Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

магістр
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему: Організація хмарної платформи «розумного міста» для уникнення повторів колекцій даних

Виконав: студент VI курсу групи СТм-61

спеціальності (напряму підготовки) 126
Інформаційні системи та технології
(шифр і назва спеціальності (напряму підготовки))

(підпис) Панасюк І.В. (прізвище та ініціали)

Керівник (підпис) Пасічник В.В. (прізвище та ініціали)

Нормоконтроль (підпис) Мацюк О.В. (прізвище та ініціали)

Рецензент (підпис) Михалик Д.М. (прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Організація хмарної платформи «розумного міста» для уникнення повторів колекцій даних // Дипломна робота освітнього рівня «Магістр» // Панасюк Ігор Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кафедра комп’ютерних наук, група СТм-61 // Тернопіль, 2019 // С. , рис. – , табл. – , кресл. – , додат. – , бібліогр. – .

Ключові слова: ДЕДУБЛІКАЦІЯ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, ІНТЕРНЕТ, ІНТЕРФЕЙС, ПЛАТФОРМА, РОЗУМНЕ МІСТО, СЕРВЕР, ХМАРНІ ОБЧИСЛЕННЯ.

Дипломна робота присв’ячена організації хмарної платформи «розумного міста» побудованої на принципах уникнення повторів колекцій даних. В першому розділі дипломної роботи досліджено інноваційні проекти класу «розумне місто». Виконано огляд та проаналізовано способи зменшення вартості зберігання даних в інноваційних проектах «розумних міст». Розкрито зміст інформаціної технології «Дедуплікація» даних в хмарних проектах «розумних міст».

В другому розділі дипломної роботи розглянуто подано порівняння способів «уникнення повторів» застосовних в проектах «розумних міст». Запропоновано архітектуру проектованої хмарної платформи «розумного міста» для уникнення повторів колекцій даних. Проаналізовано ключові особливості файлової системи «OpenDedup» з автоматичним об’єднанням дублікатів даних. Досліджено архітектуру екосистеми «Hadoop» в контексті її використання для проектів «розумних міст». Подано огляд «HBase» для проектів «розумних міст» та описано реалізацію методу уникнення повторів.

ANNOTATION

Cloud platform of “smart house” organization aimed at avoiding datasets repetition // Master's degree work // Igor Panasyuk // Ivan Puliuyi Ternopil National Technical University, Department of Computer Information Systems and Software Engineering, Department of Computer Science, STm-61 group // Ternopil, 2019 // P. , Fig. - , Table. - , Chair. - , Add. - , Biblio. - .

The diploma thesis is devoted to the organization of the cloud platform "smart city", built on the principles of avoiding repetition of data collections. The first section of the thesis examines innovative projects of the "smart city" class. The review and analysis of ways to reduce the cost of data storage in innovative smart cities projects has been completed. Content of information technology "Deduplication" of data in cloud projects of "smart cities" is revealed.

The second section of the thesis deals with the comparison of ways of avoiding repetitions applicable in projects of "smart cities". The architecture of the designed smart city cloud platform is proposed to avoid repetition of data collections. Key features of the OpenDedup file system with automatic duplication of data are analyzed. The architecture of the Hadoop ecosystem is explored in the context of its use for smart cities projects. An HBase overview of smart cities projects is provided and implementation of the avoidance method is described.

Keywords: DEDUBLICATION, INFORMATION TECHNOLOGIES, INTERNET, INTERFACE, PLATFORM, SMART PLACE, SERVER, Cloud Computing.

ПЕРЕЛІК ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – Прикладний програмний інтерфейс.

JNI (англ. Java Native Interface) – Нативний Java-інтерфейс.

JVM (англ. Java Virtual Machine) – Віртуальна машина Java.

GFS (англ. Google File System) – Файлова система Google.

FUSE (англ. Filesystem in Userspace) – Файлова система в просторі користувача.

SDFS (англ. Software Defined File System) – Файлова система, визначена програмним забезпеченням.

б (укр. байт, англ. byte) – одиниця вимірювання обсягу цифрової інформації, яка зазвичай містить вісім бітів і представлена двійковим числом.

Кб (укр. Кілобайт) – 1Кб=1 024 б.

Мб (укр. Мегабайт) – 1Мб=1 024 Кб.

Гб (укр. Гігабайт) – 1Гб=1 024 Мб.

Тб (укр. Терабайт) – 1Тб=1 024 Гб.

ОС – Операційна система.

ПАК – Програмно-алгоритмічний комплекс.

ПК – Персональний комп’ютер.

Пб (укр. Петабайти) – 1Пб-1 024 Пб.

Зб (укр. Зетабайт) – 1Зб=1 024 Тб.

ЗМІСТ

ВСТУП

Бурхливе зростання обсягів міських даних і концентрація «розумного міста» приводять до зростання вимог до витрат простору, спричинених дублюванням даних, та викликаних появою і розвитком технологічної надмірності даних. В даний час дослідницькі роботи щодо надмірності міських даних досягли серії цінних результатів з точки зору збільшення надмірності, оптимізації продуктивності і забезпечення надійності та ефективно сприяла застосуванню цієї технології.

В сервіс-орієнтованої системи хмарного зберігання великомасштабна концентрація призначених для проектів класу «розумне місто» даних робить непотрібною надмірність даних і створює нові проблеми для реалізації інформаційних технологій.

Метою уникнення повторів даних є усунення надлишкових даних в системі зберігання по всьому світу, включаючи внутрішні надлишкові дані між файлами, в той час як традиційне стиснення даних може тільки виключити внутрішню надлишковість інформації у файлі.

Актуальність теми. Провівши аналіз поточного стану досліджень було виявлено ряд аспектів які потрібно додатково вивчити. Наприклад, як ефективніше використовувати внутрішні характеристики даних, пропонувати більш ефективні технології уникнення повторів, як підвищити надійність систем уникнення повторів і як поліпшити дедуплікацію даних Продуктивність технології, інтеграція різних існуючих технологій в проектах «розумних міст», підвищення гнучкості, масштабованості і адаптованості технології уникнення повторів є актуальним напрямком сучасних досліджень.

Мета і задачі дослідження. Підвищення якості сервісів інформаційно-технологічної платформи «розумного міста» та зменшання витрат на оренду її хмарних обчислювальних ресурсів.

Для досягнення даної мети необхідно вирішити такі завдання:

– Провести огляд та проаналізовано способи зменшення вартості зберігання даних в інноваційних проектах «розумних міст».

– Дослідити використання «MapReduce», «HDFS» і «HBase» в «Hadoop» для реалізації алгоритму «Dedupe» і зберігати більш ефективні «Великі дані» для уникнення повторів.

– Розробити архітектуру проектованої хмарної платформи «розумного міста» для уникнення повторів колекцій даних.

– Запустити кластер «Hadoop» в «Docker»-контейнерах для потреб «розумних міст».

Об'єкт дослідження. Підсистема для хмарного зберігання та опрацювання даних в проектах класу «розумне місто».

Предмет дослідження. Методи та засоби зберігання, оптимізації та аналітичного опрацювання даних в міському середовищі.

Методи дослідження. Методи аналітичного опрацювання наукових публікацій та джерел. Методи системного аналізу. Методи видобування даних. Методи системного та логічного програмування. Методи дедублікації.

Наукова новизна одержаних результатів. Описано процес запуску «spark RDD» в середовищі «Spark» для проектів класу «розумне місто».

Практичне значення одержаних результатів. Реалізовано метод уникнення повторів для потреб проектів класу «розумне місто».

Апробація результатів магістерської роботи проведена на двох наукових конференціях з публікацією тез доповідей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інноваційні проекти та «розумні міста»

Концепція «розумного міста» реалізується в складному міському середовищі, включаючи множину складних інфраструктурних систем, поведінку людей, технології, соціальні та політичні структури, економіку, тощо (див. рисунок 1.1) [1]. Проекти «розумних міст» надають науково-обгрунтовану методику керування міськими компонентами та підсистемами, наприклад, ресурсними мережами, енергетикою, інформаційними ресурсами та муніципальними послугами [2].

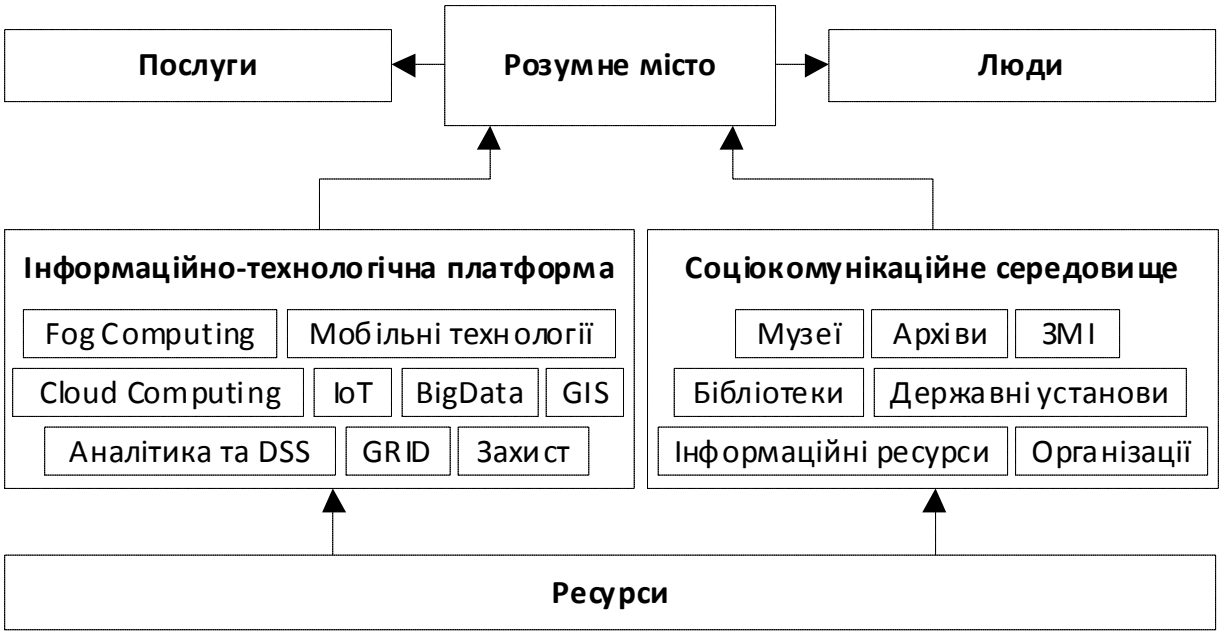

Рисунок 1.1 – Інформаційно-технологічна платформа та соціокомунікаційна концепція «розумного міста»

Є дві складових концепту «розумне місто» [3]. З одного боку, технологічно орієнтована інформаційна та комунікаційна платформа, яка забезпечує ключові обчислювальні алгоритми та сервісні сценарії і забезпечує інтеграцію вбудованих в міське середовище пристроїв [4, 5, 6]. З

іншого боку, соціокомунікаційна концепція, яка фокусується на можливостях, створених для реалізації нового, заснованого на знаннях соціуму та економіки [7], [8].

Мета муніципальних проектів подібного класу в підвищенні ефективності управління ресурсами і якості надаваних міських послуг за рахунок впровадження сучасних інформаційних та комунікаційних технологій у практично всі сфери міського буття та суспільства [9].

За допомогою сучасних інформаційних та комунікаційних технологій системи «розумних міст» забезпечать потужну, інтелектуальну та гнучку підтримку міського населення [10]. На рисунку 1.2 подано напрями формування концепту «розумне місто» [11].

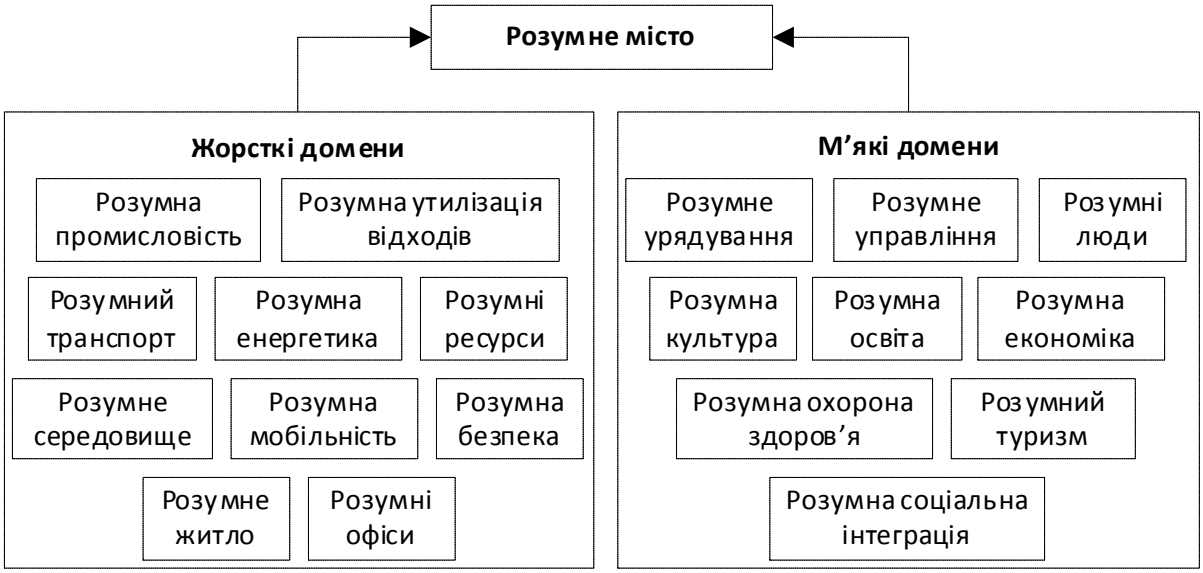

Рисунок 1.2 – Напрями формування концепту «розумне місто»

Дані, створені міськими структурними компонентами, облікуються давачами інтегрованими в міське середовище [12]. Муніципальні бездротові сенсорні мережі використовуються у багатьох промислових та сервісних застосунках, таких як моніторинг здоров'я, інтелектуальні програми для «розумних будинків», моніторинг водних ресурсів та навколишнього середовища [13]. Оперативний аналіз даних в проектах «розумних міст»

передбачає широкий спектр заходів, спрямованих на відслідковування, передачу, перетворення, зберігання та аналітичне опрацювання інформаційних потоків щодо про забруднення навколишнього середовища, погоди, накопичення та утилізації відходів, водопостачання, ресурсів та енергоносіїв, міських подій та інцидентів із використанням даних, що ініціюються громадянами та обробляються Інтернет-пристроями в режимі реального часу [14]. При цьому доцільно проводити видобування та перетворення соціальних даних, пов'язаних з міським буттям. Поєднання даних з фізичних пристроїв-давачів та соціальних джерел сприятиме формуванню повнішшої картини про міські процеси та ефективнішому застосуванню статистичних та аналітичних методів [15].

«Розумне місто» є сучасною інформаційно-технологічною та соціокомунікаційною концепцією яка визначена як місце або місто, яке прагне ефективніше використовувати доступні ресурси для підвищення якості життя та продуктивності муніципальних послуг надаваних жителям та гостям міста, при одночасному зниженні експлуатаційних витрат [16], [17]. Мета муніципальних проектів подібного класу в підвищенні ефективності управління ресурсами і якості надаваних міських послуги за рахунок впровадження сучасних інформаційних та комунікаційних технологій у практично всі сфери міського буття та суспільства [18].

Є дві складових концепту «розумне місто» [19]. З одного боку, технологічно орієнтована інформаціійна та комунікаційна платформа, яка забезпечує ключові обчислювальні алгоритми та сервісні сценарії і забезпечує інтеграцію вбудованих в міське середовище пристроїв [20, 21, 22]. З іншого боку, більш соціальна концепція, яка фокусується на можливостях, створених для реалізації нового, заснованого на знаннях соціуму та економіки [23]. Спільним мостом між ними є можливість ефективного використання даних, пов’язаних з міською діяльністю, їх комплексний аналіз з метою генерації нових інформаційних кортежів, призначених для

оптимізації процесів міського функціонування. Інтернет пристрої (IoT) розглядається як одна з ключових інформаційних технологій для забезпечення цих функцій [24], [25].

Давачі та сенсори розгорнуті в кожному домені «розумного міста», є основними джерелами даних для гетерогенної генерації інформаційних наборів (див. рисунок 1.3).

Інформація, отримана через сенсори та давачі, збирається за допомогою пристроїв IoT підключених до комунікаційних мереж. Смарфони, підключені до мобільних мереж GSM/3G/4G використовуються для збору міських даних. Для агрегації ріноманітних інформаційних наборів міських даних доцільно використовувати сучасні технологій бездротового зв'язку (GSM/3G/4G, LTE, Wi-Fi і т.п.).

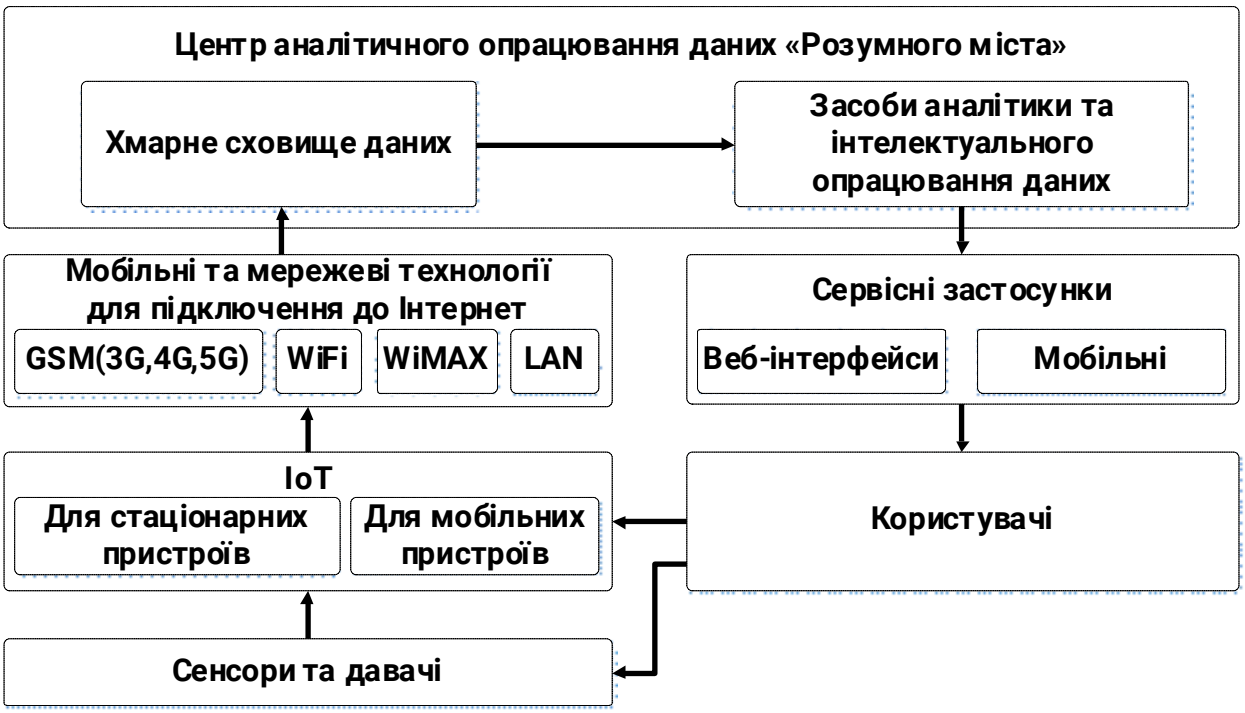

Рисунок 1.3 – Інформаційні потоки «розумного міста»

Зібрані дані обробляються та аналізуються за допомогою центу аналітичного опрацювання даних розумного міста, архітектура якого розгорнута на хмарній платформі та використовує хмарне сховище даних.

Поєднуючи дані з різних доменів «розумного міста», ця архітектура призначена для покращення сервісних характеристик «розумних» міських застосунків.

Давачі, пов'язані з різними складовими проектів класу «розумне місто», створюють великі за обсягом набори даних, які в даний час використовуються недостатньо ефективно. Завдяки використанню сучасної інфраструктури заснованої на основі інформаційних та комунікаційних технологій, згенерована неоднорідна інформація консолідується та обробляється засобами аналітики та інтелектуального опрацювання даних. В проектах «розумних міст» експлуатації мільярди вбудованих в міську інфраструктуру Інтернет-пристроїв (IoT).

Міжгалузеве планування. Оскільки ініціативи «розумних міст» охоплюють широкий спектр галузей, пов'язаних з розвитком та вирішують найважливіші міські проблеми, їх розвиток має стати продуктом міждисциплінарного планування. Незважаючи на те, що «розумні міста» розглядають горизонтальні проблеми соціального, економічного та технологічного розвитку, фахівці з містобудування частіше не включаються у їхні проекти [26]. Такої практики можна уникнути з метою формування повного підходу до міських проблем в цілому та уникненні стратегічного дефіциту.

Явні та дієздатні стратегічні рамки. Також виявляється явна стратегічна структура, що відображає чітко визначену архітектуру інтелектуальних міських ініціатив та розглядає стратегічні пріоритети та взаємодоповнюваність, як характеристику «розумного міста». Очевидно, що явне пояснення сприяє економічній ефективності, легкості реалізації, потенціалу подальшого збільшення масштабів та низьким рівнем інвестиційних ризиків. Отже, питання, які необхідно вирішити за допомогою стратегії «розумного міста», потрібно ретельно вивчати, оцінювати та визначати пріоритетом, а також інтелектуальні інвестиції в місто слід

розглядати як з точки зору їх довгого, так і короткотермінового значення [27, 28]. З іншого боку, містам слід уникати занадто широкого залучення окремих ініціатив.

Розвиток людського і соціальногокапіталу міст. Перспективи розвитку людського та соціального капіталу міст також є характеристикою «розумності» в контексті міста. Знання, інтелект і творчість є стовпами людського і соціального капіталу. Таким чином, основними складовими «розумного міста» вважаються інформовані, освічені та залучені громадяни, висока якість життя та створення «цивільного» простору вважаються основними складовими розумного міста [29]. Крім того, технології базовані на розвику знань і навпаки; два з них разом породжують міський розвиток та допомагають освоювати розумні міста. Згідно з Гіффінгером [30] «розумні люди» – це ті, хто має рівень кваліфікації та схильність до навчання протягом усього життя; є гнучкими, творчими та космополітично-відкритими; і займатися громадським життям. Також було помічено, що існуючі реалізації «розумних міст» з потужною базою людського капіталу з часом стають розумнішими[31].

Заохоченя підприємництва в «розумних містах». Підприємництво також є визначальною характеристикою розумної та конкурентоспроможної економіки, котра являє собою дуже актуальну особливість стратегічного планування розвитку «розумного міста», оскільки вважається необхідною для сталого розвитку. У розумних містах ця ціль переважно реалізується за рахунок розвитку професійного середовища, пропонуючи сучасні послуги новим та розширенням бізнесу. Ці середовища виховують соціальні та підприємницькі інновації, залучають творчих, талановитих та кваліфікованих працівників, а також служать випробувальним майданчиком для інноваційних ділових моделей «продуманих» та «зелених» технологічних продуктів. Фізичні характеристики можуть відігравати вирішальну роль у макеті та успішності включених послуг.

Глобальне співробітництво та Інтернет. Глобальне співробітництво та мережеві зв'язки також є характеристиками «розумних» міст, оскільки міста, які мають хороші можливості спостерігати, навчатися та співпрацювати, особливо з сусідніми містами, як очікується, матимуть найбільшу користь від економії за масштабами та масштабами. Переваги партнерства та співпраці з іншими містами та громадами включають: обмін знаннями та досвідом, економію на масштабах, об'єднання спільних ресурсів та обмін інфраструктурою та взаємодоповнюваність у слабких та сильних сторонах та спільне вирішення питань проблеми [32]. Крім того, існує вже декілька мереж співпраці між містами та однолітків для міст, які стикаються з подібними проблемами. Деякі області, які можуть мати взаємні інтереси для міст і можуть бути придатними для співпраці, включають житло, управління ресурсами, інфраструктуру, охорону здоров'я та безпеку.

Локально адаптовані стратегії. Ще однією характеристикою «розумних» міських стратегій є їх здатність адаптуватися до місцевих умов, тобто розглянути місцеві характеристики та ідентичність місця, а також місцеві проблеми, потреби та можливості. Хоча в багатьох випадках помилково передбачається, що потреби міст від'єднані від їх фізичної обстановки, фізичні зручності місця, його людей та їх культура значно залежать від стійкості міста. Крім того, кожне місто знаходиться на іншому етапі розвитку та має різні потреби, таким чином стає «розумним» в одному місті, може не мати такого ж значення в іншому, що робить рішення, що підходять для всіх, не має значення. Крім того, місцеві характеристики можуть стати порівняльною перевагою для міст, що приваблюють підприємства з низьким рівнем доступу, інвесторами, туристами та капіталом. Тому перед розробкою стратегії розвитку розумного міста важливо подивитися, що вже існує та як його можна.

