НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК 004.9

«До захисту допущено»
В.о. завідувача кафедри
_____ О.І. Ролік
«___»_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: «Архітектура системи комплексного аналізу даних на основі Big Data технологій»

Виконав:
студент ІІ курсу, групи ІА-61м
Рижко Борис Володимирович _____

Керівник:
Доцент, к.т.н, доцент
Катін Павло Юрійович _____

Рецензент:
Доц. каф. АПЕПС, к.т.н.
Михайлова І. Ю. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ С.Ф. Теленик
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Рижко Борису Володимировичу

1. Тема дисертації «Архітектура системи комплексного аналізу даних на основі Big Data технологій», науковий керівник дисертації Катін Павло Юрійович, доцент, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – великі дані.

4. Предмет дослідження – архітектура системи комплексного аналізу даних.

5. Перелік завдань, які потрібно розробити: дослідження відомих інфраструктур розподілених обчислень для роботи з Big Data і обрання на підставі проведених досліджень відповідної інфраструктури; розроблення сценаріїв використання програмної системи; розроблення структури програмної системи; розроблення архітектури системи комплексного аналізу даних; розроблення тестових модулів і бази даних; написання пояснювальної записки.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: структура програмної системи; діаграми варіантів використання програмної системи адміністратором; діаграми варіантів використання програмної системи оператором; архітектура системи комплексного аналізу даних; блок-схема роботи модуля збору інформації; діаграма класів модуля обробки подій; діаграма класів модуля аналізу.

7. Перелік публікацій: 1) Розробка архітектури системи автоматизованого збору, обробки та аналізу даних на основі технології Big Data : матеріали V міжнар. наук.-

практ. конф. з інформаційних систем та технологій, 1-2 грудня 2017 р., Київ / відп. ред. А. В. Писаренко. – К.: Вид-во ТОВ «Інжиніринг», 2017. – 76 с. 2) Рижко Б. Розроблення системи автоматизованого збору, оброблення та аналізу даних на основі технологій Big Data [Текст] / Б. Рижко, П. Катін // Інфокомунікаційні системи та технології. – 2018. – № 2(2). – С. 37-41.

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Дослідження відомих інфраструктур розподілених обчислень для роботи з Big Data і обрання на підставі проведених досліджень відповідної інфраструктури	15.03.18-31.03.18	
2	Розроблення сценаріїв використання програмної системи	01.04.18-10.04.18	
3	Розроблення структури програмної системи	10.04.18-15.04.18	
4	Розроблення архітектури системи комплексного аналізу даних	15.04.18-28.04.18	
5	Розроблення тестових модулів і бази даних	29.04.18-03.05.18	
6	Написання пояснювальної записки	04.05.18-14.05.18	

Студент Б.В. Рижко

Науковий керівник дисертації П.Ю. Катін

РЕФЕРАТ

Магістерська дисертація освітньо-кваліфікаційного рівня «магістр» на тему «Архітектура системи комплексного аналізу даних на основі Big Data технологій» містить: 144 с., 26 рис., 54 таб., 5 додатків, 58 джерел.

Об’єктом дослідження є великі дані, що потребують комплексного аналізу, для вилучення корисної інформації, якою компанія може скористуватися для збільшення власних прибутків.

Метою дослідження є модель ПС на основі АСКАД. Крім того, при проведені досліджень ураховувалися такі характеристики АСКАД як можливість налаштування, масштабованість у відповідності до щорічно зростаючих обсягів даних.

Методи дослідження. Емпіричні дослідження відомих ІРО дозволили обрати відповідну ІРО для основи АСКАД. За результатами дослідження було формалізовано структуру ПС на основі АСКАД, її сценарії використання та безпосередньо АСКАД.

Програмне тестування прототипу ПС на основі розробленої АСКАД дозволили перевірити практичну важливість отриманих результатів.

У ході досліджень отримана нова узагальнена структура АСКАД для побудови ПС з метою обробки великих обсягів даних, що може бути використана як основа для реалізації аналогічних рішень і продовження подальших досліджень.

Результати роботи оприлюднені на V Міжнародній науково-практичної конференції Winter InfoCom Advanced Solutions 2017, в науково-практичному виданні «Інфокомунікаційні системи та технології» № 2(2) 2018 р.

ВЕЛИКІ ДАНІ, СИСТЕМА КОМПЛЕКСНОГО АНАЛІЗУ ДАНИХ, ТЕХНОЛОГІЇ BIG DATA.

ABSTRACT

Master’s thesis "Architecture of the system of complex analysis of data on the basis of Big Data Technologies" consists of: 144 pages, 26 figures, 54 tables, 5 appendices, 58 sources.

The subject of the study is the big data that needs a comprehensive analysis to extract useful information that the company can use to increase its own profits.

The purpose of the study is a model of the program system based on architecture of the system of complex data analysis based on the results of scientific analysis. In addition, the research has taken into account such characteristics of the architecture as the ability to adjust, scalability in accordance with the annual growth of data volumes.

Research methods. Empirical studies of well-known distributed computing systems have allowed to select the appropriate system for the basis of the architecture. According to the results of the study, the structure of the program system on the basis of the developed architecture was formalized, its use scenarios and the architecture.

The software testing of the prototype of the program system on the basis of the developed architecture allowed to verify the practical importance of the results obtained.

In the course of research, a new generalized structure of big data analysis system was obtained for the construction of the program system in order to process large amounts of data, which can be used as a basis for implementing similar solutions and the continuation of further research.

The results of the work were announced at the Winter InfoCom Advanced Solutions 2017 Conference, in the scientific journal Infocommunication Systems and Technologies No. 2 (2), 2018.

BIG DATA, BIG DATA ANALYSIS, BIG DATA TECHNOLOGY, ANALYSIS SYSTEMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АІРО – апаратна інфраструктура розподілених обчислень

АСКАД – архітектура системи комплексного аналізу даних

ІРО – інфраструктура розподілених обчислень

КС – кластерна система

ОС – операційна система

ПЗ – програмне забезпечення

ПС – програмна система

СКБД – система керування базами даних

ФС – файлова система

ACL – Access Control List

API – Application program interface

BD – Big Data

BI – Business Intelligence

FIFO – First In First Out

HDFS – Hadoop distributed file system

HTML – Hypertext Markup Language

IDE – Integrated development environment

JSON – JavaScript Object Notation

JVM – Java Virtual Machine

MPI – Message Passing Interface

RMA – Remote Memory Access

RPC – Remote Procedure Call

RSS – Rich Site Summary

TCP – Transmission Control Protocol

UC – Use Case

YARN – Yet Another Resource Negotiator

ВСТУП

Актуальність теми

На теперішній час суттєво зростає обсяг даних, що потрібно аналізувати, обробляти і використовувати для комерції з метою підвищення конкурентоздатності компаній. Ця проблема характерна також для провадження конструкторських робіт, у наукових дослідженнях, тощо. Отже актуальним завданням є дослідження відповідних технологій для розробки АСКАД на основі BD технологій, що призначені для побудови ПС комплексного аналізу. Дані ПС призначені для більш швидкої і ефективної обробки великого обсягу даних.

В роботі представлена АСКАД, проведено тестування на прототипі ПС на основі АСКАД для обробки великих обсягів даних. Архітектура розроблена з використанням відкритого ПЗ. Це надає значну перевагу над комерційними рішеннями. Результати роботи дозволяють побудувати власну ефективну ПС на основі АСКАД для комплексного аналізу великих обсягів даних з мінімальними витратами. AIPO можна реалізувати у тому числі і з використанням комп’ютерів загального призначення.

Мета і задачі дослідження

Метою дослідження є модель ПС на основі АСКАД. При проведені досліджень ураховувалися такі характеристики АСКАД як можливість налаштування, масштабованість у відповідності до щорічно зростаючих обсягів даних [22]. Під час досліджень були виконані наступні задачі:

– дослідження відомих IPO для роботи з BD і обрання на підставі проведених досліджень відповідної IPO для розробки АСКАД;

– розроблення діаграми варіантів використання ПС на основі АСКАД;

– розроблення структури ПС на основі АСКАД на базі обраної IPO;

– розроблення АСКАД у відповідності зі структурою ПС;

– розроблення тестових модулів загальної ПС на основі АСКАД для обґрунтування всіх елементів АСКАД і підтвердження працездатності ПС на основі розробленої АСКАД;

– розроблення тестової схеми бази даних для модулів;

– остаточне формування структури ПС на основі АСКАД на основі результатів тестування з урахуванням інтерфейсів взаємодії між компонентами АСКАД і прототипом ПС.

При розробці АСКАД досліджено існуючі технології BD, що призначені для вирішення такого роду задач.

Об’єктом дослідження є великі дані, що потребують комплексного аналізу, для вилучення корисної інформації, якою компанія може скористуватися для збільшення власних прибутків.

Предметом дослідження є безпосередньо АСКАД, що наочно забезпечує взаємозв’язок між компонентами складної ПС.

Методи дослідження. Емпіричні дослідження відомих ІРО дозволили обрати відповідну ІРО для основи АСКАД. За результатами дослідження було формалізовано структуру ПС на основі АСКАД, її сценарії використання та безпосередньо АСКАД.

Програмне тестування прототипу ПС на основі розробленої АСКАД дозволили перевірити практичну важливість отриманих результатів.

Наукова новизна одержаних результатів

У ході досліджень отримана нова узагальнена структура АСКАД для побудови ПС з метою обробки великих обсягів даних, що може бути використана як основа для реалізації аналогічних рішень і продовження подальших досліджень.

Практичне значення одержаних результатів

Практичне значення роботи полягає в тому, що розроблена АСКАД може бути використана для комплексного аналізу великих обсягів даних без використання спеціального технічного забезпечення (серверів). Окрім відсутності необхідності в спеціальному обладнанні для побудови КС, в розробленій АСКАД використовуються

безоплатні компоненти, побудовані на принципах відкритого ПЗ. Отже, в даній роботі представлена АСКАД, що дозволяє побудувати недорогу КС, що може бути використана для комплексного аналізу даних, а добута корисна інформація сприятиме підвищенню конкурентоспроможності компанії.

Апробація результатів роботи

Результати роботи оприлюднені на V Міжнародній науково-практичній конференції Winter InfoCom Advanced Solutions 2017, в науково-практичному виданні «Інфокомунікаційні системи та технології» № 2(2) 2018 р.

Публікації

Розробка архітектури системи автоматизованого збору, обробки та аналізу даних на основі технології Big Data : матеріали V міжнар. наук.-практ. конф. з інформаційних систем та технологій, 1-2 грудня 2017 р., Київ / відп. ред. А. В. Писаренко. – К.: Вид-во ТОВ «Інжиніринг», 2017. – 76 с.

Рижко Б. Розроблення системи автоматизованого збору, оброблення та аналізу даних на основі технологій Big Data [Текст] / Б. Рижко, П. Катін // Інфокомунікаційні системи та технології. – 2018. – № 2(2). – С. 37-41.

1 ВЕЛИКІ ДАНІ

Кількість даних, що генеруються кожен день у світі, зростає. Зростає доля даних, що поступає з цифрових засобів масової інформації, соцмереж та «інтернету речей». Темпи зростання даних вражають, і ці дані надходять зі швидкістю, різним степенем структурованості і містять велику кількість інформації, яка може бути ключовою для досягнення мети конкуруючих компаній. Можливість аналізу цієї величезної кількості даних являє собою нову еру зростання продуктивності, інновацій.

Великі дані - це термін для набору масивів даних, настільки великих і складних, що їх важко обробити за допомогою традиційних інструментів управління базою даних або додатків обробки даних.[1].

1.1 Атрибути великих даних

Зазвичай, для опису різних аспектів великих даних використовуються для три атрибути (рис. 1.1):

– обсяг,

– швидкість,

– структурованість.

Ці три атрибути дозволяють визначити природу даних, доступних для аналізу[2].

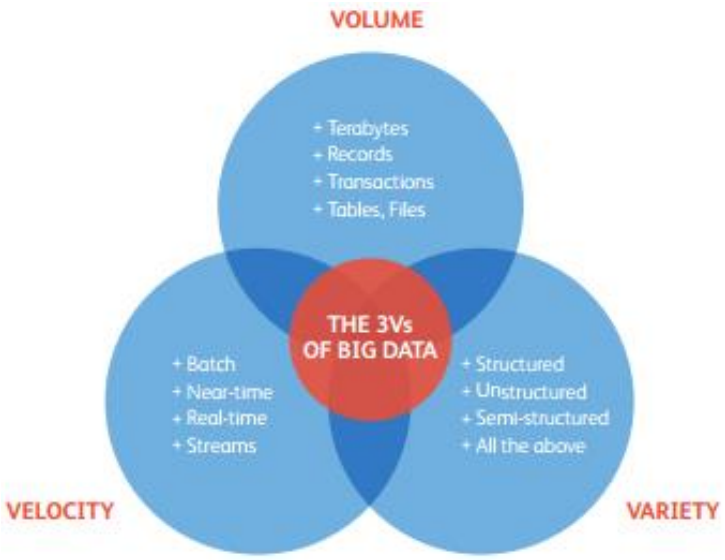


Рисунок 1.1 – Атрибути великих даних [3]

1.1.1 Обсяг даних

Обсяг є найбільш складним аспектом великих даних, оскільки вимагає необхідності масштабованого зберігання та розподіленого підходу до запитів.

Великі компанії вже володіють великою кількістю даних, накопичених протягом багатьох років. Вони можуть бути представлені у формі системних журналів, ведення обліку тощо. Ці дані сягають таких обсягів, що реляційні СКБД не впораються з ними. Рішення на базі «data warehouse» може не обов'язково мати можливість обробляти та аналізувати ці дані через брак архітектури паралельних обчислень.

Багато чого може бути отримано з текстових даних, даних геолокації або файлів журналів. Наприклад, комунікативні шаблони по електронній пошті, споживчі вподобання та тенденції, дослідження безпеки. Технології BD пропонують рішення для отримання корисної інформації з цих величезних обсягів даних.

1.1.2 Швидкість даних

Дані надходять користувачам з великою швидкістю. Веб та мобільні технології дозволили компаніям налагодити зворотній зв’язок зі своїми клієнтами.

Онлайн магазини революціонізували спосіб взаємодії споживачів та постачальників. Інтернет-магазини тепер можуть вести журнали всіх операцій користувачів, зберігати історію цих операцій, і швидко використовувати ці дані для надання рекомендацій, що підвищує конкурентоздатність компанії. Інтернет-маркетингові організації отримують багато переваг, маючи можливість швидко отримувати прогнозовану інформацію. З винаходом смартфонів в світі генерується багато геолокаційних даних, і стала важливою можливість скористатися перевагами великих обсягів цих даних.

1.1.3 Структурованість даних

Дані, що генеруються соціальними мережами та цифровими медіа, зазвичай не є структурованими.

Неструктуровані текстові документи, відео, аудіо дані, зображення, фінансові операції, взаємодії в соціальних мережах є прикладами неструктурованих даних.

Реляційні СКБД підтримують можливість збереження великих об'єктів (типу «LOB»), але мають свої обмеження. Це веде до втрати даних.

Технології BD з іншого боку, сприяють повному збереженню всіх даних для подальшого аналізу. Головний принцип технологій BD полягає в тому, що в кожному біті даних може бути прихована важлива і цінна інформація.

1.2 Потреби аналізу великих даних

З вищезгаданими атрибутами великих даних, обсяг даних величезний, вони приходять зі швидкістю і в дуже неструктурованому вигляді, що не відповідають звичайним структурам реляційних СКБД.

З такою великою кількістю прихованої інформації в цих даних, необхідний альтернативний спосіб аналізу даних. Великим корпораціям може бути достатньо ресурсів для вирішення цього завдання, але загальний обсяг даних, що генеруються щодня, легко переважить можливості компанії по обробці таких обсягів. Завдяки більш дешевому технічному обладнанню, хмарним обчисленням та відкритому

програмному забезпеченню обробка великих обсягів даних стала набагато дешевшою.

Багато даних означає багато прихованої цінної інформації. Можливість швидко аналізувати великі обсяги даних означає можливість більш детально вивчати клієнтів, тенденції ринку, драйвери маркетингу та реклами, моніторингу обладнання та аналізу продуктивності та багато іншого. І це є важливою причиною того, що багато великих компаній потребують надійних технологій та інструментів аналізу даних.

Раніше, для збереження великих обсягів даних використовувалися реляційні СКБД. Технології BD перевершують реляційні СКБД по низці критеріїв, включаючи потребу в більш складних резервних копіях, відновленні та більш швидких алгоритмів пошуку [4].

Переваги використання технологій BD можуть призвести до втрати конфіденційності даних.

1.3 Висновки за розділом

В першій частині даного розділу було визначено три атрибути великих обсягів даних (BD): обсяг, швидкість, структурованість. За допомогою даних атрибутів можна висунути вимоги до функціональності ПС на основі АСКАД.

В другій частині обґрунтовано необхідність використання BD технологій для аналізу великих даних. Можна виділити наступні категорії користувачів BD:

– наукові установи, що використовують технології BD для побудови науково-дослідних кластерів [5],

– компанії, що використовують технології BD в основі BI програмних систем [6].

Для першої категорії користувачів технології BD надають можливість проаналізувати наукові дані, зробити висновки на основі проведених досліджень, що сприяє науково-технічному прогресу.

Для другої категорії необхідність використання технологій BD ґрунтується на збільшенні власних прибутків.

Таким чином можна зробити висновок, що BD технології є дуже актуальними в наш час і дослідження цього напрямку є перспективним.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Однією з основних елементів, що є основою АСКАД на базі BD є готова інфраструктура розподілених обчислень. Апаратною основою IPO є комп'ютерний кластер, що складається з мережи комп'ютерів особливого типу. Мережа побудована у такий спосіб, що їх можна розглядати як єдину систему. Комп'ютерні кластери з'явилися як результат наявності дешевих процесорів, швидкісних мереж та відповідного ПЗ для розподілених обчислень.

Вони мають широке коло використання: від невеликих кластерів для малого бізнесу до найпродуктивніших суперкомп'ютерів у світі. Наприклад, станом на листопад 2017 року, найпродуктивнішим кластером у світі є Sunway TaihuLight [7] з наступними характеристиками:

– кількість ядер – 10649600,

– продуктивність (за Linpack тестом) – 93014,6 Терафлопс/с,

– потужність – 15371 кВт,

– кількість пам'яті – 1310720 ГБ,

– ОС – Sunway RaiseOS.

Можна виділити наступні переваги комп'ютерних кластерів, до яких відносяться [8]:

– доступність реалізації,

– високі обчислювальні здібності,

– можливості балансування навантаження,

– великі обсяги сховища.

Для вирішення поставленої задачі комп'ютерний кластер повинен поєднувати властивості обчислювальних кластерів та кластерів-сховищ.

За даними критеріями можна виділити наступні види IPO:

– Hadoop,

– Dryad,

– Message Passing Interface.

2.1 Інфраструктура розподілених обчислень Hadoop

Hadoop - проект фонду Apache Software Foundation, вільно розповсюджуваний набір утиліт, бібліотек і фреймворків для розробки і виконання розподілених програм. Ці фреймворки працюють на кластерах з сотень і тисяч вузлів. Використовується для реалізації пошукових механізмів багатьох високонавантажених веб-сайтів, в тому числі, для Yahoo! і Facebook. Розроблено на Java в рамках обчислювальної парадигми MapReduce, згідно з якою додаток поділяється на велику кількість однакових елементарних завдань, що виконуються на вузлах кластера [9].

Станом на 2018 рік проект складається з чотирьох модулів [10]:

– Hadoop Common (набір інфраструктурних програмних бібліотек і утиліт, використовуваних для інших модулів і споріднених проектів),

– HDFS (розподілена файлова система),

– YARN (система для планування завдань і управління кластером),

– Hadoop MapReduce (платформа програмування і виконання розподілених MapReduce-обчислень).

Вважається однією з основоположних технологій «великих даних». Навколо Hadoop утворилася ціла екосистема з пов’язаних проектів і технологій, багато з яких розвивалися спочатку в рамках проекту, а згодом стали самостійними.

Однією з основних цілей Hadoop спочатку було забезпечення горизонтальної масштабовності кластера за допомогою додавання недорогих вузлів (обладнання масового класу, англ. Commodity hardware), без придбання потужних серверів і дорогих мереж зберігання даних.

Функціонуючі кластери розміром в тисячі вузлів підтверджують дієздатність і економічну ефективність таких систем. Так, станом на 2011 рік, відомо про великі кластери Hadoop в Yahoo (більше 4 тис. вузлів з сумарним об’ємом для зберігання 15 Пбайт), Facebook (близько 2 тис. вузлів на 21 Пбайт)[11] і Ebay (700 вузлів на 16 Пбайт)[12]. Проте, вважається, що горизонтальна масштабовність в Hadoop-системах

обмежена. Для Hadoop до версії 2.0 максимально можлива кількість вузлів оцінювалася в 4 тис. при використанні 10 MapReduce завдань на вузол. Багато в чому даному обмеженню сприяла концентрація в модулі MapReduce функцій з контролю за життєвим циклом завдань. Вважається, що з виносом її в модуль YARN в Hadoop 2.0 і децентралізацією частини функцій з моніторингу на вузли обробки горизонтальна масштабованість підвищилася.

Ще одним обмеженням Hadoop-систем є розмір оперативної пам'яті на вузлі імен, що зберігає весь простір імен кластера для розподілу завдань обробки. Загальна кількість файлів, що здатен обробляти вузол імен, дорівнює 100 млн. Для подолання цього обмеження ведуться роботи з розподілу вузла імен, єдиного в поточній архітектурі на весь кластер, на кілька незалежних вузлів. Іншим варіантом подолання цього обмеження є використання розподілених СКБД поверх HDFS [13].

Масштабовність Hadoop-систем в значній мірі залежить від характеристик даних, що обробляються. Вона залежать від їх внутрішньої структури, особливостей вилучення необхідної інформації і складності обробки. Це, в свою чергу, диктує організацію циклів обробки, обчислювальну інтенсивність атомарних операцій, рівень паралелізму і завантаженість кластера. У конфігураціях Hadoop (перших версіях до 2.0) вказувалося, що прийнятним рівнем паралелізму є використання 10-100 примірників базових оброблювачів на вузол кластера, а для задач, які не потребують значних витрат процесорного часу до 300. Для згорток вважалося оптимальним використання їх за кількістю вузлів, з урахуванням коефіцієнту і діапазону від 0,95 до 1,75 і константи mapred.tasktracker.reduce.tasks.maximum. З великим значенням коефіцієнту найбільш швидкі вузли, закінчивши перший раунд обробки. Таким чином раніше отримають другу порцію проміжних пар для обробки. Збільшення коефіцієнта надлишково завантажує кластер, але при цьому забезпечує більш ефективне балансування навантаження. У YARN використовуються конфігураційні константи, що визначають значення доступної оперативної пам'яті і віртуальних процесорних ядер, доступних для планувальника ресурсів, на підставі яких і визначається рівень паралелізму [14].

