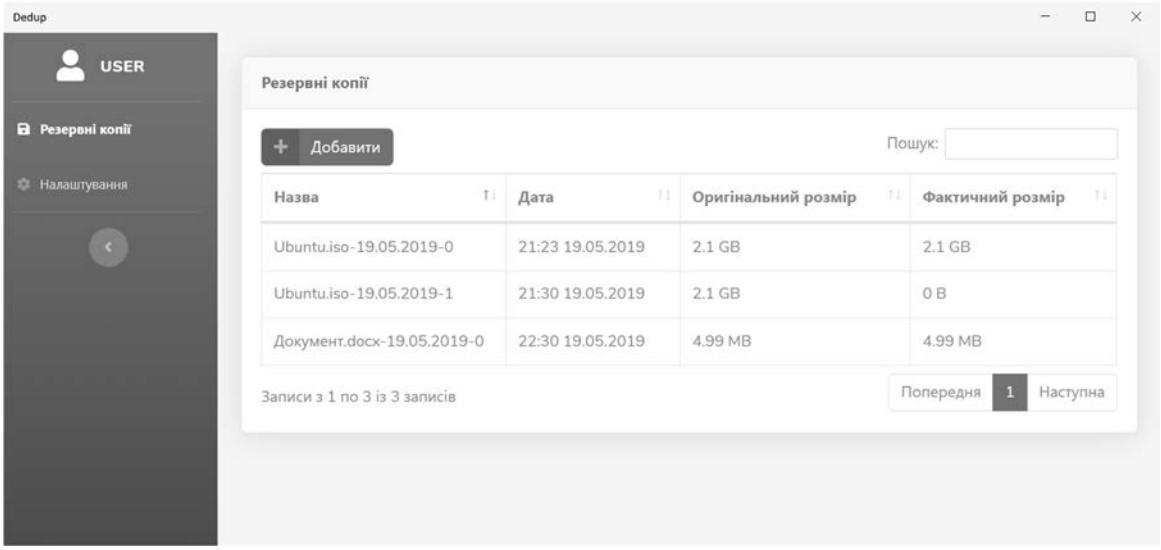

Рис. 4. Вікно резервних копій із доданим документом

```
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)
```