Універсальний підхід. Підхід до участі, який залучає зацікавлених сторін до знизу вгору у плануванні та впровадженні інтелектуальних міських

проектів, також є визначальною характеристикою розумних міських стратегій. Очевидно, підхід до знизу вгору у формуванні державної політики не є новим зовсім.3 Проте, в умовах розумного міста, користувачі міста, будь то громадяни, підприємці або громади, можуть бути задіяні в режимі реального часу та на великій основі , набравши різні ролі. Платформи Web 2.0, розумні пристрої та мережі забезпечують безпрецедентну можливість широкого залучення користувачів та кодування вхідних даних та інформації, що дозволяє вдосконалити операції та відкривати нові канали для соціального діалогу та громадських інновацій. Цей аспект "розумних" міст неодноразово цитується як основний елемент успішної стратегії розвитку міських міст [33].

Координація зверху в низ. Координація "зверху вниз" також відіграє центральну роль у розвитку розумного міста. розумні міські підприємства, оскільки великі та дорогі проекти, що впливають на все суспільство, є як сильно етичними, так і політичними, потребують короткострокових політичних інтересів у контексті довгострокових інтересів громади [34]. У цьому сенсі лідери повинні бути здатними натхненню прагнення до економічної, соціальної та екологічної стійкості, охоплюючи зміну ролі та захищаючи розумне уявлення про місто але вони також повинні бути добре сформовані для створення довгострокових екосистем знань, які сприяють співпраці між урядом, промисловістю, містами та громадянами [35].

1.2 Огляд способів зменшення вартості зберігання даних в інноваційних проектах «розумних міст»

Розглянемо ключові етапи зменшення вартості зберігання даних в інноваційних проектах «розумних міст»:

1. Регулярна перевірка процесів з метою виявлення надмірності. Регулярний перегляд процесів обміну файлами та інформацією, щоб знайти

будь-яку дубльовану або надлишкову інформацію. Зберігання даних в тонких і релевантних цілях допомагає гарантувати, що не відбуваєтья володіння надлишковими хмарними ресурсами. Одним з аспектів перевірки є визначення того, чи планується розширення або додавання нових областей для міської хмарної стратегії. Якщо це так, слід поступово збільшувати пропускну здатність хмари для обліку цього зростання.

2. Жорстка лінія для зберігання документів. Деякі муніципальні адміністрації та компанії використовують дуже оптимізований підхід до збереження корпоративної та міської інформації. Існує множина способів публікації контенту, і якщо можна отримати інформацію з інших джерел, не потрібно створювати резервні копії всього. Це дозволяє муніципалітетам та компаніям мінімізувати обсяги використаних хмарних ресурсів і зберігати тільки найважливіші дані. Цей метод значно знижує вартість.

3. Використання контейнерів та розподіл ресурсів на вимогу. Сучасні системи моніторингу та управління хмарною інфраструктурою легко розширюються без суттєвих витрат. Використання контейнерів та розподіл ресурсів на вимогу – один із способів управління хмарною пропускною спроможністю без необхідності підписання довгострокових контрактів. Але для перетворення застарілих міських програмних сервісів та застосунків потрібна значна кількість часу та досвіду їх реорганізації,

4. Розташування даних та використання класифікації для відповідності. Один з ключів до оптимізації робочого навантаження і підвищення ефективності програмно-алгоритмічних рішень в проектах класу «розумне місто» – це зіставлення розташування даних з прецедентами їх використання. З метою пошуку додатків та даних, які можна використовувати найбільш ефективно в хмарі або в міському центрі опрацювання даних. Муніципальним установам заважають декілька перешкод, які заважають переносимості даних. Реалізовані міські системи зберігання часто

використовують різні протоколи або мови, що ускладнює взаємне переміщення даних.

При використанні дедублікації даних в цій реалізації прототипу проекту класу «розумне місто» за зберігання кожного Тб файлів потрібно буде заплатити близько «30$» в місяць. При цьому кожен терабайт трафіку вартуватиме приблизно «90$» а «POST/GET»-запити до сервера оплачуватимуться додатково. Пробна версія хмарної платформи безкоштовна. Після реєстрації нові клієнти «AWS» отримують:

– «5 Гб» стандартного сховища «Amazon S3»;
– «20 000» «GET»-запитів;
– «2 000» «PUT»-запитів;
– «15 ГБ» вихідного трафіку щомісяця протягом одного року.

1.3 «Дедублікація» даних в хмарних проектах «розумних міст»

Дедуплікація даних – це інформаційна технологія, за допомогою якої виявляються і виключаються надлишкові дані в муніципальному дисковому сховищі. На даний час доступні два способи уникнення повторів:

– Онлайн дедублікація може здійснюватися безпосередньо в момент запису даних на диски в момент кешування даних.

– Офлайн дедублікація може бути реалізована у формі «постпроцесу», у фоновому режимі після запису даних на диск.

Для уникнення повторів добре підходять:

– Резервні копії.
– Віртуальні машини.
– Будь-які дані з великою кількістю повторюваних блоків.

Для уникнення повторів погано підходять:

– Рисунки.
– Музика.

– Відео.
– Стиснені дані.

1.4 Висновок до першого розділу

В першому розділі дипломної роботи освітнього рівня «Магістр» досліджено інноваційні проекти класу «розумне місто». Виконано огляд та проаналізовано способи зменшення вартості зберігання даних в інноваційних проектах «розумних міст». Розкрито зміст інформаціної технології «Дедуплікація» даних в хмарних проектах «розумних міст».

2 УСУНЕННЯ ПОВТОРІВ ДАНИХ ДЛЯ HDFS ПРОЕКТІВ КЛАСУ «РОЗУМНЕ МІСТО»

2.1 Порівняння способів «уникнення повторів» застосовних в проектах «розумних міст»

Розглянемо плюси «оффлайн»-уникнення повторів та мінуси «онлайн»-уникнення повторів в проектах «розумних міст»:

– Можна використовувати ефективніші та точніші обчислювальні алгоритми виявлення дублікатів міських даних.

– Не потрібно здійснювати компроміси, щоб не перевантажувати роботу центрального процесора інформаційної системи і не знижувати продуктивність роботи муніципальних системи зберігання в момент виконання процедур дедублікації.

– Можна аналізувати та обробляти значно більші обсяги даних.

– Можна робити дедублікацію тоді і там, коли це зручно.

Розглянемо детальніше недоліки «офлайн» уникнення повторів та переваги «онлайн» уникнення повторів. При записі сильно дублікованих даних доведеться спершу виділити місце для запису, заповнити його дублікатами, і лише потім, в ході процесу уникнення повторів, «90%» всього дискового простору буде звільнено. Як наслідок:

– Підвищується загальна вартість сервісу за рахунок використання трафіку для «90%» дублікованих даних.

– Використання додаткових запитів до даних для порівняння.

– Використання додаткових запитів на видалення.

«Онлайн» дедуплікація дешевша у використанні, хоча вона вкрай повільна. Зручно та ефективно використовувати «офлайн» дедуплікацію, але дорожче.

2.2 Архітектура проектованої хмарної платформи «розумного міста» для уникнення повторів колекцій даних

В дипломній роботі пропонується використовувати «офлайн» дедуплікацію з тимчасовим сховищем:

– Дані з клієнта записуються в тимчасове сховище. Не «S3», а наприклад «HDFS».

– Запускається «постпроцесна» дедублікація сховища.

– Дані дедублікуються.

– Дедубліковані дані завантажуються в «S3»-сховище.

– Відбувається очищення тимчасового сховища.

Таким чином, досягається оптимальне збалансоване співвідношення між двома підходами котре приводить до мінімізації вартості та до:

– Збільшення продуктивності, завдяки тому що процедуру дедублікації можна виконувати будь коли.

– Збільшення швидкості функціонування інформаційної системи інтегрованої в проект класу «розумне місто».

2.3 Файлова система «OpenDedup» з автоматичним об’єднанням дублікатів даних

«SDFS» призначено для використання переваг продуктивності та масштабованості об’єктної файлової системи з оптимізованою для зберігання даних технологією уникнення повторів. В результаті «Opendedup»/«SDFS» може видаляти більше «1Рб» дубльованих даних та підтримувати понад «3Тб» даних на Гб пам’яті з розміром блоку «128Кб», виконує вбудовану дедуплікацію з швидкістю «290Мб/с» з високою сукупною продуктивністю процедур введення-виведення. Підтримка «VMware», «Xen» та «KVM» разом з можливістю видалення даних розміру блоку «4Кб».

2.3.1 Ключові особливості «opendedup»

Дозволяє дедублікацію великих обсягів даних з високими показниками надійності. Може зберігати до семи екземплярів кожного блоку. Дозволяє збереження даних:

– локально;

– у мережі з декількома вузлами;

– в хмарі «Amazon S3» або «Azure».

В хмарі зберігаються тільки унікальні блоки даних.

Підтримувані розміри блоків:

– Змінний, котрий використовується не для кластерів. Використовується алгоритм «Rabin Window Borders».

– Постійний розмір блоку.

Дозволяє проводити дедуплікацію для блоків розміром «4Кб» з використанням хеш таблиці. Кешується в оперативній пам'яті. Використовуються алгоритми:

– Murmur3_128 – за замовчуванням.

– Tiger16.

– Tiger24.

Для зберігання всіх хешів в оперативній пам'яті потрібні блоки пам'яті розміром розділу/ блоку «25 байт». При цьому відбувається періодичне видалення невикористовуваних блоків.

2.3.2 Термінологія «opendedup»

«SDFS Volume» – змонтований дедублікований, оскільки містить один «Dedup File Engine» і одну «Fuse Based File System». «Dedup File Engine» – сервіс на стороні клієнта, всередині якого змонтованого «SDFS Volume», який управляє діяльністю на рівні операцій з файлами, зокрема читанням, записом та видаленням.

«Fuse Based File System» – файлова система користувальницького рівня, яка використовується для подання розміщених всередині «Dedup Engine» файлів. Використовується як окремий том файлової системи.

«Dedup Storage Engine» – сервіс на стороні сервера, який зберігає і видобуває блоки дедублікованих даних.

2.3.3 Концепція «Opendedup» в проектах «розумних міст»

В проектах класу «розумне місто» використовується узагальнена концепція «Opendedup» подана на рисунку 2.1.

Рисунок 2.1 – Концепція Opendedup

Структура «Opendedup»-клієнта подана на рисунку 2.2.

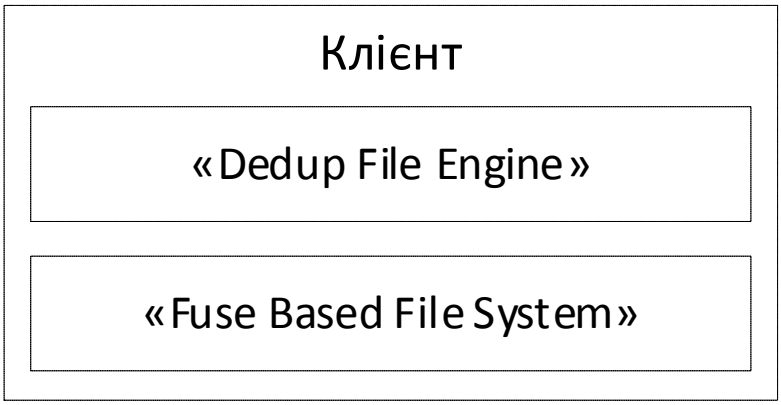

Рисунок 2.2 – Концепція клієнта «Opendedup»

Структура «Opendedup» подана на рисунку 2.3.

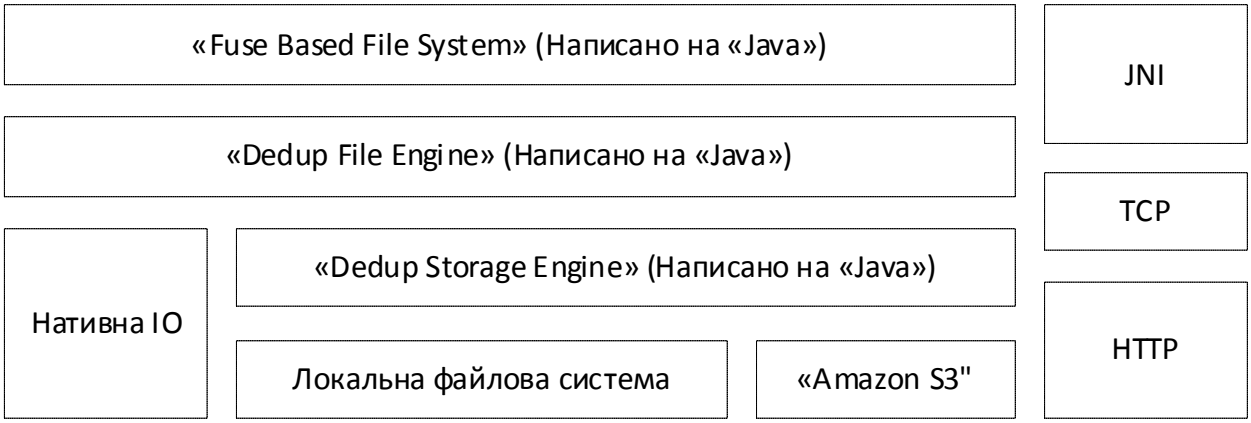

Рисунок 2.3 – Структура «Opendedup»

2.3.4 Фізична архітектура «Opendedup» в проектах «розумних міст»

Подана на рисунку 2.4 схема фізичної архітектури показує, що «SDFS» вміє працювати з великою кількістю клієнтів та розподіленим сховищем, використовуючи для обміну інформацією протоколи «TCP»/«UDP».

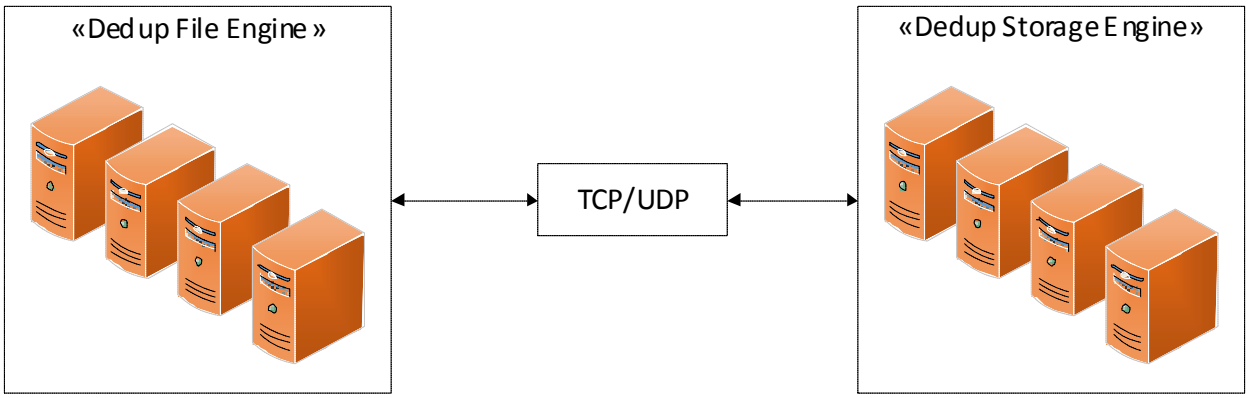

Рисунок 2.4 – Фізична архітектура «Opendedup»

«FUSE». На всіх машинах, які повинні брати участь в зберіганні даних, запускається рушій уникнення повторів. На клієнтських машинах, що використовують сховище даних, запускається сервіс «Dedup File Engine», який підключається до доступних рушіїв уникнення повторів і отримує доступ до сховища. Для взаємодії з «Dedup File Engine» файлова система «Fuse Based File System» використовує механізм «JNI».

«SDFS» – розподілена і розширювана файлова система, що забезпечує вбудовану дедуплікацію. Незважаючи на назву, «SDFS» не є файловою системою в прямому розумінні слова. Це прошарок, написано на мові програмування «Java» і реалізована з використанням механізму для створення файлових систем простору користувача «FUSE». Кожен логічний «SDFS»-файл представляється двома файлами метаданих:

– «MetaDataDedupFile» – зберігає атрибути.

– «Mapping file» – список записів наступного виду.

«SDFS» складається з наступних компонентів:

– «Fuse Based File System» – файлова система рівня користувача, що надає доступ до файлів і каталогів.

– «Dedup File Engine» – сервіс на стороні клієнта, який приймає всі запити на доступ до файлів від файлової системи і відповідає за зберігання метаданих і карти дублікатів, пов'язаних з файлами і каталогами.

– «Deduplication Storage Engine» – серверна сторона, рушій уникнення повторів. Відповідає за зберігання, вилучення та видалення повторюваних даних. Для зберігання даних використовує підпорядковану файлову систему або сховище «Amazon S3», індексуючи блоки за допомогою хеш-таблиці поданої на рисунку 2.5.

dup	hash	reserverd	hash locations(s)
1 byte	hash algo length	1 byte	8 bytes

Рисунок 2.5 – SDFS

«Dedup File Engine» складається з 4 основних компонентів:

– «MetaDataDedupFile» містить всі метадані про фото в змонтованому томі: шлях до файлу, довжина, останній доступ, дозволу.

– «DedupFile» – містить хеш-карту для розташування файлів даних щодо уникнення повторів. Зберігається з використанням постійної клієнтської «Hashtable» швидкої пам’яті.

– «Dedup File Channel» – інтерфейси між «Fuse-J» та іншими командами «Dedup File» для підсистеми вводу-виводу.

– «Writable Cache Buffer» містить фрагмент даних, який читається або записується в рядок. «DedupFile» кешують багато з них під час процедур введення-виведення.

«MetaDataDedupFiles» зберігаються в «MetaFileStore». «MetaFileStore» продовжує використовувати «JDBM». «DedupFiles» зберігаються в «DedupFileStore» незалежно в якості хеш-таблиці на основі окремих дисків. Основні елементи архітектури «Dedup Storage Engine»:

«HashStore» зберігає вміст у списку всіх об’єктів де розташовані дані. Хеш-накопичувач зберігається із застосуванням користувальницької хеш-таблиці. При цьому швидка та ефективна пам'ять «ChunkStore» відповідає за зберігання даних, пов'язаних з визначеним хешем. В даний час в проектах класу «розумне місто» використосується три реалізації «ChunkStores»:

– «FileChunkStore» – зберігає всі дані у одночасно великому файлі.

– «FileBasedChunkStore» – зберігає всі куски як окремі файли.

– «ChunkStore» – зберігає всі дані в «AWS (s3)».

Частина даних передається у базі масивів байтів між об’єктами в «Dedup Storage Engine».

2.4 Архітектура екосистеми «Hadoop» в контексті проектів «розумних міст»

«Hadoop» – це програмне середовище, яке дозволяє розподілене опрацювання великих за обсягом наборів даних, надаючи при цьому надійні, ефективні та масштабовані функції (див. рисунок2.6).

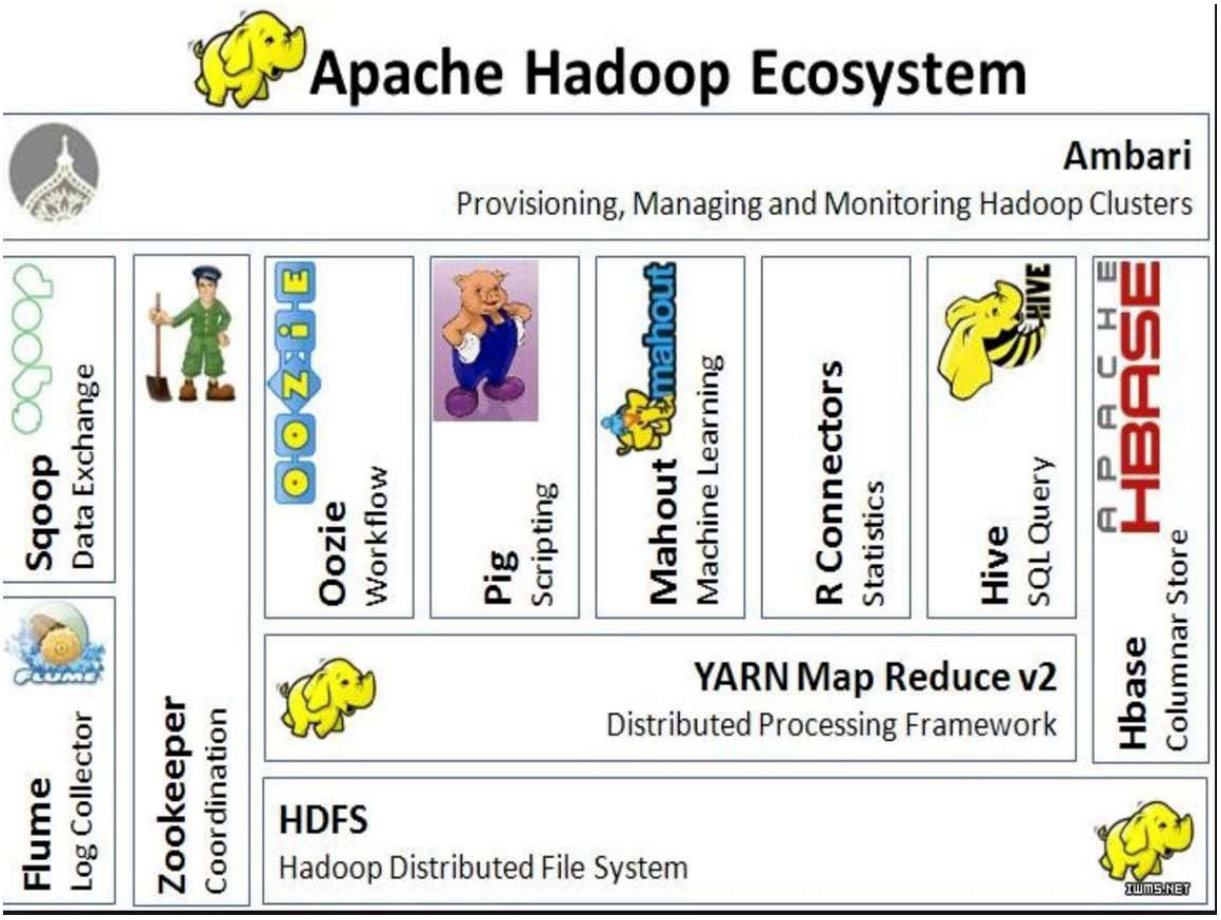

Рисунок 2.6 – Узагальнена структура екосистеми «Hadoop»

Огляд файлової системи «HDFS». Документ «GFS» від компанії «Google», опублікований в жовтні 2003 року. «HDFS» – це клон «GFS» та основа управління зберіганням даних в системі «Hadoop». Це відмовостійка система з високими показниками відмовостійкості, яка виявляє і справляється з апаратними збоями для роботи на недорогому поширеному обладнанні. «HDFS» спрощує моделі узгодженості документів, надаючи можливості доступу до даних з високою пропускною здатністю через потоковий доступ до даних для міських застосунків з великими за обсягом наборами даних.

«Client» – цн розділені файли, доступні до «HDFS», котрі використовують взаємодію з «NameNode» для одержання інформації щодо місцезнаходження файлу та взаємодію з «DataNode» для виконання процедур читання і запису даних (див. рисунок 2.7).

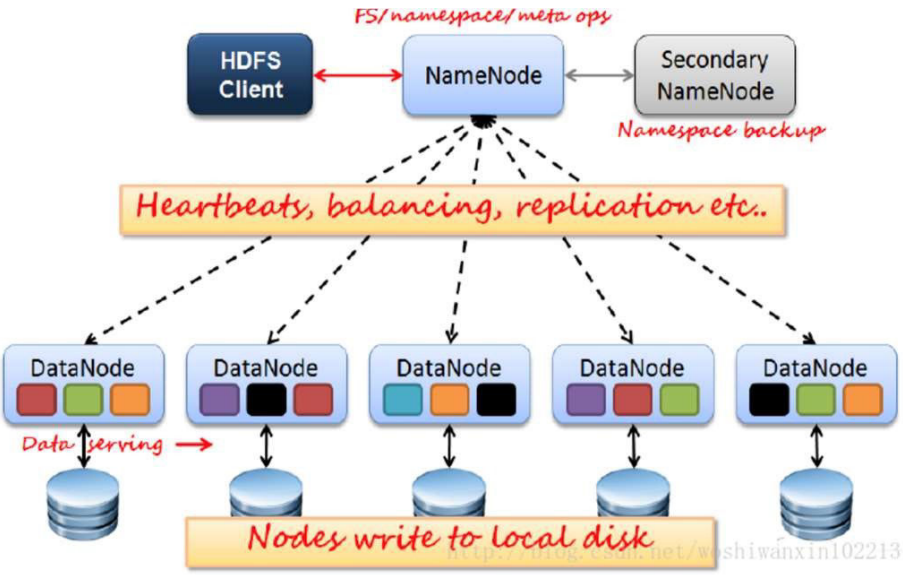

Рисунок 2.7 – Взаємодія складових сутностей «HDFS»

«NameNode» – це головний вузол, котрий управляє інформацією щодо зіставлення простору імен та даних в «HDFS», налаштовуючи політику копіювання та обробляє запити клієнтів. «DataNode» – це «Slave»-вузол, котрий зберігає фактичні дані та подає збережену інформацію в «NameNode».