На рис. 2.1 представлена структура IPO Hadoop.

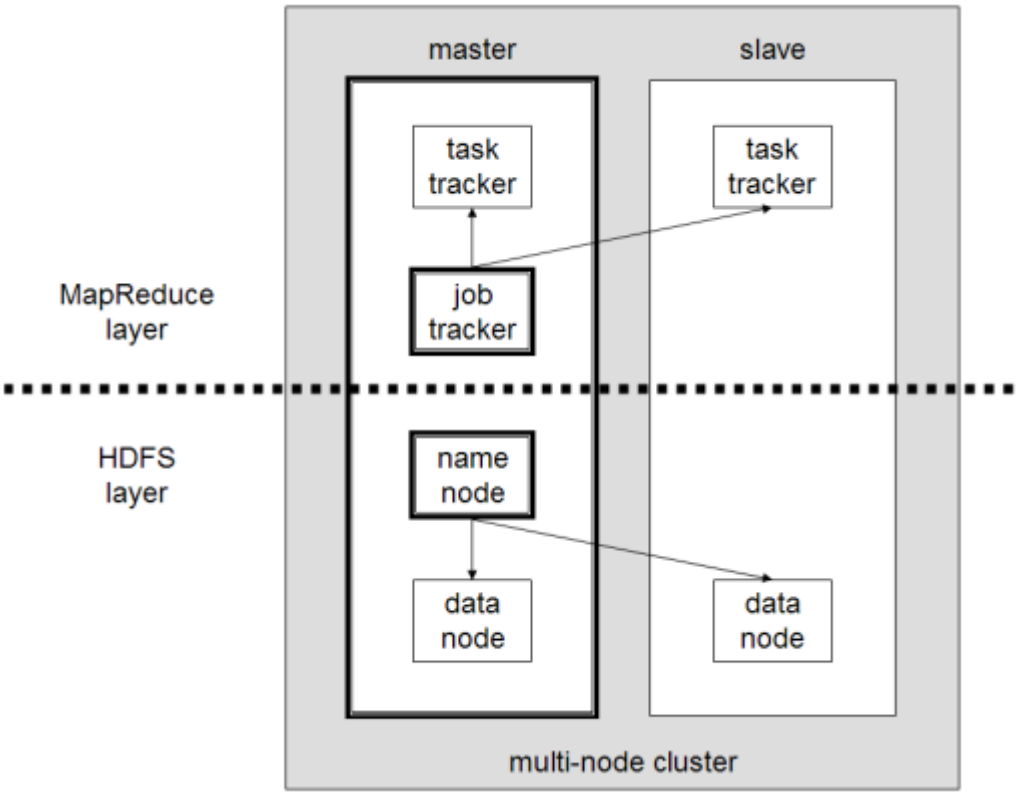


Рисунок 2.1 – Структура IPO Hadoop [15]

2.2 Інфраструктура розподілених обчислень Dryad

Dryad – програмний фреймворк загального призначення для розподілених обчислень. Dryad є проектом підрозділу Microsoft Research. Центральної концепцією проекту Dryad є побудова орієнтованого ациклічного графу (directed acyclic graph, DAG) для кожного розподіленого завдання. Вершини графа представляють собою операції над даними (по суті, програми), а ребра графа канали даних.

Абстракція на основі моделі орієнтованого ациклічного графу дає можливість ефективно реалізовувати плани виконання великої кількості паралельних алгоритмів, ітеративних алгоритмів, алгоритмів машинного навчання. Таким чином, єдина (до появи YARN) програмна модель MapReduce, що реалізована в Hadoop, по суті є лише окремим випадком моделі розподілених обчислень, яку надає Dryad.

Dryad оптимізований для запуску на середньому або великому обчислювальному кластері (від 100 до 10000 обчислювальних вузлів) і націлений,

головним чином, на тривалі пакетні завдання, які не потребують частих взаємодій [16].

Структура Dryad показана на рисунку 2.2.

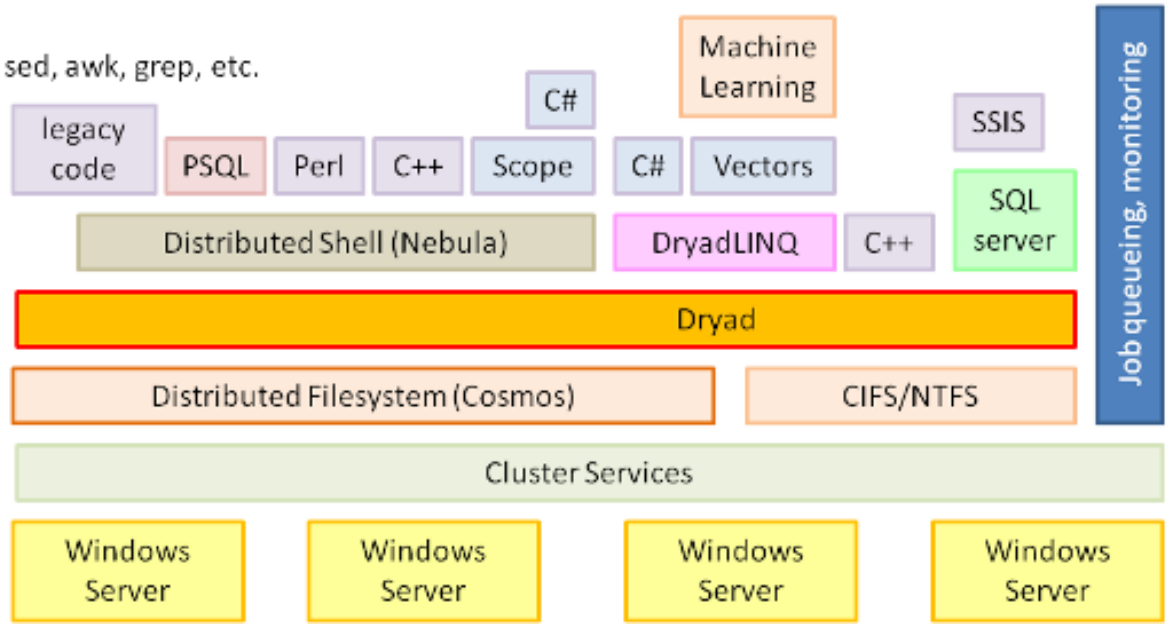


Рисунок 2.2 – Структура Dryad [17]

Проект Dryad складається з 3-ох ключових компонентів:

– Dryad середовище виконання розподілених обчислень (Dryad Runtime);

– DryadLINQ високорівнева мова запитів, що базується на програмній моделі .NET Language Integrated Query (LINQ);

– Distributed Storage Catalog (DSC) розподілена файлова система з надмірністю.

Dryad Runtime виконує наступні функції:

– планування і управління розподіленими завданнями,

– управління ресурсами,

– забезпечення відмовостійкості,

– моніторинг.

Завдання в Dryad Runtime представляє собою орієнтований ациклічний граф, де вершини представляють собою програми, а ребра графа канали даних. Цей логічний граф розділяється (mapped) середовищем виконання на фізичні ресурси, що

знаходяться в кластері. Загалом, кількість вершин в графі перевищує кількість фізичних обчислювальних вузлів в кластері [18].

2.3 Інфраструктура розподілених обчислень Message Passing Interface

Message Passing Interface (інтерфейс передачі повідомлень) - програмний інтерфейс (API) для передачі інформації, що дозволяє обмінюватися повідомленнями між процесами, що виконують одну задачу [19].

MPI є найбільш поширеним стандартом інтерфейсу обміну даними в паралельному програмуванні, існують його реалізації для різних комп'ютерних платформ. Використовується при розробці програм для кластерів і суперкомп'ютерів. Основним засобом комунікації між процесами в MPI є передача повідомлень між вузлами.

Стандартизацією MPI займається MPI Forum. У стандарті MPI описаний інтерфейс передачі повідомлень, що має підтримуватися на платформі, і в додатках користувача. В даний час існує велика кількість безкоштовних і комерційних реалізацій MPI. Існують реалізації для мов Фортран 77/90, Java, C і C++.

В першу чергу MPI орієнтований на системи з розподіленою пам'яттю, тобто коли витрати на передачу даних великі, в той час як OpenMP орієнтований на системи із загальною пам'яттю (багатоядерні із загальним кешем). Обидві технології можуть використовуватися спільно, щоб оптимально використовувати в кластері багатоядерні системи [20].

На рисунку 2.3 зображена структура програми MPI.

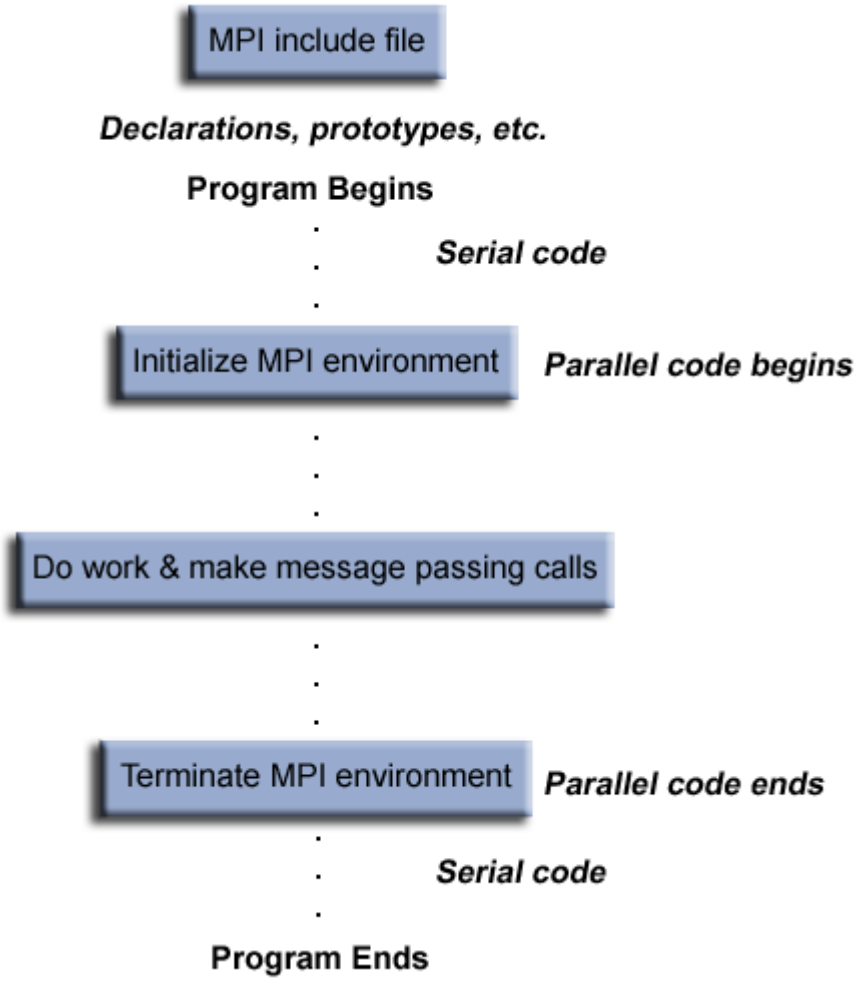


Рисунок 2.3 – Структура програми MPI [21]

Базовим механізмом зв’язку між MPI процесами є передача і прийом повідомлень. Повідомлення несе в собі необхідні дані та додаткову інформацію, що дозволяє приймаючій стороні здійснювати їх вибірковий прийом. Додаткова інформація включає:

– ранг відправника (номер в групі) повідомлення;

– ранг одержувача;

– тип, що може використовуватися для поділу різних видів повідомлень;

– комунікатор, код групи процесів.

Операції прийому і передачі можуть бути блокуючі і неблокуючі. Для неблокуючих операцій визначені функції перевірки готовності і очікування виконання операції [22].

Іншим способом зв’язку є віддалений доступ до пам’яті (RMA), що дозволяє читати і змінювати область пам’яті віддаленого процесу. Локальний процес може переносити область пам’яті віддаленого процесу (всередині зазначеного процесами вікна) в свою пам’ять. Комбінувати дані, що передаються до віддаленого процесу з наявними в його пам’яті даними (наприклад, шляхом підсумовування). Всі операції віддаленого доступу до пам’яті не блокуються до і після їх виконання необхідна синхронізація [23].

На основі отриманих даних можна побудувати порівняльну таблицю IPO (табл. 2.1).

Таблиця 2.1 – Порівняльна таблиця IPO

IPO	Програмна модель	Сховище даних	Канали зв’язку	Мови програмування	Ліцензія
Hadoop	MapReduce, YARN	HDFS	TCP/IP	Java, C++ та ін.	Відкрита
Dryad	Direct acyclic graph	DSC	TCP/IP, temp files,shared memory FIFO	C++, C#	Пропрієтарна
MPI	Залежить від топології	Shared file system	Low latency канали	C, C++, C#, Java, Fortran	Пропрієтарна, відкрита

2.4 Висновки за розділом

Таким чином у розділі проведений аналіз базових IPO для побудови ПС на базі АСКАД. Розглянуті рішення найбільш відомих технологій BD. Відповідно до узагальненого результату у таблиці 2.1, кожна з IPO має свої недоліки і переваги у контексті реалізації АСКАД для ПС обробки даних. Проведений аналіз дає можливість зробити висновок, що для нашого випадку потрібно провести

дослідження і реалізувати АСКАД на основі ІРО Hadoop. Він має наступні переваги для малого та середнього бізнесу з метою аналізу розподілених даних, а саме:

– можливість розподілити загальний обсяг даних на певні частини за рахунок використання MapReduce і виконати обчислення над цими частинами;

– можливість провадити роботу над розподіленими у мережі даними, що потрібно зібрати з використанням, наприклад, Apache Flume;

– надає можливість побудувати АІРО з використанням звичайних комп’ютерів (commodity hardware);

– відносно невисока вартість практичної реалізації АСКАД для ПС обробки даних для однакової виробничої здібності у порівнянні з іншими ІРО.

3 ВИМОГИ ДО ПРОГРАМНОЇ СИСТЕМИ НА ОСНОВІ АРХІТЕКТУРИ СИСТЕМИ КОМПЛЕКСНОГО АНАЛІЗУ ДАНИХ

3.1 Опис предметного середовища

Системи збору, обробки й аналізу даних з використанням АСКАД на основі технологій BD потрібні для розробки, тестування і дослідження складних систем. Прикладом може бути наукові дослідження, конструкторські розробки, рішення проблем у певних сферах бізнесу. Такі системи потребують інструменти для обробки структурованих і неструктурованих даних великих обсягів.

Необхідність побудови ПС на основі АСКАД і з використанням IPO BD ґрунтується на щорічному зростанні обсягів інформації, що потребують обробки (рис. 3.1).

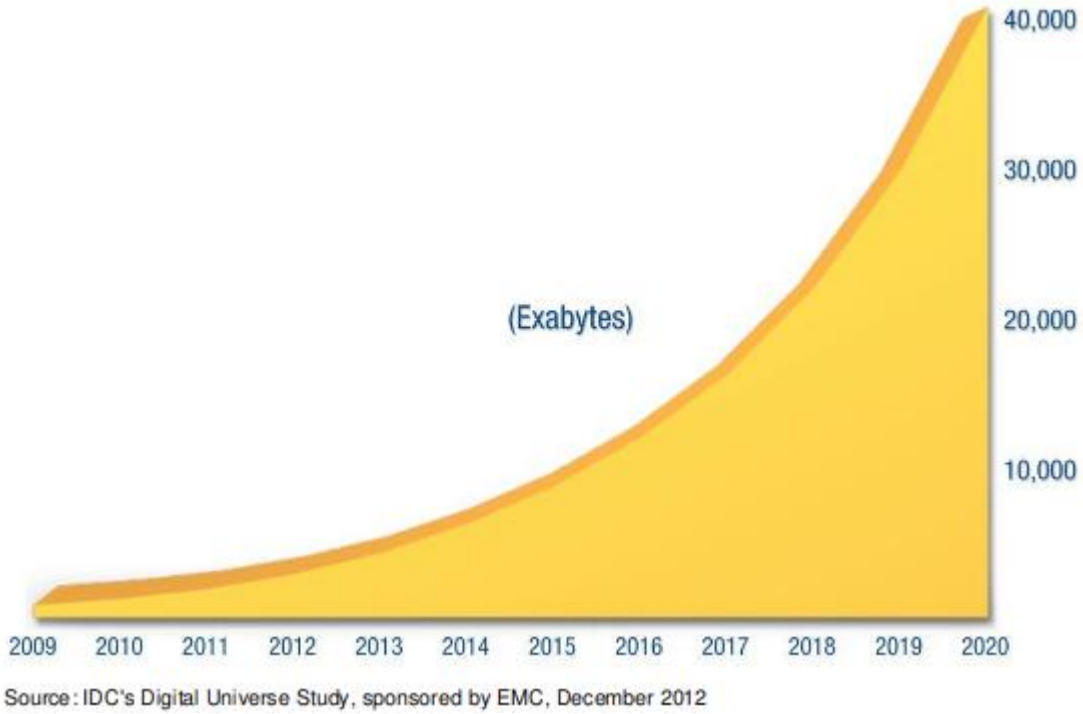


Рисунок 3.1 – Зростання обсягів інформації [24]

Аналіз рис. 3.1 показує, що кількість даних для обробки збільшується за експоненціальним законом. Показана майбутня перспектива, що також демонструє експоненціальне збільшення обсягу даних. В заданих умовах необхідно забезпечити можливості їх обробки.

Саме для цього мною обраний компонент ІРО для ПС на основі АСКАД на основі Hadoop. Зважаючи на її архітектуру, до системи можна встановити наступні вимоги [25]:

– масштабованість,

– можливість збирати структуровані та неструктуровані дані,

– високу швидкість обробки великих обсягів даних (петабайти даних [26]),

– можливість зберігати дані у кластерній ФС,

– відмовостійкість,

– представлення результатів аналізу у вигляді, зручному для подальшого відображення.

Продуктивність ІРО цілком залежить від технічного та програмного забезпечення.

3.2 Засоби розробки

При побудові ПС на основі АСКАД було використано ряд програмного забезпечення для графічного оформлення, програмування.

Основною мовою всієї системи є мова Java. Модулі були описані з використанням даної мови програмування. Вона має ряд переваг порівняно з іншими мовами, а саме:

– стандартна бібліотека вміщує багато корисних інструментів,

– кросплатформеність,

– автоматичний розподіл пам’яті,

– велика кількість допоміжних бібліотек.

Модулі ПС на основі АСКАД були розроблені у середовищі NetBeans IDE з використанням пакетного менеджеру Maven. Він спрощує підключення додаткових бібліотек до проекту і дозволяє скомпілювати виконавчий файл Java з усіма залежностями і без них, вказуючи відносні шляхи до них.

Для написання модулю АСКАД збору інформації було також використано мову програмування Python. Вона дозволяє швидко писати нескладні скрипти і виявилася дуже зручною для поставленої мети.

Розроблення UML діаграм [27] відбувалося з використанням Web додатку ’draw.io’.

3.3 Вимоги до технічного забезпечення

Апаратні вимоги IPO дозволяють визначити швидкість обробки даних, кількість оброблюваних потоків даних за одиницю часу, розмір кластерної файлової системи для збереження даних та розгортання бази даних. Таким чином, необхідними складовими апаратних вимог є:

– характеристики окремих вузлів кластеру,

– характеристики локальної мережі для передачі даних між окремими вузлами.

Кластери будуються на базі спеціалізованих комп’ютерів (серверів), що мають більшу надійність (резервування інформації, підвищення стійкість від помилок пам’яті, резервування блоків живлення і т.д.), спеціально розроблені форм-фактори для вбудови у телекомунікаційні стійки, більшу продуктивність. Дана IPO може використовувати як сервери, так і обладнання масового класу для зменшення затрат на технічні засоби.

IPO Hadoop виділяє наступні ролі в системі:

– мастер-вузол, що координує потік даних у кластері та розподіл задач;

– вторинний вузол, що виконує задану задачу, а також надає сховище для побудови кластерної файлової системи.

Для вибору компонентів мастер-вузла можна скористатися таблицею 3.1 з рекомендованими характеристиками у збалансованому кластері [28, 29].

Таблиця 3.1 – Вибір компонентів для мастер-вузла

Компонент	Характеристика
Форм-фактор	1U-4U

Продовження таблиці 3.1

Компонент	Характеристика
CPU	Мінімум двоядерний процесор тактовою частотою 2-2.5 ГГц
RAM	64-512 ГБ
Диски	12-24 по 1-4 ТБ
Мережева карта	Gigabit або 10Gigabit Ethernet

Характеристики окремих компонентів для вторинних вузлів надані у таблиці 3.2.

Таблиця 3.2 – Вибір компонентів для вторинних вузлів

Компонент	Характеристика
Форм-фактор	1U-4U
CPU	Мінімум двоядерний процесор тактовою частотою 2-2.5 ГГц
RAM	64-128 ГБ
Диски	4-6 по 1 ТБ
Мережева карта	Gigabit або 10Gigabit Ethernet

Вибір характеристик тих чи інших компонентів, що вказуються у проміжках, залежить від задач, що виконуватиме IPO. Таким чином, можна виділити 2 типи задач за наявними обмеженнями:

– що обмежені кількістю зчитувань-записів,

– що обмежені продуктивністю.

Відповідно, для збільшення швидкодії необхідно для 1-го роду задач збільшувати пропускну спроможність мережі та обирати такі диски, що мають найменшу затримку в операціях зчитування-запису; для 2-го роду задач необхідно обирати більш продуктивні процесори, збільшувати їх кількість і т.д.

Для комунікації між вузлами використовуються протоколи зі стеку протоколів TCP/IP [30].

3.4 Вимоги до програмного забезпечення

IPO Hadoop, як і інші комерційні рішення, рекомендується будувати на системі Linux. Існує велика кількість дистрибутивів Linux, але перевагу слід віддавати таким, що перелічені далі [31].

3.4.1 Операційна система Debian

Debian [32] використовується в таких великих комерційних підприємствах, як Boeing, T-Mobile і т.д. [33]. Debian підтримується користувачами, має швидкий сервіс відповідей на запитання, що стосуються безпосередньо системи. Окрім того, Debian легко встановлюється з CD, DVD, Blu-ray, USD або з мережі.

Для Debian існує довготривала підтримка програмного забезпечення, дана ОС стабільна і надійна в роботі. Це досягається шляхом створення спеціальних репозитаріїв, що містять стабільні версії програм та ядра, вони пройшли тривалі тестування і їх можна вважати надійними.