Рис. 5. Журнал повідомлень для першої резервної копії

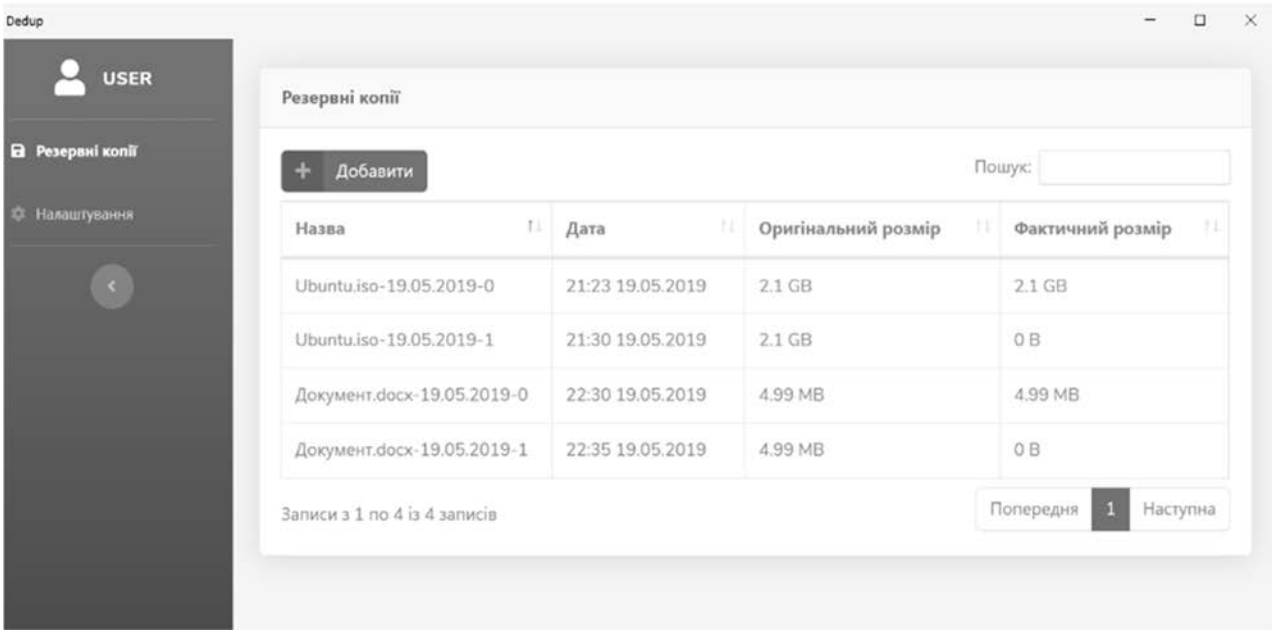

Рис. 6. Вікно резервних копій із вдруге доданим документом

```
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)
```

Рис. 7. Журнал повідомлень для другої резервної копії

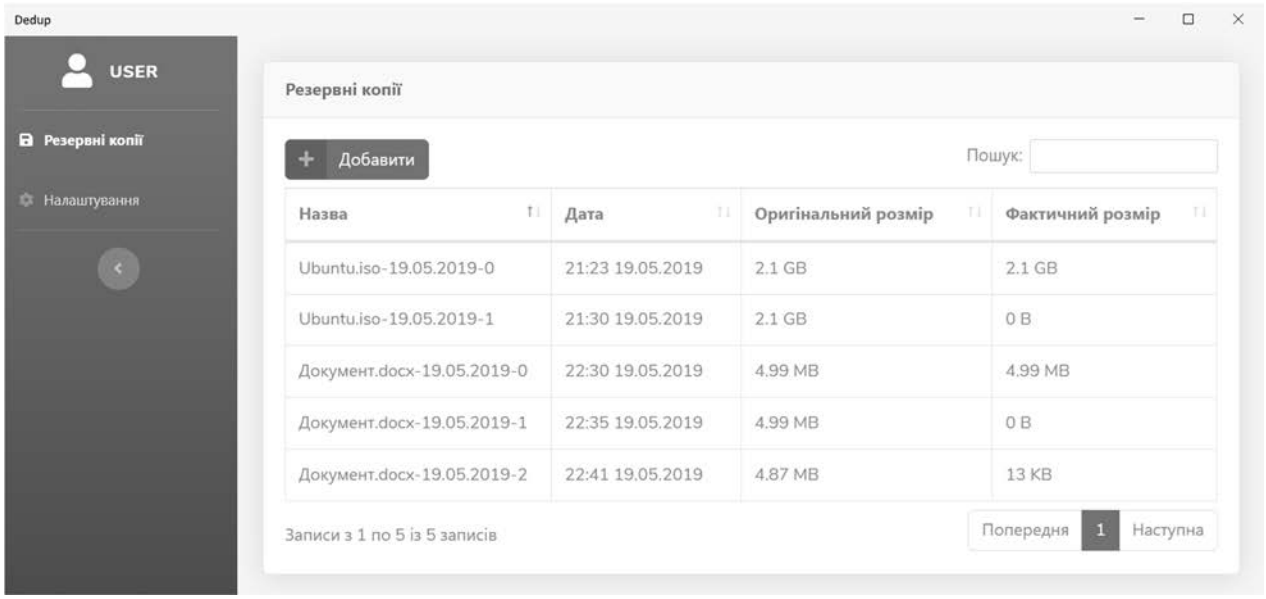

Рис. 8. Вікно резервних копій із втретє доданим документом

На рис. 9 подано останні повідомлення кластера для резервної копії *Документ.docx-19.05.2019-2*. Згідно з цими повідомленнями, додано лише два нові блоки (разом 13 кілобайт), які охоплюють те місце, де було видалено дані з оригінального документа. Зменшилась також загальна кількість байтів та блоків, що відповідає даним рис. 9.

```
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)
```

Рис. 9. Журнал повідомлень для другої резервної копії

Отже, під час тестового запуску файл *Документ* було тричі надіслано для резервного копіювання, копії завдяки дедублікації займають у сховищі даних лише 4,99 МБ + 131 КБ = 5,1 МБ. Якщо би копіювання відбувалось без дедублікації, то фактичний розмір становив би 4,99 МБ + +4,99 МБ + 4,87 МБ = 14,85 МБ.