«Secondary NameNode» – допоміжний «Assist NameNode» для спільного використання свого робочого навантаження. Він періодично об’єднує «fsimage» та «fsedits» і передає його в «NameNode». У разі виникнення надзвичайної ситуації він може допомогти у відновленні «NameNode», але «Secondary NameNode» не є точкою доступу для «NameNode».

2.5 Огляд «HBase» для проектів «розумних міст»

«HBase» – це клон «Google Bigtable», масштабована, високонадійна, високопродуктивна, розподілена і орієнтована на стовпці база даних динамічних схем для структурованих даних. На відміну від традиційних реляційних баз даних, «HBase» використовує модель даних «BigTable»,

зокрема «Enhanced Sparse Sorted Maps (Key/Value)», де ключі складаються з рядків, стовпців та тимчасових міток (див. рисунок 2.8).

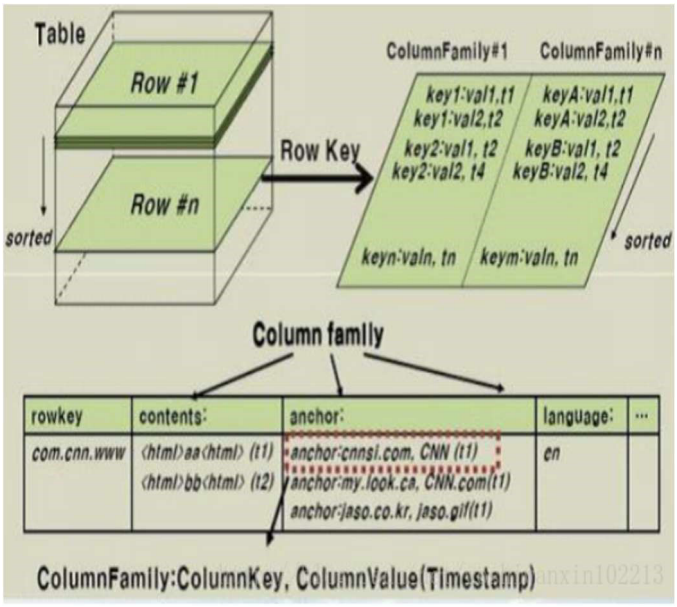

Рисунок 2.7 – Ключі «HDFS»

«HBase» забезпечує випадковий доступ в режимі реального часу з метою читання і запису до великомасштабних даних [36]. Дані, збережені в «HBase», можуть оброблятися за допомогою «MapReduce» [37], який відмінно поєднує зберігання даних і паралельні обчислення:

```
Schema->Table->Column Family->Column->RowKey->TimeStamp->Value
```

Для «HDFS» не існує ефективних методів та засобів уникнення повторів.

2.6 Розроблення архітектури хмарної платформи «розумного міста» для уникнення повторів колекцій даних

В дипломній роботі освітнього рівня «Магістр» пропонується використовувати «оффлайн»-уникнення повторів з тимчасовим сховищем:

– Дані клієнта записуються в тимчасове хмарне сховище «розумного міста», не «S3» а, наприклад, на основі «HDFS».

– Запускається «постпроцесне» уникнення повторів у сховищі.

– Дані уникають повторів.

– Дедубльовані дані завантажуються в «S3».

– Тимчасове сховище очищається.

Таким чином, досягається оптимальне співвідношення між двома підходами.. Один з яких мінімізація вартості і як наслідок:

– збільшення продуктивності при якому можна запускати алгоритм уникнення повторів по мірі необхідності.

– комплексне збільшення швидкості роботи системи.

Другим підходом є традиційний алгоритм уникнення повторів. Вхідні дані розбиваються на блоки по певних алгоритмах та певного розміру. Для кожного блоку обчислюється хеш-функція, для якої використовуються різні алгоритми обчислення. Обчислений хеш порівнюється з існуючими та зберігається в таблицях. Якщо хеш існує, то дані замінюються на посилання.

2.7 Використання «HBase» для зберігання таблиці блоків після процедури дедублювання

2.7.1 Запис даних

Для запису даних використовується «FUSE»-бібліотека:

– Дані розбиваються на блоки, котрі кешуються та записуються спочатку в «FIFO»-буфер [38] розміром якого за замовчуванням «1 Мб», а потім в «flushing» буфер.

– «Flushing»-буфер очищається потоками, які використовуються для обчислення хешу, порівнюють отриманий хеш з уже наявними, підтверджують наявність або успішний запис і записують інформацію в «mapping file» (див. рисунок 2.8).

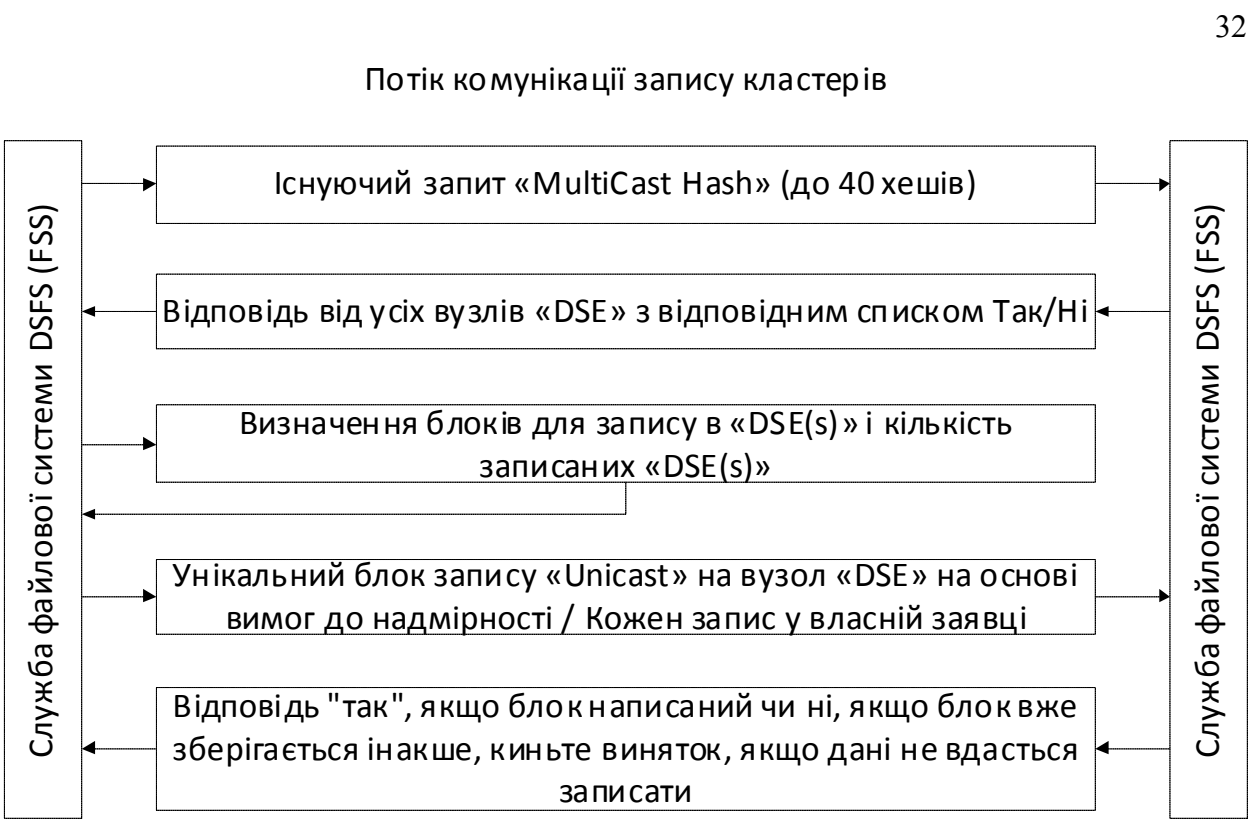

Рисунок 2.8 – Запис даних

2.7.2 Читання даних

Запитувані дані передаються через «fuse»-рівень:

– У сервера запитується блок даних з необхідним хешом.

– Якщо інформацію не знайдено на поточному сервері, тоді вона шукається на іншому до тих пір, поки блок не буде знайдений або список місць розташування блоків вичерпається.

Сучасний тренд в області зберігання даних спрямований на створення розподілених сховищ, які можуть бути добре масштабуються, мають високу надійність, безпекою, продуктивністю і дозволяють зберігати і обробляти колосальний обсяг вхідних даних. Такі розподілені муніципальні сховища даних розміщують в хмарній інфраструктурі, яка в свою чергу має величезний потенціал для створення і використання різних сервісів обслуговування даних.

Одним з важливих аспектів технологій зберігання даних є мінімізація вартості зберігання даних за рахунок таких підходів як архівація, багаторівневе зберігання, «thin-provision» і уникнення дублікації. [39]

2.8 Реалізація методу уникнення повторів даних в проектах класу «розумне місто»

2.8.1 Реалізація алгоритму

Мета дипломної роботи полягає у виборі оптимального стеку інформаційних технологій для організації хмарного сервісу організованого з метою уникнення повторів даних. Для досягнення мети було потрібно вирішити завдання створення конфігурації, архітектури, прототипу сервісу уникнення повторів і настройки випробувального стенду для вимірювання продуктивності створеного прототипу ПАК і вибору найбільш оптимальних складових сервісу.

У дипломній роботі було розглянуто існуючу інформаційну систему «OpenDedup» [40], що дозволяє виробляти зберігання з уникненням повторів даних в хмарі Однак з’ясувалося, що ця система має ряд архітектурних недоліків, пов’язаних із створенням розподіленої конфігурації, яка не дозволяє ефективно використовувати горизонтальне масштабування. Якби алгоритм уникнення повторів був сформований на основі обчислення «hash»-функції для блоків вхідного потоку даних. Вхідний потік даних, позначений блоком 1, на рисунку 2.9, розбивається на блоки по «4-16 Кб» в залежності від вимог завдання і досягнення оптимального рівня уникнення повторів. Потім для кожного блоку даних обчислюється «hash»-функція (блок 2). На основі значення «hash»-функції приймається рішення (блок 3), чи вже в системі існує вхідний блок даних чи ні. Якщо блок не існує, то він записується в таблицю «hash»-значень (блок 4), а якщо існує, то він замінюється посиланням на вже існуючий у сховищі (блок 5).

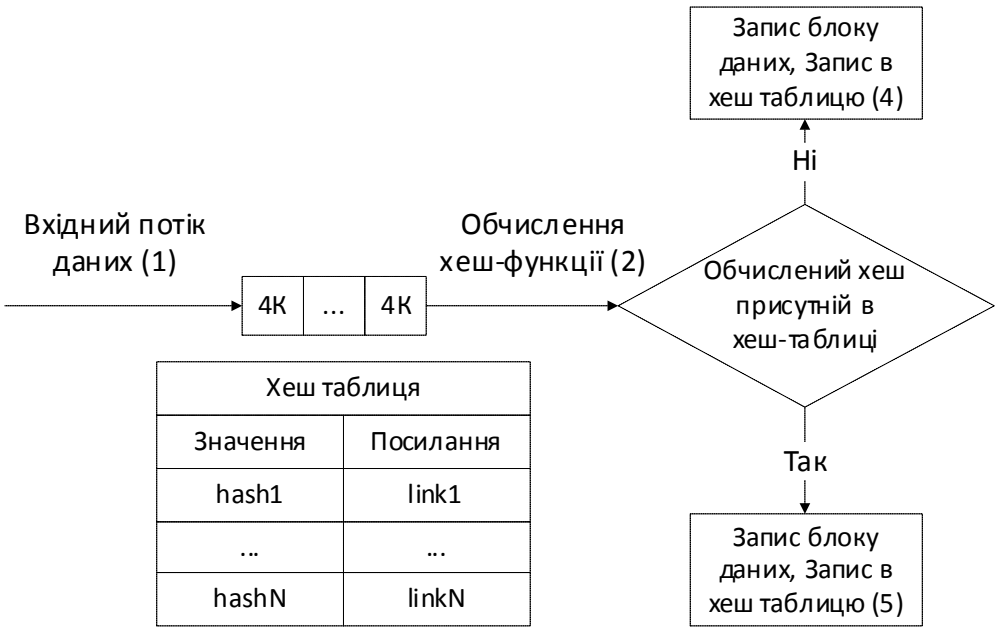

Рисунок 2.9 – Алгоритм уникнення повторів

Хеш-таблиця містить значення «hash»-функції та посилання, які вказують на місцезнаходження відповідних блоків даних. Ключем в такій таблиці є хеш певного блоку даних, а значенням є безпосереднє посилання на сам блок даних в сховищі. Для дослідницького проекту в дипломній роботі подані нижче дві складові мають особливу цінність.

Перша складова – вхідні дані. Предметна область «розумного міста» вхідного потоку даних, самі дані визначають спосіб розбиття вхідних інформаційних наборів на блоки і використовувану для блоку даних «hash»-функція не матиме великого значення. Важливими для проекту є характеристики вхідного потоку: обсяг («Тб/Пб»), швидкість («10-100 Гб/с»), ступінь повторення блоків даних («10% -90%»).

Для абстрагування від типу даних вхідного потоку і джерела даних в інтегрованого в міському середовищі, на якому вимірюється продуктивність роботи сховища «hash»-таблиці, в роботі запропоновано використовувати генерацію «hash»-значень. Для цього створений «hash»-генератор, яким можна задавати вхідні характеристика потоку даних для моделювання різного навантаження в системі «розумного міста», режимів роботи і значень,

які забезпечили б різний набір експериментів для обчислення продуктивності роботи з конкретним муніципальним «in-memory key-value» сховищем.

Друга складова – сховище дедублікованих даних. Існують готові, добре масштабовані рішення, такі як: «Amazon S3» [41], «HDFS» [42], «Cassandra» [43], «MongoDB» [44], а завдання взаємодії з їх «API» також є очевидним для дипломної роботи. Модуль, який відповідає за зберігання дедублікованих даних, зроблений заглушкою «Mock»-об'єкт, замість якого, при необхідності можна інтегрувати конкретну реалізацію хмарного сховища інтегрованого в ПАК «розумного міста». [45]

Найбільшу цінність в даній роботі представляє дослідження можливостей використання розподілених «in-memory key-value» сховищ даних для зберігання «hash»-таблиці, вимір продуктивності читання даних для операції «lookup» і можливості компресії таблиці. Особливу увагу було приділено дослідженню можливості використання «Spark» [46] для реалізації «in-memory» сховища, яке дозволяє не тільки виконувати обчислення над даними в пам'яті, а зберігати при цьому «key-value» таблиці. Були обрані традиційні «in-memory» сховища, зокрема «GemFire» [47], «Redis» [48].

Для кожного з обраних типів сховищ були проведені однотипні експерименти, котрі включали запис «5*105 хешів», кожен з яких займає по «64б». При цьому вимірювався час читання («lookup»). Вимірювання продуктивності було виконано на кластері «ІКС», що містить «64Гб» розподіленої загальної оперативної пам'яті (було налаштовано «4 вузли», у кожного з яких по «16Гб» оперативному пам'яті) [49]. З огляду перелічене, проводилося порівняння по операціях запису і читання заданого числа хешів, а кожен хеш за замовчуванням співставлений з розміром блоку рівним «4Кб» реальних даних, то можна помітити, що порівнювалися системи по вхідному значенню близько «32 Мб» неповторних даних, проте випробовувалися значно більші навантаження на «key-value» сховища. Для одного «1Кб» актуальних даних в пам'яті створюється об'єкт-обгортка, яка займає в кілька

разів більше від реальних даних місце. Це пов’язано з принципами зберігання об’єктів в пам'яті «JVM». [50]

У таблиці 2.1 наведені тимчасові характеристики проведених операції по вимірюванню «lookup» в «hash»-таблицю для перевірки наявності конкретного значення.

Таблиця 2.1 – Результати експериментів щодо практичного випробування реалізації розробленої архітектури

	Обсяг Актуальних даних, «Мб»	Обсяг даних в сховищі, «Мб»	Час «lookup» для «1 hash», мл сек
«Spark»	«32»	«348»	«3210»
«Redis»	«32»	«255»	«0.823»
«GemFire»	«32»	«268»	«0.75»

Можна помітити, що для «Spark» час «look-up» в кілька десятків разів більше, ніж у інших інструментів. Це пов'язано з особливістю «Spark API» для проведення операції «lookup». [51]

Результати дослідження показали, що «Spark» не підходить в якості «key-value» сховища, тому що має занадто великий час пошуку «1 хешу» через архітектурних особливостей реалізації сховища в пам’яті. Але при цьому він дозволяє використовувати операцію «reduceByKey()», яка приймає на вхід набір хеш-значень і показує вагомий виграш у часі в порівнянні з проведенням «look-up» для одного хеш-, котрий сягає десятків разів. На основі операції «reduceByKey()» можна створити алгоритм уникнення повторів, але через його складність реалізації, час його виконання вимагатиме значного більше часу в порівнянні з самою операцією «lookup». Алгоритм поданий на рисунку 2.10.

При невеликій кількості даних «Redis» [52] виконує операцію «lookup» на кластері приблизно за «0.3 мс», з ростом кількості даних «lookup»-час зростає.

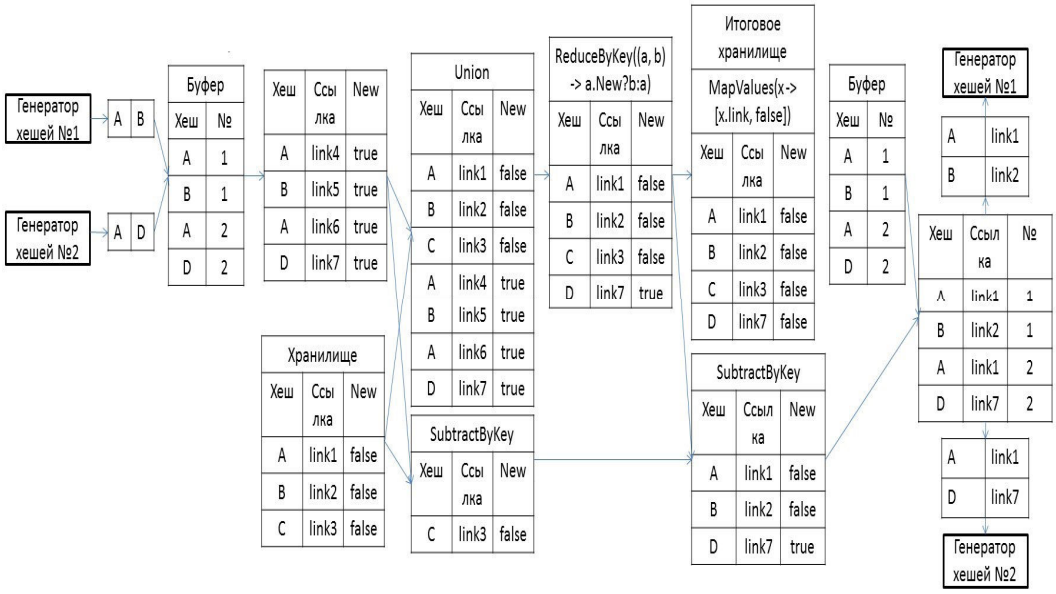

Рисунок 2.10 – Алгоритм уникнення повторів на основі функції «reduceByKey ()»

Між «GemFire» і «Redis» особливої різниці в продуктивності не спостерігається, результати приблизно однакові. Максимальний обсяг даних, який вдалося записати в «GemFire» [53] і «Redis» [54] – кілька «Гб» хешів, що відповідає кільком «Тб» реальних даних.

За результатами досліджень у «Spark» була виявлена пропріонарна залежність часу запису в сховище від кількості вузлів. Для інструментів «Redis» та «GemFire» така проблема відсутня, оскільки збільшення вузлів збільшує час запису на фіксовану часову затримку.

Так як «GemFire» базується на системі взаємопов'язаних «JVM» і використовує в якості контейнера «Java HashMap», то можна помітити, що в середньому «HashMap» читає і записує дані за фіксований час, що вірно для всієї системи «GemFire» в цілому. Це також показує велику перевагу «GemFire» в порівнянні з іншими системами. [55]

Також було відмічено, що «GemFire» використовує мета-дані, що зберігаються в сховищі, для опису моделі даних, які збільшують обсяг реальних даних в кілька разів. Причому до зростання обсягу мета-даних призводить не тільки зростання обсягу реальних даних, але і збільшення

кількості вузлів «GemFire». Було виявлено, що найчастіше один або кілька серверів відмовляються працювати через переповнення «JVM Heap», так як всі дані зберігаються саме там.

В результаті досліджень можна оцінити, який з фреймворків найбільше підходить для «key-value» сховища хешу в системі уникнення повторів. «Spark» створювався для складних обчислень на розподілених високонавантажених системах, однак вимірювання показали, що ефективність роботи в якості «key-value» сховища у «Spark» [56] значно нижча , ніж у інших, призначених для цього, інструментів – «Redis» та «GemFire». Ці інструменти є першокласними рішеннями для завдання розподіленого зберігання хешу в сенсі швидкості роботи, проте використовують надлишкову пам'ять. Крім того, «Redis» не має можливості динамічного розширення системи вузлів, поверх якого він працює. Тому можна зробити висновок, що найкращим з розглянутих рішень для «in-memory key-value» сховища є «GemFire».

2.9 Підготовка та налаштування операційного середовища «Hadoop»

Для створення операційного середовища «Hadoop» використано «Docker» [57]. Сутності «Docker» беруться з контейнера. Для цього використовується наступна метафора. На великому судні товари можна впорядкувати певним чином. Всі види товарів стандартизируются контейнерами, а контейнери не впливають один на одного. Тоді не потрібен спеціальний корабель, який перевозить фрукти і корабель, який спеціалізується на транспортуванні хімічних речовин. Поки ці товари добре упаковуються в контейнерах, можна використовувати великий корабель, щоб переміщувати їх усіх. «Docker», як нова форма віртуалізації, має багато переваг у порівнянні з традиційними методами віртуалізації (див. рисунок 2.11), зокрема:

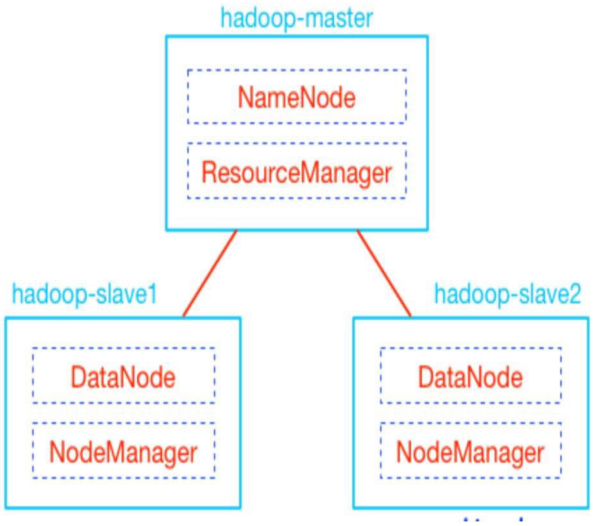

Рисунок 2.11 – Віртуалізація «Hadup»

1. Більш ефективне використання системних ресурсів. Оскільки для контейнера не потрібно апаратна віртуалізація і накладні витрати на запуск повної ОС, «Docker» використовує більше системних ресурсів. Незалежно від того, чи є швидкість виконання додатків, втрата пам'яті або швидкість зберігання файлів, вона більш ефективна, ніж традиційні технології віртуальних машин. [58] Тому, в порівнянні з технологіями віртуальних машин, ідентично конфігурований хост може часто запускати більшу кількість застосунків.

2. Більш швидкий час запуску. Традиційні технології віртуальних машин часто займають кілька хвилин, щоб почати роботу з застосунками. Застосунки-контейнери «Docker», тому що вони запускаються безпосередньо на ядрі хоста, не потрібно запускати повну ОС, тому вони можуть досягти часу запуску в секундах або навіть мілісекундах. Значна економія часу на розробку, тестування і час розгортання.

3. Послідовна операційне середовище. Загальною проблемою в процесі розробки є проблема узгодженості навколишнього середовища. Через непослідовну середовища розробки, середовища тестування та виробничого середовища деякі помилки не були виявлені в процесі розробки. Зображення «Docker» забезпечує повну середу виконання на додаток до ядра. Це

гарантує, що в середовищі контейнера застосунки будуть узгоджено працювати і ця проблема більше не буде виникати.