3.4.2 Операційна система CentOS

CentOS – це операційна система, націлена на використання всереедині компаній та в середовищі розробників, що не потребують останніх версій забезпечення, а потребують стабільну та захищену операційну систему [34]. Використовується для віртуалізації [35], розгортання SaaS платформ і т.д. Дистрибутив використовується такими великими компаніями, як Godaddy Inc, Vistronix Intelligence & Technology Solutions LLC [36]. Даний дистрибутив є неофіційним безоплатним клоном RHEL. Перед кожним релізом дистрибутив проходить довготривале тестування.

3.4.3 Операційна система Ubuntu

Операційна система Ubuntu – це відкритий та безоплатний дистрибутив на основі ОС Linux [37]. Ubuntu є однією з найбільш популярних ОС у хмарних сервісах [38].

Окрім версії для комп'ютерів загального призначення, користувачам доступна серверна версія даного дистрибутиву.

Дана ОС відрізняється простотою налаштувань системи, встановлення додаткового програмного забезпечення.

3.4.4 Операційна система RedHat Enterprise Linux

ОС RedHat Enterprise Linux націлена на комерційне використання і має платну підписку ціною 349 доларів за базову версію [39]. Даний дистрибутив має довготривалу підтримку версій. Сервіс відповідей на питання користувачів включений до підписки, але окрім офіційного сервісу дистрибутив має велику користувацьку базу, що також можуть надати відповіді на питання.

Дана ОС використовується в таких відповідальних сферах [40] як, наприклад:

– банкінгу (Barclays, Bharatiya Stock Exchange, Deutsche Bank, Key Bank),

– авіаперевезень (Lufthansa),

– розробки програмного забезпечення (CA technologies).

Даний список не обмежується вище переліченими прикладами.

Розробники ОС впроваджують власні зміни в ядро базової ОС Linux, що сприяє більш високому ступеню захищеності даного дистрибутиву та його більш стабільної роботи.

3.4.5 Операційна система SUSE

ОС SUSE також націлена на комерційне використання і має платну підписку [41].

Компанія-розробник постачає версії загального користування та серверні. Даний дистрибутив, як і RedHat Enterprise Linux, розробляється з метою створення стабільної та високо захищеної системи, що має як професійну підтримку користувачів, так і підтримку зі сторони спільноти користувачів.

ОС використовується в багатьох індустріях і має низку специфічних продуктів (наприклад, для високопродуктивних обчислень) [42].

Практика показала, що хмарні провайдери найбільш часто використовують ОС Ubuntu [43].

Оскільки Hadoop та інші модулі системи написані з використанням мови програмування Java, необхідно також встановити Java Development Kit (JDK). JDK містить в собі бібліотеки розробки та середовище виконання Java Runtime Environment (JRE).

Бібліотеки розробки для Java необхідні для написання модульних програм для підсистем збору й обробки та аналізу даних.

Середовище виконання JRE необхідно для запуску ІРО, окремих підсистем (збору й обробки та аналізу даних) та модулів для заданих підсистем.

Рекомендована версія для запуску всієї системи – реліз Oracle JDK 1.7 [44].

3.5 Висновки за розділом

В даному розділі обґрунтовано вимоги до ІРО для ПС на основі АСКАД, описано засоби розроблення модулів для ІРО для ПС, визначені вимоги до програмного та технічного забезпечення.

Рекомендовані [28, 29] характеристики для технічного забезпечення для побудови АІРО дозволяють підібрати компоненти для коректної роботи ПС на основі АСКАД. Огляд рекомендованих характеристик показує, що описані компоненти мають невисоку ціну, що дозволяє побудувати недорогий продуктивний кластер.

Серед перелічених ОС було обрано ОС Debian, як надійну та стабільну у роботі систему, що швидко встановлюється і конфігурується. Як було показано вище, найбільш часто використана ОС Ubuntu, але вона потребує додаткових конфігурацій

і встановлення програмного забезпечення, чого не потребує ОС Debian після повної інсталяції.

4 РОЗРОБЛЕННЯ СЦЕНАРІЇВ ВИКОРИСТАННЯ

На початковій стадії були проаналізовані різновиди використання ПС на основі АСКАД. Було визначено, що при розробці сценаріїв використання, необхідно врахувати наявність двох акторів у системі: оператор та адміністратор.

Для побудови діаграм невід'ємними процесами є:

– виділення діючих суб'єктів АСКАД;

– розроблення варіантів використання АСКАД з боку актора, які визначають функціональність ПС на основі АСКАД;

– короткий детальний опис визначених варіантів використання або їх детальний табличний опис.

Розглянемо сценарії використання для кожного з акторів окремо.

4.1 Розроблення діаграми варіантів використання адміністратором

Розроблена діаграма варіантів використання адміністратором складається з 3 складових частин:

– взаємодія з ОС,

– взаємодія з підсистемою ІРО,

– взаємодія з СКБД.

Діаграма варіантів використання адміністратором у контексті взаємодії з ОС представлена на рис. 4.2.

Дана діаграма відображає базовий набір функцій, що необхідні для актора «адміністратор» для коректної роботи всієї ПС на основі АСКАД.

Більш детальний опис кожного UC представлений у 6 розділі.

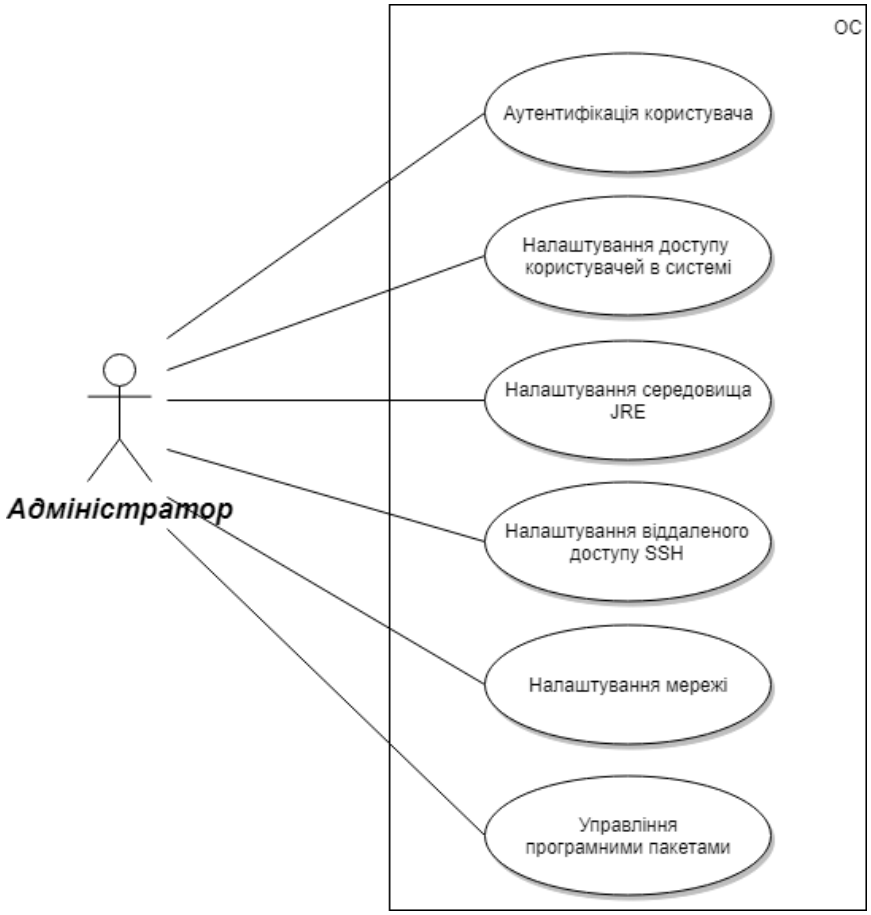


Рисунок 4.1 – Діаграма варіантів використання адміністратором у контексті взаємодії з ОС

Діаграма варіантів використання адміністратором у контексті взаємодії з підсистемою ІРО представлена на рис. 4.2.

Оскільки ПС на основі АСКАД є кластерною, то за необхідності збільшення її продуктивності можна підключити додаткові вузли. Дана підсистема є масштабованою, тому адміністратор має права конфігурації вузлів у складі ІРО. Таким чином, запущена підсистема має змогу підключити усі наявні в системі вузли для максимальної швидкодії та збільшення обсягів ФС.

Можливість налаштування шляхів до бібліотек та програм дозволяє адміністратору відокремити бібліотеки та виконавчі файли в окремих папках. Це призводить до зменшення розмірів програм (деякі програми можуть використовувати однакові бібліотеки, але кожна з них буде включати дану бібліотеку на етапі компіляції) та спрощує запуск і відлагодження системи.

Налаштування фреймворку MapReduce дозволяє обрати фреймворк для обробки даних, виставити ліміти наявних для IPO ресурсів (пам’яті тощо).

Процес налаштування ФС являє собою визначення параметрів кластерної ФС у відповідності з вимогами до відмовостійкості, швидкості роботи ФС (можлива конфігурація розміру блоків реплікації і т.д.). Форматування ФС дозволяє створити кластерну ФС з розподіленням даних на всі наявні вузли у складі IPO.

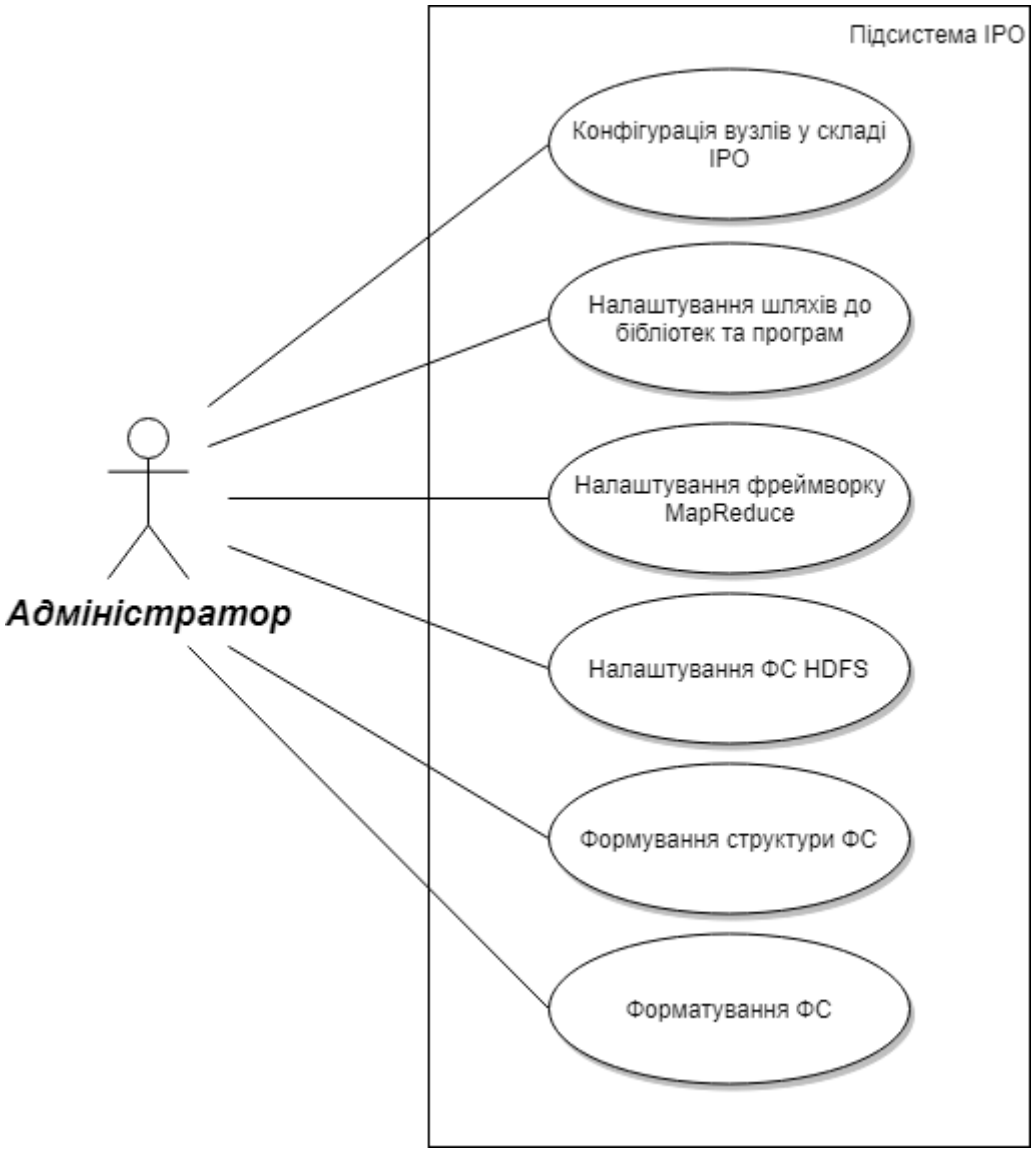


Рисунок 4.2 – Діаграма варіантів використання адміністратором у контексті взаємодії з IPO

Опишемо більш детально деякі з варіантів використання, представлених на рисунку 4.2.

Прецедент конфігурація вузлів у складі IPO детально описано в таблиці 4.1.

Таблиця 4.1 – UC Конфігурація вузлів у складі IPO

Назва	Конфігурація вузлів у складі IPO	
Мета або контекст використання	Логічне додавання або видалення вузлів КС. Використовується для управління кількістю активних вузлів всередині КС.	
Область видимості та рівень	Для адміністратора.	
Передумова	Аутентифікація користувача на комп’ютері, що виступає в ролі master всередині КС.	
Умова успішного виконання	Правильно прописана конфігурація у файлах конфігурації IPO у відповідності з заданим форматом опису вузлів КС.	
Умова невиконання	Некоректна робота ОС, IPO, відсутність віддаленого доступу до комп’ютеру, некоректно налаштовані права доступу (відмова в редагуванні файлів у зв’язку з недостатніми правами), некоректно описані вузли у конфігураційних файлах, некоректна мережева конфігурація	
Первинний, вторинний актор	Первинний актор – користувач платформою (адміністратор).	
Тригер	Не передбачені	
Опис	Кроки	Дія
	1	Аутентифікація користувача в системі.
	2	Відкриття на редагування конфігураційного файлу зі списком вузлів.
	3	Редагування списку вузлів.
	4	Збереження файлу та перезапуск компонентів АСКАД.

Прецедент налаштування шляхів до бібліотек та програм детально описано в таблиці 4.2.

Таблиця 4.2 – UC налаштування шляхів до бібліотек та програм

Назва	Налаштування шляхів до бібліотек та програм	
Мета або контекст використання	Налаштування шляхів до бібліотек та програм надає можливість запускати задачі на IPO.	
Область видимості та рівень	Для адміністратора.	
Передумова	Аутентифікація користувача на комп’ютері, що виступає в ролі master всередині КС.	
Умова успішного виконання	Прописані шляхи відповідають наявній на накопичувачі структурі, у відповідних папках зберігаються актуальні бібліотеки та програми для аналізу даних, що їх використовують.	
Умова невиконання	Некоректна робота ОС, IPO Hadoop, відсутність віддаленого доступу до комп’ютеру, некоректно налаштовані права доступу (відмова в редагуванні файлів у зв’язку з недостатніми правами), некоректно описані шляхи до програм та бібліотек у конфігураційних файлах.	
Первинний, вторинний актор	Первинний актор – користувач платформою (адміністратор).	
Тригер	Не передбачені	
Опис	Кроки	Дія
	1	Аутентифікація користувача в системі.
	2	Відкриття на редагування конфігураційного файлу IPO.
	3	Редагування параметрів, що задають шлях до файлів.

Продовження таблиці 4.2

Назва	Налаштування шляхів до бібліотек та програм	
	4	Збереження файлу та перезапуск компонентів АСКАД.

Прецедент форматування ФС детально описано в таблиці 4.3.

Таблиця 4.3 – UC форматування ФС

Назва	Форматування ФС	
Мета або контекст використання	Первинне форматування ФС після розґортання ПС на основі АСКАД дозволяє включити всі наявні IPO вузли до складу кластерної ФС, або, якщо вже включені, виконати очищення ФС.	
Область видимості та рівень	Для адміністратора.	
Передумова	Аутентифікація користувача на комп’ютері, що виступає в ролі master всередині КС.	
Умова успішного виконання	Коректна конфігурація IPO у відповідності з форматами конфігураційних файлів.	
Умова невиконання	Некоректна конфігурація IPO.	
Первинний, вторинний актор	Первинний актор – користувач платформою (адміністратор).	
Тригер	Не передбачені	
Опис	Кроки	Дія
	1	Аутентифікація користувача в системі.
	2	Відключення IPO.
	3	Ініціювання команди форматування HDFS, що входить до складу утиліт ПС.
	4	Перезапуск IPO.

Діаграма варіантів використання адміністратором у контексті взаємодії з СКБД представлена на рис. 4.3.

Налаштування СКБД має на увазі визначення всіх необхідних параметрів конфігураційних файлів. Оскільки ПС на основі АСКАД є кластерною, то за необхідності збільшення її продуктивності можна підключити додаткові вузли.

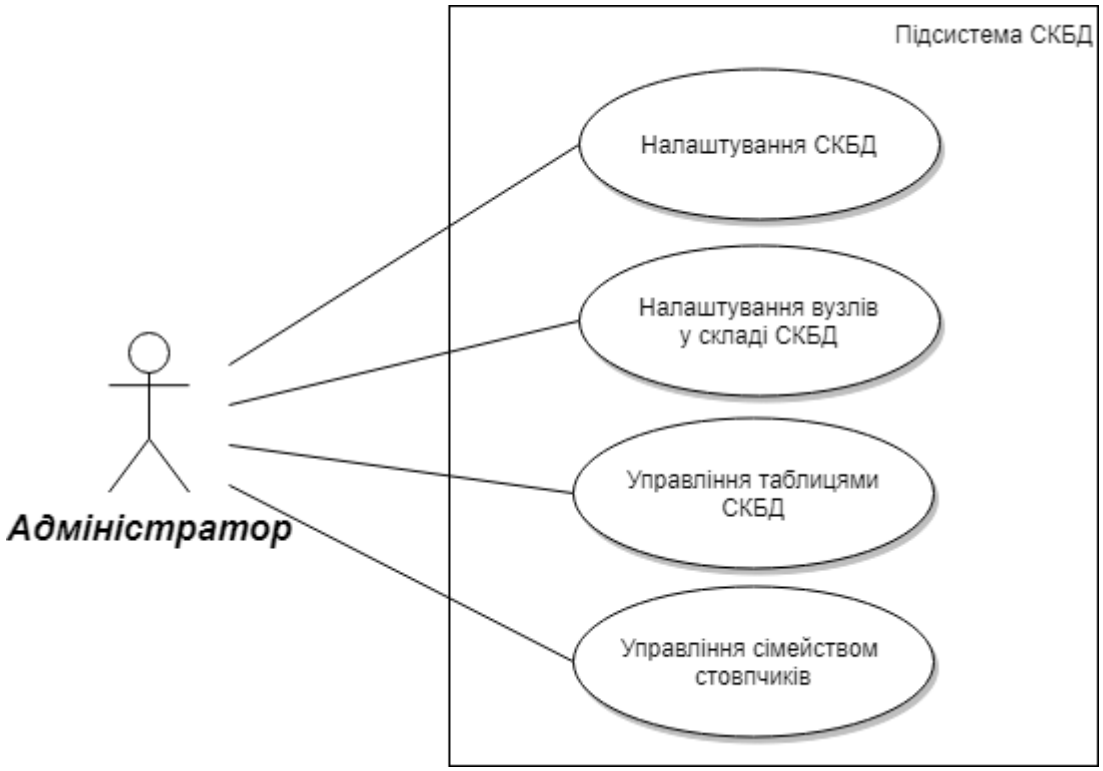


Рисунок 4.3 – Діаграма варіантів використання адміністратором у контексті взаємодії з СКБД

Прецедент налаштування вузлів у складі СКБД детально описано в таблиці 4.4.

Таблиця 4.4 – UC налаштування вузлів у складі СКБД

Назва	Налаштування вузлів у складі СКБД
Мета або контекст використання	Логічне додавання або видалення вузлів розподіленої СКБД. Використовується для управління кількістю активних вузлів всередині розподіленої СКБД.
Область видимості та рівень	Для адміністратора.

Продовження таблиці 4.4

Назва	Налаштування вузлів у складі СКБД	
Передумова	Аутентифікація користувача на комп’ютері, що виступає в ролі master всередині КС.	
Умова успішного виконання	Правильно прописана конфігурація у файлах конфігурації СКБД у відповідності з заданим форматом опису вузлів СКБД.	
Умова невиконання	Некоректна робота ОС, СКБД, відсутність віддаленого доступу до комп’ютеру, некоректно налаштовані права доступу (відмова в редагуванні файлів у зв’язку з недостатніми правами), некоректно описані вузли у конфігураційних файлах, некоректна мережева конфігурація	
Первинний, вторинний актор	Первинний актор – користувач платформою (адміністратор).	
Тригер	Не передбачені	
Опис	Кроки	Дія
	1	Аутентифікація користувача в системі.
	2	Відкриття на редагування конфігураційного файлу зі списком вузлів.
	3	Редагування списку вузлів.
	4	Збереження файлу та перезапуск СКБД.

Прецедент управління таблицями СКБД детально описано в таблиці 4.5.

Таблиця 4.5 – UC управління таблицями СКБД

Назва	Управління таблицями СКБД
Мета або контекст використання	Можливість створювати, редагувати та видаляти таблиці бази даних.

Продовження таблиці 4.5

Назва	Управління таблицями СКБД	
Область видимості та рівень	Для адміністратора.	
Передумова	Аутентифікація користувача будь-якому з комп'ютерів КС. Запущена ПС на основі АСКАД.	
Умова успішного виконання	Команди оболонки СКБД відповідають формату подачі аргументів; в якості аргументів вказані існуючі таблиці (у випадку редагування або видалення), назви таблиць не містять заборонених символів.	
Умова невиконання	Некоректна робота ОС, СКБД, відсутність віддаленого доступу до комп'ютеру, некоректно налаштовані права доступу, некоректно описані команди оболонки, помилки при визначенні аргументів.	
Первинний, вторинний актор	Первинний актор – користувач платформою (адміністратор).	
Тригер	Не передбачені	
Опис	Кроки	Дія
	1	Аутентифікація користувача в системі.
	2	Запуск оболонки СКБД.
	3	Виклик команд редагування, створення або видалення таблиць.
	4	Закінчення роботи з оболонкою СКБД.

4.2 Розроблення діаграми варіантів використання системи оператором

Визначені операторські UC можна побачити на діаграмі прецедентів, що зображена на рисунку 4.4.

Оператор має змогу запускати модулі для аналізу зібраних даних окремо або за встановленим розкладом запусків модулів.