Висновки

Розроблено метод дедублікації та розподілу даних у хмарних сховищах, на основі якого отримано програмний продукт зі зручним інтерфейсом, що дає змогу виконувати резервне копіювання та відновлення даних. Проведено аналітичний огляд методологічних засад дослідження, проаналізовано підходи до резервного копіювання даних із використанням дедублікації та розподілу даних у хмарних сховищах, виділено їх переваги та недоліки. Детально розглянуто переваги та недоліки сучасних технологій дедублікації даних. Здійснений аналіз довів ефективність проєктування та реалізації інтелектуальної системи дедублікації та розподілу даних у хмарних сховищах. Виконано системний аналіз предметної області досліджуваної роботи. Сформульовано мету функціонування та розроблення системи, призначення та місце функціонування системи, визначено очікувані ефекти від упровадження програмного продукту. Розроблено та детально описано концептуальну модель системи. Наведено деталізовані діаграми прецедентів, переходів станів, послідовностей, компонентів та

класів, що в сукупності дало змогу визначити поведінку системи, виявити та сформулювати необхідні бізнес-процеси.

Список літератури

1. Understanding Data Deduplication. (2018). Retrieved 28, 2019, from https://www.druva.com/understanding-data-deduplication

2. Explaining deduplication rates and single-instance storage to clients. (2008). Retrieved 28, 2019, from https://searchitchannel.techtarget.com/tip/Explaining-deduplication-rates-and-single-instance-storage-to-clients

3. Inline vs. post-processing deduplication appliances. (2008). Retrieved 28, 2019, from https://searchdatabackup.techtarget.com/tip/Inline-vs-post-processing-deduplication-appliances

4. Introduction to Data Deduplication. (2008). Retrieved 28, 2019, from https://www.petri.com/data-deduplication-introduction

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

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

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

8. Using StorReduce for cloud-based data deduplication. (2008). Retrieved 28, 2019, from https://cloud.google.com/solutions/partners/storreduce-cloud-deduplication

9. OpenDedup Overview. (2008). Retrieved 2019, from https://opendedup.org/odd/overview/

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

11. Rolling hash, Rabin Karp, palindromes, rsync and others. (2008). Retrieved 28, 2019, from https://www.infoarena.ro/blog/rolling-hash

12. Vysotska, V., Chyrun, L., & Lytvyn, V. (2016). Methods based on ontologies for information resources processing. LAP Lambert Academic Publishing.

13. Vysotska, V., & Shakhovska, N. (2018). Information technologies of gamification for training and recruitment. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

14. Висоцька, В. А. (2008). Особливості проєктування та впровадження систем електронної комерції.

15. Vysotska, V., & Lytvyn, V. (2018). Web resources processing based on ontologies. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

16. Vysotska, V. (2018). Tekhnolohiyi elektronnoyi komertsiyi ta Internet-marketynhu. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

17. Vysotska, V. (2018). Internet systems design and development based on Web Mining and NLP. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

18. Vysotska, V. (2018). Computer linguistics for online marketing in information technology: monograph. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

19. Lytvyn, V., Vysotska, V., Wojcik, W., & Dosyn, D. (2017). A method of construction of automated basic ontology. *In Computational linguistics andintelligent systems (COLINS 2017).* National Technical University “KhPI”.

20. Lytvyn, V., Vysotska, V., Chyrun, L., Smolarz, A., & Naum, O. (2017). Intelligent system structure for Web resources processing and analysis. *In Computational linguistics andintelligent systems (COLINS 2017).* National Technical University “KhPI”.

21. Berko, A., Vysotska, V., & Chyrun, L. (2014). Features of information resources processing in electronic content commerce. *Applied Computer Science*, 10.

22. Берко, А. Ю., Висоцька, В. А., & Рішняк, І. В. (2008). Методи та засоби оцінювання ризиків безпеки інформації в системах електронної комерції.

23. Vysotska, V., & Chyrun, L. (2013). Web Content Processing Method for Electronic Business Systems. *International Journal of Computers & Technology*, 12(2), 3211–3220.

24. Висоцька, В. А., Чирун, Л. Б., & Чирун, Л. В. (2011). Моделювання етапів життєвого циклу комерційного web-контенту.