4. Безперервна доставка і розгортання. Для співробітників «DevOps» найбільш бажаним є одне створення або конфігурація, які можуть працювати в будь-якому місці.

Слід використовувати «Docker» для забезпечення безперервної інтеграції, доставки і розгортання через зображення для користувача застосунків. Розробники можуть використовувати «Dockerfile» для тестування зображень та інтеграції з системою безперервної інтеграції, в той час як співробітники за операціями і обслуговування можуть швидко розгорнуті зображення безпосередньо в робочому середовищі, навіть з системою безперервної доставки та розгортання. [59]

Автоматичне розгортання. Крім того, використання «Dockerfile» для забезпечення прозорості зображення не тільки дозволяє команді розробників зрозуміти середовище виконання програми, але також допомагає команді експлуатації та обслуговування зрозуміти умови, необхідні для запуску програм. Це допомагає розгорнути образ в кращому виробничому середовищі.

5. Проста міграція. Оскільки «Docker» забезпечує узгодженість середовища виконання, міграція додатків спрощується. «Docker» може працювати на багатьох платформах, фізичних чи віртуальних машинах, публічних хмарах, приватних хмарах або навіть ноутбуках. При цьому узгоджуються результати. Таким чином, користувач може легко перенести застосунок, що працює на одній платформі, на іншу платформу, не турбуючись про ситуацію, коли програма не працює належним чином через зміни операційного середовища.

6. Простота обслуговування і розширення. Багаторівнева технологія зберігання і віддзеркалення, яка використовується «Docker», спрощує повторне використання декількох частин застосунка, а також спрощує

обслуговування і оновлення додатків. Також дуже просто розширити зображення на основі основних зображень. Крім того, команда «Docker» разом з різними проектними командами з відкритим вихідним кодом підтримує велику кількість високоякісних офіційних зображень, які можуть бути використані безпосередньо у виробничому середовищі і можуть бути додатково налаштовані в якості основи, що значно знижує витрати на дзеркальне відображення застосунків.

2.10 Висновок до другого розділу

В цьому розділі магістерської роботи подано порівняння способів «уникнення повторів» застосовних в проектах «розумних міст». Запропоновано архітектуру проектованої хмарної платформи «розумного міста» для уникнення повторів колекцій даних. Проаналізовано ключові особливості файлової системи «OpenDedup» з автоматичним об’єднанням дублікатів даних. Досліджено архітектуру екосистеми «Hadoop» в контексті її використання для проектів «розумних міст». Подано Огляд «HBase» для проектів «розумних міст» та описано реалізацію методу уникнення повторів даних в проектах класу «розумне місто». В розділі розроблений алгоритм MapReduce для уникнення повторів великих даних. Описано підготовку та налаштування операційного середовища «Hadoop».

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОВЕДЕНИХ ДОСЛІДЖЕНЬ

3.1 Запуск кластера «Hadoop» в «Docker»-контейнерах для потреб «розумних міст»

Запуск кластера «Hadoop» в «Docker»-контейнерах для потреб «розумних міст» відбувається з використанням команди:

```
zhangyu@zhangyudeMacBook-Pro:~ $sudo docker network create -
p
```

При цьому відбуваються наступні етапи:

1. Видобування Docker-зображення.
2. Клонування «github»-сховища. [60]
3. Створення «Hadoop»-мережі для запущеного контейнера:

```
zhangyu@zhangyudeMacBook-Pro:~ $git clone https://github.com
ster-docker
zhangyu@zhangyudeMacBook-Pro:~ $cd hadoop-cluster-docker
zhangyu@zhangyudeMacBook-Pro:~/hadoop-cluster-docker $sudo ./start-container.sh
Password:
Sorry, try again.
Password:
start hadoop-master container...
start hadoop-slave1 container...
start hadoop-slave2 container...
```

4. Запуск «Hadoop»:

```
hangyu@zhangyudeMacBook-Pro:~ $sudo docker pull kiwe
```

```
root@hadoop-master:~# ./start-hadoop.sh


Starting namenodes on [hadoop-master]
hadoop-master: Warning: Permanently added 'hadoop-master,172.18.0.2' (ECDSA) to the list of known hosts.
hadoop-master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-hadoop-master.out
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.18.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.18.0.4' (ECDSA) to the list of known hosts.
hadoop-slave2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave2.out
hadoop-slave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-hadoop-master.out


starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemanager-hadoop-master.out
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.18.0.4' (ECDSA) to the list of known hosts.
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.18.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave2.out
hadoop-slave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave1.out


root@hadoop-master:~#
```

Етапи побудови «Hbase»: [61]

1. Завантаження «HBase» (подано на рисунку 3.1).

```
wget http://mirrors.hust.edu.cn/apache/hbase/0.98.13/hbase-0.98.13-hadoop2-bin.tar.gz
tar -zxvf hbase-0.98.13-hadoop2-bin.tar.gz
cd hbase-0.98.13-hadoop2
```

Рисунок 3.1 – Завантаження «HBase»

2. Редагування:

«conf/hbase-site.xml»

Результат виконаної операції подано на рисунку 3.2.

```
<property>
        <name>hbase.rootdir</name>
        <value>hdfs://master.kiwenlau.com:9000/hbase</value>
</property>
<property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
</property>
<property>
        <name>hbase.master</name>
        <value>hdfs://master.kiwenlau.com:60000</value>
</property>
<property>
        <name>hbase.zookeeper.quorum</name>
        <value>master.kiwenlau.com,slave1.kiwenlau.com,slave2.kiwenlau.com</value>
</property>
```

Рисунок 3.2 – Приклад редагування «conf/hbase-site.xml»

3. Редагування «conf/regionservers» (див. рисунок 3.3).

```
master.kiwenlau.com
slave1.kiwenlau.com
slave2.kiwenlau.com
```

Рисунок 3.3 – Приклад редагування «conf/regionservers»

4. Редагування «conf/hbase-env.sh» (див. рисунок 3.4).

```
export HBASE_MANAGES_ZK=true
```

Рисунок 3.4 – Приклад редагування «conf/hbase-env.sh»

5. «Scp to slave1» і «slave2», також можна використовувати «docker –v» для монтування того ж каталогу «hbase» на хості (див. рисунок 3.5).

```
scp -r /root/hbase-0.98.13-hadoop2 root@slave1:/root/hbase-0.98.13-hadoop2
scp -r /root/hbase-0.98.13-hadoop2 root@slave2:/root/hbase-0.98.13-hadoop2
```

Рисунок 3.5 – Приклад редагування «conf/hbase-env.sh»

6. Запуск «hbase» (див. рисунок 3.6).

Рисунок 3.6 – Запуск «hbase»

7. Тестування «HBase» (див. рисунок 3.7).

Рисунок 3.7 – Тестування «hbase»

Після запуску та тестування «Hadoop»-кластера в «Docker»-контейнерах для потреб «розумних міст» доцільно виконати запуск «spark RDD» в «Spark».

3.2 Запуск «spark RDD» в середовищі «Spark» для проектів класу «розумне місто»

«Spark RDD» це незмінна колекція розподілених об’єктів, котра може бути ефективно використана для обчислювальних потреб проектів класу «розумне місто». [62] Кожен «RDD» розділений на кілька розділів, які виконуються на різних вузлах кластера. «RDD» може містити об’єкти будь-якого типу написані з використанням «Python», «Java», «Scala» або навіть певні задані користувачем сутності.

Існує два способи створення «RDD», це читання зовнішнього набору даних або поширення колекції об'єктів в програмі драйверів, наприклад, список і набір в програмі драйвера. Ми бачили приклади використання «SparkContext.textFile()» для читання текстового файлу у вигляді рядка «RDD» в попередніх розділах цієї книги.

«Spark» пропонує два способи створення «RDD» – читання зовнішніх наборів даних і розпаралелювання набору в програмі драйвера.

Найпростіший спосіб створення «RDD» – передати існуючу колекцію в програмі методу «parallelize () SparkContext». Цей метод дуже корисний при вивченні «Spark», він дозволяє швидко створити власний «RDD» в оболонці, а потім працювати з цими «RDD». Однак слід зазначити, що цей метод не використовується набагато більше, ніж при розробці прототипів і тестів. Зрештою, цей підхід вимагає, щоб весь ваш набір даних був поміщений в пам'ять машини першим «parallelize()» метод в «Java».

```
JavaRDD<String> lines = sc.parallelize(Arrays.asList("pandas", "i like pandas"));
```

Подамо поетапний опис використаних класів «spark»-програми.

На першому етапі відбувається ініціалізація «Spark» в «Java». В програмному коді поданому нижче продемонстровано найпростіший спосіб створення «SparkContext». Котрий реалізовано простою передачею двох параметрів:

– «URL»-адрес кластера. Інформує «Spark», про адресу для підключення до кластера.

```
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaSparkContext;

SparkConf conf = new SparkConf().setMaster("local").set
JavaSparkContext sc = new JavaSparkContext(conf);
```

В даному випадку використано «local», «this». Таe спеціальне значення дозволяє «Spark» працювати в автономному одиночному потоці без необхідності підключення до кластера.

– Назва програми: У даному прикладі використано «застосунок». При підключенні до кластера це значення може допомогти знайдіть застосунок в інтерфейсі менеджера.

На другому етапі відбувається створення автономного застосунка. Наприклад «Застосунок» для підрахунку слів поточної використаної версії «Java»-коду:

```
// 创建一个Java版本的Spark Context
SparkConf conf = new SparkConf().setAppName("wordCount");
JavaSparkContext sc = new JavaSparkContext(conf);
// 读取我们的输入数据
JavaRDD<String> input = sc.textFile(inputFile);
// 切分为单词
JavaRDD<String> words = input.flatMap(
  new FlatMapFunction<String, String>() {
    public Iterable<String> call(String x) {
      return Arrays.asList(x.split(" "));
    }});
```

Продовження реалізованого програмного коду програмного застосунку:

```
// 转换为键值对并计数
JavaPairRDD<String, Integer> counts = words.mapToPair(
  new PairFunction<String, String, Integer>(){
    public Tuple2<String, Integer> call(String x){
      return new Tuple2(x, 1);
    }}).reduceByKey(new Function2<Integer, Integer, Integer>(){
        public Integer call(Integer x, Integer y){ return x + y;}});
// 将统计出来的单词总数存入一个文本文件,引发求值
counts.saveAsTextFile(outputFile);
```

Вміст першого файлу використаної збірки «Maven» подано в наступному програмному коді:

```
<project>
  <groupId>com.oreilly.learningsparkexamples.mini</groupId>
  <artifactId>learning-spark-mini-example</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <name>example</name>
  <packaging>jar</packaging>
  <version>0.0.1</version>
  <dependencies>
    <dependency> <!-- Spark依赖 -->
      <groupId>org.apache.spark</groupId>
```

Вміст другого файлу використаної збірки «Maven» подано в наступному програмному коді:

```
      <artifactId>spark-core_2.10</artifactId>
      <version>1.2.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
  <properties>
    <java.version>1.6</java.version>
  </properties>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>   <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.1</version>
          <configuration>
            <source>${java.version}</source>
            <target>${java.version}</target>
          </configuration> </plugin> </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

Файли використаної для реалізації прототипу ПАК «розумного міста» збірки «Maven» завершено.

На рисунку 3.8 подано запуск «wordcount» за допомогою «hadoop».

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>data</groupId>
    <artifactId>data</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-core</artifactId>
            <version>1.2.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.7.2</version>
        </dependency>
    </dependencies>

</project>
```

Рисунок 3.8 – Запуск «wordcount» за допомогою «hadoop»

Для введення значень використано команду «input» (див. рисунок 3.9).

```
i love you
i like you
like
you
```

Рисунок 3.9 – Введення значень за допомогою команди «input»

Приклад програмного застосунку для динамічного підрахунку слів версії «Java»-коду подано на рисунку 3.10.

```java
public class WordCount {

    public static class TokenizerMapper extends
            Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable( value: 1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context)
                throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends
            Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
                           Context context) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf,  jobName: "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

Рисунок 3.10 – Приклад програмного застосунку для динамічного підрахунку слів версії «Java»-коду

Для виведення значень використано подану на рисунку 3.11 команду «output».

```
i   2
like    2
love    1
you 3
```

Рисунок 3.11 – Виведення значень за допомогою команди «output»

3.3 Практична реалізація методу уникнення повторів для потреб проектів класу «розумне місто»

На першому етапі опишемо прототип «Хеш-генератора». Модуль «HashGenerator» складається з множини «classeModule». «HashGenerator» складається з множини класів та інтерфейсів. Гнучке рішення для хеш-генератора:

```
«HashGenerator (ByteGenerator byteGenerator, HashLength
hashLength, Destribution destribution)».
```

Можна побудувати «HashGenerator», передаючи наступні компоненти:

1. Інтерфейс «ByteGenerator» – генерація байтів. Цей компонент дозволяє генерувати лише один байт різними способами.

2. Лічильник «HashLength». Цей компонент може бути ініціалізований, але «HashGenerator» повинен мати унікальну назву.

3. Відкритий інтерфейс:

```
«int [] getDistribution (int numberDuplicates, int
numberUniques)».
```

Розподілено отриману кількість дубльованих хешів та кількість унікальних хешів. Він распределяє повтори хешів на кожному унікальному.

За замовчуванням «HashGenerator» використовує стандартний випадковий «ByteGenerator», використовуючи «java.util.Random», дистрибуцію реалізує за допомогою простого алгоритму, який використовує «java.util.Random» і «HashLength» по замовчуванню рівний «32 байтам».

Таким чином, «HashMap» надає новий елемент і отримує елемент, використовуючи інші операції. «HashGenerator» генерує «HashMap», який створює історію в якості маркерів та вказує кількість хешів, для яких цей

ключ взаємодіє з і інтерфейсами. У поданому дослідницькому проекті надано певну гнучкість рішення для хеш-генератора:

```
«HashGenerator (ByteGenerator byteGenerator, HashLength
hashLength, Destribution destribution)».
```

Виконання генератора хешування для проектів класу «розумне місто». Цей клас дозволяє встановити швидкість генерації. Він містить «HashGenerator» всередині і обов'язково потребує ініціалізований для запуску клас «HashGeneratorExecutor», котрий реалізує «Runnable Builder (Consumer <String []> споживач)».

При цьому розширені можливості пошуку довжини хешу, котра за замовчуванням рівна «32 байта», швидкості дублювання, котра за замовчуванням рівна «50%» та часу генерації, котрий за замовчуванням рівний «30 секунд».

Місцеві експерименти виконаємо використовуючи «HashGeneratorExecutor», та порівнюючи при цьому різні методи, що дозволяють задіяти «Spark». При цьому довжина хешу становить «32 байта».

Результати вимірювання швидкості запису по одному хешу за допомогою «Hash Generator Executor» подані в таблиці 3.1.

Таблиця 3.1 – Вимірювання запису по одному хешуванню за допомогою «Hash Generator Executor»

Швидкість генерації, «Мбіт/с»	Час генерації, «сек»	Швидкість запису по одному хэшу, «Мбіт/с»	Швидкість генерації, «Мбіт/с»	Час генерації, «сек»	Швидкість запису по одному хэшу, «Мбіт/с»
1	2	3	1	2	3
1	1	0.52336480387156	01 20	1	1.749934535796657
	2	0.6169703540370237		2	2.195532389957502
	3	0.2995645414795755		3	1.9891131302424566
2	1	0.85856494958810	71 30	1	2.1826251431176955
	2	0.7742907353750926		2	2.1993053294031367
	3	0.7692765648482128		3	2.1728410593522217

Продовження таблиці 3.1

1	2	3	1	2	3
3	1	0.990311976349081	40	1	2.24435875195499
	2	0.9372081539665547		2	2.4677463137553713
	3	0.9342533864137653		3	1.9881169950593145
4	1	1.121455940015520	50	1	2.360550490732735
	2	1.0948056390892946		2	2.5963878567080805
	3	1.1053493718865593		3	2.4204398065140443
5	1	1.1368612503289244	100	1	2.6012853523413126
	2	1.1028768947941253		2	2.5390646896846537
	3	1.2516944075634542		3	2.6759903746588744
7	1	1.3990475383611944	500	1	2.8385382955646346
	2	1.3177132130937212		2	«Waiting more than 30 min»
	3	1.3013699473259334			
10	1	1.3948393422501841		3	-
	2	1.6422814545478017			
	3	1.3721431515872552			

Швидкість запису різного часу генерації має лінійну залежність з невеликим стрибком швидкості (див. рисунок 3.12).

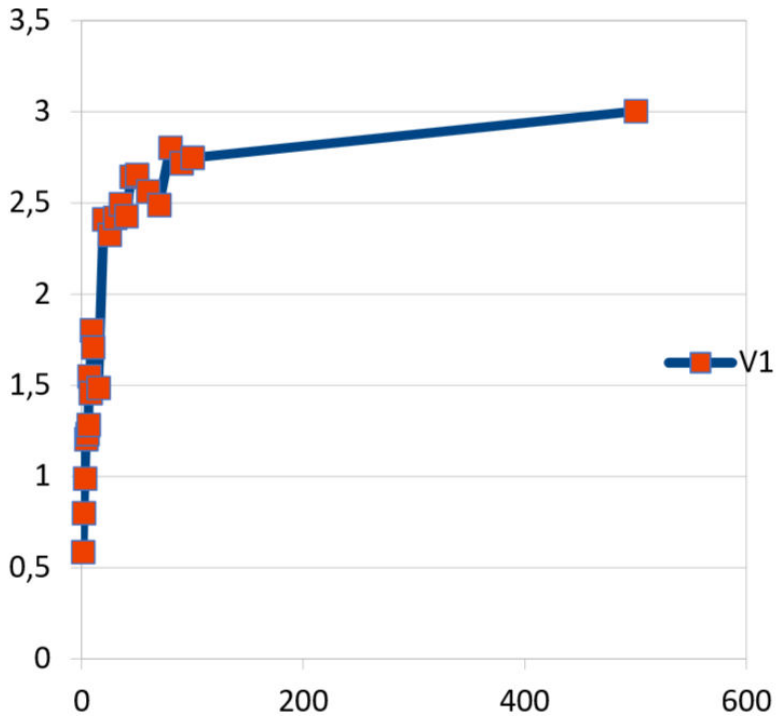

Рисунок 3.12 – Усереднена швидкість запису по одному хешуванню, в залежності від швидкості генерації («Мбіт»/«Мбіт»)

Результати вимірювання швидкості запису всіх хешів за один раз подано в таблиці 3.2.

Таблиця 3.2 – Вимірювання швидкості запису всіх хешів за один раз

Генерація «Мбіт/с»	Швидкість запису всіх хешів за один раз, «Мбіт/с»	Генерація «Мбіт/с»	Швидкість запису всіх хешів за один раз, «Мбіт/с»
1	2,385875606	6	11,24587567
2	5,516546649	7	14,03785483
3	5,987254851	8	14,41358737
4	8,830845634	9	15,38757484
5	9,593978514	10	16,33986486

Графік вимірювання швидкості запису всіх хешів за один раз подано на рисунку 3.13.

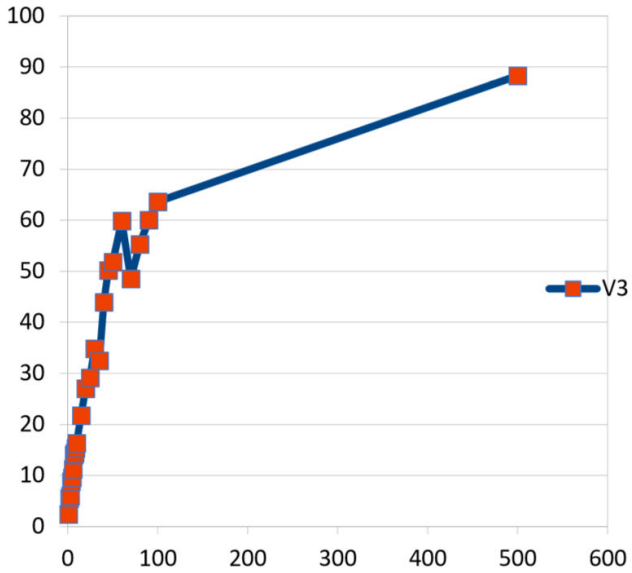

Рисунок 3.13 – Швидкість запису залежностей пропорційно до швидкості генерації («Мбіт»/«Мбіт»)

Результати вимірювання швидкості запису по одному хешу за допомогою пошуку. Цей експеримент досліджувався з використанням «API Java Thread». Кожне завдання, кожен рядок запускається в окремому потоці. Результати експерименту для такого випадку подано в таблиці 3.3 та рисунку 3.14.

Таблиця 3.3 – Вимірювання швидкості запису при перевірці за допомогою пошуку

Швидкість генерації, Мбіт/с	Швидкість запису по одному хешу з використанням пошуку, Мбіт/с	Швидкість генерації, Мбіт/с	Швидкість запису по одному хешу з використанням пошуку, Мбіт/с
1	0,034565344	25	0,017985449
2	0,03478153	30	0,017345231
3	0,032325634	35	0,016253529
4	0,031654876	40	0,01582548
5	0,030874266	45	0,015254548
6	0,029474673	50	0,014925435
7	0,027257295	60	0,017863722
8	0,028692426	70	0,015087662
9	0,026356354	80	0,015276898
10	0,026481034	90	0,015817817
15	0,023187255	100	0,016576892
20	0,020625785		

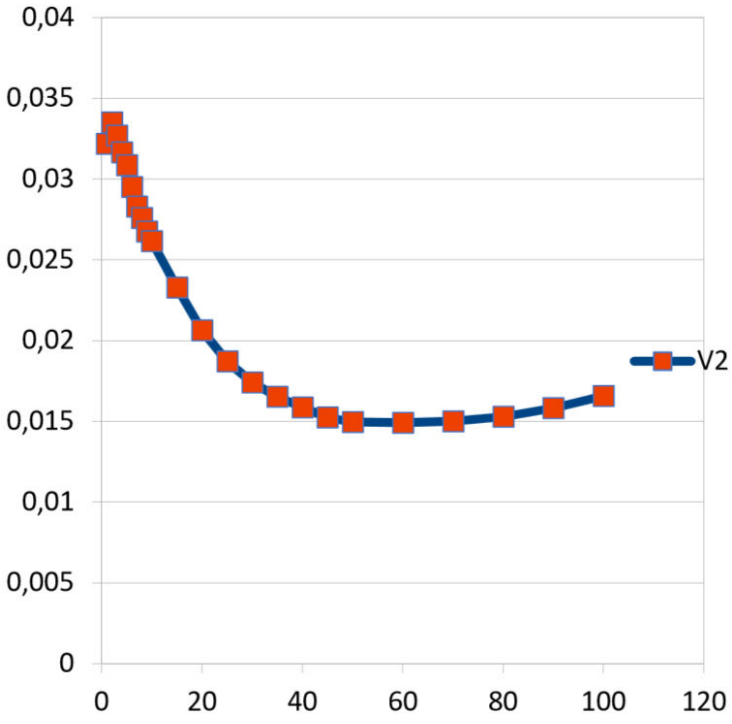

Рисунок 3.14 – Швидкість запису залежностей залежно від швидкості генерації («Мбіт»/«Мбіт»).

На рисунку чітко виражена нелінійна залежність.

Числові результати написання всіх значень хешу протягом одного разу за допомогою процедури «reduceByKey ()» подано в таблиці 3.4. та рисунку 3.15.

Таблиця 3.4 – Вимірювання швидкості запису зі зменшенням по ключу.

Швидкість генерації, «Мбіт/с»	Швидкість запису послідовності хешів з використанням «reduceByKey», «Мбіт/с»	Швидкість генерації, «Мбіт/с»	Швидкість запису послідовності хешів з використанням «reduceByKey», «Мбіт/с»
1	1,034359867	30	4,187589365
2	5,178632318	35	6,634683458
3	3,433458855	40	5,416478153
4	4,487591563	45	7,897356113
5	2,230348896	50	8,034789186
6	2,958784432	60	6,256432291
7	3,243560111	70	9,024836604
8	3,014586763	80	6,057348996
9	3,314034833	90	7,818869247
10	4,648712102	100	7,102345038
15	2,423457291	500	8,087567412
20	5,041247526	1000	9,273599116
25	5,157434385		

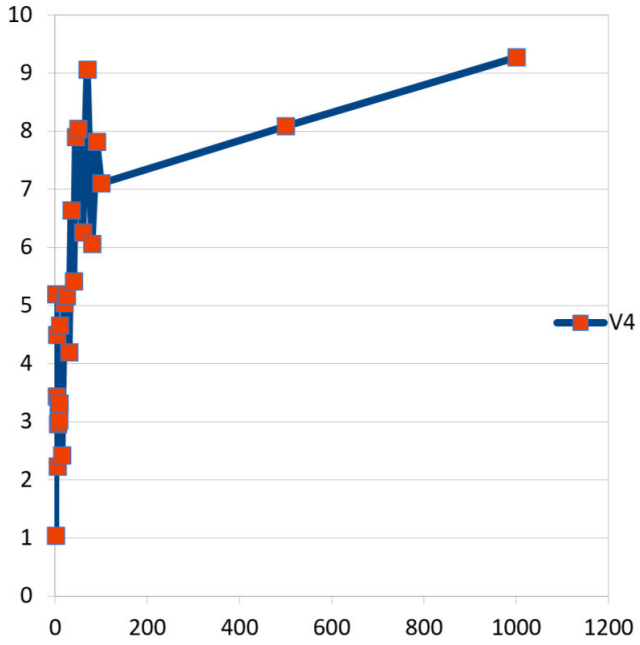

Рисунок 3.15 – Швидкість запису залежностей пропорційно від швидкості генерації («Мбіт»/«Мбіт»)

Графік порівняння всіх методів подано на рисунку 3.16.

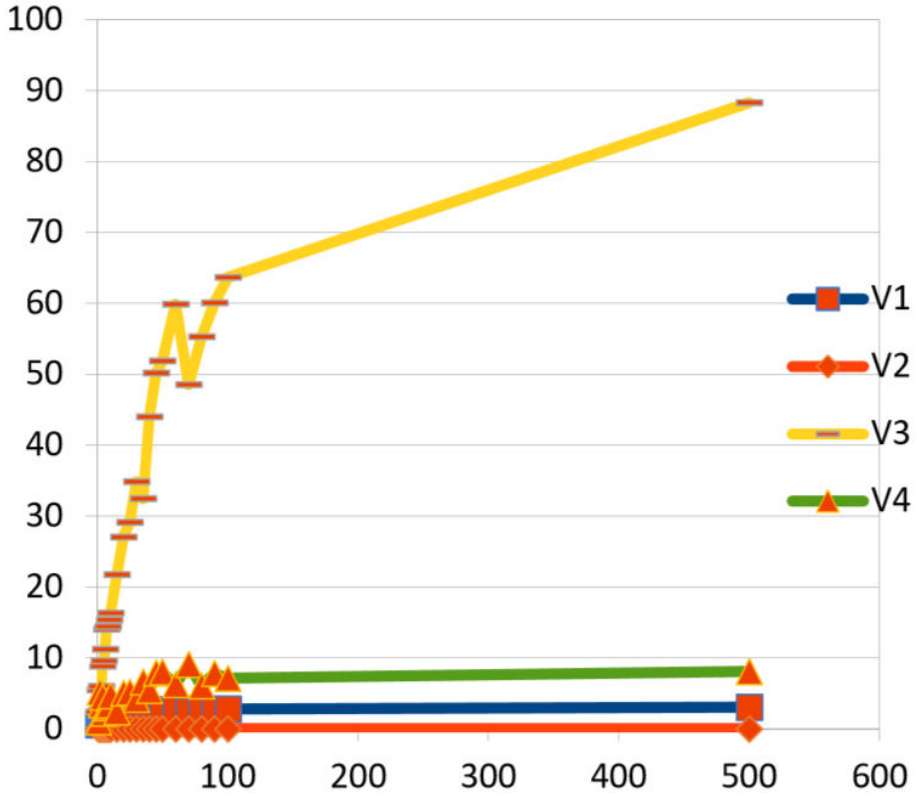

Рисунок 3.16 – Графік порівняння всіх методів запису («Мбіт»/«Мбіт»)

Написання по одному з пошуком має найнижчу швидкість та буде найцікавішим для використання в проектах класу «розумне місто».

3.4 Висновок до третього розділу

В цьому розділі дипломної роботи освітнього рівня «Магістр» побудовано середовище «Hadoop» для прототипу ПАК «розумного міста». Зокрема описано процес запуску кластера «Hadoop» в «Docker»-контейнерах для потреб «розумних міст» та розглянуто процес запуску «spark RDD» в середовищі «Spark» для проектів класу «розумне місто». Висвітленоу практичну реалізацію методу уникнення повторів для потреб проектів класу «розумне місто».

4 СПЕЦІАЛЬНА ЧАСТИНА

4.1 Кластерні експерименти тестування хмарної платформи «розумного міста»

При знаходженні точок насичення кластера запитується велика кількість інших даних «Spark». В процесі запитів знаходиться точна кількість кластерів у супроводі проблем з «Java Garbage Collector». При цьому алгоритм використовує всі «4Gb» пам'яті виділеної для «JVM» при вимкненому «OutOfMemory». Профілювальник показав, в процесі тестування використовувалась вся пам'ять (див. рисунок 4.1).

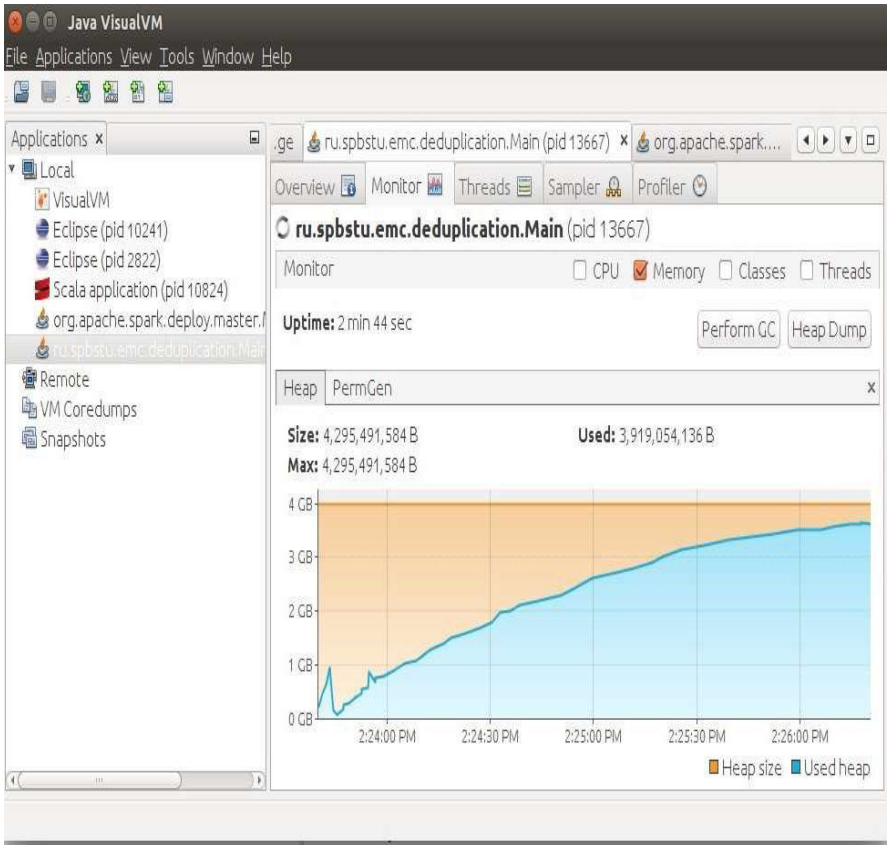

Рисунок 4.1 – Графік використання пам'яті виділеної для «JVM»

Завдяки використаному вихідному коду удосконаленого алгоритму зі збирачем сміття, отримано очищення пам'яті (див. рисунок 4.2).

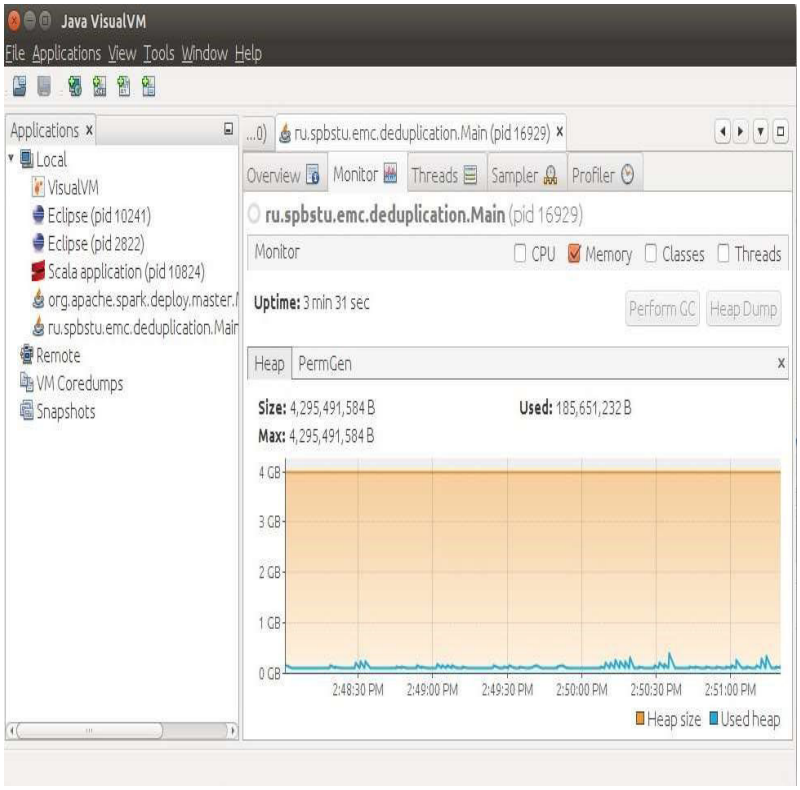

Рисунок 4.2 – Результат тестування вихідного коду удосконаленого алгоритму зі збирачем сміття

В процесі тестування виявлено наступну проблему, один із видів GUI-інтерфейс показав, що «Spark» працює, а інший «GUI»-показав, що створені нові ресурси не використовуються (див. рисунок 4.3).

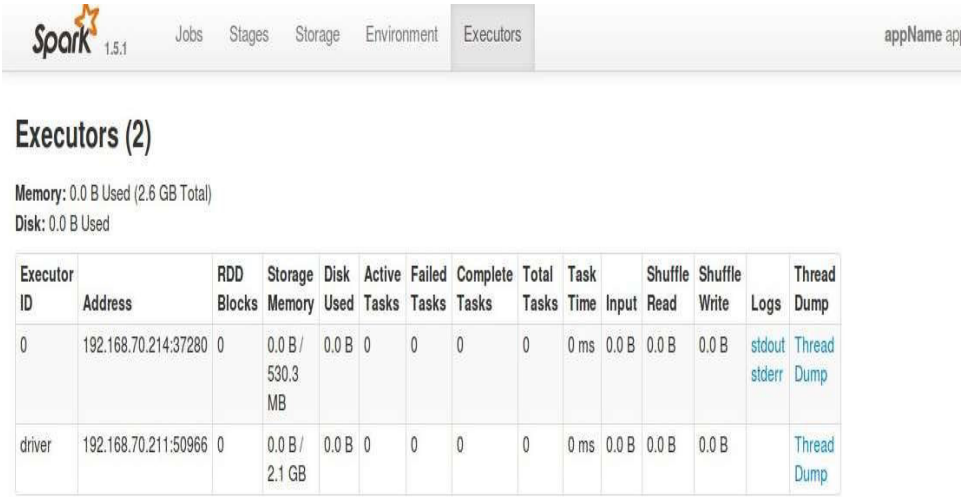

Рисунок 4.3 – Результати тестування інтерфейсів

«Java»-профілювальник показав як спроектований застосунок використовує доступні ресурси пам'яті (див. рисунок 4.4).

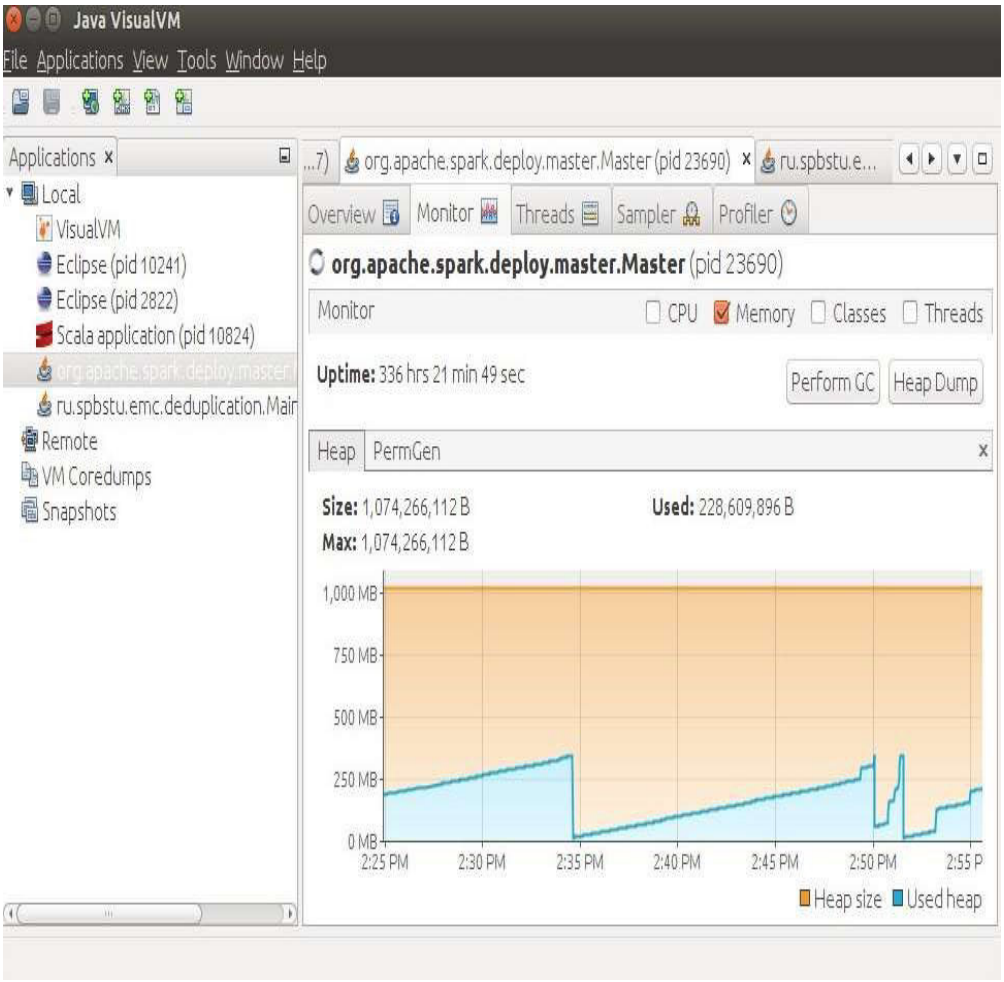

Рисунок 4.4 – Результат «Spark Deploy Master»

Розглянемо час запису одного з елементів «1000 хешів» у «Spark» на ПК (див. таблицю 4.1). Кожен хеш має довжину «16 байт».

Таблиця 4.1 – Результати експериментів

Дубльована швидкість = 0,5						
Кількість хешів у сховищі	Час запису 1000 хешів без перевірки унікальності, с		Час запису 1000 хешів з перевіркою унікальності, с		Час пошуку 1000 хешів один за іншим, с	Час запису 1000 хешів у Java HashMap, с
	Один за одним	По 1000	Один за одним	По 1000 (разом з ReduceByKey())		
0	0,6	0,002	25	0,002	35	0,02
1000	0,5	0,002	25	0,002	27	0,02
10000	0,5	0,0013	43,5	0,002	39	0,013

Час запису хешів без перевірки унікальності досить великий. Більша частина часу запису виконується пошуком. Використання методу «RedubyKey ()» набагато швидше чим пошук по одному критерію.

Під час наступного експерименту було вимкнуто час пошуку «1000 хешів» у залежності від об’ємів даних завантажених в пам'яті на ПК (див. рисунок 4.5).

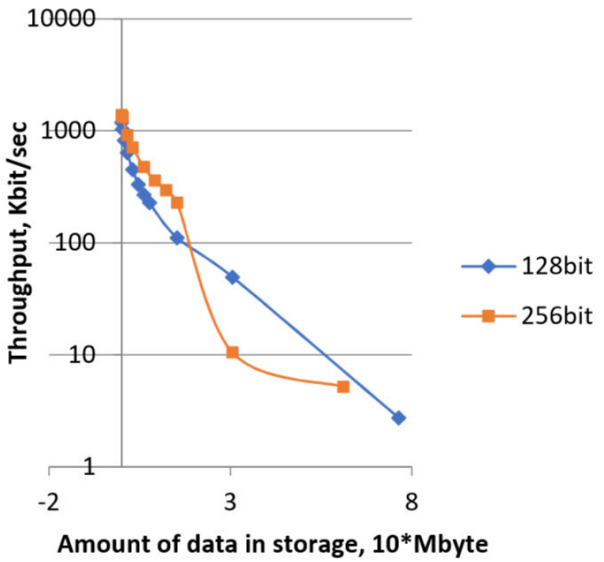

Рисунок 4.5 – Обсяг даних, що зберігаються

«Spark» є жадібним до пам'яті і створює «RDD», який для реальних даних по «1Кб» створює об'єкт для «10Кб» або більше. Оскільки порожній рядок «Java» займає близько «40 байт» в пам'яті.

При наступному експерименті вимірюється час пошуку «1000 хешів» і час виконання «reduceByKey» на кластері «IUS». Це час пошуку схожий на час пошуку на ПК, якщо кількість даних менше «3 Мбайт». Результати різних експериментів практично абсолютно однакові.

При наступному експерименті вимірюється час пошуку «1000 хешів» в залежності від кількості «Java»-потоків в кластері «IUS». Використовуючи «Java»-потоки, скорочується час пошуку.

Не дивлячись на те, що «Spark» виконує автоматичне паралельне виконання, якщо користувач створює пошук в пам'яті, то використання «Java»-потоків прискорює виконання пошуку «.Redis Pipelining».

Сервер запиту/відповіді може бути реалізований так, щоб він міг обробляти нові запити, навіть якщо клієнт ще не прочитав старі відповіді. Таким чином, можна відправити декілька команд на сервер, не чекаючи відповідей взагалі, і, нарешті, прочитати відповіді за один крок.

Тест «Redis» включає утиліту «redis-benchmark», яка імітує виконання команд, виконуваних «N»-клієнтами, одночасно відправляючи «M» найбільших запитів. Це схоже на утиліту «ab Apache».

«Redis Cluster» забезпечує спосіб запуску установки «Redis», де дані автоматично пересилаються через кілька вузлів «Redis».

«Redis Cluster» також забезпечує певний рівень доступності під час поділу, що на практиці означає можливість продовження операцій, коли деякі вузли зазнають невдачі або не можуть взаємодіяти. Однак кластер перестає працювати в разі великих збоїв, наприклад, коли більшість майстрів є недоступною.

Отже, з практичної точки зору з «Redis Cluster» буде отримано:

– Можливість автоматично розбивати набір даних на кілька вузлів.

– Можливість продовжити роботу, коли підмножина вузлів помічає збої або не може взаємодіяти з іншою частиною кластера.

4.2 Висновок

В цьому розділі отримано результати використанням алгоритму «MapReduce». Проаналізувавши результати отримано:

1. «Redis» використовує пам'ять 3-4 рази більше на ПК і в 9 разів більше в кластері, ніж пам'ять, зайнята хешами.

2. Можна написати тільки «2 Гб» хешу на кластері.

3. Використання конвеєрної обробки може підвищити продуктивність на ПК на одному вузлі.

4. Використання декількох підключень або «Java»-потоків може підвищити продуктивність.

5. Використання «Byte[]» замість «String» може підвищити продуктивність

6. «Redis» працює повільніше на кластері без конвеєрної обробки та декількох з'єднань.

5 ОБҐРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Метою дипломної роботи освітнього рівня «Магістр» є дослідження щодо організації хмарної платформи «розумного міста» для уникнення повторів колекцій даних. Головною метою розділу є встановлення економічної доцільності проведення даної розробки.

Щоб виконати оцінку економічної ефективності необхідно розрахувати трудомісткість дослідження, витрати на оплату праці найманим працівникам, витрати апаратного і програмного забезпечення, амортизаційні відрахування, витрати енергоресурсів та інші витрати які є основними пунктами виконання обчислень, а також показники економічної ефективності дослыдження.

5.1 Розрахунок норм часу на виконання науково-дослідної роботи

Реалізація проекту організації хмарної платформи «розумного міста» для уникнення повторів колекцій даних складається з низки послідовних та взаємопов’язаних етапів.

Кожен із етапів дослідження характеризується метою та змістом, оцінкою часу виконання, кількістю та спеціалізацією виконавців, а також приблизною оцінкою вартості.

Реалізація хмарної платформи «розумного міста» для уникнення повторів колекцій даних складається із підготовчого етапу, етапу технічної пропозиції, створення технічного завдання, проектування системи, практичної реалізації, тестування, верифікації та заключного етапу.

Норми часу на виконання науково-дослідницької роботи розраховуватимуться на основі середнього часу виконання стадії в годинах, що наведені в таблиці 5.1 разом із інформацією про виконавців і сумарною кількості затраченого часу.

Таблиця 5.1 – Операції науково-дослідного процесу та час їх виконання

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1.	Підготовча стадія	Проектний менеджер	4
		Інженер-програміст	
2.	Технічна пропозиція	Проектний менеджер	74
		Інженер-програміст	
3.	Створення технічного завдання	Проектний менеджер	80
		Інженер-програміст	
4.	Проектування системи	Проектний менеджер	76
5.	Практична реалізація	Інженер-програміст	50
6.	Тестування системи	Тестувальник	45
7.	Верифікація системи	Проектний менеджер	27
		Інженер-програміст	
		Тестувальник	
8.	Створення документації	Інженер-програміст	28
9.	Заключна стадія	Проектний менеджер	18
Разом			402

В підсумку на реалізацію хмарної платформи «розумного міста» для уникнення повторів колекцій даних необхідно 402 людино-годин, залучення трьох спеціалістів та виконання дев’яти різноманітних стадій реалізації проекту.

5.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

Визначення витрат на оплату праці та відрахувань на соціальні заходи прямо залежить від кількості витраченого працівниками часу на роботу,

ставки в годину чи місяць, кількість відрахувань на соціальні заходи встановлених в законному порядку на час розрахунку.

В результаті розрахунку потрібно визначити основну та додаткову заробітну плату, витрати на соціальні заходи та на основі цих даних визначити сумарні витрати на оплату праці.

Основна заробітна плата нараховується за виконану роботу за тарифними ставками, відрядними розцінками чи посадовими окладами.

Додаткова заробітна плата – це складова заробітної плати працівників, до якої включають витрати на оплату праці, не пов'язані з виплатами за фактично відпрацьований час.

При розрахунку заробітної плати кількість робочих днів у місяці слід в середньому приймати – 24,5 дні/міс., або ж 196 год./міс. (тривалість робочого дня – 8 год.).

Наймані працівники для розробки інформаційної системи управління доступом з використанням інформаційних технологій розпізнавання образів працюють згідно контракту, який в якому вказано їхню погодинну ставку. Тобто розрахунок заробітної плати працівників відбуватиметься на базі тарифної ставки та кількості відпрацьованих годин.

У штаті найманих працівників для розробки інформаційної системи залучено проектного менеджера, інженера-програміста і тестувальника.

Тарифні ставки учасників процесу розробки інформаційної системи управління доступом з використанням інформаційних технологій розпізнавання образів:

– Проектний менеджер – 150 грн./год.

– Інженер-програміст – 130 грн./год.

– Тестувальник – 100 грн./год.

Основна заробітна плата розраховується за формулою:

$$З_{осн.} = Т_с \cdot К_г, \qquad (5.1)$$

де $Т_с$ – тарифна ставка, грн.;

$К_г$ – кількість відпрацьованих годин.

Оскільки всі види робіт в виконує три спеціалісти, то основна заробітна плата буде розраховуватись за даною формулою 5.1.

$$З_{осн.} = 150 \cdot 182 + 130 \cdot 166 + 100 \cdot 54 = 54280{,}00 \text{ грн.}$$

Додаткова заробітна плата становить 10–15 % від суми основної заробітної плати й визначається за формулою 5.2. Коефіцієнт додаткових виплат працівникам становить 0,1 .

$$З_{дод.} = З_{осн.} \cdot К_{допл.}, \quad (5.2)$$

де $К_{допл}$ – коефіцієнт додаткових виплат працівникам.

$$З_{дод.} = 54280{,}00 \cdot 0{,}1 = 5428{,}00 \text{ грн.}$$

Звідси загальні витрати на оплату праці ($В_{о.п.}$ – фонд заробітної плати) визначаються за формулою 5.3:

$$В_{о.п.} = З_{осн.} + З_{дод.}. \quad (5.3)$$

$$В_{о.п.} = 54280{,}00 + 5428{,}00 = 59708{,}00 \text{ грн.}$$

З цієї суми утримуються обов’язкові відрахування на заробітну плату:

– Єдиний соціальний внесок (ЄСВ), що становить – 22%.

– Військовий збір (ВЗ), що становить – 1,5%.

Сума відрахувань становить 23,5 % від фонду оплати праці та визначається за формулою 5.4:

$$B_{с.з.}=Ф_{ОП}\ 0{,}235, \qquad (5.4)$$

де $Ф_{ОП}$ – фонд оплати праці, грн.

$$B_{с.з.} = 59708{,}00 \cdot 0{,}235 = 14031{,}38 \text{ грн.}$$

Усі витрати обчислюються детально наведені в таблиці 5.2 та обчислюються за формулою 5.5.

$$B_{З.П.}= Ф_{ЗП}+Ф_{ОП}, \qquad (5.5)$$

$$B_{З.П.} = 54280{,}00 + 5428{,}00 + 14031{,}38 = 73739{,}38 \text{ грн.}$$

Таблиця 5.2 – Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	Нарахув. на ФОП, грн.	Всього витрати на оплату праці, грн. *6=3+4+5*
		Тарифна ставка, грн.	К-сть відпрацьов. год.	Фактично нарах. з/пл., грн.			
А	Б	1	2	3	4	5	6
1	Проектний менеджер	150	182	27300,00	2730,00		
2	Інженер-програміст	130	166	21580,00	2158,00		
3	Тестувальник	100	54	5400,00	540,00		
Разом		380	402	54280,00	5428,00	14031,38	73739,38

Опираючись на розрахунки витрат на оплату та зведену таблицю результатів 5.2 видно, що всього витрати на плату праці становлять 73739,38 грн.

5.3 Розрахунок матеріальних витрат

Матеріальні витрати є невід'ємною частиною розроблення хмарної платформи «розумного міста» для уникнення повторів колекцій даних та визначаються як добуток кількості витрачених матеріалів та їх ціни за формулою 5.6:

$$M_{Bi} = q_i \cdot p_i, \tag{5.6}$$

де: q_i – кількість витраченого матеріалу і-го виду;

p_i – ціна матеріалу і-го виду.

Звідси, загальні матеріальні витрати можна визначити за формулою 5.7:

$$З_{мв.} = \sum M_{Bi}. \tag{5.7}$$

Результати проведених розрахунків наведено у таблиці 5.3.

Таблиця 5.3 – Результати розрахунків матеріальних витрат

1	Найменування матеріальних ресурсів	Одиниця виміру	Фактично витрачено матеріалів	Ціна за одиницю, грн	Загальна сума витрат, грн
1	2	3	4	5	6
1. Основні матеріали					
1.1	Використання мережі Інтернет, місячна абонплата	міс		150	300,00
2. Допоміжні витрати					
2.1	Папір	уп.	0,2	85,00	17,00
2.2	Тонер	уп.	1	50,00	50,00
2.3	CD диск	шт.	2	10	20,00
Разом:					387,00

Загальні матеріальні витрати на Інтернет, папір формату А4, тонер та CD-диски становлять 387,00 грн.

5.4 Розрахунок витрат на електроенергію

Однією із статей витрат є витрати на електроенергію під час проходження усіх етапів реалізації кінцевого продукту. Затрати на електроенергію одиниці обладнання визначаються за формулою 5.8:

$$З_е = W \cdot T \cdot S \ , \qquad (5.8)$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії.

Вартість кіловат-години електроенергії слід приймати згідно існуючих на даний час тарифів. Отже, 1 кВт з ПДВ коштує 2,42 грн.

Потужність комп'ютерів для реалізації кінцевого продукту – 400 Вт, кількість годин роботи обладнання згідно таблиці 5.1 – 402 годин.

Визначимо витрати на електроенергію згідно формули 5.8

$$З_е = 0{,}4 \cdot 402 \cdot 2{,}42 = 389{,}14 \text{ грн.}$$

Отже, затратами на електроенергію для організації хмарної платформи «розумного міста» для уникнення повторів колекцій даних буде 389,14 грн.

5.5 Розрахунок суми амортизаційних відрахувань

Для будь якої діяльності характерною є властивість зношування на зниження якості властивостей інструментарію та фондів за допомогою яких

ведеться діяльність. Для вирішення проблеми із відновленням даних фондів використовується амортизація, що являє собою процес трансформації вартості основних фондів на вартість продукції, яка щойно була створена, задля повного відновлення основних фондів. Для визначення амортизаційних відрахувань використовується формула 5.9:

$$A = \frac{Б_В \cdot Н_А}{100\,\%}, \qquad (5.9)$$

де A – амортизаційні відрахування за звітний період, грн.;

$Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, грн.;

$Н_А$ – норма амортизації, %.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %). Річний робочий фонд становитиме 2352 годин, так як робочий день становить 8 годин, а кількість робочих днів в місяці становить 24,5 годин.

Для даної розробки засобом розробки є комп'ютер. Його сума становить 18580 грн. Отже, амортизаційні відрахування будуть рівні:

$$A = 18580 \cdot 5\ \% / 100\ \% = 929{,}00 \text{ грн.}$$

Згідно проведених обчислень амортизаційні відрахування становлять 929,00 грн.

5.6 Обчислення накладних витрат

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління спілкою та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20-60 % від суми основної та додаткової заробітної плати працівників.

$$H_в = B_{о.п.} \cdot 0{,}2...0{,}6, \quad (5.10)$$

де $H_В$ – накладні витрати.

Отже, накладні витрати становлять згідно формули 5.10:

$$H_в = \ 59708{,}00 \ \cdot \ 0{,}2 \ = \ 11941{,}60 \text{ грн.}$$

Отже, накладні витрати для науково-дослідних робіт щодо організації хмарної платформи «розумного міста» для уникнення повторів колекцій даних будуть становити 11941,60 грн.

5.7 Складання кошторису витрат та визначення собівартості науково-дослідницької роботи

Результати проведених вище розрахунків зведемо у таблицю 5.4.