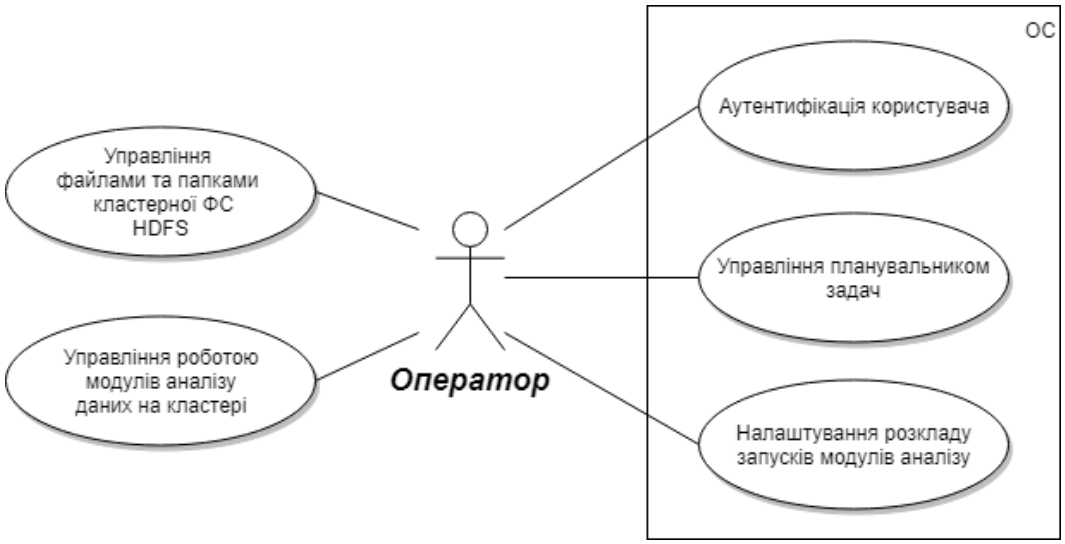


Рисунок 4.4 – Діаграма варіантів використання оператором

Опишемо більш детально деякі з варіантів використання, представлених на рисунку 4.4.

Прецедент управління роботою модулів аналізу даних на кластері детально описано в таблиці 4.6.

Таблиця 4.6 – UC управління роботою модулів аналізу даних на кластері

Назва	Управління роботою модулів аналізу даних на кластері
Мета або контекст використання	Можливість вручну запускати, зупиняти модулі аналізу даних ПС на основі АСКАД.
Область видимості та рівень	Для оператора.
Передумова	Аутентифікація користувача на будь-якому з комп'ютерів КС. Запущена ПС на основі АСКАД.
Умова успішного виконання	Команди оболонки IPO відповідають формату подачі аргументів; вказані коректні аргументи програм.
Умова невиконання	Некоректна робота ОС, IPO, відсутність віддаленого доступу до комп'ютеру, некоректно налаштовані права

Продовження таблиці 4.6

Назва	Управління роботою модулів аналізу даних на кластері	
	доступу, некоректно описані команди оболонки, помилки при визначенні аргументів.	
Первинний, вторинний актор	Первинний актор – користувач платформою (оператор).	
Тригер	Не передбачені	
Опис	Кроки	Дія
	1	Аутентифікація користувача в системі.
	2	Запуск оболонки ОС.
	3	Виклик команди запуску або зупинки роботи модуля.
	4	Закінчення роботи з оболонкою ОС.

Прецедент налаштування розкладу запусків модулів аналізу детально описано в таблиці 4.7.

Таблиця 4.7 – UC налаштування розкладу запусків модулів аналізу

Назва	Налаштування розкладу запусків модулів аналізу
Мета або контекст використання	Налаштування планувальника на періодичний запуск модулів аналізу.
Область видимості та рівень	Для оператора.
Передумова	Аутентифікація користувача на будь-якому з комп’ютерів КС. Запущена ПС на основі АСКАД.
Умова успішного виконання	Конфігурація відповідає заданому формату.
Умова невиконання	Некоректна робота ОС, ПС на основі АСКАД, відсутність віддаленого доступу до комп’ютеру, некоректно

Продовження таблиці 4.7

Назва	Налаштування розкладу запусків модулів аналізу	
	налаштовані права доступу, некоректно описані команди оболонки, помилки при визначенні аргументів.	
Первинний, вторинний актор	Первинний актор – користувач платформою (оператор).	
Тригер	Не передбачені	
Опис	Кроки	Дія
	1	Аутентифікація користувача в системі.
	2	Запуск оболонки СКБД.
	3	Виклик команд редагування, створення або видалення таблиць.
	4	Закінчення роботи з оболонкою СКБД.

Діаграма варіантів використання оператором в контексті підсистеми збору даних представлена на рисунку 4.5.

Оператор системи повинен мати можливість конфігурувати підсистему збору та обробки інформації, таким чином він має змогу визначити наступні складові конфігурації:

– джерела інформації,

– канали передачі інформації,

– сховища даних.

Для кожного з вузлів системи можна визначити конфігурацію і розподілити завдання збору й обробки інформації.

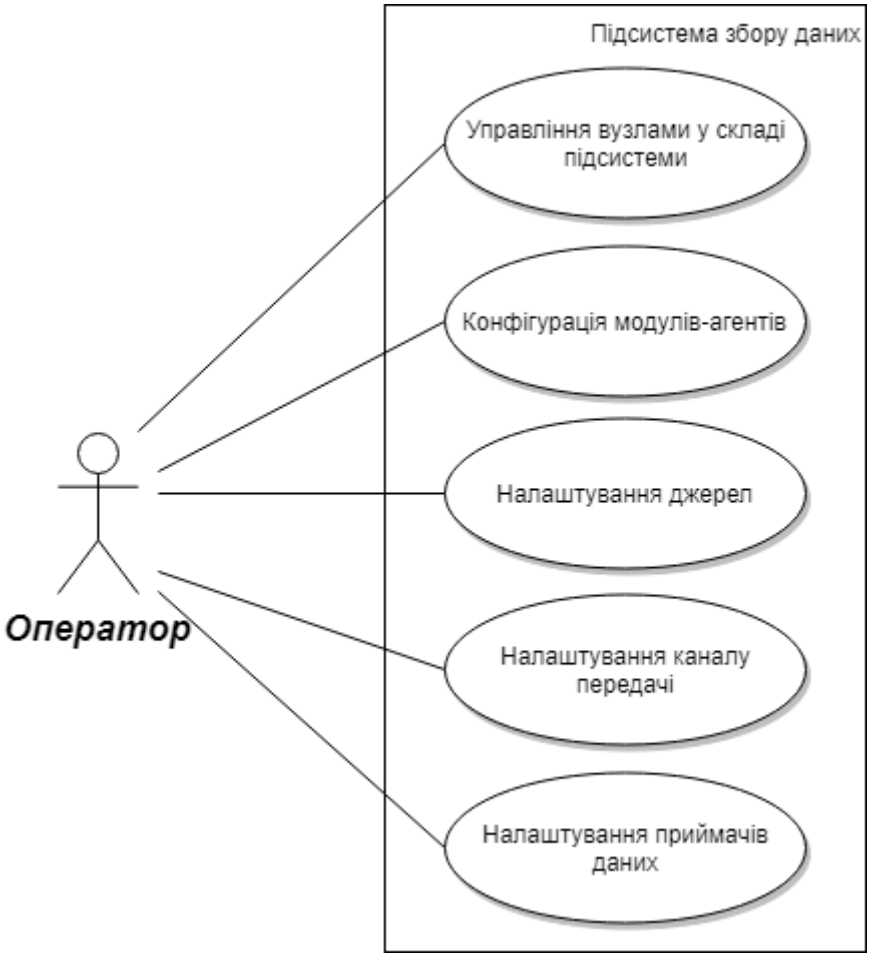


Рисунок 4.5 – Діаграма варіантів використання оператором в контексті підсистеми збору даних

Діаграма варіантів використання оператором в контексті СКБД представлена на рисунку 4.6.

Для сховища даних, як правило, використовується розподілена база даних, що розгорнута поверх IPO. Тому окрім визначення параметрів модулів-агентів збору й обробки інформації, оператор має змогу додавати, змінювати або видаляти таблиці з бази даних, що використовуються підсистемами. Він також може визначати сімейство стовпчиків. За необхідності оператор може змінювати окремі рядки в базі даних.

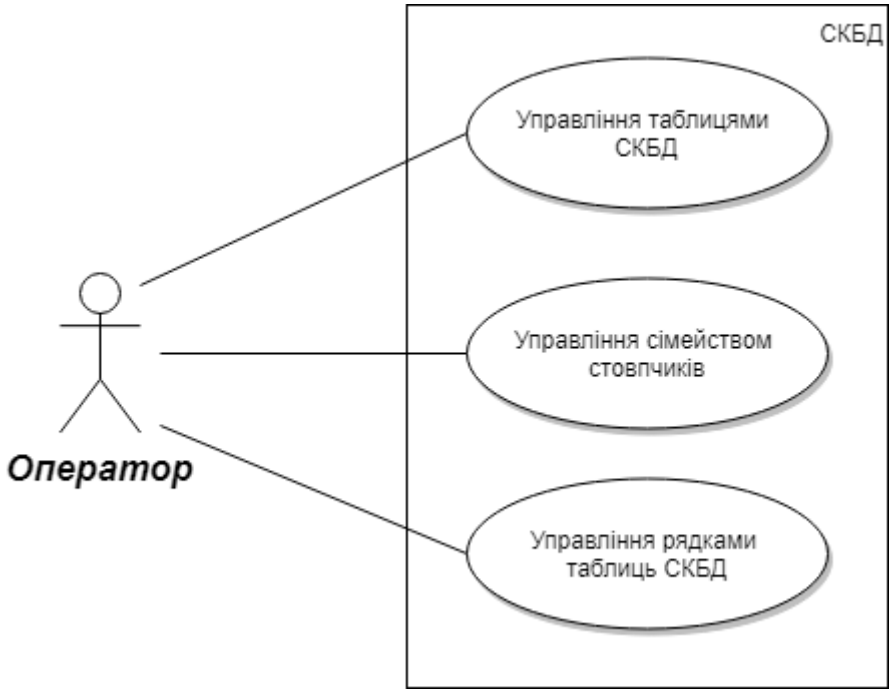


Рисунок 4.6 – Діаграма варіантів використання оператором в контексті СКБД

Прецедент управління таблицями СКБД детально описано в таблиці 4.8.

Таблиця 4.8 – UC управління таблицями СКБД

Назва	Управління таблицями СКБД
Мета або контекст використання	Можливість створювати, редагувати та видаляти таблиці бази даних.
Область видимості та рівень	Для оператора.
Передумова	Аутентифікація користувача на будь-якому з комп’ютерів КС. Запущена ПС на основі АСКАД.
Умова успішного виконання	Команди оболонки СКБД відповідають формату подачі аргументів; в якості аргументів вказані існуючі таблиці (у випадку редагування або видалення), назви таблиць не містять заборонених символів.
Умова невиконання	Некоректна робота ОС, СКБД, відсутність віддаленого доступу до комп’ютеру, некоректно налаштовані права

Продовження таблиці 4.8

Назва	Управління таблицями СКБД	
	доступу, некоректно описані команди оболонки, помилки при визначенні аргументів.	
Первинний, вторинний актор	Первинний актор – користувач платформою (оператор).	
Тригер	Не передбачені	
Опис	Кроки	Дія
	1	Аутентифікація користувача в системі.
	2	Запуск оболонки СКБД.
	3	Виклик команд редагування, створення або видалення таблиць.
	4	Закінчення роботи з оболонкою СКБД.

4.3 Висновки за розділом

В даному розділі було розроблено діаграми варіантів використання системи на основі аналізу різновидів використання ПС на основі АСКАД. Визначені два актора в ПС: оператор та адміністратор. Виділені діючі суб’єкти АСКАД – актор та ПС на основі АСКАД.

При розробці діаграм було використано UML нотацію UC діаграм, що наочно відображає функціональність системи, взаємодію акторів з системою.

Деякі з UC були більш детально описані в таблиці, інші були коротко описані словесно.

Для оператора і адміністратора системи визначено контексти їх взаємодії із ПС на основі АСКАД, що описані в окремих діаграмах варіантів використання. Розподіл на контексти взаємодії дає можливість відокремити функціональність за підсистемами, що спрощує наочне представлення UC для ПС на основі АСКАД.

5 РОЗРОБЛЕННЯ СТРУКТУРИ ПРОГРАМНОЇ СИСТЕМИ НА ОСНОВІ АРХІТЕКТУРИ СИСТЕМИ КОМПЛЕКСНОГО АНАЛІЗУ ДАНИХ

Виходячи з зазначених вимог до ПС на основі АСКАД, визначено наступні структурні елементи [45]:

– ІРО,

– підсистема збору та обробки інформації,

– СКБД.

Структурна схема ПС на основі АСКАД наведена на рисунку 5.1.

Рисунок 5.1 – Структура системи

Визначені елементи дозволяють розділити функції всієї системи, що значно полегшує відлагодження системи і закладає принципи модульності. Така структура також визначає взаємозамінність кожної з підсистем у випадку необхідності.

В свою чергу, кожна з підсистем повинна також мати модульний принцип побудови, що є одним з критеріїв їх вибору.

5.1 Обґрунтування вибору інфраструктури розподілених обчислень

ІРО повинна відповідати наступним вимогам:

– горизонтальна масштабовність,

– наявність кластерної ФС,

– обробка великих даних,

– наявність інтерфейсів доступу до ФС та обчислювальних модулів,

– відмовостійкість.

З огляду на поставлені критерії, було обрано ІРО Hadoop. За необхідності дана ІРО може бути розгорнута на хмарному рішенні, що дає змогу зекономити на придбанні технічного обладнання.

5.1.1 Опис компонентів

Розглянемо докладно основні компоненти Hadoop (рис. 5.2) [46].

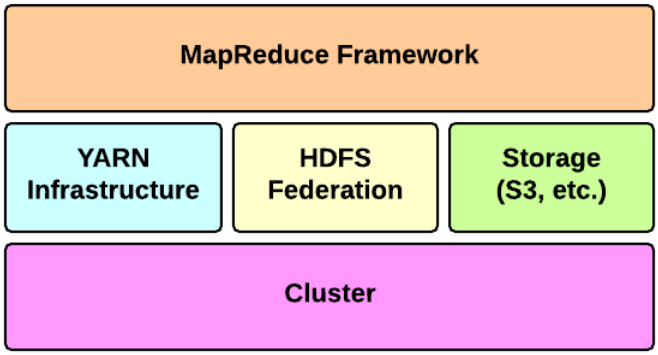


Рисунок 5.2 – Компоненти Hadoop [47]

Найнижчий рівень – кластер (cluster) – складається з декількох комп’ютерів (вузлів). Вузли можуть бути розділені на стеки вузлів у випадку, якщо кількість вузлів на сегмент мережі дуже велике.

HDFS – це фреймфорк, що відповідає за надання доступу до розподіленої ФС.

Інфраструктура YARN відповідає за виділення обчислювальних ресурсів серед вузлів, необхідних для виконання програм. Важливими складовими цієї інфраструктури є:

– менеджер ресурсів (Resource Manager) – знаходиться на master-вузлі та має необхідну інформацію про розміщення усіх вторинних вузлів і їх ресурси;

– менеджер вузла (Node Manager) – усі наявні вторинні вузли кластеру. Надає власні ресурси кластеру та виконує частину обчислень, що назначається менеджером ресурсів, при запуску завдання.

Storage представляє собою альтернативні сховища, що можуть бути використані системою.

MapReduce фреймворк, що побудований на вершині архітектури, визначає абстрактні методи розподілу завдань між вузлами та збору обчислених даних у єдиний приймач інформації, використовуючи усі наявні нижче за архітектурою інтерфейси.

5.1.2 Парадигма розподілених обчислень

Як вже було згадано вище (див. 5.1.1) в Hadoop реалізовано MapReduce парадигму обчислень [48], загальну діаграму потоку даних якої наведено на рисунку 5.3.

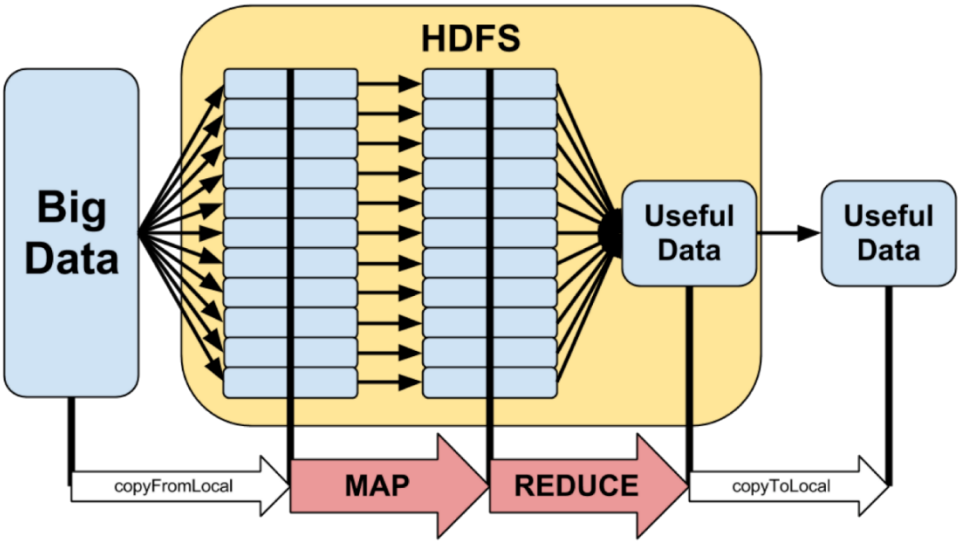


Рисунок 5.3 – Діаграма потоку даних MapReduce [48]

З діаграми можна побачити, що перед запуском завдання необхідно розмістити файли для аналізу на кластерній ФС HDFS.

Подальшим кроком є запуск завдання, що містить перевизначені методи ’Map’ та ’Reduce’. Системою буде визначена необхідна кількість “розподілів” та “збиральників”, і завдання почнеться з методу ’Map’. Кожному з вузлів призначається певний об’єм даних, який він оброблятиме, а після закінчення завдання передасть результати до одного зі “збиральників”.

Після того як усі вузли закінчать обробку даних, і буде виконаний повний збір результатів обробки з усіх вузлів, тоді завдання можна вважати закінченим, а результуючі дані розміщуються у розподіленій ФС.

5.1.3 Опис кластерної файлової системи

Головним компонентом HDFS є NameNode, що зберігає усі метадані (інформація про папки, файли і т.д.). Даний компонент розміщується на master-вузлі (рис. 5.4).

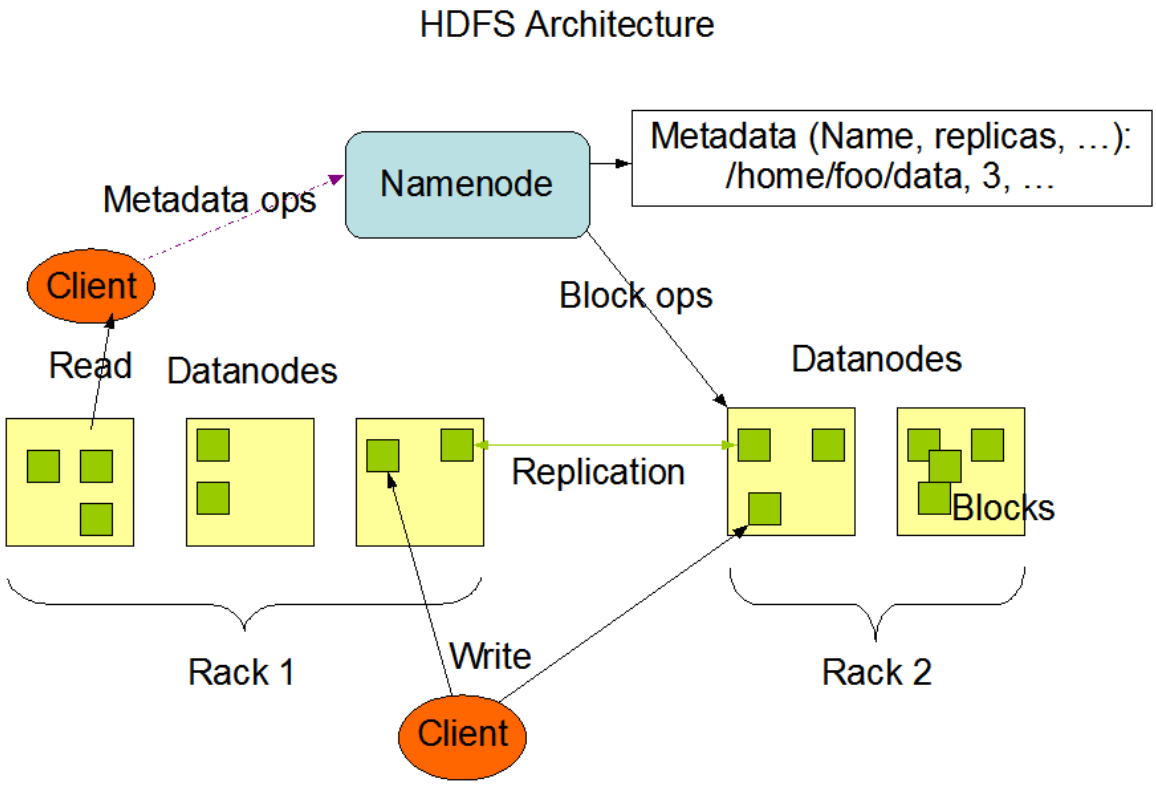


Рисунок 5.4 – Архітектура HDFS [49]

DataNode, що розміщуються на вторинних вузлах, зберігають дані, що записані до ФС.

Усі файли розбиваються на великі блоки, що розміщуються на вторинних вузлах (DataNode). Ті ж самі блоки можуть бути розміщені й на інших вузлах, що захищає ФС від втрати даних у випадку, якщо один з вторинних вузлів вийде з ладу – даний процес контролюється NameNode, в разі необхідності блок буде відновлено.

DataNode може управляти компонентами NameNode на вторинних вузлах, використовуючи команди:

– розповсюдити блок на інші вузли,

– видалити локальні копії блоку,

– перепідключення,

– відключення вузла.

Доступ до HDFS можливий за допомогою:

– команд Hadoop в оболонці системи,

– інтерфейсу браузера.

Інтерфейс браузера має обмежені можливості:

– перегляд папок,

– завантаження файлів.

Команди Hadoop надають повний доступ до ФС. У таблиці 5.1 наведено деякі типи команд.