25. Берко, А. Ю., Висоцька, В. А., & Чирун, Л. В. (2004). Алгоритми опрацювання інформаційних ресурсів в системах електронної комерції.

26. Vysotska, V., & Chyrun, L. (2011). Commercial Web Content Lifecycle Model.

27. Берко, А., & Висоцька, В. А. (2009). Проектування навігаційного графу web-сторінок бази даних систем електронної контент-комерції.

28. Берко, А. Ю., & Висоцька, В. А. (2009). Семантична інтеграція неповних та неточних даних. *Системи обробки інформації*, (7), 93–98.

29. Берко, А. Ю., & Висоцька, В. А. (2007). Моделі та методи проектування інформаційних систем електронної комерції. *Автоматизированные системы управления и приборы автоматики*, (138).

30. Алєксєєва, К. А., Берко, А. Ю., & Висоцька, В. А. (2015). Управління Web-ресурсами за умов невизначеності. *Технологический аудит и резервы производства*, (2 (2)), 4–7.

31. Vysotska, V., & Chyrun, L. (2014). Designing features of architecture for e-commerce systems [Electronic resource]. *MEST Journal*, 2(1), 57–70.

32. Vysotska, V., & Chyrun, L. (2014). Set-theoretic models and unified methods of information resources processing in e-business systems. *Applied Computer Science*, 10.

References

1. Understanding Data Deduplication. (2018). Retrieved 28, 2019, from https://www.druva.com/understanding-data-deduplication

2. Explaining deduplication rates and single-instance storage to clients. (2008). Retrieved 28, 2019, from https://searchitchannel.techtarget.com/tip/Explaining-deduplication-rates-and-single-instance-storage-to-clients

3. Inline vs. post-processing deduplication appliances. (2008). Retrieved 28, 2019, from https://searchdatabackup.techtarget.com/tip/Inline-vs-post-processing-deduplication-appliances

4. Introduction to Data Deduplication. (2008). Retrieved 28, 2019, from https://www.petri.com/data-deduplication-introduction

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

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

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

8. Using StorReduce for cloud-based data deduplication. (2008). Retrieved 28, 2019, from https://cloud.google.com/solutions/partners/storreduce-cloud-deduplication

9. OpenDedup Overview. (2008). Retrieved 2019, from https://opendedup.org/odd/overview/

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

11. Rolling hash, Rabin Karp, palindromes, rsync and others. (2008). Retrieved 28, 2019, from https://www.infoarena.ro/blog/rolling-hash

12. Vysotska, V., Chyrun, L., & Lytvyn, V. (2016). Methods based on ontologies for information resources processing. LAP Lambert Academic Publishing.

13. Vysotska, V., & Shakhovska, N. (2018). Information technologies of gamification for training and recruitment. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

14. Vysotska, V. (2008). Osoblyvosti proektuvannya ta vprovadzhennya system elektronnoyi komertsiyi.

15. Vysotska, V., & Lytvyn, V. (2018). Web resources processing based on ontologies. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

16. Vysotska, V. (2018). Tekhnolohiyi elektronnoyi komertsiyi ta Internet-marketynhu. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

17. Vysotska, V. (2018). Internet systems design and development based on Web Mining and NLP. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

18. Vysotska, V. (2018). Computer linguistics for online marketing in information technology: Monograph. Saarbrücken, Germany: LAP LAMBERT Academic Publishing.

19. Lytvyn, V., Vysotska, V., Wojcik, W., & Dosyn, D. (2017). A method of construction of automated basic ontology. *In Computational linguistics andintelligent systems (COLINS 2017).* National Technical University “KhPI”.

20. Lytvyn, V., Vysotska, V., Chyrun, L., Smolarz, A., & Naum, O. (2017). Intelligent system structure for Web resources processing and analysis. *In Computational linguistics andintelligent systems (COLINS 2017).* National Technical University “KhPI”.

21. Berko, A., Vysotska, V., & Chyrun, L. (2014). Features of information resources processing in electronic content commerce. Applied Computer Science, 10.

22. Berko, A., Vysotska, V., & Rishnyak, I. (2008). Metody ta zasoby otsinyuvannya ryzykiv bezpeky informatsiyi v systemakh elektronnoyi komertsiyi.