Таблиця 5.4 – Кошторис витрат на НДР

Зміст витрат	Сума, грн.	В % до загальної суми
Витрати на оплату праці (основну і додаткову заробітну плату)	59708,00	68,3
Відрахування на соціальні заходи	14031,38	16,1
Матеріальні витрати	387	0,44
Витрати на електроенергію	389,136	0,45
Амортизаційні відрахування	929,00	1,06

Накладні витрати	11941,60	13,7
Собівартість	87386,12	100

Собівартість ($C_в$) дослідження розрахуємо за формулою:

$$C_В = B_{о.п.} + B_{с.з.} + З_{м.в.} + З_е + A + H_в. \quad (5.11)$$

Отже, собівартість дослідження дорівнює:

$$C_В = 59708{,}00 + 14031{,}38 + 387 + 389{,}136 + 929{,}00 + 11941{,}60 =$$

$$= 87386{,}12 \text{ грн.}$$

Загальний кошторис витрат та визначення собівартості науково-дослідницької роботи становить 87386,12 грн.

5.8 Розрахунок ціни проведених науково-дослідних робіт

Ціну науково-дослідної роботи можна визначити можна визначити за формулою:

$$Ц = \frac{C_В \cdot (1 + P_{рен}) + K \cdot B_{н.і.}}{K} \cdot (1 + ПДВ), \quad (5.12)$$

де $P_{рен.}$ – рівень рентабельності, 30 %;

K – кількість замовлень, од.;

$B_{н.і.}$ – вартість носія інформації, грн.;

$ПДВ$ – ставка податку на додану вартість, (20 %).

Оскільки розробка є науково-дослідною, і використовуватиметься тільки один раз, то для розрахунку ціни не потрібно вказувати коефіцієнти К та $B_{н.і}$, оскільки їх в даному випадку не потрібно.

Тоді, формула для обчислення ціни розробки буде мати вигляд:

$$Ц = C_B \cdot (1 + P_{рен}) \cdot (1 + ПДВ). \quad (5.13)$$

Звідси ціна проведення науково-дослідних робіт складе:

$$Ц = 87386,12 \ (1+ \ 0,3 \ ) \cdot (1+ \ 0,2 \ ) = 136322,34 \text{ грн.}$$

Отже, для організації хмарної платформи «розумного міста» для уникнення повторів колекцій даних необхідно 136322,34 грн.

5.9 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (Ер) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_p = \frac{П}{C_B}, \quad (5.14)$$

де $П$ – прибуток;

C_B – собівартість.

Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = Ц - С_В. \quad (5.15)$$

Розраховуємо плановий прибуток:

$$П_{пл} = 136322{,}34 - 87386{,}12 = 48936{,}22 \text{ грн.}$$

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_р = \frac{П_{пл}}{Св}. \quad (5.16)$$

Тоді,

$$E_р = 48936{,}22 \ / \ 87386{,}12 = 0{,}56 \ .$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_р$):

$$T_р = \frac{1}{E_р}, \quad (5.17)$$

Термін окупності дорівнює:

$$T_р = 1 / \ 0{,}56 = 1{,}8 \text{ р.}$$

Згідно формул плановий прибуток від проведених науково-дослідних робіт становить 48936,22 грн., економічна ефективність дорівнює 0,56 , а термін окупності становить 1,8 роки що вважається доцільним та економічно вигідним.

5.10 Висновок

В розділі «Обґрунтування економічної ефективності» дипломної роботи освітнього рівня «магістр» розраховано основні техніко-економічні показники проведених досліджень щодо організації хмарної платформи «розумного міста» для уникнення повторів колекцій даних (див. таблицю 5.5).

Розраховане значення економічної ефективності становить 0,56 , що є високим значенням.

Так само нормальним є термін окупності, який повинен коливатися від 1 до 3 років. Для проведених в дипломній роботі досліджень він становить 1,8 років.

Таблиця 5.5 – Техніко-економічні показники НДР

№ п/п	Показник	Значення
1.	Собівартість, грн.	87386,12
2.	Плановий прибуток, грн.	48936,22
3.	Ціна, грн.	136322,34
4.	Економічна ефективність	0,56
5.	Термін окупності, рік	1,8

Отже, отримані в рамках дипломного проектування результати науково-дослідних робіт можуть бути впроваджені та мати подальший розвиток, оскільки вони є економічно вигідним за всіма основними техніко-економічними показниками.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Основні особливості стандарту OHSAS 18001 щодо ведення та управління документацією з охорони праці

Особливості OHSAS 18001. OHSAS 18001: 2007 був розроблений відповідно до правил Директив ISO / IEC з урахуванням структури стандартів ISO 9001: 2000 (система менеджменту якості), ISO 14001: 2004 (система менеджменту навколишнього середовища) та ILO-OSH. Таким чином даний стандарт сприяє інтеграції систем менеджменту якості, навколишнього середовища та професійного здоров'я та безпеки праці в організаціях.

OHSAS 18001: 2007 тепер називається стандартом, а не специфікацією або документом, як це було в попередніх редакціях. У даній редакції OHSAS 18001 значно поліпшена узгодженість з ISO 14001: 2004 по структурі стандарту, а також покращена сумісність з ISO 9001: 2000, що відображає тенденцію прийняття OHSAS 18001 в якості основи для національних стандартів по системах менеджменту професійного здоров'я та безпеки праці (СУОП).

Слід підкреслити, що існує велика різниця між керівними вказівками щодо впровадження СУОП та даним стандартом (OHSAS 18001), який не тільки встановлює вимоги до системи управління охороною праці організації, але і може бути використаний як для сертифікації, так і/або для самодекларування системи управління охороною праці. Керівні вказівки, навіть у випадку якщо вони розроблені такою авторитетною організацією, як МОП, непридатні для цілей сертифікації і призначені лише для надання допомоги при організації, розробці, впровадженні та вдосконаленні системи управління охороною праці.

Останнім часом організації все більш зацікавлені в досягненні і демонстрації вагомої результативності в сфері охорони праці за рахунок управління професійними ризиками у відповідності з політикою та цілями в галузі охорони праці. Здійснюється це в умовах розвитку економічної політики та інших заходів, спрямованих на належне виконання заходів з охорони праці, а також в умовах загального зростання зацікавленості питаннями охорони праці.

Стандарт OHSAS 18001 якраз і призначений для забезпечення організацій елементами ефективної та результативної системи управління охороною праці, які можуть бути інтегровані з іншими елементами управління для сприяння організаціям в досягненні поставлених цілей у сфері охорони праці та економіки.

Як відомо, управління охороною праці охоплює весь діапазон проблем, включаючи проблеми, що стосуються стратегії і конкурентоспроможності, і демонстрація успішного впровадження цього стандарту може бути використана організацією для того, щоб зацікавлені сторони впевнилися в наявності у неї належної системи управління охороною праці.

Особливістю стандарту OHSAS 18001 є те, що в ньому використана методологія, відома як "Plan-Do-Check-Act" (PDCA), або "Плануй-Роби-Перевіряй-Коригуй" (ПРПК). [63]

Коротко методологія PDCA / ПРПК може бути описана таким чином:

„Плануй” – це встановлення цілей і процесів, необхідних для отримання результатів у відповідності з політикою організації в області охорони праці;

„Роби” – це реалізація процесів;

„Перевіряй” – це моніторинг та оцінка процесів по відношенню до політики в сфері охорони праці, цілям, завданням, законодавчим і іншим вимогам, а також запис результатів;

„Коригуй” – це здійснення заходів стосовно безперервного поліпшення результативності охорони праці.

Необхідно відзначити, що багато організацій управляють своїми операціями за допомогою системи процесів та їх взаємодій, і ця методологія називається «процесний підхід». Саме таку методологію «процесного підходу» рекомендує використовувати ISO 9001. Однак, оскільки методологія PDCA/ПРПК може бути застосовна до всіх без виключення процесів, то обидві ці методології є сумісними.

Ще однією особливістю стандарту OHSAS 18001 є те, що він містить лише ті вимоги, які можуть бути піддані об'єктивному аудиту.

Також слід зазначити, що стандарт OHSAS 18001 не встановлює абсолютних вимог у питаннях результативності охорони праці. Виняток становлять лише зобов'язання щодо відповідності законодавчим, нормативно-правовим та іншим вимогам, які містяться у політиці в сфері охорони праці і поширюються на організацію, а також зобов'язання щодо попередження нещасних випадків на виробництві та професійних захворювань, та зобов'язання щодо постійного поліпшення ефективності СУОП.

OHSAS 18001 не містить вимог, характерних для інших систем управління, наприклад, вимог до управління якістю, навколишнім середовищем, безпекою, фінансовими ресурсами, і елементи системи управління охороною праці можуть бути узгоджені або інтегровані з відповідними елементами інших систем управління. Таким чином, організація може використовувати вже існуючі системи управління при розробці системи управління охороною праці відповідно до вимог стандарту OHSAS 18001.

Сфера застосування. Стандарт OHSAS 18001 встановлює вимоги до системи управління охороною праці, які можуть бути застосовні до організацій будь-якого типу і розміру. Успіх впровадження системи залежить

від зобов'язань, прийнятих на всіх рівнях і всіма підрозділами організації, особливо вищим керівництвом. Впровадження такої системи дозволить організації сформулювати політику в галузі охорони праці, встановити цілі та процеси для виконання зобов'язань, передбачених політикою, а також здійснити заходи для поліпшення результативності та продемонструвати відповідність СУОП вимогам OHSAS 18001.

Організація повинна розробити, документально оформити, впровадити, підтримувати в робочому стані і постійно поліпшувати систему управління охороною праці відповідно до вимог даного стандарту, а також визначити механізми виконання цих вимог.

Рівень деталізації та складності системи управління охороною праці, обсяг документації та витрачених на неї ресурсів залежить від ряду факторів, таких як сфера застосування системи, розмір організації, характер її діяльності, вид продукції, що виготовляється, або послуг, що надаються.

OHSAS 18001 встановлює вимоги до системи управління охороною праці з метою надання допомоги організаціям в управлінні ризиками та підвищення результативності даного управління. Стандарт не встановлює конкретних критеріїв результативності системи управління охороною праці і не містить вказівок щодо її розробки.

Даний стандарт може бути застосовний до будь-якої організації, яка має намір:

а) розробити систему управління охороною праці для усунення або мінімізації ризиків для працюючих (працівників) та інших зацікавлених сторін, які можуть піддаватися небезпекам, що пов'язані з діяльністю організації;

б) впровадити, підтримувати в робочому стані і поліпшувати систему управління охороною праці;

в) упевнитися, що СУОП відповідає вимогам політики в галузі охорони праці, яка розроблена організацією;

г) продемонструвати відповідність стандарту OHSAS 18001 шляхом проведення самооцінки і самодекларування, абоотримання підтвердження своєї відповідності сторонами, зацікавленими діяльністю організації, такими як замовники, або отримання підтвердження самодекларування зовнішньою стороною, або проведення сертифікації системи управління охороною праці зовнішньої організацією.

Всі вимоги стандарту OHSAS 18001 можуть бути застосовні до будь-якій системі управління охороною праці. Ступінь їх застосування залежить від таких факторів, як політика в галузі охорони праці організації, характеру її діяльності та ризиків, а також складності процесів.

6.2 Заходи щодо відвернення пожежі на робочих місцях користувачів ПК

Пожежна безпека підприємств та організацій забезпечується шляхом проведення організаційних, технічних та інших заходів, спрямованих на попередження пожеж, забезпечення безпеки людей, зниження можливих матеріальних збитків, зменшення негативних екологічних наслідків, створення умов для швидкого виклику пожежних підрозділів та успішного гасіння пожеж, а також евакуації з зони виникнення і можливого розповсюдження пожежі людей, документів і матеріальних цінностей. [64]

Відповідно до Закону України "Про пожежну безпеку" відповідальними за протипожежний стан енергетичних підприємств є керівники цих підприємств і організацій, а також відповідальні особи, уповноважені наказом керівників та посадовою інструкцією.

Забезпечення пожежної безпеки є складовою виробничої та іншої діяльності посадових осіб і персоналу ІТ-підприємств. Це повинно бути відображено у трудових договорах (контрактах) та статутах ІТ-підприємств.

Керівник ІТ-підприємства повинен визначити обов'язки посадових осіб щодо забезпечення пожежної безпеки, призначати своїм наказом

відповідальних за пожежну безпеку окремих будівель цехів, споруд, приміщень, технологічного та інженерного обладнання, а також за утримання й експлуатацію технічних засобів протипожежного захисту.

З метою залучення працівників до проведення заходів щодо запобігання пожежам, організації гасіння їх на ІТ-підприємствах створюються добровільні пожежні дружини (далі – ДПД) і команди (далі – ДПК), діяльність яких повинна здійснюватися відповідно до Положення про добровільні пожежні дружини (команди), затвердженого наказом МНС України від 11.02.2004 N 70 та зареєстрованого в Міністерстві юстиції України 19.02.2004 за N 221/8820.

На підприємствах з кількістю працівників 50 осіб і більше за рішенням трудового колективу створюються пожежно-технічні комісії (далі – ПТК). Їх роботу необхідно організовувати згідно з Типовим положенням про пожежно-технічну комісію, затвердженим наказом МНС України від 11.02.2004 N 70 та зареєстрованим у Міністерстві юстиції України 19.02.2004 за N 222/8821.

На кожному ІТ-підприємстві повинна бути розроблена загальнооб'єктова інструкція про заходи пожежної безпеки та інструкції для усіх вибухопожежонебезпечних та пожежонебезпечних приміщень (цехів, складів, майстерень, лабораторій, дільниць тощо) відповідно до основних вимог щодо документації з пожежної безпеки.

Інструкції повинні періодично переглядатися на основі аналізу протипожежного стану об'єкта та відповідних наказів не рідше одного разу на 3 роки.

На всіх ІТ-підприємствах наказом повинен бути встановлений відповідний протипожежний режим, яким визначені:

– можливість і місця паління, застосування відкритого вогню, побутових нагрівальних приладів;

– порядок проведення тимчасових пожежонебезпечних робіт;

– правила проїзду та стоянки транспортних засобів;

– місця для зберігання і допустима кількість сировини, напівфабрикатів та готової продукції, які можуть одночасно бути у виробничих приміщеннях і на території;

– порядок прибирання горючого пилу й відходів, зберігання промасленого ганчір'я та спецодягу, очищення повітропроводів вентиляційних систем від горючих відкладень;

– порядок відключення від мережі електрообладнання в разі пожежі;

– порядок огляду й зачинення приміщень після закінчення роботи;

– порядок проходження посадовими особами навчання і перевірки знань з питань пожежної безпеки, а також проведення з працівниками протипожежних інструктажів та занять з пожежно-технічного мінімуму з призначенням відповідальних осіб за їх проведення;

– порядок організації експлуатації і обслуговування наявних технічних засобів протипожежного захисту (протипожежного водогону, насосних станцій, установок пожежної сигналізації, автоматичного пожежогасіння, димовидалення, вогнегасників тощо);

– порядок проведення планово-попереджувальних ремонтів та оглядів електроустановок, опалювального, вентиляційного, технологічного та іншого інженерного обладнання;

– дії працівників у разі виявлення пожежі;

– порядок збирання членів добровільної пожежної дружини та відповідальних посадових осіб у разі виникнення пожежі, виклику вночі, у вихідні та святкові дні.

Відповідальними за пожежну безпеку окремих цехів, лабораторій, відділів, складів і інших виробничих і допоміжних споруд підприємств є керівники цих структурних підрозділів або посадові особи, які виконують їхні обов'язки.

Керівники структурних підрозділів ІТ-підприємств, а також інші посадові особи, відповідальні за пожежну безпеку, зобов'язані:

– забезпечити дотримання протипожежного режиму і виконання в установлені строки заходів, які підвищують пожежну безпеку;

– забезпечити справність і нормальну роботу технологічного обладнання відповідно до технологічних вимог і проектних рішень та негайно вживати заходів до усунення виявлених несправностей, які можуть призвести до пожежі або загорання;

– організувати пожежно-технічне навчання персоналу і вимагати від нього дотримання протипожежного режиму і виконання встановлених вимог пожежної безпеки, особливо щодо технології виробництва;

– забезпечити контроль за виконанням вимог пожежної безпеки при проведенні ремонтних робіт, а також відключення електромережі після закінчення роботи;

– забезпечити утримання в справному стані і постійну готовність до дії всіх засобів виявлення та гасіння пожежі;

– при виявленні пожежі негайно викликати пожежний підрозділ, оповістити керівництво підприємства та організувати гасіння пожежі й евакуацію персоналу (при потребі).

6.3 Оцінка події, що сталася або може статися у прогнозований термін, та визначення ступеня реагування на відповідному рівні управління

Впровадження ефективного механізму оцінки аварійної події, що сталася або може статися у прогнозований термін, обґрунтування віднесення цієї події до рангу НС та визначення рівня реагування, що відповідає масштабу цієї події, на нашу думку, повинно провадитись за допомогою класифікатора надзвичайних ситуацій.

Виходячи з цього та з метою створення єдиної системи класифікації надзвичайних ситуацій і визначення їх рівнів, забезпечення оперативного і адекватного реагування на такі ситуації розроблено Положення про класифікацію надзвичайних ситуацій, затверджене постановою Кабінету Міністрів України «Про порядок класифікації надзвичайних ситуацій» від 15 липня 1998 р. №1099.

У зазначеному Положенні сформульовано і затверджено ряд єдиних термінів та понять.

Так аварію визначено як небезпечну подію техногенного характеру, що створює на об'єкті, території або акваторії загрозу для життя і здоров'я людей і призводить до руйнування будівель, споруд, обладнання і транспортних засобів, порушення виробничого або транспортного процесу чи завдає шкоди довкіллю.

Надзвичайні ситуації на території України поділяються за такими основними ознаками [65]:

– у сфері виникнення;

– за галузевою ознакою;

– за масштабами можливих наслідків.

За першою ознакою «у сфері виникнення» надзвичайні ситуації розподіляються за характером виникнення на: техногенні, природні, соціально-політичні та воєнні.

Техногенні надзвичайні ситуації класифікуються за типами аварій (катастроф) як представлено на рисунку 6.1.

Надзвичайні ситуації техногенного характеру за характеристиками явищ, що визначають особливості дії факторів ураження на людей, навколишнє середовище та об'єкти господарської діяльності, поділяються на аварії (катастрофи), які супроводжуються викидами (виливами) небезпечних речовин, пожежами, вибухами, затопленнями, аваріями на інженерних

мережах і системах життєзабезпечення, руйнуванням будівель і споруд, аваріями транспортних засобів та інші. [66]

Рисунок 6.1 – Класифікація НС в Україні

Аварії (катастрофи), що пов’язані з викидом небезпечних речовин, додатково поділяються на радіаційні, хімічні, біологічні. Крім цього, поділяються ще за видами розповсюдження речовин в навколишньому середовищі, як представлено на рисунку 6.2.

Природні надзвичайні ситуації класифікують за видами можливих природних явищ, що приводять до їх виникнення: небезпечні геологічні, метеорологічні, гідрологічні морські та прісноводні явища, деградація ґрунтів чи надр, природні пожежі, зміна стану повітряного басейну, інфекційна захворюваність людей, сільськогосподарських тварин, масове ураження сільськогосподарських рослин хворобами і збудниками, зміна стану водних ресурсів та біосфери тощо.

Рисунок 6.2 – Класифікація НС техногенного характеру

Кожний клас стихій класифікується за характеристиками явища, що визначають особливості дії факторів ураження на людей, навколишнє середовище та об’єкти господарської діяльності, як представлено на рисунку 6.3.

Рисунок 6.3 – Класифікація НС природного характеру

Надзвичайні ситуації соціально-політичного характеру, які пов’язані з протиправними діями терористичного і антиконституційного спрямування, поділяються на: здійснення або реальна загроза терористичного акту (збройний напад, захоплення і утримання важливих об’єктів, ядерних установок і матеріалів, систем зв’язку та телекомунікації, напад чи замах на екіпаж повітряного або морського судна), викрадення (спроба викрадення) чи знищення суден, захоплення заручників, встановлення вибухових пристроїв у громадських місцях, викрадання або захоплення зброї, виявлення застарілих боєприпасів тощо, як представлено на рисунку 6.4.

Рисунок 6.4 – Класифікація НС соціально-політичного характеру

Надзвичайні ситуації воєнного характеру, які пов’язані з наслідками застосування зброї масового ураження, під час яких виникають вторинні фактори ураження населення, внаслідок руйнування атомних і гідроелектричних станцій, складів і сховищ радіоактивних і токсичних речовин та відходів, нафтопродуктів, вибухівки, транспортних та інженерних комунікацій тощо, як представлено на рисунку 6.5.