Таблиця 5.1 – Команди Hadoop для управління ФС

Дія	Команда
Створити папку /bar	hadoop dfs -mkdir /bar
Видалити папку /bar	hadoop dfs -rmr /bar
Завантажити файл до ФС	hadoop fs -put <local-src> ... <HDFS_dest_path>
Завантажити файл з ФС	hadoop fs -get <hdfs-src> <localdst>
Проглянути зміст папки	hadoop fs -ls <args>

Таким чином, обрана IPO задовольняє поставлені до неї вимоги:

– горизонтально масштабовна,

– має власну кластерну ФС,

– призначена для обробки великих даних,

– є інтерфейси доступу до ФС та обчислювальних модулів,

– має базові механізми відмовостійкості.

5.2 Обґрунтування вибору підсистеми збору та обробки інформації

Вибір підсистеми збору та обробки інформації базується на наступних критеріях:

– модульний дизайн,

– масштабовність,

– сумісність з СКБД,

– сумісність з ІРО,

– гнучкі налаштування,

– розподіленість виконання задач,

– багатопоточність,

– відкрите програмне забезпечення.

В якості підсистеми збору та обробки інформації було обрано відкрите програмне забезпечення Apache Flume.

Flume є розподіленим та надійним сервісом збору, обробки та переміщень великих обсягів даних. Він має просту та гнучку архітектуру, що базується на потоках даних [50]. Спрощена структура Flume зображена на рисунку 5.5.

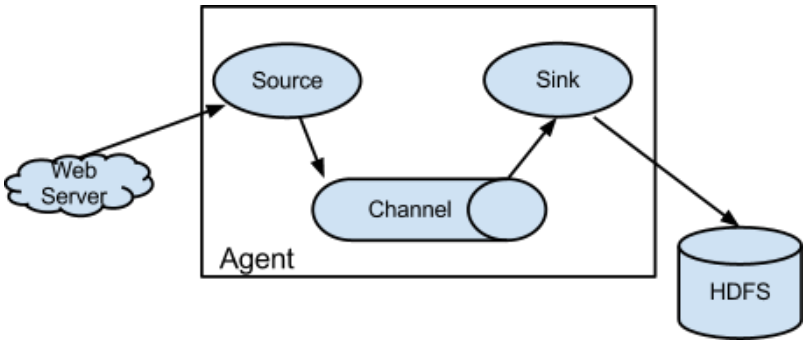


Рисунок 5.5 – Структура Flume [50]

Властивості обраного програмного забезпечення перелічені у таблиці 5.2.

Таблиця 5.2 – Властивості Flume

Властивість	Опис
Горизонтальна масштабовність	Для збільшення кількості об’ємів збираємих даних можна підключити додаткові агенти
Гарантована доставка даних	Канальний принцип передачі даних в купі з подійно-орієнтованою архітектурою системи гарантує доставку повідомлень
Ізольованість модулів	Наявність проміжного каналу передачі даних, що захищає від перехідних збурювань, коли кількість даних, що поступають, більша за кількість даних, що може бути записана до приймача
Потоки даних	Направлення потоків даних з кількох джерел до сховища Hadoop, бази даних, файлів та ін. приймачів

5.2.1 Опис компонентів

Конфігурація агентів Flume зводиться до налаштування трьох основних компонентів:

– джерел (source),

– каналів зв’язку (channel),

– приймачів (sink).

Розглянемо послідовно кожний з компонентів та їх властивості.

Flume підтримує наступні вбудовані типи джерел:

– Avro Source,

– Thrift Source,

– Exec Source,

– JMS Source,

– Spooling Directory Source,

– Twitter Source,

– Kafka Source,

– NetCat Source,

– Sequence Generator Source,

– Syslog Sources,

– HTTP Source,

– Stress Source,

– Scribe Source.

Опишемо деякі найбільш вживані типи джерел.

Exec Source запускає вказану в конфігурації Unix команду та обробляє інформацію з потоку стандартного виводу вказаної програми. Якщо робота програми припиняється, тоді зупиняється й робота. Дане джерело має параметри, що вказані в таблиці 5.3.

Таблиця 5.3 – Властивості Exec Source

Властивість	Опис
channels	Вказує на канали передачі даних
type	Тип джерела, має бути exec
command	Команда для виконання
shell	Додатковий виклик оболонки (/bin/bash -c)
restartThrottle	Затримка до перезапуску програми
restart	Якщо програма завершена, перезапустити
logStdErr	Додаткова обробка потоку stderr
batchSize	Максимальна кількість рядків зчитування і відправки за раз
batchTimeout	Затримка перед записом даних
selector.type	Мультиплексування або точна копія
selector.*	Залежить від selector.type
interceptors	Список перехоплювачів
interceptors.*	-

Spooling Directory Source дозволяє аналізувати дані з файлів у вказаній папці. Дане джерело зчитує дані з файлів та змінює їх назву в разі, якщо файл було

оброблено, або видаляє проаналізований файл. Перелік параметрів даного джерела наведено у таблиці 5.4.

Таблиця 5.4 – Властивості Spooling Directory Source

Властивість	Опис
channels	Вказує на канали передачі даних
type	Тип джерела, має бути spoolDir
spoolDir	Папка, з якої зчитуються дані
fileSuffix	Суфікс, що додається до проаналізованих файлів
deletePolicy	Коли видаляти файли: never або immediate
fileHeader	Чи додавати заголовок абсолютного шляху до файлу
fileHeaderKey	Ключ заголовку, див. попередній параметр
basenameHeader	Чи додавати заголовок з назви файлу
basenameHeaderKey	Ключ заголовку, див. попередній параметр
ignorePattern	Шаблон ігнорування файлів
consumeOrder	Порядок аналізу файлів, що базується на даті останньої модифікації: випадковий, найстаріші та найновіші файли.
inputCharset	Кодування символів у файлі

HTTP Source дозволяє обробляти події, що поступають через запроси HTTP GET або POST. Запроси поступають в оброблювач, який виконує парсинг даних і структурує їх для подальшого запису до каналу зв’язку. Параметри джерела перелічені у таблиці 5.5.

Таблиця 5.5 – Властивості HTTP Source

Властивість	Опис
channels	Вказує на канали передачі даних
type	Тип джерела, має бути http
port	Порт прив’язки
bind	Назва хосту або IP адреса джерела
handler	Оброблювач запитів

Продовження таблиці 5.5

Властивість	Опис
handler.*	Додаткові параметри оброблювача
enableSSL	Чи включати SSL
excludeProtocols	Список протоколів SSL/TLS, що виключаються
keystore	Шлях до сховища ключів
keystorePassword	Пароль від сховища

Flume підтримує наступні приймачі інформації:

– HDFS Sink,

– Hive Sink,

– Logger Sink,

– Avro Sink,

– Thrift Sink,

– IRC Sink,

– File Roll Sink,

– Null Sink,

– HBase Sinks,

– MorphlineSolr Sink,

– ElasticSearch Sink,

– Kite Dataset Sink,

– Kafka Sink.

Опишемо деякі з них.

HDFS Sink – це приймач інформації, що дозволяє записувати дані напряму до HDFS. Джерело підтримує створення текстових файлів на файлів послідовностей. Підтримується також стискання даних. Наявна гнучка конфігурація порядку створення та завершення запису до файлів: в залежності від часу, що пройшов від моменту початку запису або від кількості подій, що записані до файлу. Параметри наведені у таблиці 5.6.

Таблиця 5.6 – Властивості HDFS Sink

Властивість	Опис
channel	Вказує на канал передачі даних
type	Тип приймача, має бути hdfs
hdfs.path	Шлях до папки HDFS
hdfs.filePrefix	Префікс назви файлів
hdfs.fileSuffix	Суфікс назви файлів
hdfs.inUsePrefix	Використання префіксу для тимчасових файлів
hdfs.rollInterval	Час, через який система буде писати в інший файл
hdfs.rollSize	Розмір файлу перед переключенням
hdfs.rollCount	Кількість подій перед переключенням
hdfs.idleTimeout	Затримка, після якої неактивний файл буде закрито
hdfs.batchSize	Кількість подій перед записом у HDFS
hdfs.codeC	Метод стискання
hdfs.fileType	Тип файлу
hdfs.maxOpenFiles	Максимальна кількість файлів, що можуть бути відкриті
hdfs.callTimeout	Допустима затримка операцій HDFS
hdfs.threadsPoolSize	Кількість потоків на єдиний HDFS приймач
hdfs.round	Округлення часового штампу
hdfs.roundUnit	Одиниця, яка округлюється
hdfs.timeZone	Назва часової зони
hdfs.useLocalTimeStamp	Використання місцевого часу
hdfs.closeTries	Кількість спроб переіменувати файл перед його закриттям
hdfs.retryInterval	Затримка між спробами закрити файл
hdfs.serializer	Серіалізатор даних

Hive Sink дозволяє записувати дані (події) до таблиць або розділів Hive. Hive – це SQL база даних. Таким чином, даний приймач реалізує прямий запис даних до бази даних за допомогою транзакцій Hive. Параметри приймача наведені у таблиці 5.7.

Таблиця 5.7 – Властивості Hive Sink

Властивість	Опис
channel	Канал зв’язку
type	Тип компоненту, hive
hive.metastore	Шлях до Hive metastore
hive.database	Назва бази даних
hive.table	Назва таблиці
hive.partition	Перелік розділів для запису
hive.txnsPerBatchAsk	Кількість транзакцій в пакеті
heartBeatInterval	Затримка оновлення часу закінчення неактивних транзакцій
autoCreatePartitions	Автоматичне створення розділів
batchSize	Розмір пакетної транзакції
maxOpenConnetions	Максимальна кількість відкритих каналів
callTimeOut	Максимально дозволена затримка для Hive та HDFS
serializer	Серіалізатор
roundUnit	Одиниця округлення
timeZone	Часова зона
useLocalTimeStamp	Чи використовувати місцевий час

Logger Sink надає можливості налагодження системи та власних написаних модулів. Записана до приймача інформація записується до системного логеру у стандартний потік виводу. Параметри перелічені у таблиці 5.8.

Таблиця 5.8 – Властивості Logger Sink

Властивість	Опис
channel	Канал зв’язку
type	Тип компоненту, logger
maxBytesToLog	Максимальна кількість байтів тіла події для запису

File Roll Sink дозволяє записувати зібрану інформацію до файлів на локальному вузлі. Параметри наведені у таблиці 5.9.

Таблиця 5.9 – Властивості File Roll Sink

Властивість	Опис
channel	Канал зв’язку
type	Тип компоненту, file_roll
sink.directory	Папка, де розміщуватимуться файли
sink.rollInteral	Інтервал, через який система почне записувати в інший файл
sink.serializer	Серіалізатор
batchSize	Розмір пакету даних

HBaseSink – це приймач, що записує дані до СКБД HBase. Приймач автоматично зчитує параметри з файла конфігурації. В нього є підтримка захищеного запису до СКБД. Гарантується атомарність операцій над рядками СКБД. Для даного приймача існує декілька типів серіалізаторів, що аналізують дані і конвертують їх у формат, прийнятний для запиту до СКБД HBase:

– SimpleHbaseEventSerializer – записує отримані дані за принципом “як є” у вказані колонки; є можливість налаштувати вигляд ідентифікатору рядку;

– RegexHbaseEventSerializer – записує отримані дані у колонки, що вказані у конфігурації; дані аналізуються за допомогою шаблону і розбиваються згідно його вигляду.

Перелік параметрів приймача наведений у таблиці 5.10.

Таблиця 5.10 – Властивості HBase Sink

Властивість	Опис
channel	Канал зв’язку
type	Тип компоненту, hbase
table	Назва таблиці в СКБД
columnFamily	Сімейство колонок
zookeeperQuorum	Параметр quorum у налаштуваннях HBase

Продовження таблиці 5.10

Властивість	Опис
znodeParent	Шлях до головного регіону СКБД
batchSize	Розмір пакету даних
serializer	Серіалізатор
serializer.*	Додаткові параметри серіалізатора
kerberosPrincipal	Ідентифікатор користувача Kerberos для захищеного доступу
kerberosKeytab	Kerberos keytab

Канали зв’язку Flume дозволяють передавати інформацію від джерела до приймача. Визначені наступні вбудовані типи каналів:

– Memory Channel,

– JDBC Channel,

– Kafka Channel,

– File Channel,

– Spillable Memory Channel,

– Pseudo Transaction Channel.

Розглянемо деякі з них.

Memory Channel дозволяє передавати події за допомогою черги, що організована у пам’яті. Максимальний розмір черги можна налаштувати. Перелік параметрів можна подивитись у таблиці 5.11.

Таблиця 5.11 – Властивості Memory Channel

Властивість	Опис
type	Тип компоненту, memory
capacity	Максимальна кількість подій, що можуть бути збережені у пам’яті
transactionCapacity	Максимальна кількість подій, що канал прийме від джерела та передасть до приймача за транзакцію
keep-alive	Затримка додавання або видалення події

Продовження таблиці 5.11

Властивість	Опис
byteCapacityBufferPercentage	Відсоток буферу, що буде відведений на заголовки
byteCapacity	Максимально можлива кількість байт, що передаються пам’яттю

File Channel дозволяє передавати інформацію, використовуючи файли. Параметри наведені у таблиці 5.12.

Таблиця 5.12 – Властивості File Channel

Властивість	Опис
type	Тип компоненту, file
checkpointDir	Папка для збереження файлу checkpoint
useDualCheckpoints	Чи резервувати попередню папку?
backupCheckpointDir	Резервна папка checkpoint
dataDirs	Список папок для збереження файлів логів
transactionCapacity	Максимальний розмір транзакції каналу
checkpointInterval	Затримка між checkpoint
maxFileSize	Максимальний розмір одного файлу
minimumRequiredSpace	Мінімальний вільній простір
capacity	Максимальний розмір каналу
keep-alive	Максимальний час очікування запису
checkpointOnClose	Контролює створення checkpoint, коли канал закривається
encryption.activeKey	Назва ключа для шифрування даних
encryption.cipherProvider	Тип шифру
encryption.keyProvider	Тип ключу
encryption.keyProvider. keyStoreFile	Шлях до файлу ключів
encrpytion.keyProvider. keyStorePasswordFile	Шлях до файлу пароля до файлу ключів

Продовження таблиці 5.12

Властивість	Опис
encryption.keyProvider. keys	Список усіх ключів

5.2.2 Опис взаємодії компонентів

Розглянемо взаємодію компонентів системи за допомогою діаграми послідовності, що наведена на рисунку 5.6.

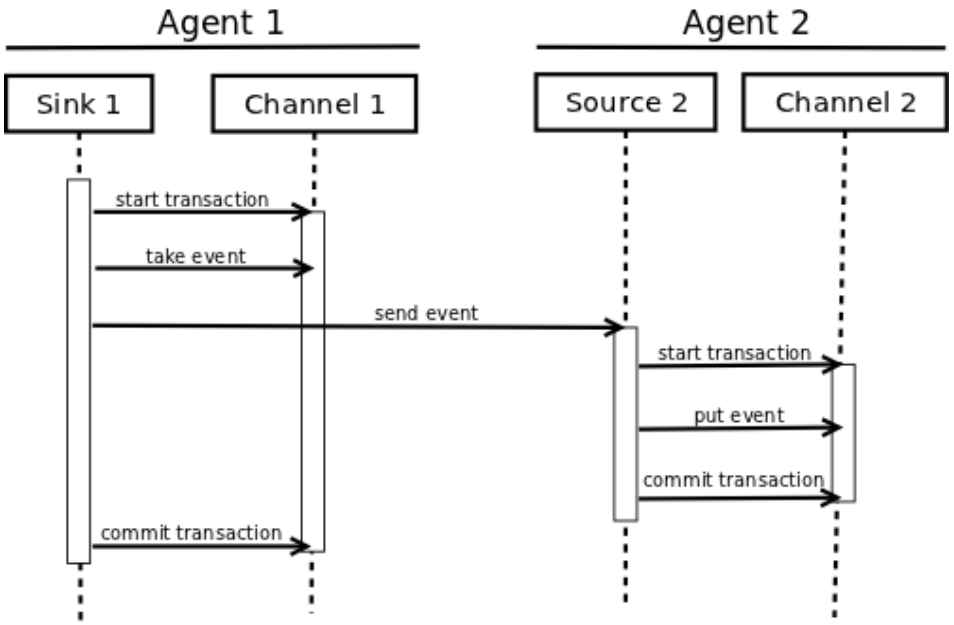


Рисунок 5.6 – Діаграма послідовності взаємодії компонентів [51]

Як можна побачити з діаграми послідовності, в даному прикладі розглядається взаємодія компонентів двох агентів – Agent 1 та Agent 2.

Для правильної роботи системи спочатку запускається Agent 1, оскільки він містить компонент Sink 1, що надає доступ до приймача. Якщо запустити Agent 2 першим, тоді виникне помилка через те, що наявне джерело даного агенту Source 2 не матиме доступу до приймача інформації, а сумісний канал передачі даних Channel 1 - Channel 2 буде обірваний.

Після запуску Agent 1, компонент Sink 1 починає транзакцію прийому даних з каналу зв’язку. Компонент Source 2 агенту Agent 2 приймає інформацію про початок

транзакції та готов до роботи. Коли у джерелі Source 2 виникає подія (наприклад, це може бути рядок інформації логів, результати роботи парсерів і так далі), то воно починає власну транзакцію з розміщення отриманих даних до каналу зв’язку Channel 2. В разі успішного виконання транзакції вона підтверджується, а приймач Sink 1 агенту Agent 1 підтверджує власну транзакцію з прийому даних з каналу зв’язку [52].

Такий підхід в системі, коли взаємодії між компонентами “загорнуті” у транзакції дозволяє отримувати достовірну інформацію, адже у випадку помилок при прийнятті або передачі інформації транзакція не буде підтвердженою. Це забезпечує надійність системи.

Для розуміння того, як об’єднуються потоки інформації у єдиному агенті, розглянемо діаграму потоку даних, що зображена на рисунку 5.7.

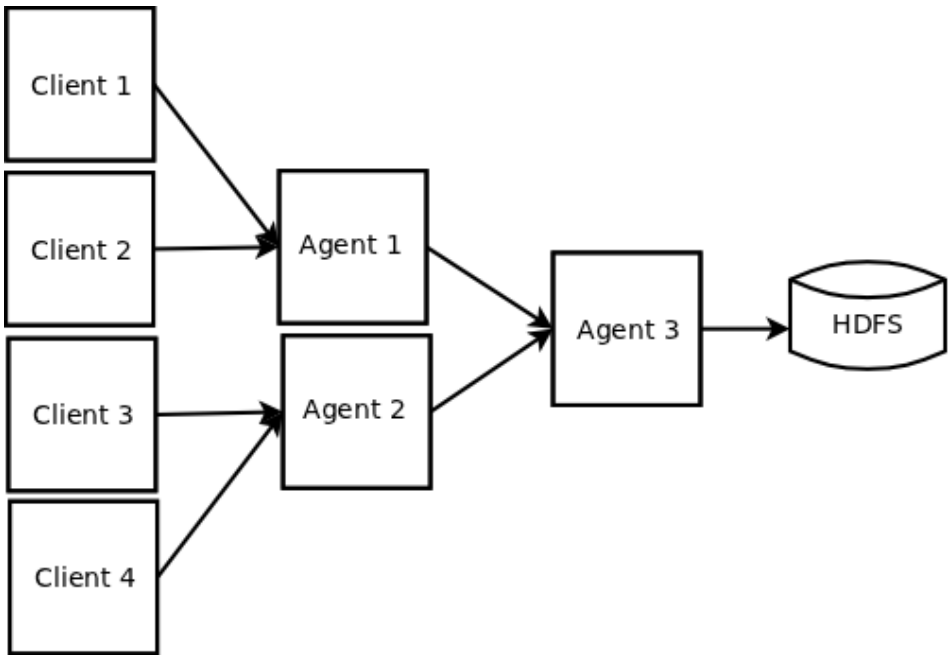


Рисунок 5.7 – Діаграма суміщених потоків Flume [51]

Оскільки кожен з агентів може вміщувати кілька джерел інформації, то зручно розподіляти декілька джерел між кількома вузлами IPO. З діаграми можна побачити, що Agent 1 та Agent 2 вміщують в собі два джерела, що передаються єдиним суміщеним каналом зв’язку до Agent 3, який прослуховує його.

Agent 3 містить компонент приймача, таким чином всі дані, що зчитуються з каналу зв’язку, записуються до приймача (за діаграмою приймачем є розподілена

файлова система HDFS). Таким чином, дана архітектура реалізує відношення “багато-до-одного”.

Розглянуті діаграми дозволяють зрозуміти взаємодію між компонентами системи для подальшої розробки більш складних архітектур (наприклад, взаємодія “багато-до-багатьох” і так далі). Розроблення архітектури залежить від кількості інформації, джерел, характеристик вузлів. Більш складні діаграми взаємодії дозволяють розподіляти навантаження між окремими вузлами ІРО.

Таким чином, підсистема збору та обробки інформації Flume надає можливості:

– масштабовності,

– розподіленого виконання збору та обробки інформації,

– багатопоточності.

Крім того, вона сумісна з базою даних та ІРО, що гарантує правильну роботу всієї системи.

5.3 Обґрунтування вибору системи керування базами даних

З урахуванням загальних вимог до системи було встановлено критерії вибору СКБД:

– масштабовність,

– сумісність з ІРО,

– сумісність з підсистемою збору та обробки інформації,

– можливість використання інтерфейсу доступу до СКБД зовнішніми модулями,

– надійність,

– відкрите програмне забезпечення.

Заданим вимогам відповідає обрана СКБД Apache HBase. Це стовпчико-орієнтована NoSQL СКБД. NoSQL означає, що СКБД відмінна від традиційного представлення даних у реляційних СКБД. Дані у HBase зберігаються у вигляді структурованих та напівструктурованих пар ключ-значення. Дана СКБД надає наступні можливості:

– запис-зчитування у режимі реального часу,

– інтерфейс управління СКБД,

– конфігурація розподіленої СКБД між вузлами,

– додаткові інтерфейси для доступу зі сторонніх програм.

5.3.1 Опис компонентів

Розглянемо архітектуру СКБД HBase [53], що показана на рисунку 5.8.