23. Vysotska, V., & Chyrun, L. (2013). Web Content Processing Method for Electronic Business Systems. *International Journal of Computers & Technology*, 12(2), 3211–3220.

24. Vysotska, V., Chyrun, L., & Chyrun, L. (2011). Modelyuvannya etapiv zhyttyevoho tsyklu komertsiynoho web-kontentu.

25. Berko, A., Vysotska, V., & Chyrun, L. (2004). Alhorytmy opratsyuvannya informatsiynykh resursiv v systemakh elektronnoyi komertsiyi.

26. Vysotska, V., & Chyrun, L. (2011). Commercial Web Content Lifecycle Model.

27. Berko, A., & Vysotska, V. (2009). Proektuvannya navihatsiynoho hrafu web-storinok bazy danykh system elektronnoyi kontent-komertsiyi.

28. Berko, A., & Vysotska, V. (2009). Semantychna intehratsiya nepovnykh ta netochnykh danykh. Systemy obrobky informatsiyi, (7), 93–98.

29. Berko, A., & Vysotska, V. (2007). Modeli ta metody proektuvannya informatsiynykh system elektronnoyi komertsiyi. *Avtomatyzyrovannye systemy upravlenyya y prybory avtomatyky*, (138).

30. Alekseeva, K., Berko, A., & Vysotska, V. (2015). Upravlinnya Web-resursamy za umov nevyznachenosti. *Tekhnolohycheskyy audyt y rezervy proyzvodstva*, (2 (2)), 4–7.

31. Vysotska, V., & Chyrun, L. (2014). Designing features of architecture for e-commerce systems [Electronic resource]. *MEST Journal*, 2(1), 57–70.

32. Vysotska, V., & Chyrun, L. (2014). Set-theoretic models and unified methods of information resources processing in e-business systems. *Applied Computer Science*, 10.

METHOD OF DATA DEDUPLICATION AND DISTRIBUTION IN CLOUD WAREHOUSES DURING DATA BACKUP

Bohdan Rusyn[1], Liubomyr Pohreliuk[2], Victoria Vysotska[3], Mykhailo Osypov[4]

[1, 2] Karpenko Physico-Mechanical Institute of the NAS Ukraine,
[3, 4] Information Systems and Networks Department, Lviv Polytechnic National University,
[1] b.rusyn.prof@gmail.com, ORCID: 0000-0001-8654-2270
[2] liubomyr@inoxoft.com, ORCID: 0000-0003-1482-5532
[3] victoria.a.vysotska@lpnu.ua, ORCID: 0000-0001-6417-3689
[4] mykhailo.osypov@gmail.com, ORCID: 0000-0002-6611-724X

An intellectual system of data deduplication and distribution in cloud storage facilities was developed. The resulting software has a user-friendly interface that allows you to backup and restore data. An analytical review of the methodological principles of the research is carried out, existing approaches to data backup with the use of data deduplication and distribution in cloud storage are analyzed, their advantages and disadvantages are highlighted. The advantages and disadvantages of modern data deduplication technologies are considered in details. This analysis has proved the efficiency of the design and implementation of the intellectual system of data deduplication and distribution in cloud storage. A systematic analysis of the subject domain is performed. The purpose of functioning and development of the system, purpose and place of functioning of the system is formulated, the expected effects from the introduction of the software product are determined. A conceptual model of the system has been developed and described in detail. The detailed diagrams of precedents, states of transitions, sequences, components and classes are given, which together allowed to determine the system's behavior, to define and formulate the necessary business processes. It is analyzed the disadvantages and advantages of using different approaches and the effective methods of solving problems are selected: hybrid deduplication at the block level, data splitting on the basis of Rabin's digital imprint, data distribution based on the hash values of the duplication units, and the use of the distributed index. During the analysis of task solutions, the Rust programming language for writing a client part, Scala programming language for the server part, Akka for distributed computing management and Amazon S3 as cloud storage are selected. The intellectual system of deduplication and distribution of data in cloud storage is developed, the software description is described, the steps for the user's operation are considered. The testing of the work of the designed system is carried out and several control samples were given, the results are analyzed.

Key words: data deduplication, data sharing, cloud environment, cloud computing, Rabbin algorithm, data hashing, hybrid deduplication.