За другою основною ознакою «галузевою» надзвичайні ситуації поділяються на такі, які можуть статися: в будівництві, промисловості, в житлово-комунальній та побутовій сферах, на транспорті, в сільському та лісовому господарстві.

Рисунок 6.5 – Класифікація НС воєнного характеру

Надзвичайні ситуації на транспорті додатково поділяються, в залежності від виду транспорту, на надзвичайні ситуації на повітряному, водному, наземному та на підземному транспорті, як представлено на рисунку 6.6.

Рисунок 6.6 – Класифікація НС на транспорті

За третьою основною ознакою «масштабом можливих наслідків» надзвичайні ситуації поділяються, з урахуванням територіального поширення, характеру та виду сил і засобів, що залучаються для ліквідації наслідків, на:

– НС загальнодержавного рівня.
– НС регіонального рівня.
– НС місцевого рівня.
– НС об’єктового рівня.

6.4 Функціонування державної системи спостереження, збирання, оброблення та аналізу інформації про стан довкілля під час надзвичайних ситуацій мирного та воєнного часу.

Законом України „Про охорону навколишнього природного середовища" (ст.20, 22) передбачено створення державної системи моніторингу довкілля (далі – ДСМД) та проведення спостережень за станом навколишнього природного середовища, рівнем його забруднення. Виконання цих функцій покладено на Мінприроди та інші центральні органи виконавчої влади, які є суб'єктами державної системи моніторингу довкілля, а також підприємства, установи та організації, діяльність яких призводить або може призвести до погіршення стану довкілля.

Основні принципи функціонування ДСМД визначені у постанови Кабінету Міністрів України від 30.03.1998 № 391 „Про затвердження Положення про державну систему моніторингу довкілля".

На даний час, у державній системі моніторингу довкілля (далі – ДСМД) функції і задачі спостережень та інформаційного забезпечення виконують 8 суб`єктів системи моніторингу: Мінприроди, МНС, МОЗ, Мінагрополітики, Мінжитлокомунгосп, Держводгосп, Держкомлісгосп, Держкомзем.

Кожний із суб`єктів ДСМД здійснює моніторинг тих об`єктів довкілля, що визначаються Положенням про державну систему моніторингу довкілля та порядками і положеннями про державний моніторинг окремих складових довкілля.

Основні нормативні акти, що регламентують моніторинг об'єктів довкілля:

– постанова Кабінету Міністрів України від 09.03.1999 № 343 «Про затвердження Порядку організації та проведення моніторингу в галузі охорони атмосферного повітря»;

– постанова Кабінету Міністрів України від 20.07.1996 № 815 «Про затвердження Порядку здійснення державного моніторингу вод»;

– останова Кабінету Міністрів України від 20.08.1993 № 661 «Про затвердження Положення про моніторинг земель»;

– постанова Кабінету Міністрів України від 26.02.2004 № 51 «Про затвердження Положення про моніторинг ґрунтів на землях сільськогосподарського призначення».

З метою координації діяльності міністерств та відомств, визначення основних принципів державної політики з питань розвитку системи моніторингу навколишнього середовища, забезпечення її функціонування на основі єдиного нормативно-методологічного забезпечення постановою Кабінету Міністрів України від 17.11.2001 № 1551 утворено Міжвідомчу комісію з питань моніторингу довкілля.

Мінприроди здійснюється організаційно-технічне забезпечення роботи комісії та її профільних секцій.

Існуюча система моніторингу довкілля базується на виконанні розподілених функцій її суб'єктами і складається з підпорядкованих їм підсистем. Кожна підсистема на рівні окремих суб'єктів системи моніторингу має свою структурно-організаційну, науково-методичну та технічну бази.

Функціонування ДСМД здійснюється на трьох рівнях, що розподіляються за територіальним принципом:

– загальнодержавний рівень, що охоплює пріоритетні напрямки та завдання моніторингу в масштабах всієї країни;

– регіональний рівень, що охоплює пріоритетні напрямки та завдання в масштабах територіального регіону;

– локальний рівень, що охоплює пріоритетні напрямки та завдання моніторингу в масштабах окремих територій з підвищеним антропогенним навантаженням.

Моніторинг якості повітря. Державною гідрометеорологічною службою (МНС) здійснюються спостереження за забрудненням атмосферного повітря у 53 містах України на 162 стаціонарних, двох маршрутних постах спостережень та двох станціях транскордонного переносу.

Ведуться спостереження за хімічним складом атмосферних опадів та за кислотністю опадів.

Програма обов`язкового моніторингу якості атмосферного повітря включає сім забруднюючих речовин: пил, двоокис азоту (NO2), двоокис сірки (SO2), оксид вуглецю, формальдегід (H2CO), свинець та бенз(а)пірен. Деякі станції здійснюють спостереження за додатковими забруднюючими речовинами. Проводиться аналіз наявності забруднюючих речовин в опадах та сніговому покриві.

Державна екологічна інспекція (Мінприроди) здійснює вибірковий відбір проб на джерелах викидів. Вимірюється понад 65 параметрів.

Санітарно-епідеміологічна служба (МОЗ) здійснює спостереження за якістю атмосферного повітря у житловій та рекреаційній зонах, зокрема поблизу основних доріг, санітарно-захисних зон та житлових будинків, на території шкіл, дошкільних установ та медичних закладів в містах та в робочий зоні. Крім того, здійснюється аналіз якості повітря у житловій зоні за скаргами мешканців.

Моніторинг стану вод суші. Державна гідрометеорологічна служба (МНС) проводить моніторинг гідрохімічного стану вод на 151 водному об`єкті, а також здійснює гідробіологічні спостереження на 45 водних

об`єктах. Отримуються дані по 46 параметрах, що дають можливість оцінити хімічний склад вод, біогенні параметри, наявність зважених часток та органічних речовин, основних забруднюючих речовин, важких металів та пестицидів. На 8 водних об`єктах проводяться спостереження за хронічною токсичністю води. Визначаються показники радіоактивного забруднення поверхневих вод.

Державна екологічна інспекція (Мінприроди) відбирає проби води та отримує дані по 60 вимірюваних параметрах.

Державний комітет по водному господарству проводить моніторинг річок, водосховищ, каналів, зрошувальних систем і водойм у межах водогосподарських систем комплексного призначення, систем водопостачання, транскордонних водотоків та водойм у зонах впливу атомних електростанцій. Контроль якості води за фізичними та хімічними показниками здійснюється на 72 водосховищах, 164 річках, 14 зрошувальних системах, 1 лимані та 5 каналах комплексного призначення. Крім того, у рамках радіаційного моніторингу вод водогосподарськими організаціями здійснюється контроль вмісту радіонуклідів у поверхневих водах.

Санітарно-епідеміологічна служба (МОЗ) проводить спостереження за джерелами централізованого та децентралізованого постачання питної води, а також місцями відпочинку вздовж річок та водосховищ.

Підприємствами Державної геологічної служби (Мінприроди) здійснюється моніторинг стану підземних вод. У місцях моніторингу проводиться оцінка рівня залягання підземних вод (наявність), їх природного геохімічного складу. Проводяться визначення 22 параметрів, в тому числі концентрації важких металів та пестицидів.

Санітарно-епідеміологічна служба (МОЗ) здійснює хімічний аналіз підземних вод, які призначаються для питного споживання.

Моніторинг прибережних вод. Державна гідрометеорологічна служба (МНС) управляє мережею моніторингу стану прибережних вод, яка

складається з станцій моніторингу у місцях скиду стічних вод та науково-дослідних станцій, що розташовані на прибережних територіях Чорного та Азовського морів. На існуючих станціях проводяться вимірювання від 16 до 26 гідрохімічних параметрів вод та донних відкладів.

Державні інспекції охорони Чорного та Азовського морів (Мінприроди) мають власні системи спостережень. До їх повноважень відносяться:

– щомісячні відбори проб та аналіз впливу джерел забруднення, які розташовані на узбережжі;

– моніторинг скидів з кораблів;

– забруднення від діяльності з пошуку та видобування нафти, газу і будівельних матеріалів на морському шельфі;

– нагляд за використанням живих ресурсів моря.

Державна санітарно-епідеміологічна служба (МОЗ) здійснює моніторинг якості морської води в зонах рекреаційного та оздоровчого водокористування.

Моніторинг стану ґрунтів. Державна гідрометеорологічна служба (МНС) здійснює моніторинг забруднення ґрунтів сільськогосподарських земель пестицидами та важкими металами у населених пунктах. Проби відбираються раз у п`ять років, проби на важкі метали у містах Костянтинівка та Маріуполь відбираються щороку.

Державна екологічна інспекція (Мінприроди) здійснює відбір проб на промислових майданчиках в межах країни. Загальна кількість параметрів, що вимірюються 27.

Установи МОЗ здійснюють моніторинг стану ґрунтів на територіях їх можливого негативного впливу на здоров`я населення. Найбільше охоплені території вирощення сільськогосподарської продукції, території в місцях застосування пестицидів, ґрунти в зоні житлових масивів, дитячих майданчиків та закладів. Досліджуються проби ґрунту в місцях зберігання

токсичних відходів на території підприємств та поза територією підприємств у місцях їх складування або захоронення.

Мінагрополітики здійснює спостереження за ґрунтами сільськогосподарського використання. Здійснюються радіологічні, агрохімічні та токсикологічні визначення, залишкова кількість пестицидів, агрохімікатів і важких металів.

Моніторинг показників біологічного різноманіття. Через обмежене бюджетне фінансування моніторинг здійснюється тільки за видами, які представляють промисловий інтерес (дерева, риба, дичина).

Підприємства Держкомлісгоспу проводять моніторинг лісової рослинності у 24 областях країни. Здійснюється оцінка біомаси, пошкодження її біотичними та абіотичними чинниками; мисливської фауни, біорізноманіття; радіологічні визначення.

Деякі дослідження здійснюються через надання міжнародної допомоги, або в рамках міжнародних програм.

Моніторинг радіаційного випромінювання. Державна гідрометеорологічна служба (МНС) здійснює спостереження за радіоактивним забрудненням атмосфери шляхом щоденних замірів доз гамма-радіаційної експозиції (ГРЕ), осідання радіоактивних частинок з атмосфери та вмісту радіоактивного аерозолю в повітрі. Здійснюються заміри радіоактивного забруднення поверхневих вод на 8 водних об`єктах. Поблизу атомних електростанцій Державна гідрометеорологічна служба здійснює заміри радіоактивного забруднення поверхневих вод цезієм-137 у та забруднення ґрунтів.

Лабораторії моніторингу Мінагрополітики проводять контроль у місцях концентрації радіоактивних речовин у ґрунтах та харчових продуктах.

МНС здійснює моніторинг доз ГРЕ на 10 автоматизованих пунктах поблизу атомних електростанцій. У межах 30-кілометрової зони навколо Чорнобильської АЕС (зони відчуження), МНС здійснює спостереження за

концентрацією радіонуклідів; радіонуклідами в атмосферних опадах, а також концентрацією «гарячих» частинок у повітрі. Міжнародна радіоекологічна лабораторія Чорнобильського центру атомної безпеки, радіоактивних відходів та радіоекології у Славутичі, здійснює моніторинг впливу радіації на біоту у зоні відчуження.

Інформаційна взаємодія. Суб`єктами ДСМД створені, або розробляються відомчі бази даних моніторингової інформації. Існуюча система інформаційної взаємодії відомчих підсистем моніторингу довкілля передбачає обмін інформацією на загальнодержавному та регіональному рівнях. Організаційна інтеграція суб`єктів моніторингу довкілля на всіх рівнях здійснюється Мінприроди та його територіальними органами.

Для упорядкування процесу обміну інформацією за показниками та термінами надання екологічної інформації між Мінприроди та суб'єктами ДСМД укладено двохсторонні угоди про співробітництво у сфері моніторингу навколишнього природного середовища, до яких розроблені відповідні регламенти обміну екологічною інформацією.

Оперативна моніторингова інформація передається територіальними органами суб'єктів ДСМД до регіональних центрів моніторингу довкілля, або державних управлінь охорони навколишнього природного середовища в регіонах.

Узагальнена аналітична інформація надається міністерствами та відомствами-суб'єктами ДСМД Мінприроди.

Отримані дані передаються до Інформаційно – аналітичного центру Мінприроди та накопичується у банках екологічних даних.

На основі отриманої щомісячної та щоквартальної інформації Мінприроди видається інформаційно – аналітичний огляд „Стан довкілля в Україні ", який розповсюджується серед заінтересованих користувачів.

Функціонування Інформаційно-аналітичного центру Мінприроди забезпечує інформаційний обмін з регіональними центрами моніторингу

довкілля, суб'єктами державної системи моніторингу довкілля, створення уніфікованого банку екологічних даних, проведення комплексного аналізу стану довкілля, тощо.

Постановою Кабінету Міністрів України від 05.12.2007 № 1376 затверджено Державну цільову екологічну програму проведення моніторингу навколишнього природного середовища.

Програма спрямована на поєднання зусиль усіх суб`єктів системи моніторингу щодо виключення дублювання та включення додаткових функцій з моніторингу, створення єдиної мережі спостережень після оптимізації її елементів та програм спостережень, вдосконалення технічного, методичного, метрологічного та наукового забезпечення функціонування єдиної мережі спостережень. З метою забезпечення інтеграції інформаційних ресурсів суб`єктів системи моніторингу довкілля передбачено створення та забезпечення функціонування єдиної автоматизованої підсистеми збору, оброблення, аналізу і збереження даних та інформації, отриманих в результаті здійснення моніторингу.

В межах Державної цільової екологічної програми проведення моніторингу навколишнього природного середовища, у тому числі, передбачено розширення мережі автоматизованих постів спостережень за забрудненням атмосферного повітря в екологічно небезпечних містах.

6.5 Висновок

В розділі описано основні особливості стандарту OHSAS 18001 щодо ведення та управління документацією з охорони праці та заходи щодо відвернення пожежі на робочих місцях користувачів ПК.

Розглянуто оцінку події, що сталася або може статися у прогнозований термін, та визначення ступеня реагування на відповідному рівні управління. Досліджено функціонування державної системи спостереження, збирання, оброблення та аналізу інформації про стан довкілля під час надзвичайних ситуацій мирного та воєнного часу.

7 ЕКОЛОГІЯ

7.1 Методи узагальнення екологічної інформації

Як вже відомо, екологічне дослідження складається з трьох етапів:

– збору інформації;

– обробки інформації;

– узагальнення інформації.

Останній етап – це та стадія роботи з екологічною інформацією, коли оброблений екологічний матеріал потребує узагальнення, наочного подання і відображення складних екологічних ситуацій. Методами наочного подання та викладання фізичних величин, що використовують для більш раціонального та систематизованого викладення цифрової інформації є статистичні таблиці і статистичні графіки, тобто табличний і графічний метод. [67]

7.1.1 Табличний метод в екологічних дослідженнях

Статистичні таблиці – це форма раціонального та систематизованого викладення цифрової інформації. Основною перевагою цифрової інформації, зведеної в таблиці, є компактність, наочність, виразність. Інформація стає легкодоступною і рельєфною, компактною і раціональною. Мета побудови таблиць багатогранна:

– систематизація цифрової інформації;

– полегшення і прискорення ефекту сприйняття;

– інтенсифікація пізнавального процесу;

– економія місця при викладенні інформації.

Таблиці складають не лише на заключному етапі дослідження. В процесі обробки статистичних даних користуються допоміжними, робочими таблицями. їх слід відрізняти від допоміжних розрахункових

таблиць (логарифмічних, таблиць коефіцієнтів). Статистичними таблицями вважають тільки ті, що містять наслідки статистичного аналізу еколого-економічних явищ і процесів. [68]

Таблиця за своїм логічним змістом розглядається як «статистичне речення», що має свій підмет і присудок. Підмет таблиці характеризує об'єкт дослідження, а присудок – це система показників, що відображує підмет як об'єкт.

Статистична таблиця має ряд горизонтальних рядків і вертикальних граф. Перетин рядків і граф утворює клітини таблиці. Ліві бічні і верхні клітини призначені для словесних заголовків, а решта – для числових. Схему статистичної таблиці див. на рисунку 7.1.

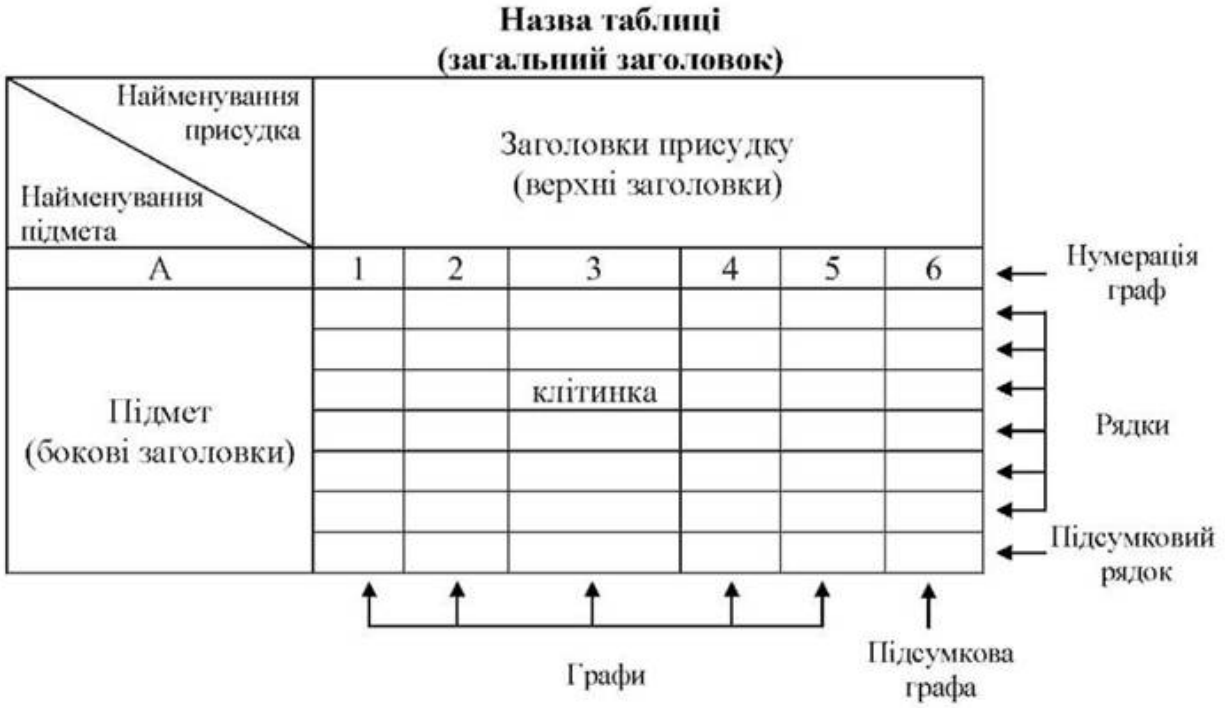

Рисунок 7.1 – Схема статистичної таблиці

Кожна таблиця має три заголовки:

– загальний – відображає зміст таблиці (його місце над таблицею);

– ліві (бічні) – найменування рядків, розкривають зміст підмета;

– верхні – найменування граф, розкривають зміст присудка.

Оптимальний за розміром об'єм таблиці складає 25-30 графо-клітин, який є добутком рядків і граф.

7.1.2 Графічний метод в екологічних дослідженнях

Статистичний графік являє собою рисунок, який описує статистичні сукупності умовною мовою геометричних знаків тієї чи іншої форми: крапок, ліній, площин, фігур та різних їх комбінацій. [69]

Статистичні графіки – це спосіб умовного зображення цифрової інформації у вигляді крапок, ліній, стовпчиків, кругів або фігур. Мета побудови потрійна:

– популяризація цифрової інформації;

– забезпечення доступності сприйняття інформації;

– узагальнення цифрової інформації.

Призначення графіків багатогранне:

– порівняння між собою різних величин;

– характеристика складу, структури і структурних зрушень сукупностей;

– з'ясування ступеня розповсюдження явищ в просторі;

– вивчення взаємозв'язку між явищами і їх ознаками;

– виявлення хронологічних явищ і їх ознак;

– дослідження темпів, тенденцій, закономірностей і перспектив розвитку явищ.

Елементами графіка є:

– графічний образ – це сукупність геометричних або графічних знаків (крапки, лінії, фігури), що замінюють числові дані і використовуються для зображення статистичних даних;

– допоміжні елементи – складові частини, що роз'яснюють суть графічного образу:

– загальний заголовок, який розкриває зміст графіка;

– осі координат, шкали;

– числові дані на шкалах, які потрібні для уточнення значень величин;

– пояснювальні надписи.

Графічні зображення в статистиці можуть бути представлені і негеометричними знаками – силуетами чи малюнками. Наприклад, динаміку книжкової продукції на графіку можна зобразити у вигляді книжкових полиць, інфляційні процеси – у вигляді банкнотів тощо.

У більшості випадків статистичних графіків використовують не об'ємне зображення, складне за побудовою, а площинне. Площинне зображення досить різноманітне за формою і водночас має ті ж самі складові елементи. Просторові орієнтири в статистичних графіках використовують для визначення порядку розміщення геометричних знаків у полі графіка. Вони задаються системою координатних сіток контурних ліній, які ділять це поле на частини. Як правило, в статистиці використовується система прямокутників координат, але іноді може застосовуватися і полярна система (колові графіки).

Експлікація графіка являє собою словесне пояснення основних елементів графіка та його змісту. Вона включає назву графіка, надписи вздовж масштабних шкал, окремі пояснювальні надписи, що розкривають зміст елементів графічного образу. Статистичний графік – це знакова модель, без експлікації його не можна зрозуміти, тобто перенести знання із формалізованої системи характеристики дійсності на саму дійсність.

7.2 Отримання енергії за рахунок альтернативних джерел

До альтернативних джерел енергії належать відновлювальні – вітер, сонячне випромінювання, енергія морів і океанів тощо (див. рисунок 7.2). Їх перевагою є те, що всі вони екологічно чисті.

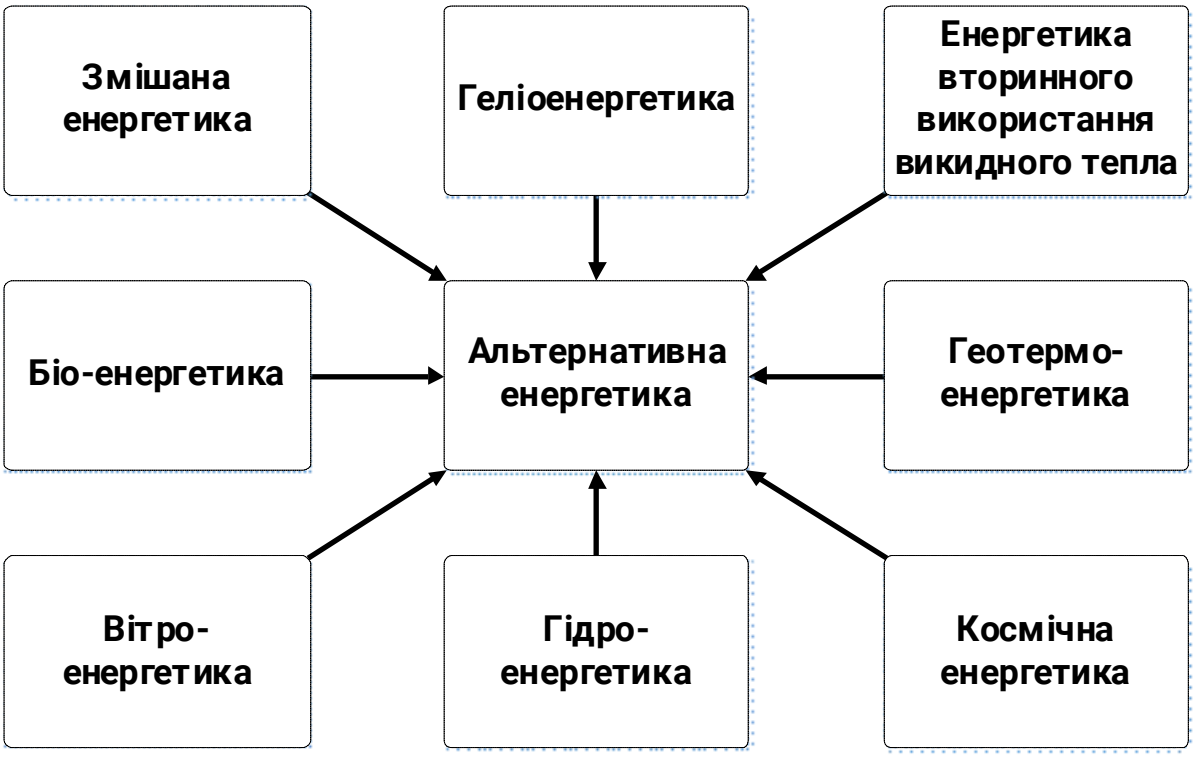

Рисунок 7.2 – Альтернативні джерела енергії

Енергія вітру. За підрахунками вчених, загальний вітроенергетичний потенціал Землі в тридцять разів перевищує річне споживання електроенергії в усьому світі. Однак, використовується лише мізерна частка цієї енергії. Але так було не завжди. За даними статистики, в дореволюційній колишній Російській імперії налічувалось близько тридцять тисяч вітряків. Ця нехитра установка була також атрибутом майже кожного другого села в Україні. Проте парова машина, а потім двигун внутрішнього згоряння витіснили цих скромних трудівників.

Можливості використання цього виду енергії в різних місцях Землі неоднакові. Для нормальної роботи вітрових двигунів швидкість вітру не повинна в середньому за рік падати нижче 4-5 м/с, а краще, коли вона становить 6-8 м/с. Для цих установок шкідливі і надто великі швидкості вітру (урагани), які можуть їх поламати. Найбільш сприятливі зони для використання вітрової енергії – узбережжя морів і океанів, степи, тундра,

гори. В межах України такими ділянками є узбережжя Чорного моря, особливо Крим, а також Карпати, південні степові райони.

Піонером будівництва вітрових електростанцій (ВЕС) у нашій країні до війни був видатний український вчений та інженер, один з основоположників космонавтики Ю.Кондратюк. Побудована ним у 1931 р. поблизу Севастополя ВЕС потужністю 100 кВт, забезпечувала струмом міську мережу понад десять років. Ю.Кондратюк проектував більш потужні ВЕС на 5 і 10 тис. кВт, та розпочалась війна, Кондратюк пішов добровольцем на фронт і загинув у 1941 р., а проекти його ВЕС було покладено "під сукно".