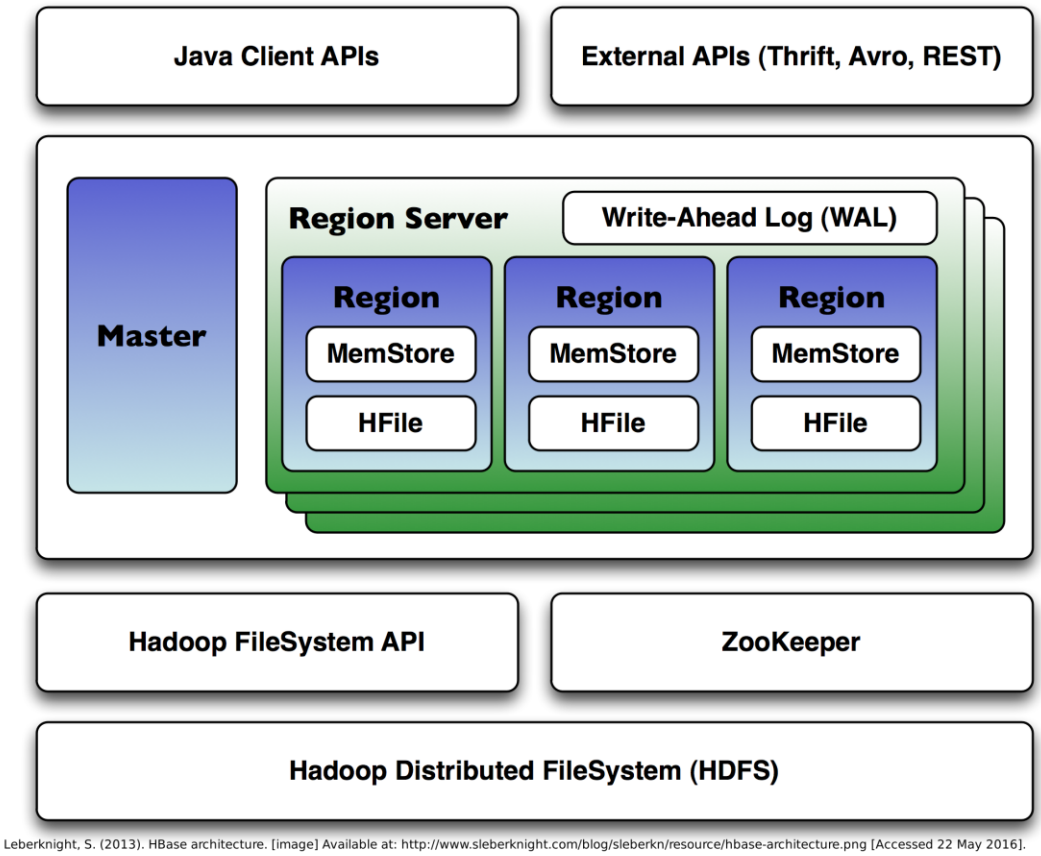


Рисунок 5.8 – Архітектура СКБД HBase [53]

HBase реалізує централізовану архітектуру Master-Slave, побудовану поверх шару HDFS, який контролюється за допомогою Hadoop.

HMaster є провідним вузлом і відповідальне за наступні операції:

– контроль над усіма HRegionServer вузлів,

– призначення регіонів,

– операції над метаданими.

На розподіленому кластері HMaster запускається на NameNode вузлі IPO.

HRegionServer – вторинні вузли, що відповідальні за обслуговування регіонів. Дані вузли розгортаються на DataNode вузлах IPO.

Zookeeper також контролює статуси HRegionServer. Він надає клієнтський інтерфейс для доступу до шляху розміщення головної таблиці.

Верхній шар діаграми показує можливі інтерфейси доступу до бази даних.

5.3.2 Модель даних

Для розуміння моделі даних СКБД HBase, розкриємо сутність складових цієї моделі [54].

Рядок в HBase складається з ключа рядка і одного або декількох стовпців зі значеннями, пов’язаними з ними. Рядки сортуються в алфавітному порядку по ключу, як вони зберігаються. З цієї причини конструкція ключа рядку є дуже важливою. Мета полягає в тому, щоб зберігати дані таким чином, щоб відповідні рядки знаходились поруч один з одним.

Стовпець в HBase складається з сімейства стовпців і специфікатора, які розмежовуються за допомогою двокрапки.

Сімейство стовпців фізично об’єднує набір стовпців і їх значень.

Стовпець специфікатор додається до сімейства стовпців для забезпечення доступу до даних.

Комірка є поєднанням рядку, сімейства стовпців і специфікатора і містить значення і часовий штамп, який є версією значення.

Розглянемо рисунок 5.9, на якому зображено логічне представлення таблиці СКБД.

Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor	ColumnFamily people
"com.cnn.www"	t9		anchor:cnnsi.com = "CNN"	
"com.cnn.www"	t8		anchor:my.look.ca = "CNN.com"	
"com.cnn.www"	t6	contents:html = "<html>..."		
"com.cnn.www"	t5	contents:html = "<html>..."		
"com.cnn.www"	t3	contents:html = "<html>..."		

Рисунок 5.9 – Приклад таблиці HBase [53]

Необхідно зауважити, що кількість стовпців у сімействі стовпців є необмеженою, як і кількість рядків.

З рисунку можна побачити 3 сімейства колонок:

– contents,

– anchor,

– people.

У кожному з сімейства стовпців можна створити колонки з вказаними специфікаторами. В даному прикладі визначені специфікатори ’cnnsi.com’ та ’my.look.ca’ у сімействі колонок ’anchor’; специфікатор ’html’ у сімействі колонок ’contents’, а також кожному з них надано відповідне значення.

Часовий штамп кожної з комірок виставляється системою автоматично згідно з дійсним часом запису. Також часовий штамп може змінюватись за допомогою інтерфейсу зв’язку з базою даних і виставлятися вручну.

5.3.3 Опис операцій над даними

В даній СКБД не існує суворої типізації і усі дані зберігаються у вигляді послідовності байтів. Тому перед початком роботи з даними необхідно конвертувати їх. Це є обов’язковою операцією як для запису даних, так і для зчитувань.

Основні операції з даними СКБД HBase перелічені у таблиці 5.13.

Таблиця 5.13 – Операції над даними HBase

Операція	Опис
Get/Scan	Операція визначена для заданого ідентифікатора рядку, сім'ї стовпців та специфікатора. Якщо не вказується необхідна версія значення, тоді повертається значення з часовим штампом останньої модифікації. Якщо необхідно, можна вказати часовий проміжок. Повертається масив байт у першому випадку або список масивів байт у другому.
Put	Операція завжди створює нову версію комірки. Вона також визначення для заданого ідентифікатора рядку, сім'ї стовпців та специфікатора. За замовчуванням, якщо не вказане інше, часовий штамп відповідає поточному значенню часу сервера. Може бути використана для зміни значення будь-якої версії комірки, для чого необхідно вказати точний часовий штамп необхідної комірки.
Delete	Операція визначена для специфічної версії комірки, цілого стовпчика або сімейства стовпчиків.

Усі операції повертають дані у сортованому вигляді послідовно: за ідентифікатором рядку, за сімейством стовпчиків, за специфікаторами та за часовими штампами.

Обрана стовпчико-орієнтована СКБД HBase задовольняє критеріям:

– масштабовності,

– сумісності з ІРО,

– сумісності з підсистемою збору та обробки інформації,

– наявності інтерфейсів доступу до СКБД зовнішніми модулями,

– швидкого доступу до великих обсягів даних,

– надійності.

5.4 Висновки за розділом

В даному розділі було розроблено структуру ПС на основі АСКАД і проаналізовано кожний структурний елемент.

За результатами аналізу вимог, що висуваються до такого роду ПС на основі АСКАД (див. 1, 3 розділи) було виділено три основних компоненти, на базі яких розгортається ПС:

– IPO,

– підсистема збору та обробки даних,

– СКБД.

В такий спосіб побудови ПС на основі АСКАД дозволяє дотримуватися модульного принципу побудови систем, що має ряд переваг при подальшій роботі з ПС на основі АСКАД, а саме:

– кожен структурний елемент має чіткий функціонал в загальній ПС, що закладає принцип єдиної відповідальності;

– поділ системи на підсистеми дозволяє більш ефективно відлагоджувати окремі підсистеми;

– кожна з підсистем може бути замінена на аналогічну функціональну одиницю за умов реалізації необхідних інтерфейсів зв’язку між ними.

В результаті аналізу окремих компонентів в якості IPO було обрано Hadoop, в ролі підсистеми збору та обробки даних виступає Apache Flume, а в якості СКБД обрано HBase. Кожний компонент задовольняє поставленим до всієї ПС на основі АСКАД вимогам.

.

6 РОЗРОБЛЕННЯ АРХІТЕКТУРИ СИСТЕМИ КОМПЛЕКСНОГО АНАЛІЗУ ДАНИХ

Підставою для побудови АСКАД є її структурна схема (див. рис. 5.1), що відображає наявні в ПС компоненти і зв’язок між ними.

АСКАД в UML нотації діаграми розгортання показана на рисунку 6.1 [55].

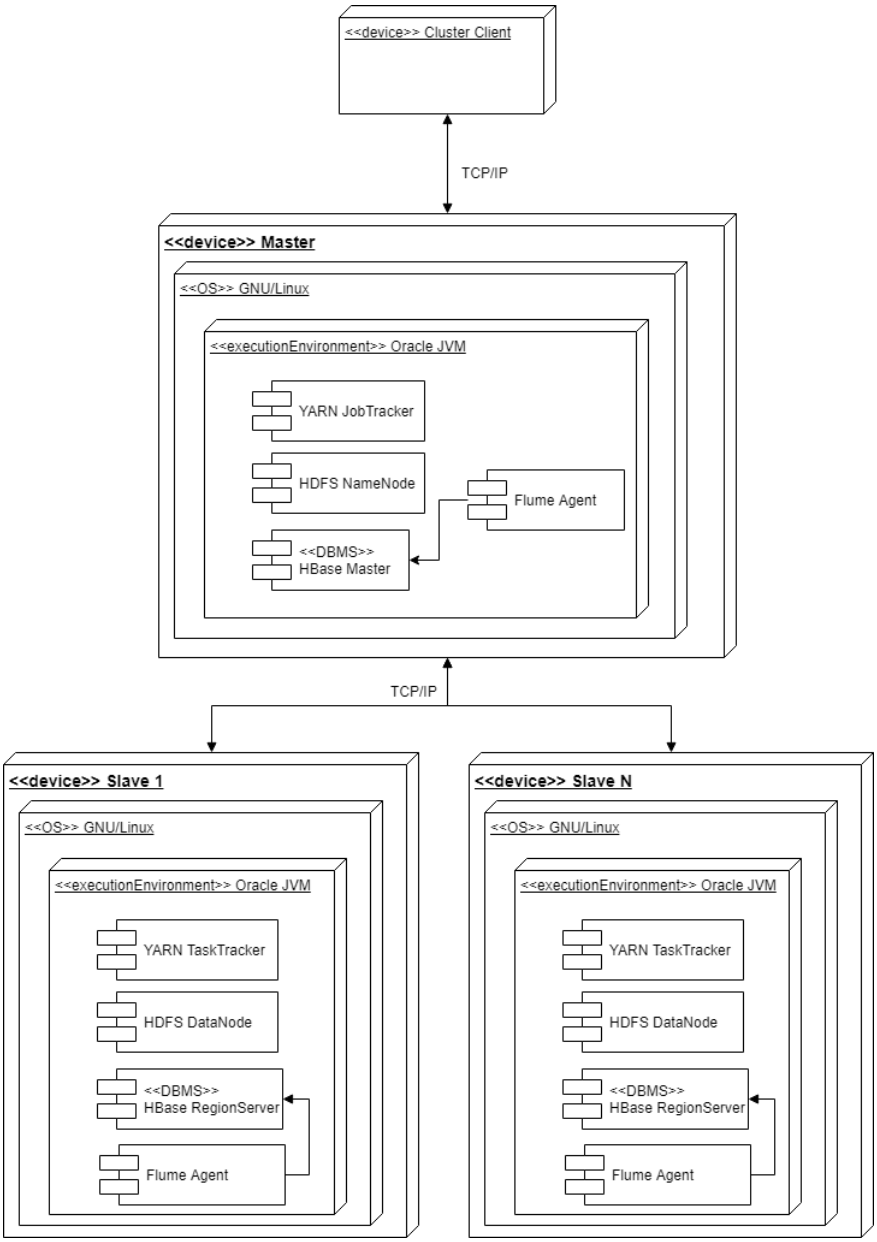


Рисунок 6.1 – Архітектура системи комплексного аналізу даних

Розроблення АСКАД передбачає повну підготовку ОС і відповідне налаштування усіх підсистем. Згідно визначених вимог до ПС на основі АСКАД, ОС

середовища виконання – Linux, і всі наведені і описані нижче команди відносяться виключно до даної ОС і не залежать від обраного дистрибутиву.

Усі описані налаштування мають бути однаковими серед вузлів системи.

6.1 Підготовка операційної системи

Перед запуском усіх модулів системи необхідно провести налаштування середовища виконання. Цей процес включає в себе налаштування:

– часової зони – забезпечує базову синхронізацію процесів між вузлами, модулями підсистем; правильно встановлений час надає можливість аналізувати логи окремих підсистем;

– користувачів та груп системи – забезпечує захист важливих для роботи файлів від втручання сторонніх осіб або процесів; виконує функцію розподілу прав між операторами та адміністратором системи;

– мережі – забезпечує мережевий зв’язок між вузлами системи для синхронізації завдань, розподілу ресурсів системи;

– доступу до вузлів – надає права безпарольного доступу до вузлів системи для нормальної роботи окремих підсистем;

– середовища виконання – надає можливість правильного запуску усіх модулів з коректним підключенням всіх наявних та необхідних бібліотек.

6.1.1 Налаштування часової зони

Для налаштування часової зони в ОС Linux спочатку необхідно визначити зону, до якої належить система. Усі можливі варіанти часових зон наведені у файлі ’/usr/share/zoneinfo’. Після визначення зони необхідно зробити прив’язку місцевого часу до обраної зони, що можна зробити двома шляхами:

– копіюванням файлу обраної зони з ’/usr/share/zoneinfo’ до папки ’etc’ з назвою ’localtime’;

– створенням символічного посилання файлу ’/etc/localtime’ на файл обраної зони.

Наступним кроком є встановлення синхронізатору часу, що надає сервіс автоматичного оновлення і коригування часу згідно з встановленими параметрами часового поясу. Можливе використання наступного ПЗ:

– NTP (Network Time Protocol),

– rdate.

Для остаточного налаштування часу системи необхідно вказати сервери синхронізації та перезапустити процеси.

6.1.2 Налаштування користувачів

Системою передбачено 2 типи користувачів:

– адміністратор,

– оператор.

Оскільки обліковий запис адміністратора вбудований в ОС, то необхідно додати запис оператора і налаштувати права відповідних папок і файлів, де розміщуються модулі системи.

Для створення облікового запису використовується команда ’adduser’. Параметри команди наведені в таблиці 6.1.

Таблиця 6.1 – Параметри команди «adduser» [56]

Ключ	Опис
-m	Створення домашньої папки користувача у ’/home/’. В даній папці користувач має права запису, видалення, зміни файлів, встановлення програм і так далі.
-g	Назва початкової групи користувача. Якщо параметр вказується у вигляді номеру групи або її назви, вона повинна існувати. Якщо параметр не вказується, тоді поведінка команди регулюється змінною

Продовження таблиці 6.1

Ключ	Опис
	’USERGROUPS_ENAB’ у файлі ’/etc/login.defs’. За замовчуванням, буде створено групу з аналогічною назвою.
-G	Додаткові групи, до яких включається користувач. Назви груп перелічуються через кому.
-s	Оболонка користувача за замовчуванням. Запускається після повного завантаження системи. За замовчуванням ’/bin/bash’.

Права та власник файлів або папок встановлюються за допомогою команд ’chmod’ та ’chown’ відповідно.

Усі файли системи повинні мати власника адміністратора, групу новоствореного оператора і визначені з наступними правами:

– для файлів – виконання, читання, запис;

– для власника – виконання, читання, запис;

– для групи – читання, виконання.

Таким чином, оператор системи матиме змогу проглядати конфігураційні файли та запускати окремі підсистеми.

6.1.3 Налаштування мережі

Наявність мережевого зв’язку між вузлами є важливою умовою роботи системи.

Конфігураційні файли різних дистрибутивів наведені в таблиці 6.2.

Таблиця 6.2 – Файли для конфігурації мережі

Файл	Опис
/etc/hosts	Перелік хостів
Red Hat/CentOS: /etc/sysconfig/network	Параметри підключення мережі, глобальне налаштування

Продовження таблиці 6.2

Файл	Опис
Red Hat/CentOS: /etc/sysconfig/network-scripts/ifcfg-device	Мережеві параметри окремого пристрою
Ubuntu/Debian: /etc/network/interfaces	Вказується конфігурація та пристрої, наприклад, статична адреса, DHCP

До файлу з переліком хостів мають бути прописані назви з відповідною IP адресою усіх вузлів для зручності подальшого налаштування окремих підсистем.

6.1.4 Налаштування доступу до вузлів

Для доступу до вторинних вузлів використовується Secure Shell (SSH) мережевий протокол. SSH надає захищений канал для підключення SSH клієнту до серверу.

SSH використовує криптографічну систему з відкритим ключем для аутентифікації до віддалених вузлів та користувачів вузлів. Існує два підходи до використання SSH:

– шифрування мережевого трафіку з використанням парольного доступу до віддаленого вузла;

– створення пари ключів для аутентифікації, що дозволяє програмам та користувачам отримати безпарольний доступ до вузла.

В даній ПС на основі АСКАД використовується безпарольний доступ. Для коректної роботи необхідно створити пару ключів на master-вузлі системи і розповсюдити публічний ключ серед вторинних вузлів.

6.1.5 Налаштування середовища виконання

В якості середовища виконання використовується Java.

Налаштування середовища Java зводиться до встановлення дійсного шляху до головного каталогу, де було встановлено середовище, що контролюється за допомогою змінної ’JAVA_HOME’. Якщо шлях буде встановлений не вірно, Java додатки не матимуть змогу завантажити необхідні стандартні бібліотеки і буде оголошено виключення.

Для того, щоб можна було запустити середовище виконання, необхідно також встановити зміну ’PATH’, що відповідає за всі шляхи розміщення виконавчих файлів.

6.2 Підготовка окремих компонентів програмної системи

Перед налаштуванням компонентів ПС на основі АСКАД необхідно упевнитися, що встановлені сумісні версії, інакше може виникнути багато помилок в процесі роботи. Налаштування кожного з компонентів проводиться тільки з використанням конфігураційних файлів.

6.2.1 Налаштування інфраструктури розподілених обчислень

Для IPO Hadoop налаштування проходить за допомогою чотирьох конфігураційних файлів, кожен з яких відповідає за окремий модуль інфраструктури:

– ядра IPO,

– ФС HDFS,

– менеджера ресурсів YARN,

– фреймворку MapReduce.

Параметри конфігурації ядра IPO наведені в таблиці 6.3 [57]. Окремі модулі IPO використовують надану конфігурацію ядра і можуть перезаписувати власні параметри, якщо не задані інші.

Таблиця 6.3 – Параметри ядра IPO

Параметр	Опис
fs.defaultFS	Шлях до хосту NameNode у вигляді ’hdfs://host:port/’
io.file.buffer.size	Розмір буфера запису-читання для файлів послідовностей

Перед налаштуванням HDFS треба створити всі необхідні папки на локальній ФС кожного з вузлів для коректної роботи. Параметри конфігурації наведені в таблиці 6.4.

Таблиця 6.4 – Параметри ФС HDFS

Параметр	Опис
dfs.namenode.name.dir	Шлях до папки на локальній ФС, де NameNode зберігатиме простір імен та логи транзакцій. Задається у вигляді переліку через кому
dfs.hosts	Перелік дозволених DataNode
dfs.blocksize	Розмір блоку даних HDFS для файлів
dfs.namenode.handler.count	Кількість потоків NameNode серверу для обробки RPC, що поступають від DataNode
dfs.datanode.data.dir	Шлях до папки на локальній ФС, де DataNode зберігатиме блоки даних. Задається переліком через кому

Налаштування YARN дозволяє запускати програми у розподіленому режимі. Інакше всі програми запускатимуться на master-вузлі. Параметри конфігурації YARN наведені в таблиці 6.5.

Таблиця 6.5 – Параметри YARN

Параметр	Опис
yarn.acl.enable	Чи включено ACL?
yarn.admin.acl	Перелік ACL груп доступу до кластеру
yarn.log-aggregation-enable	Включення або відключення агрегації логів
yarn.resourcemanager.address	Адреса менеджера ресурсів для запуску задач
yarn.resourcemanager.scheduler.address	Адреса менеджера ресурсів для ApplicationMasters для виділення ресурсів

Параметр	Опис
yarn.resourcemanager.resource-tracker.address	Адреса менеджера ресурсів для NodeManager
yarn.resourcemanager.admin. address	Адреса менеджера ресурсів для команд адміністратора
yarn.resourcemanager.webapp. address	Адреса до веб-додатку
yarn.resourcemanager.hostname	Хост менеджера ресурсів
yarn.resourcemanager.scheduler. class	Клас планувальника менеджера ресурсів
yarn.scheduler.minimum-allocation-mb	Мінімальний розмір пам’яті для виділення на запит контейнера
yarn.scheduler.maximum-allocation-mb	Максимальний розмір пам’яті для виділення на запит контейнера
yarn.resourcemanager.nodes. include-path	Перелік дозволених NodeManager
yarn.nodemanager.resource. memory-mb	Встановлює загальну кількість дозволених до використання ресурсів на NodeManager
yarn.nodemanager.vmem-pmem-ratio	Максимальне відношення використання віртуальної пам’яті до наявної фізичної
yarn.nodemanager.local-dirs	Шлях до папок на локальній ФС для збереження проміжних даних. Перелік через кому
yarn.nodemanager.log-dirs	Шлях до папок на локальній ФС для збереження логів
yarn.nodemanager.log.retain-seconds	Час зберігання логів на NodeManager
yarn.nodemanager.remote-app-log-dir	Шлях до папки HDFS, куди записуватимуться логи виконання програм
yarn.nodemanager.remote-app-log-dir-suffix	Суфікс, що додається до назви папки логів у HDFS

Продовження таблиці 6.5

Параметр	Опис
yarn.nodemanager.aux-services	Додаткові сервіси NodeManager. Встановлюється ’mapreduce_shuffle’
yarn.log-aggregation.retain-seconds	Час зберігання агрегованих логів до їх видалення
yarn.log-aggregation.retain-check-interval-seconds	Затримка між перевірками агрегованих логів

Параметри конфігурації MapReduce наведені в таблиці 6.6.

Таблиця 6.6 – Параметри MapReduce

Параметр	Опис
mapreduce.framework.name	Назва фреймворку обробки. Встановлюється ’yarn’
mapreduce.map.memory.mb	Ліміт ресурсів при розподіленні
mapreduce.map.java.opts	Додаткові параметри JVM
mapreduce.reduce.memory.mb	Ліміт ресурсів при зборі даних
mapreduce.reduce.java.opts	Додаткові параметри JVM
mapreduce.task.io.sort.mb	Ліміт пам’яті при сортуванні даних
mapreduce.task.io.sort.factor	Кількість потоків, що об’єднуються за раз, в процесі сортування файлів
mapreduce.reduce.shuffle. parallelcopies	Кількість паралельних копій, що запускаються при зборі даних з розподілених обчислювачів
mapreduce.jobhistory.address	Адреса серверу збереження задач MapReduce
mapreduce.jobhistory.webapp. address	Адреса до веб-додатку серверу збереження задач
mapreduce.jobhistory.intermediate-done-dir	Шлях до папки з проміжними файлами історії виконання задач

Продовження таблиці 6.6

Параметр	Опис
mapreduce.jobhistory.done-dir	Шлях до папки з файлами історії виконання задач серверу

Результат роботи правильно налаштованої IPO можна побачити на рис. А.1 додатка А.