Нині на Заході, особливо в Данії та США, серійно випускаються невеликі ВЕС потужністю від 1,5 до 100 кВт. Побудовано кілька експериментальних ВЕС потужністю до 30 тис. кВт. Втілюється інша технічна ідея Ю.Кондратюка, який запропонував свого часу будувати ВЕС разом з установками по виробництву водню шляхом електролізу води. Тоді, коли потреба в електроенергії нижча, "зайва" потужність ВЕС спрямовується на виробництво надзвичайно цінного енергетичного продукту – водню. Водень може використовуватися як пальне для автомобілів, а також замість природного газу у багатьох інших установках, причому внаслідок його згоряння не утворюються шкідливі речовини, а лише водяна пара.

Під час роботи ВЕС навколишнє середовище не зазнає жодних забруднень. Єдині негативні впливи – це низькочастотний шум (гудіння) працюючих вітряків та ще гибель птахів, що потрапляють у лопасті двигунів.

Енергія морів і океанів. Світовий океан містить велетенський енергетичний потенціал. Це, по-перше, енергія Сонця, поглинута океанською водою, що виявляється в енергії морських течій, хвиль, прибою, різниці температур різних шарів морської води і, по-друге, енергія тяжіння Місяця й Сонця, яка спричиняє морські припливи й відпливи. Використовується цей великий і екологічно чистий потенціал ще вкрай мало.

Одну з перших електростанцій, що використовує енергію морських хвиль, було побудовано ще в 1970 р. поблизу норвезького міста Бергена. Вона має потужність 350 кВт і забезпечує енергією селище з 100 будинків. Можливості створення більш потужних хвильових станцій досліджуються вченими Великобританії, США та Японії. А румунські вчені провели вдалі досліди з установками для перетворення енергії морських хвиль на електроенергію на Чорному морі, яке поблизу узбережжя Румунії нічим не відрізняється (з енергетичної точки зору) від того, що омиває береги України.

Усі типи морських хвильових електростанцій, що будуються і діють сьогодні, побудовані за єдиним принципом: у спеціальному буї-поплавку під дією хвилі коливається рівень води. Це призводить до стискання в ньому повітря, яке рухає турбіну. В експериментальних електростанціях навіть невеликі хвилі висотою 35 см примушують турбіну розвивати швидкість понад 2 тис. обертів за хвилину. Метрової висоти хвиля забезпечує від 25 до 30 кВт енергії, а в деяких частинах Світового океану, наприклад, у Тихому океані, можна одержати до 90 кВт.

Іншим різновидом морських електростанцій с установки, що перетворюють енергію морського прибою. Крім згаданого поплавкового принципу, такі станції використовують також принцип накачки сильним прибоєм морської води в резервуар, розташований вище рівня моря. Звідти вода спускається вниз, крутячи турбіни енергоустановок.

У океані подекуди досить близько розташовані шари води з різною температурою. Найбільш значною (до 22С) різниця температури є в тропічній зоні світового океану. На цьому явищі базується принцип одержання електроенергії. В спеціальний теплообмінник закачується насосами холодна глибинна вода і нагріта Сонцем поверхнева. Робочий агент (фреон), яку домашньому холодильнику, почергово випаровується та переходить у рідкий стан у різних частинах теплообмінника. Пара фреону рухає турбіну генератора. Нині така установка потужністю 100 кВт працює

на тихоокеанському острові Науру, забезпечуючи енергопотреби населення цього острова.

Нарешті, розроблені і вже діють електростанції, що використовують енергію морських припливів. Вигідними вони є в таких ділянках узбережжя Світового океану, де припливи бувають найвищими. До таких ділянок належать канадська затока Франції (висота припливу становить 17м). протока Ла-Манш (15м), Пенжинська затока Охотського моря (13м) тощо. На узбережжі Чорного моря висота припливу дуже незначна. Нині споруджено і працює кілька припливний станцій: у гирлі р. Рані на узбережжі Ла-Маншу (Франція) потужністю 240 тис. кВт і Кислогубська в Кольській затоці (Росія) потужністю 400 кВт.

Широке впровадження морських електростанцій різних типів стримується відносно високою їх вартістю. Проте, вчені дійшли висновку, що їх енергетичний баланс (співвідношення одержаної та затраченої енергії) може бути більш високим, ніжу деяких АЕС і ТЕС, що працюють на вугіллі та нафті. Розрахунки й проекти інженерів свідчать, що в найближчому майбутньому можливе спорудження великих електростанцій такого типу. Привертають увагу проекти електростанцій, розташованих на плавучих установках вдалині від берега. В деяких проектах пропонується одержувати енергію на таких станціях комплексним способом (наприклад, за рахунок хвиль, різниці температур, а також вітру та Сонця). Ця енергія може використовуватися для виробництва водню або передаватися на берег по підводному кабелю.

Робота згаданих електростанцій не спричиняє забруднення навколишнього середовища, зокрема й теплового, бо вони лише перетворюють акумульовану в хвилях, припливах тощо енергію Сонця й Місяця на інші види енергії, зокрема електричну.

7.3 Висновок до розділу

В даному розділі описано та проаналізовано методи узагальнення екологічної інформації. Зокрема досліджено табличний та графічний методи в екологічних дослідженнях.

Окремо подано дослідження щодо отримання енергії за рахунок альтернативних джерел. Зокрема розглянуто відновлювальні джерела енергії, вітер, сонячне випромінювання, енергія морів і океанів тощо.

ВИСНОВКИ

В процесі виконання дипломної роботи освітнього рівня «магістр» була виконана організація хмарної платформи «розумного міста» для уникнення повторів колекцій даних. В процесі аналізу предметної області:

– досліджено інноваційні проекти класу «розумне місто»;

– виконано огляд та проаналізовано способи зменшення вартості зберігання даних в інноваційних проектах «розумних міст»;

– розкрито зміст інформаціної технології «Дедуплікація» даних в хмарних проектах «розумних міст».

В другому розділі дипломної роботи:

– подано порівняння способів «уникнення повторів» застосовних в проектах «розумних міст»;

– запропоновано архітектуру проектованої хмарної платформи «розумного міста» для уникнення повторів колекцій даних;

– проаналізовано ключові особливості файлової системи «OpenDedup» з автоматичним об’єднанням дублікатів даних;

– досліджено архітектуру екосистеми «Hadoop» в контексті її використання для проектів «розумних міст»;

– подано Огляд «HBase» для проектів «розумних міст» та описано реалізацію методу уникнення повторів даних в проектах класу «розумне місто»;

– в розділі розроблений алгоритм MapReduce для уникнення повторів великих даних;

– описано підготовку та налаштування операційного середовища «Hadoop».

В третьому розділі дипломної роботи:

– описано процес запуску кластера «Hadoop» в «Docker»-контейнерах для потреб «розумних міст»;

– розглянуто процес запуску «spark RDD» в середовищі «Spark» для проектів класу «розумне місто»;

– висвітленоу практичну реалізацію методу уникнення повторів для потреб проектів класу «розумне місто».

В розділі «Спеціальна частина» описано процеси виконання кластерних експериментів.

В розділі «Обґрунтування економічної ефективності» розраховано основні техніко-економічні показники проведених досліджень.

В розділі «Охорона праці та безпека в надзвичайних ситуаціях» описано основні особливості стандарту OHSAS 18001 щодо ведення та управління документацією з охорони праці та розглянуто заходи щодо відвернення пожежі на робочих місцях користувачів ПК. Висвітлено питання оцінки події, що сталася або може статися у прогнозований термін, та визначення ступеня реагування на відповідному рівні управління. Описано процес пункціонування державної системи спостереження, збирання, оброблення та аналізу інформації про стан довкілля під час надзвичайних ситуацій мирного та воєнного часу.

В розділі «Екологія» описано методи узагальнення екологічної інформації та отримання енергії за рахунок альтернативних джерел.

ПЕРЕЛІК ДЖЕРЕЛ

1 O. Duda, N. Kunanets, O. Matsiuk, and V. Pasichnyk, "Cloud-based IT Infrastructure for “Smart City” Projects", in *Dependable IoT for Human and Industry: Modeling, Architecting, Implementation*. River Publishers, pp. 389-410, 2018. ISBN: 978-87-7022-013-2.

2 N. Kunanets, V. Pasichnyk, H. Lypak, and O. Duda, "Modeling of consolidated information resource for social data institutions", *Econtechmod an international quarterly journal*, vol. 6, №. 3, pp. 25-30, 2017. ISSN:2084–5715.

3 Kitchin, Rob. "The real-time city? Big data and smart urbanism." GeoJournal 79.1 (2014): 1-14.

4 Greenfield, Adam. Everyware: The dawning age of ubiquitous computing. New Riders, 2010.

5 Hancke, Gerhard P., and Gerhard P. Hancke Jr. "The role of advanced sensing in smart cities." Sensors 13.1 (2012): 393-425.

6 Schaffers, Hans, et al. "Smart cities and the future internet: Towards cooperation frameworks for open innovation." The future internet assembly. Springer, Berlin, Heidelberg, 2011.

7 О. М. Дуда та ін., "Актори та діаграми прецедентів системи консолідації соціокомунікаційних інформаційних ресурсів «розумних міст»", *Науковий вісник НЛТУ України*, вип. 27(10), с. 129-136, 2017. ISSN 2519-2477.

8 Kourtit, Karima, Peter Nijkamp, and Daniel Arribas. "Smart cities in perspective–a comparative European study by means of self-organizing maps." Innovation: The European journal of social science research 25.2 (2012): 229-246.

9 D. Tabachyshyn, N. Kunanets, M. Karpinski, O. Duda, and O. Matsiuk, "Information Systems for Processes Maintenance in Socio-communication and

Resource Networks of the Smart Cities", in *Advances in Intelligent Systems and Computing III*, vol. 871, pp 192-205, 2019. ISSN 2194-5365.

10 Gaur, Aditya, et al. "Smart city architecture and its applications based on IoT." Procedia computer science 52 (2015): 1089-1094.

11 O. Duda, N. Kunanets, O. Matsiuk, and V. Pasichnyk, "Information-Communication Technologies of IoT in the "Smart Cities" Projects", *CEUR Workshop Proceedings*, vol. 2105, pp. 317-330, 2018. ISSN 1613-0073.

12 O. Duda, N. Kunanets, O. Matsiuk, V. Pasichnyk, and I. Popyk., "Geoinformational components of mobile appliances for “Smart City” problem solution: current state and prospects", *Econtechmod an international quarterly journal*, vol. 7, №. 2, pp. 31-38, 2018. ISSN:2084–5715.

13 V. Pasichnyk et al., "Telecommunication Infrastructures for Telemedicine in Smart Cities”, *IDDM 2018 Informatics & Data-Driven Medicine*, vol. 2255, pp. 256-266, 2018. ISSN 1613-0073.

14 O. Duda, O. Matsiuk, M. Karpinski, N. Veretennikova, N. Kunanets, and V. Pasichnyk, "Information Technologies of Internet Devices and BigData in the “Smart Cities” Projects", in Proc. *13 Intern Scientific and Techn. Conf. on Computer Science and Information Technologies (CSIT)*, vol. 2, Lviv, 2018, pp. 72-75. ISBN: 978-1-5386-6465-0.

15 N. Shakhovska, O. Duda, O. Matsiuk, Y. Bolyubash, and R. Vovnyanka "Analysis of the Activity of Territorial Communities Using Information Technology of Big Data Based on the Entity-Characteristic Mode", in *Advances in Intelligent Systems and Computing III*, vol 871, pp. 155-170, 2019. ISSN 2194-5365.

16 A. Kharchenko, et al., "Multicriteria Choice of Software Architecture Using Dynamic Correction of Quality Attributes", *Advances in Computer Science for Engineering and Education II*, vol. 938, 419-427, 2019. ISSN 2194-5365.

17 Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of Things for SmartCities. IEEE Internet Things J. 1(1), 22–32 (2014).

18 Hofer, S.: Smart Cities and the Internet of Things, pp. 485–494 (2014).

19 Kitchin, R. The real-time city? Big data and smart urbanism. GeoJournal 2014, 79, 1–14.

20 Greenfield, A. Everyware: The Dawning Age of Ubiquitous Computing; New Riders: San Francisco, CA, USA, 2010.

21 Hancke, G.P.; Hancke, G.P., Jr. The role of advanced sensing in smart cities. Sensors 2012, 13, 393–425.

22 Schaffers, H.; Komninos, N.; Pallot, M.; Trousse, B.; Nilsson, M.; Oliveira, A. Smart cities and the future internet: Towards cooperation frameworks for open innovation. In Future Internet Assembly; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6656, pp. 431–446.

23 Kourtit, K.; Nijkamp, P.; Arribas, D. Smart cities in perspective—A comparative European study by means of self-organizing maps. Innov. Eur. J. Soc. Sci. Res. 2012, 25, 229–246.

24 O. Duda, V. Kochan, N. Kunanets, O. Matsiuk, V. Pasichnyk, and A. Sachenko, "Data Processing in IoT for Smart City Systems", in *Proc. 10th IEEE Intern. Conf. on. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2019)*, Metz, 2019. pp. 96-99.

25 V. Kochan et al, N. Kunanets, V. Pasichnyk, O. Roshchupkin, Anatoliy Sachenko, Iryna Turchenko, Oleksij Duda, Vita Semaniuk, Svitlana Romaniv, Oleksandr Matsiuk Sensing in IoT for Smart City Systems in *Proc. 10th IEEE Intern. Conf. on. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2019)*, Metz, 2019 pp. 579-586.

26 A. Aurigi, “New Technologies, Same Dilemmas: Policy and Design Issues for the Augmented City,” Journal of Urban Technology 13: 3 (2006) 5–28.

27 J. Bélissent, “Smart City Leaders Need Better Governance Tools,” Forrester (2011) <https://www. forrester.com/report/Smart+City+Leaders+ Need+Better+Governance+Tools/-/E-RES58966> Accessed April 20, 2017.

28 A. Caragliu and C. Del Bo, “Smartness and European Urban Performance: Assessing the Local Impacts of Smart Urban Attributes,” The European Journal of Social Science Research 25: 2 (2012) 97–113.

29 M. Angelidou, N. Gountaras, and P. Tarani, “Engaging Digital Services for the Creation of Urban Knowledge Ecosystems: The Case of Thermi, Greece,” International Journal of Knowledge-Based Development 3: 4 (2012) 331–350.

30 G. Giffinger, C. Fertner, H. Kramar, et al., Smart Cities; Ranking of European Medium-sized Cities, Report of the Centre of Regional Science, Vienna University of Technology (Vienna: 2007) <http://www.smart-cities.eu/download/smart_cities_final_report.pdf> Accessed April 20, 2017.

31 E.L. Glaeser, and C.R. Berry, Why are Smart Places Getting Smarter? Taubman Centre Policy Brief 2006-2 (Cambridge, MA: Taubman Centre, 2006).

32 J. Bélissent, “Getting Clever About Smart Cities: New Opportunities Require New Business Models,” Forrester (2010) <https://www.forrester.com/report/Getting+Clever+About+Smart+Cities+New +Opportunities+Require+New+Business+Models/-/E-RES56701> Accessed April 20, 2017.

33 T. Bakici, State of the Art: Open Innovation in Smart Cities, Report of the ICT Policy Support Programme of the European Commission “Open Innovation Mechanisms in Smart Cities” (2012) <http://opencities.net/sites/opencities.net/files/content-files/repository/D1.1%20 State% 20of%20the%20Art_Open%20Innovation.pdf> Accessed April 20, 2017.

34 F. Cugurullo, “How to Build a Sandcastle: An Analysis of the Genesis and Development of Masdar City,” Journal of Urban Technology 20: 1 (2013) 23–37.

35 A. Townsend, A. S. K. Pang, and R. Weddle, Future Knowledge Ecosystems: The Next Twenty Years of Technology-Led Economic Development, Report of the Institute for the Future (2009) <http:// www.iftf.org/our-work/people-technology/technology-horizons/future-knowledge-ecosystems/> Accessed July 12, 2017.

36 Osman, Ahmed M. Shahat. "A novel big data analytics framework for smart cities." *Future Generation Computer Systems* 91 (2019): 620-633.

37 Osman, Ahmed M. Shahat. "A novel big data analytics framework for smart cities." *Future Generation Computer Systems* 91 (2019): 620-633.

38 Wang, Tian, et al. "Coupling resource management based on fog computing in smart city systems." *Journal of Network and Computer Applications* 135 (2019): 11-19.

39 Oh, A. S. "Designing smart supplier chain management model under big data and internet of things environment." *International Journal of Recent Technology and Engineering* 8.2 (2019): 290-294.

40 Русин, Богдан Павлович, et al. "АРХІТЕКТУРА СИСТЕМИ ДЕДУБЛІКАЦІЇ ТА РОЗПОДІЛУ ДАНИХ У ХМАРНИХ СХОВИЩАХ ПІД ЧАС РЕЗЕРВНОГО КОПІЮВАННЯ." *Інформаційні технології та комп'ютерна інженерія* 45.2 (2019): 40-63.

41 Hedau, Khushboo, et al. "Smart Cloud based Healthcare Application having Data Authentication Control using Time Constraint." *IJIRMPS-International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences* 7.2 (2019).

42 Ulusar, Umit Deniz, Deniz Gul Ozcan, and Fadi Al-Turjman. "Open Source Tools for Machine Learning with Big Data in Smart Cities." *Smart Cities Performability, Cognition, & Security*. Springer, Cham, 2020. 153-168.

43 Alsharkawy, Nader, Eyad Nawar, and Bassem Mokhtar. "Smart Cloud Platform for Data Management in the Age of the Internet of Vehicles." *2019 Novel*

Intelligent and Leading Emerging Sciences Conference (NILES). Vol. 1. IEEE, 2019.

44 Khare, Abhinav, et al. "Toward a Trustless Smart City: the# SmartME Experience." *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2019.

45 González-Briones, Alfonso, et al. "Internet of things platform to encourage recycling in a smart city." (2019).

46 Osman, Ahmed M. Shahat. "A novel big data analytics framework for smart cities." *Future Generation Computer Systems* 91 (2019): 620-633.

47 ur Rehman, Muhammad Habib, et al. "The role of big data analytics in industrial Internet of Things." *Future Generation Computer Systems* 99 (2019): 247-259.

48 Alsaig, Alaa, et al. "Characterization and Efficient Management of Big Data in IoT-Driven Smart City Development." *Sensors* 19.11 (2019): 2430.

49 Del Esposte, Arthur de M., et al. "Design and evaluation of a scalable smart city software platform with large-scale simulations." *Future Generation Computer Systems* 93 (2019): 427-441.

50 Mohebali, Behshad, et al. "A big data inspired preprocessing scheme for bandwidth use optimization in smart cities applications using Raspberry Pi." *Big Data: Learning, Analytics, and Applications*. Vol. 10989. International Society for Optics and Photonics, 2019.

51 Bhattacharjya, Aniruddha, et al. "Secure IoT structural design for smart homes." *Smart Cities Cybersecurity and Privacy*. Elsevier, 2019. 187-201.

52 Brisebois, Ronald, et al. "LUCY: A Smart City Mobile Application Architecture Model for Multidimensional Interests Spaces and Personas using Machine Learning Models." *Asian Journal of Science and Technology* 10.09 (2019): 10067-10078.

53 Szubbocsev, Zoltan. "Re-routing autonomous vehicles using dynamic routing and memory management." U.S. Patent No. 10,495,474. 3 Dec. 2019.

54 Kalaitzakis, Manos, et al. "Building a Smart City Ecosystem for Third Party Innovation in the City of Heraklion." *Mediterranean Cities and Island Communities*. Springer, Cham, 2019. 19-56.

55 Szubbocsev, Zoltan. "Power management, dynamic routing and memory management for autonomous driving vehicles." U.S. Patent Application No. 15/933,260.

56 Singh, Praveen Kumar, Rajesh Kumar Verma, and PESN Krishna Prasad. "IoT-Based Smartbots for Smart City Using MCC and Big Data." *Smart Intelligent Computing and Applications*. Springer, Singapore, 2019. 525-534.

57 Muralidharan, Shapna, Gyuwon Song, and Heedong Ko. "Monitoring and Managing IoT Applications in Smart Cities Using Kubernetes." *CLOUD COMPUTING 2019* (2019): 11.

58 Ogawa, Keigo, et al. "IoT Device Virtualization for Efficient Resource Utilization in Smart City IoT Platform." *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2019.

59 Onet, Razvan, et al. "Automatic Deployment of a Network Overlay in an Intelligent Transportation System: Docker and Open Baton Approach." *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2019.

60 Kalluri, Balaji, et al. "A cyber-physical middleware platform for buildings in smart cities." *Advances in Informatics and Computing in Civil and Construction Engineering*. Springer, Cham, 2019. 645-652.

61 Wang, Ke, et al. "Building an efficient storage model of spatial-temporal information based on HBase." *Journal of Spatial Science* 64.2 (2019): 301-317.

62 Zhao, Hang, et al. "A DAG Refactor Based Automatic Execution Optimization Mechanism for Spark." *IFIP International Conference on Network and Parallel Computing*. Springer, Cham, 2019.

63 Зеркалов, Д. В. "Охорона праці в галузі (загальні вимоги)." (2011).

64 Жидецький, В. Ц., В. С. Джигирей, and О. В. Мельников. "Основи охорони праці." *Львів: Афіша* 350 (2000): 132-136.

65 Желібо, Євген Петрович, and І. С. Сагайдак. "Безпека життєдіяльності." (2011).

66 Депутат, О. П., І. В. Коваленко, and І. С. Мужик. "Цивільна оборона. Підручник/За ред. Полковника ВС Франчука.–2-ге вид., доп." *Львів, Афіша* (2001).

67 Бобильов, Ю. П., et al. "Екологія: базовий підручник для студентів вищих навчальних закладів." (2014).

68 Юрченко, Любов Іванівна. "Екологія." (2009).

69 Малимон, С. С. Основи екології: Підручник.–Вінниця. Нова Книга, 2009.