6.2.2 Налаштування підсистеми збору та обробки інформації

Налаштування даної підсистеми включає в себе налаштування окремих компонентів, що описані у підрозділі 5.2.1.

Для запуску розподіленого збору інформації та обробки мають бути запущені не менше двох агентів на різних вузлах. Один з агентів має розміщуватися на master-вузлі системи і виконуватиме наступні задачі:

– збір інформації з других агентів,

– запис зібраних даних до СКБД,

– власний збір інформації з джерел (опціонально).

Інші агенти розміщуються по вузлах системи і виконують збір інформації. Для досягнення поставленої мети необхідно налаштувати інтерфейс з базою даних агенту на master-вузлі та канали зв’язку з вторинними вузлами.

Для кожного агенту визначається його назва та відповідні параметри. Спочатку перелічуються компоненти, що використовуватимуться в даному агенті, після чого йде опис їх параметрів.

Важливими параметрами для компонентів є розміри буферів, оскільки за умови їх переповнення система може відкинути дані, що продовжують поступати у кількості, більшій ніж можливо передати.

Приймач агенту master-вузла встановлюється в якості СКБД з відповідними налаштуваннями. Приймачі інших агентів налаштовуються на передачу інформації до master-вузла [58].

6.2.3 Налаштування системи керування базами даних

СКБД може бути запущена у двох режимах:

– розподіленому,

– нерозподіленому.

Необхідний режим роботи для даної системи – розподілений. Окрім вибору режиму роботи СКБД, необхідно також перелічити всі вторинні вузли, між якими розподілятимуться дані та виконання операцій над БД.

Важливо перед запуском сконфігурованої СКБД створити папку на кластерній ФС HDFS, що вказується в конфігураційному файлі, інакше система працюватиме некоректно. Головні параметри ядра СКБД наведені у таблиці 6.7.

Таблиця 6.7 – Параметри ядра СКБД

Параметр	Опис
hbase.tmp.dir	Тимчасова папка на локальній файловій системі
hbase.rootdir	Папка, що розділяється між вузлами СКБД. Має знаходитись у розподіленій ФС HDFS
hbase.cluster.distributed	Відповідає за режим роботи СКБД. Можливі значення – ’true’ та ’false’ відповідно для розподіленого та нерозподіленого режимів
hbase.zookeeper.quorum	Перелік вузлів у составі ZooKeeper, через кому. Наприклад, ’host1.domain.com,host2.domain.com’. На усіх перелічених вузлах буде запущений сервіс ZooKeeper в разі запуску СКБД.

На рисунках А.2 та А.3 додатку А можна побачити результати налаштування розподіленої між вузлами СКБД та створених таблиць для тестових модулів відповідно.

6.3 Висновки за розділом

В даному розділі було розроблено АСКАД, описано необхідні налаштування ОС та окремих підсистем для правильної роботи ПС на основі АСКАД.

Розроблена в розділі АСКАД задовольняє поставленим вимогам щодо функціоналу ПС, що був описаний при розробці діаграм варіантів використання ПС у розділі 4.

7 РОЗРОБЛЕННЯ ТЕСТОВОЇ НЕРЕЛЯЦІЙНОЇ БАЗИ ДАНИХ

Для тестування ПС на основі АСКАД було розроблено модулі збору, обробки й аналізу новин з наявних відкритих джерел. Вибір джерела інформації ґрунтувався на наступних засадах:

– відкритість,

– безоплатність,

– великі об’єми.

Для даної тестової ПС на основі АСКАД використовуються дві таблиці:

– для збереження новин,

– інвертованого індексу;

Таблиця для збереження новин схематично представлена на рисунку 7.1.

Row ID	news_data					
	category	title	summary	link	published	indexed

Рисунок 7.1 – Схематичний вигляд таблиці у СКБД

В даній таблиці ’news_data’ представляє сімейство стовпців.

Стовпець ’Row ID’ призначений для унікальної ідентифікації новини і будується з трьох складових:

– час, у мілісекундах;

– випадковий ключ, унікальний для кожного агенту підсистеми збору й обробки даних;

– інкременту.

Такий ідентифікатор виключає неунікальність даних між двома і більше агентами.

Стовпчик ’indexed’ призначений для встановлення ознаки “проіндексована новина” модулем аналізу і має значення 0 або 1. Усі інші описані стовпці відповідають (відповідно до рис. 17 зліва направо) за:

– категорію,

– назву,

– короткий опис,

– посилання,

– дату публікації.

Схематичний вигляд таблиці інвертованого індексу наведено на рисунку 7.2.

Row ID	i			
	row_id_1	row_id_2	. . .	row_id_n

Рисунок 7.2 – Схематичний вигляд таблиці інвертованого індексу

Наведена таблиця представляє собою структуру, де в якості ’Row ID’ виступає відповідний терм з безліччю стовпців у сімействі стовпців ’i’, що зазначають відповідний ’Row ID’ новини, де даний терм наявний, з зазначенням кількості входжень терму у даній новині.

8 РОЗРОБЛЕННЯ ТЕСТОВИХ МОДУЛІВ

В якості формату джерела даних було обрано RSS. Такий вхідний формат надає наступні переваги:

– надає короткий опис новин,

– включає метадані,

– має напівструктурований вигляд,

– уніфікований для всіх джерел.

8.1 Підсистема збору й обробки інформації

Дана підсистема повинна забезпечити неперервний збір інформації з різних джерел новин (наприклад, BBC, Reuters і т.д.).

Після зчитування даних RSS, модуль збору розбиває метадані і комплектує необхідні у формат JSON для подальшої передачі у модуль обробки подій.

Оскільки вбудованого оброблювача даних у форматі JSON у складі приймачів HBase немає, було розроблено власний.

8.1.1 Розроблення модуля збору інформації

Для реалізації безперервного збору інформації було використано цикл з таймаутом. Після закінчення затримки, повторюється головне тіло програми – запит до RSS джерел (див. лістинг Б.1).

Отримані дані оброблюються парсером і визначається час новини. Якщо дана інформація вже була оброблена (час публікації старший від часу останнього запиту), тоді запис ігнорується.

Відібрані дані форматуються згідно стандарту JSON і передаються до стандартного потоку виводу. Блок-схема алгоритму наведена на рисунку 8.1.

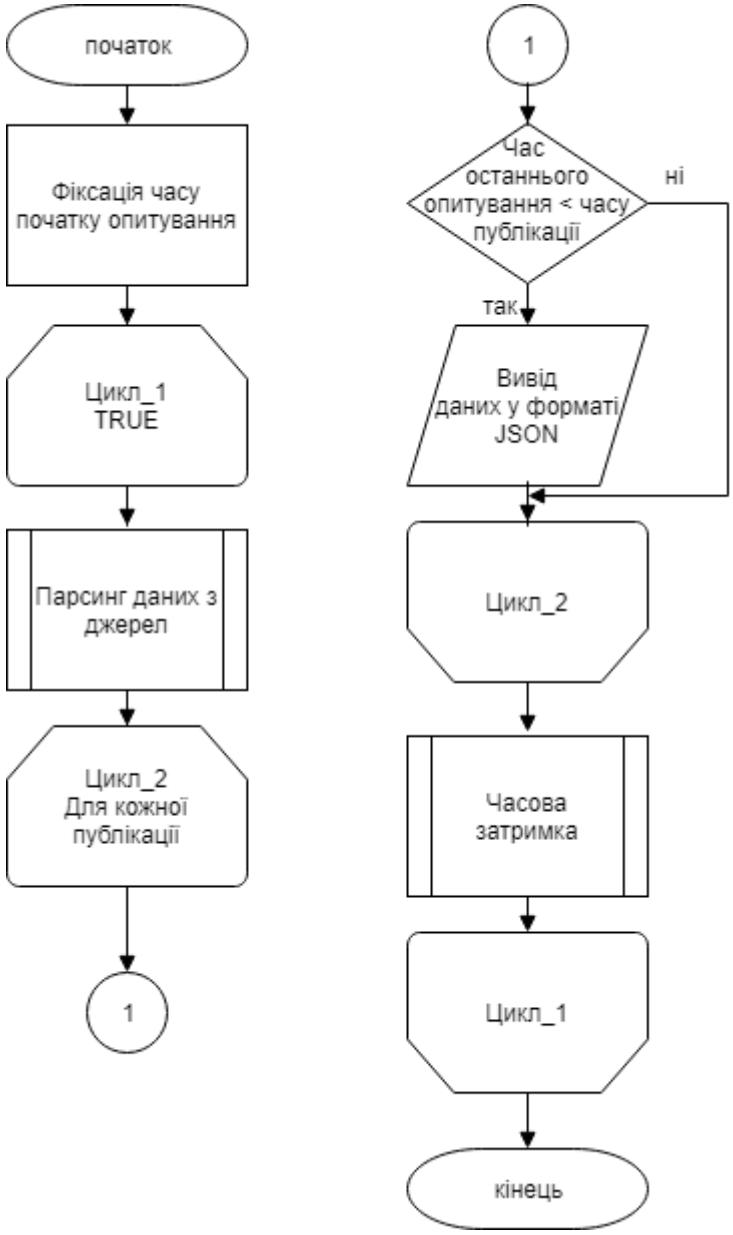


Рисунок 8.1 – Блок-схема роботи модуля збору інформації

8.1.2 Розроблення модуля обробки подій

В процесі розробки даного модулю було реалізовано інтерфейс Hbase-EventSerializer. Для цього були визначені всі методи, що наявні в інтерфейсах (див. лістинг Б.2).

Діаграму класів розробленого модулю наведено на рисунку 8.2.

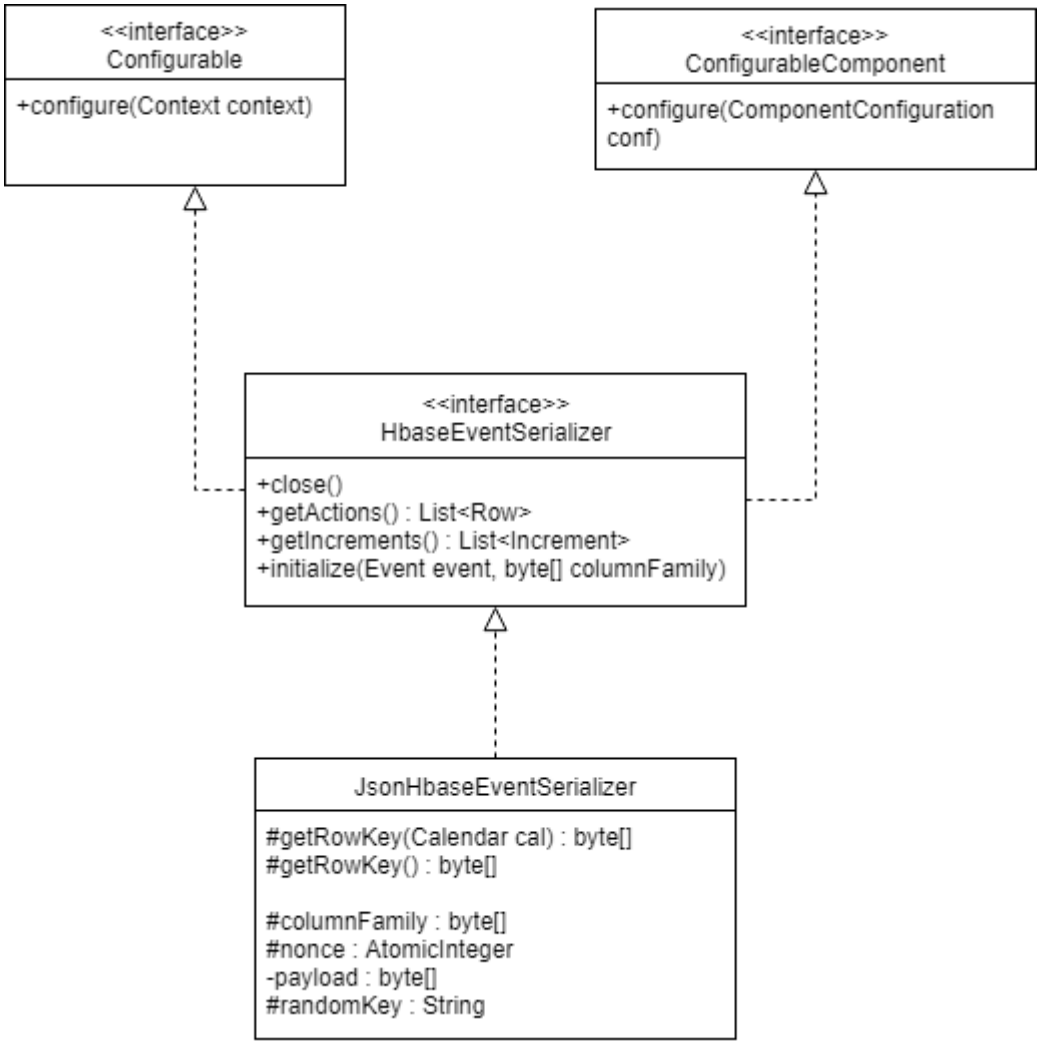


Рисунок 8.2 – Діаграма класів модуля обробки подій

HbaseEventSerializer надає інтерфейс серіалізатора подій, що обробляє заголовки та тіло подій для запису їх у СКБД HBase. Оскільки даний інтерфейс розширює інтерфейс Configurable, то він також надає можливість передати окремі параметри у конфігураційному файлі до оброблювача.

Пояснення методів надані у таблиці 8.1.

Таблиця 8.1 – Опис методів інтерфейсу HbaseEventSerializer

Метод	Опис
initialize(Event event, byte[] columnFamily)	Ініціалізація серіалізатора подій

Продовження таблиці 8.1

Метод	Опис
getActions()	Отримує перелік дій, що будуть виконані над СКБД HBase як результат даної події. Перелік оброблюється за допомогою HBase API у пакетному режимі
getIncrements()	Повертає список інкрементів, що будуть записані до СКБД у відповідь на подію

8.2 Розроблення модуля аналізу даних

Даний модуль дозволяє побудувати таблицю інвертованого індексу для швидкого пошуку новин за фразами (дивись лістинг Б.3).

Для цього у методі ’Map’ реалізовано підрахунок числа входжень усіх термів обраного запису таблиці. Якщо даний терм входить до списку “стоп-слів”, тоді він пропускається.

В даному алгоритмі немає методу ’Reduce’, оскільки немає вихідної інформації для агрегації.

В результаті запуску модуля виконане завдання зберігається в історії запуску завдань (рис. В.1). Результати роботи модуля аналізу можна побачити на рис. В.2.

Діаграму класів модуля наведено на рисунку 8.3.

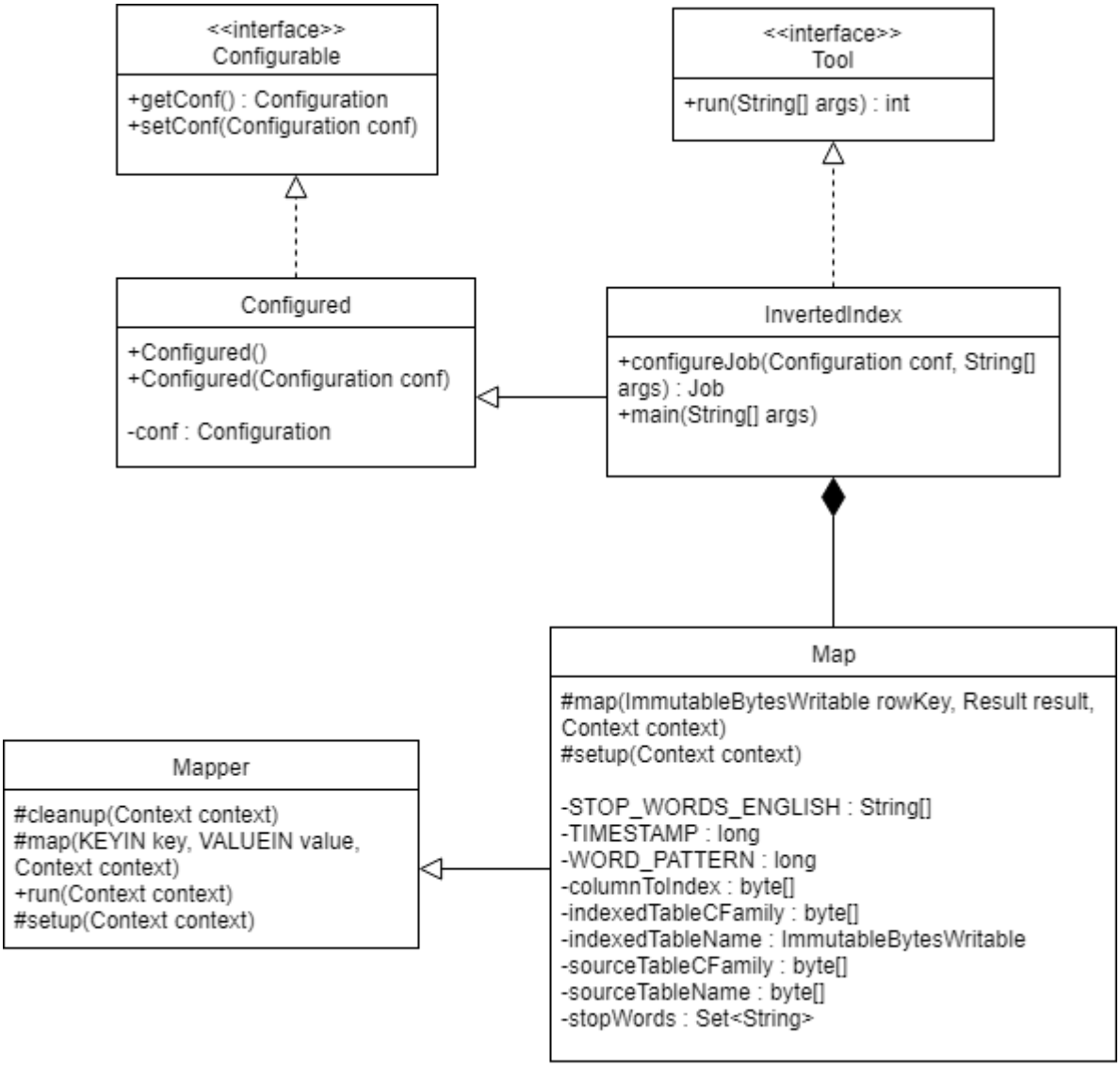


Рисунок 8.3 – Діаграма класів модуля аналізу

8.3 Висновки за розділом

В даному розділі було розроблено діаграми класів окремих модулів ПС на основі АСКАД для підтвердження працездатності отриманої АСКАД з використанням UML нотації опису.

В якості вхідних даних для підсистеми збору та обробки було обрано RSS стрічку новин, що представляє собою напівструктурований формат даних.

Оскільки СКБД працює тільки зі структурованими даними, одним з етапів роботи модуля є формування структури даних, приведення їх до придатного формату

для збереження до нереляційної бази даних, процес розроблення якої описаний в розділі 7.

9 СТАРТАП ПРОЕКТ

Завдання розділу полягає в маркетинговому аналізі перспектив реалізації запропонованих науково-технічних рішень та пропозицій, оцінювання можливостей їх ринкового впровадження.

9.1 Опис ідеї проекту

Опис ідеї проекту наведений в таблиці 9.1.

Таблиця 9.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система автоматизованого збору, обробки та аналізу великих обсягів даних	Високопродуктивні обчислення даних	Конфіденціальність інформації, власні обчислювальні потужності для збору, обробки та аналізу, ін. потреб (вирішення задач, які потребують великі потужності)
	Збереження файлів	Власне хмарне файлове сховище, можливості обчислювання з попереднім структуруванням даних безпосередньо у сховищі; немає необхідності в синхронізації даних з віддаленим центром обробки даних
	Структуризація даних	Огляд даних, вибір методів аналізу для даного набору

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

– визначення переліку техніко-економічних властивостей та характеристик ідеї;

– визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

– проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (табл. 9.2).

Таблиця 9.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Хмарні обчислення	Хмарні обчислення зі сховищем			
1.	Вартість обслуговування	середня	висока	висока	Вартість частково перекладається на клієнта		Вартість порівняно низька з конкурентами
2.	Вартість утилізації	немає	висока	висока	Перекладається повністю на клієнта	Якщо технічне забезпечення працює, можна додати до інфраструктури	Вартість утилізації відсутня

Продовження таблиці 9.2

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Хмарні обчислення	Хмарні обчислення зі сховищем			
3.	Надійність	надійний	Менш надійний	Менш надійний	Залежить від стану технічного забезпечення клієнту		Надійність за рахунок власної інфраструктури
4.	Технологічні (потреба у виділеному каналі зв'язку з ЦОД)	немає	є	є			Не потрібен дорогий виділений канал зв'язку

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

9.2 Технологічний аудит ідеї проекту

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (табл. 9.3):

– за якою технологією буде виготовлено товар згідно ідеї проекту?

– чи існують такі технології, чи їх потрібно розробити/доробити?

– чи доступні такі технології авторам проекту?

Таблиця 9.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Система автоматизованого збору, обробки та аналізу великих обсягів даних	Розгортання системи вручну	+	+
2		Розгортання системи автоматизованими скриптами з контролюванням процесу людиною	+	+ Необхідна розроблення скриптів (не потрібно багато часу)
3		Автоматичний інсталятор, віддалений контроль процесу	+	+/- Необхідна розроблення інсталятору, що займає велику кількість часу
Обрана технологія реалізації ідеї проекту: Розгортання системи автоматизованими скриптами з контролюванням процесу людиною				

9.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Аналіз попиту наведений у таблиці 9.4.

Таблиця 9.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	0
2	Загальний обсяг продаж, грн/ум.од	100000$
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	13,1%

Середня норма рентабельності проекту більша за середній банківський відсоток на вкладення, тому ринок є привабливим для входження за попереднім оцінюванням.

Визначені потенційні групи клієнтів, їх характеристики, та сформований орієнтовний перелік вимог до товару для кожної групи наведений у таблиці 9.5.

Таблиця 9.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Збір, обробка та аналіз великих обсягів даних, прогнозування	Всі види бізнесу (прогнозування, аналіз і так далі)	Споживачі з готовою, налаштованою інфраструктурою мережі	- підтримка ПЗ - підтримка працездатності системи

Продовження таблиці 9.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
			Інфраструктура відсутня	- розроблення ПЗ - оновлення інфраструктури

Проведений аналіз ринкового середовища, складено таблицю факторів, що перешкоджають ринковому впровадженню проекту (табл. 9.6). Фактори в таблиці подано в порядку зменшення значущості.

Таблиця 9.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Попит діючого бізнесу	Неусвідомленість користі інструменту для подальшого зростання бізнесу, з чого випливає можливий низький попит на даному етапі	Донесення інформації про користь інструменту, практичні приклади можливого використання
2	Поява нових конкурентів	Зменшення долі ринку	Цінова конкуренція

Складений перелік факторів, що сприяють ринковому впровадженню проекту, наведено в таблиці 9.7.

Таблиця 9.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Збільшення попиту	Усвідомлення можливостей інструменту може призвести до збільшення попиту	Впровадження глибшої автоматизації процесів (скорочення часу розгортання), розширення штату розробників
2	Технічний прогрес	Придбання нових технічних засобів призведе до необхідності утилізації старих, але їх можна використати для збільшення обчислювальних можливостей інфраструктури	Впровадження більш простих процесів інтеграції в наявну інфраструктуру

Проведений аналіз пропозиції: визначені загальні риси конкуренції на ринку описані в таблиці 9.8.

Таблиця 9.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції	монополія	-

Продовження таблиці 9.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
- монополія/олігополія/ монополістична/чиста		
2. За рівнем конкурентної боротьби - локальний/національний/…	національний	-
3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	внутрішньогалузева	-
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - між бажаннями	-	-
5. За характером конкурентних переваг - цінова / нецінова	-	-
6. За інтенсивністю - марочна/не марочна	-	-

М. Портер вирізняє п’ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції. Це:

– конкурент, що вже є у галузі,

– потенційні конкуренти,

– наявність товарів-замінників,

– постачальники, що конкурують за ринкову владу,

– споживачі.

Сильні позиції компанії за кожним з факторів означають її можливості забезпечити необхідні темпи обороту капіталу та її здатність впливати на інших агентів ринку, диктуючі їм власні умови співпраці. Характеристики факторів моделі відрізняються для різних галузей та змінюються із часом. Сила кожного фактору є функцією від структури галузі та її техніко-економічних характеристик.

На основі аналізу складових моделі 5 сил М. Портера розроблено перелік факторів конкурентоспроможності, що наведений у таблиці 9.9.

Таблиця 9.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Немає	Немає бар’єрів	Немає необхідності в постачальниках	Надійність інфраструктури, точність аналізу, підтримка системи та ПЗ	На даний час відсутні
Висновки:	Конкуренції немає, монополія	Можливості входу на ринок є, також можлива поява конкурентів	Постачальники відсутні, на ринок не впливають	Клієнтам необхідна якісна, надійна система	Відсутні обмеження на ринку через товари замінники, але в

Продовження таблиці 9.9

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Немає	Немає бар'єрів	Немає необхідності в постачальниках	Надійність інфраструктури, точність аналізу, підтримка системи та ПЗ	На даний час відсутні
		(мала кількість спеціалістів, швидкість появи невисока)			майбутньому можливі загрози

На основі аналізу конкуренції, наведеного в таблиці 9.9, а також із урахуванням характеристик ідеї проекту (табл. 9.2), вимог споживачів до товару (табл. 9.5) та факторів маркетингового середовища (таблиці 9.6 та 9.7), визначено та обґрунтовано перелік факторів конкурентоспроможності, що наведений у таблиці 9.10.

Таблиця 9.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Підтримка інфраструктури та	Немає необхідності узгоджувати технічне завдання з різними підприємствами

Продовження таблиці 9.10

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
	програмного забезпечення	
2	Забезпечення надійності	Впровадження надійності системи, збитковості даних
3	Масштабованість	Швидка інтеграція технічних засобів в існуючу інфраструктуру

За визначеними факторами конкурентоспроможності (табл. 9.10) проведено аналіз сильних та слабких сторін стартап-проекту (табл. 9.11).

Таблиця 9.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з потенційними конкурентами						
			–3	–2	–1	0	+1	+2	+3
1	Підтримка інфраструктури та програмного забезпечення	18							+
2	Забезпечення надійності	18						+	
3	Масштабованість	20							+

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 9.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 9.11).

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища.

Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 9.12 – SWOT-аналіз стартап-проекту

Сильні сторони: надійність, інтеграція та підтримка програмного забезпечення в системі	Слабкі сторони: деякі витрати повністю або частково переносяться на клієнта
Можливості: повна автоматизація процесів аналізу, інтерфейс для обчислень і виводу результатів	Загрози: поява конкурентів, низька швидкість росту ринку

На основі SWOT-аналізу розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (табл. 9.9).

Визначені альтернативи проаналізовано з точки зору строків та ймовірності отримання ресурсів (табл. 9.13).

Таблиця 9.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Створення команди	90%	3-4 міс.
2	Розроблення базових скриптів автоматизації процесів розгортання	100%	1 міс.
3	Тестування	100%	1 міс.
4	Вихід на ринок	60%	2-4 міс.

9.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 9.14).

Таблиця 9.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Малий бізнес	неповна	5%	Немає	Складно
2	Середній бізнес	так	25%	Немає	середня
3	Великий бізнес	так	40%	Немає	Просто
Які цільові групи обрано: середній/великий бізнес					

За результатами аналізу потенційних груп споживачів (сегментів) обирають цільові групи, для яких пропонуватиметься товар, та визначається стратегію ох оплення ринку:

– якщо компанія зосереджується на одному сегменті – обирається стратегія концентрованого маркетингу;

– якщо працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу – використовується стратегія диференційованого маркетингу;

– якщо компанія працює із всім ринком, пропонуючи стандартизовану програму (включно із характеристиками товару/послуги) – використовується масовий маркетинг.

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (табл. 9.15).

Таблиця 9.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
	Стратегія лідерства по витратах	Стратегія диференційованого маркетингу	Низька кількість спеціалістів разом з низькими витратами створюють бар'єр для входу на ринок	Стратегія лідерства по витратах

Вибір стратегії конкурентної поведінки наведено у таблиці 9.16.

Таблиця 9.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопроходьцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Так	Шукати нових	Ні	Стратегія заняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 9.5), а також в залежності від обраної базової стратегії розвитку (табл. 9.15) та стратегії конкурентної поведінки (табл. 9.16) розроблено стратегію позиціонування (табл. 9.17), що полягає у формуванні ринкової

позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 9.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Підтримка ПЗ	Стратегія диференціації	Задоволення потреб в різних видах аналізу даних	Функціональність
2	Розроблення ПЗ			
3	Розгортання та підтримка працездатності системи	Стратегія лідерства по витратах	Низькі ціни на розгортання системи, швидкість	Низька ціна
4	Оновлення інфраструктури			

9.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 9.18 підсумовано результати попереднього аналізу конкурентоспроможності товару.

Таблиця 9.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Аналіз даних	Інфраструктура обчислень	Низькі інфраструктурні витрати, відкрита система, наявність інтерфейсів доступу
2	Велике сховище файлів, що доступне з будь-якого комп'ютера мережі, де розміщені результати аналізу і т.д.	Хмарне файлове сховище	Є складовою інфраструктури, що розгортається, масштабується до будь-яких розмірів, надійна, можливість налаштування політик безпеки, шифрування даних
3	Структуризація даних	Додаткові модулі у вигляді програмного забезпечення	Можливість переглянути структуру даних одразу, обробка даних наживо, перегляд стратегій структуризації

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту, його фізичні складові, особливості процесу його надання (табл. 9.19).

Таблиця 9.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Система автоматизованого збору, обробки та аналізу великих обсягів даних		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор

Продовження таблиці 9.19

Рівні товару	Сутність та складові		
II. Товар у реальному виконанні	Інфраструктура високопродуктивних обчислень Хмарне файлове сховище Бази структурованих даних		
	Марка: BigDataAnalysis + DAS (Data Analysis System)		
III. Товар із підкріпленням	До продажу: демонстрація роботи		
	Після продажу: підтримка за ліцензією		
За рахунок чого потенційний товар буде захищено від копіювання: ключі активації, ліцензії			

Наступним кроком визначено цінові межі, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари-субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 9.20).

Таблиця 9.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	7,000$ / рік	-	100,000$	2,000$ - 5,000$

Визначено оптимальну системи збуту, в межах якої приймається рішення (табл. 9.21):

– проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);

– вибір та обґрунтування оптимальної глибини каналу збуту;

– вибір та обґрунтування виду посередників.

Таблиця 9.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Оформлення заявок через веб-сайт, придбання ліцензій	Постачальник відсутній	6 систем в місяць	За допомогою веб-сайту компанії

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 9.22).

Таблиця 9.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Недостатньо інформований споживач: частково відоме	Інтернет, e-mail, телефонія	Ринок високо-продуктивних обчислень	Донесення до потенційних клієнтів переваг інструменту	Make your business grow faster!

Продовження таблиці 9.22

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	призначення інструменту				

9.6 Висновки за розділом

На основі проведеного аналізу можна стверджувати, що є можливість ринкової комерціалізації проекту.

Визначені сильні, слабкі та нейтральні характеристики ідеї проекту, проаналізована технологічна здійсненність ідеї проекту і обрана технологія реалізації ідеї проекту: розгортання системи автоматизованими скриптами з контролюванням процесу людиною.

За результатами попередньої характеристики потенційного ринку стартап-проекту, потенційних клієнтів проекту, ступеневого аналізу конкуренції на ринку можна сказати, що попит наявний, а динаміка ринку відображає стійке зростання попиту. Середня норма рентабельності по ринку визначена на рівні 13,1%, але визначені фактори конкуретноспроможності сприятимуть рентабельності стартап-проекту вище середнього по ринку.

Серед обраних потенційних груп клієнтів обрано середній/великий бізнес.

Обрана альтернатива розвитку проекту – стратегія лідерства по витратах; стратегія охоплення ринку – стратегія диференційованого маркетингу; ключові конкурентоспроможні позиції відповідно до обраної альтернативи – низька кількість

спеціалістів разом з низькими витратами створюють бар’єр для входу на ринок; базова стратегія розвитку – стратегія лідерства по витратах.

Отже, подальша імплементація проекту є доцільною.

ВИСНОВОК

В першому розділі даної роботи було визначено атрибути великих даних і проаналізовано необхідність використання технологій BD. В результаті проведеного аналізу можна стверджувати, що BD технології є дуже актуальними в наш час і дослідження цього напрямку є перспективним.

Далі було проведено аналіз базових ІРО для побудови ПС на базі АСКАД. Обрана ІРО Hadoop має наступні переваги перед аналогічними рішеннями: можливість розподілити загальний обсяг даних на певні частини за рахунок використання MapReduce і виконати обчислення над цими частинами; можливість провадити роботу над розподіленими у мережі даними; надає можливість побудувати AIPO з використанням звичайних комп'ютерів (commodity hardware); відносно невисока вартість практичної реалізації АСКАД для ПС обробки даних для однакової виробничої здібності у порівнянні з іншими ІРО.

Розроблено діаграми варіантів використання системи ПС на основі АСКАД. Визначені два актора ПС: оператор та адміністратор. Для оператора і адміністратора системи визначено контексти їх взаємодії із ПС на основі АСКАД, що описані в окремих діаграмах варіантів використання.

Розроблена структура ПС на основі АСКАД повністю задовольняє висунутим вимогам до такого роду ПС. В процесі розроблення було виділено три основних компоненти, на базі яких розгортається ПС: ІРО, підсистема збору та обробки даних, СКБД. Обрана структура має наступні переваги при подальшій роботі з ПС: кожен структурний елемент має чіткий функціонал в загальній ПС, що закладає принцип єдиної відповідальності; поділ системи на підсистеми дозволяє більш ефективно відлагоджувати окремі підсистеми; кожна з підсистем може бути замінена на аналогічну функціональну одиницю за умов реалізації необхідних інтерфейсів зв'язку між ними.

При розробці АСКАД було враховано дослідження та розробки попередніх розділів. Отримана АСКАД задовольняє поставленим вимогам щодо функціональності ПС.

Для тестування архітектури були розроблені тестові модулі для збору, обробки та аналізу новин з відкритих джерел. В ході практичного тестування були отримано позитивні результати, що підтверджує працездатність побудованої моделі ПС на основі АСКАД.

За співвідношенням вартості до ефективності розроблена ПС має переваги перед іншими аналогічними комерційними рішеннями.

На основі проведеного аналізу останнього розділу даної роботи можна стверджувати, що є можливість ринкової комерціалізації проекту. Визначені сильні, слабкі та нейтральні характеристики ідеї проекту, проаналізована технологічна здійсненність ідеї проекту. За результатами попередньої характеристики потенційного ринку даного проекту, потенційних клієнтів проекту, ступеневого аналізу конкуренції на ринку можна сказати, що попит наявний, а динаміка ринку відображає стійке зростання попиту. Отже, подальша імплементація проекту є доцільною.

ПЕРЕЛІК ПОСИЛАНЬ

1. Viktor Mayer-Schönberger. Big Data: A Revolution that Will Transform how We Live, Work, and Think / Viktor Mayer-Schönberger, Kenneth Cukier. – New York: Houghton Mifflin Harcourt, 2013. – 242 с.

2. Shubham Sharma. Big Data Landscape / Shubham Sharma. // International Journal of Scientific and Research Publication. – 2013. – №6.

3. Big data sends cybersecurity back to the future [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://community.mis.temple.edu/mis520817/2017/03/26/big-data-sends-cybersecurity-back-to-the-futur/ (дата звернення 03.03.2018) – Назва з екрана.

4. Stonebraker M. Object-relational DBMSs, tracking the next great wave / M. Stonebraker, P. Brown, D. Moore. – San Francisco, California: Morgan Kauffman Publishers, Inc, 1998. – 216 с.

5. Hadoop's adolescence: an analysis of Hadoop usage in scientific workloads / K.Ren, Y. Kwon, M. Balazinska, B. Howe. // VLDB Endowment. – 2013. – №6. – С. 853–864.

6. Sharda R. Business Intelligence: A Managerial Perspective on Analytics / R. Sharda, D. Delen, E. Turban. – New Jersey: Prentice Hall Press Upper Saddle River, 2013. – 416 с.

7. November 2017 | TOP500 Supercomputer Sites [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://www.top500.org/lists/2017/11/ (дата звернення 04.03.2018) – Назва з екрана.

8. Managing data transfers in computer clusters with orchestra / [M. Chowdhury, M. Zaharia, J. Ma та ін.]. // SIGCOMM '11 Proceedings of the ACM SIGCOMM 2011 conference. – 2011. – №41 (4). – С. 98–109.

9. White T. Hadoop: The Definite Guide / Tom White. – Boston: O'Reilly Media, Inc., 2012. – 657 с.

10. Zikopoulos P. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data / P. Zikopoulos, C. Eaton. – Pennsylvania: McGraw-Hill Osborne Media, 2011. – 176 с.

11. Dhruba Borthakur. HDFS: Facebook has the world's largest Hadoop cluster! [Електронний ресурс] : [Веб-сайт]. / Dhruba Borthakur. – Електронні дані. – 2010. – Режим доступу до ресурсу: http://hadoopblog.blogspot.com/2010/05/facebook-has-worlds-largest-hadoop.html (дата звернення 04.03.2018) – Назва з екрана.

12. Anil Madan. Hadoop - The Power of the Elephant [Електронний ресурс] : [Веб-сайт]. / Anil Madan. – Електронні дані. – 2010. – Режим доступу до ресурсу: https://www.ebayinc.com/stories/blogs/tech/hadoop-the-power-of-the-elephant/ (дата звернення 08.03.2018) – Назва з екрана.

13. The Hadoop Distributed File System / K.Shvachko, H. Kuang, S. Radia, R. Chansler. // Mass Storage Systems and Technologies (MSST). – 2010. – №26.

14. Apache Hadoop YARN: yet another resource negotiator / [V. Vavilapalli, A. Murthy, C. Douglas та ін.]. // SOCC '13 Symposium on Cloud Computing. – 2013. – №4.

15. Hadoop Quiz: Hadoop Introduction 2 | Hadoop Developer Self Learning [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://hadoopquiz.blogspot.com/2017/06/hadoop-introduction-2-hadoop-developer.html (дата звернення 08.03.2018) – Назва з екрана.

16. Christophe Poulain. Dryad [Електронний ресурс] : [Веб-сайт]. / Christophe Poulain. – Електронні дані. – Режим доступу до ресурсу: https://www.microsoft.com/en-us/research/project/dryad/ (дата звернення 08.03.2018) – Назва з екрана.

17. { Machine Learning, Cloud Computing Big Data }: Dryad vs Hadoop [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://www.codeinstinct.pro/2013/05/dryad-vs-hadoop.html (дата звернення 15.03.2018) – Назва з екрана.

18. Dryad: distributed data-parallel programs from sequential building blocks / [M. Isard, M. Budiu, Y. Yu та ін.]. // EuroSys '07 Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems. – 2007. – №41. – С. 59–72.

19. Mikhail M. Separate Testing Inputs vs. Linear Programming Relaxation / M. Mikhail, G. Paul, S. Hanai. // INFORMATION THEORY AND INFORMATION PROCESSING. – 2015. – №15. – С. 351–376.

20. Pacheco P. Parallel Programming with MPI / Peter Pacheco. – Burlington: Morgan Kauffman Publishers, Inc, 1997. – 418 с.

21. A high-performance, portable implementation of the MPI message passing interface standard / W.Gropp, E. Lusk, N. Doss, A. Skjellum. // Parallel Computing. – 1996. – №22. – С. 789–828.

22. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation / [E. Gabriel, G. Fagg, G. Bosilca та ін.]. // EuroPVM/MPI 2004: Recent Advances in Parallel Virtual Machine and Message Passing Interface. – 2004. – №11. – С. 97–104.

23. MPI The Complete Reference / [M. Snir, S. Otto, S. Huss-Lederman та ін.]. – Cambridge: The MIT Press, 1996. – 336 с.

24. Gantz J. THE DIGITAL UNIVERSE IN 2020: BIG DATA, Bigger Digital Shadow s, and Biggest Grow th in the Far East [Електронний ресурс] : [Веб-сайт] / J. Gantz, D. Reinsel. – Електронні дані. – 2012. – Режим доступу до ресурсу: https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf (дата звернення 18.03.2018) – Назва з екрана.

25. Fan J. Challenges of Big Data analysis / J. Fan, F. Han, H. Liu. // National Science Review. – 2014. – №1 (2). – С. 293–314.

26. Madden S. From Databases to Big Data / Sam Madden. // IEEE Internet Computing. – 2012. – №16 (3). – С. 4–6.

27. Hardware requirments – SwiftStack Documentation [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://swiftstack.com/docs/admin/hardware.html (дата звернення 20.03.2018) – Назва з екрана.

28. How-to: Select the Right Hardware for Your New Hadoop Cluster [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу:

https://blog.cloudera.com/blog/2013/08/how-to-select-the-right-hardware-for-your-new-hadoop-cluster/ (дата звернення 20.03.2018) – Назва з екрана.

29. Гради Буч Джеймс Рамбо, Ивар Якобсон Язык UML. Руководство пользователя М.: – ДМК Пресс, 2007. – 496 с.

30. RFC 793: TRANSMISSION CONTROL PROTOCOL, 1981. – 85 с. – (Information Sciences Institute, University of Southern California).

31. Singh T. Distributions of Linux and its Comprehensive Comparison with Windows Vasishath Kaushal / T. Singh, A. Kumar. // IJCST. – 2012. – №3.

32. Murdock I. Overview of the Debian GNU/Linux System / Ian Murdock. // Linux Journal. – 1994. – №6.

33. Who's using Debian? [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.debian.org/users/#com (дата звернення 25.03.2018) – Назва з екрана.

34. Negus C. CentOS Bible / C. Negus, T. Bonorczyk. – New Jersey: Wiley Publishing, 2009. – 984 с.

35. Deploying OpenStack on CentOS using the KVM Hypervisor and GlusterFS distributed file system [Електронний ресурс] / A.Beloglazov, S. Piraghaj, M. Alrokayan, R. Buyya // University of Melbourne. – 2012. – Режим доступу до ресурсу: http://www.cloudbus.org/reports/OpenStack-CentOS-KVM-glusterfs-guide-Aug2012.pdf (дата звернення 25.03.2018) – Назва з екрана.

36. Companies using CentOS [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://idatalabs.com/tech/products/centos (дата звернення 02.04.2018) – Назва з екрана.

37. Our mission | Ubuntu [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.ubuntu.com/community/mission (дата звернення 02.04.2018) – Назва з екрана.

38. The standard OS for cloud computing [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.ubuntu.com/cloud (дата звернення 02.04.2018) – Назва з екрана.

39. Buy Red Hat Enterprise Linux [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.redhat.com/en/store/linux-platforms (дата звернення 02.04.2018) – Назва з екрана.

40. Our customers [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.redhat.com/en/customers (дата звернення 02.04.2018) – Назва з екрана.

41. Current Price Lists for Partners and Employees [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.suse.com/licensing/price/ (дата звернення 02.04.2018) – Назва з екрана.

42. Companies using SUSE Linux Enterprise Server [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://idatalabs.com/tech/products/suse-linux-enterprise-server (дата звернення 02.04.2018) – Назва з екрана.

43. Usage statistics and market share of Linux for websites [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://w3techs.com/technologies/details/os-linux/all/all?rel=outbound (дата звернення 02.04.2018) – Назва з екрана.

44. Hardware and Software for Hadoop [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://hadoopilluminated.com/hadoop_illuminated/Hardware_Software.html (дата звернення 05.04.2018) – Назва з екрана.

45. Розробка архітектури системи автоматизованого збору, обробки та аналізу даних на основі технології Big Data : матеріали V міжнар. наук.-практ. конф. з інформаційних систем та технологій, 1-2 грудня 2017 р., Київ / відп. ред. А. В. Писаренко. – К.: Вид-во ТОВ «Інжиніринг», 2017. – 76 с.

46. Coppa E. Hadoop Architecture Overview [Електронний ресурс] : [Веб-сайт]. / Emilio Coppa. – Електронні дані. – Режим доступу до ресурсу: http://ercoppa.github.io/HadoopInternals/HadoopArchitectureOverview.html (дата звернення 10.04.2018) – Назва з екрана.

47. Conceptual Overview of Map-Reduce and Hadoop [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://www.glennklockwood.com/data-intensive/hadoop/overview.html (дата звернення 10.04.2018) – Назва з екрана.

48. HDFS Architecture Guide [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (дата звернення 10.04.2018) – Назва з екрана.

49. Welcome to Apache Flume – Apache Flume [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://flume.apache.org/ (дата звернення 10.04.2018) – Назва з екрана.

50. Arvind Prabhakar. Apache Flume – Architecture of Flume NG [Електронний ресурс] : [Веб-сайт]. / Arvind Prabhakar. – Електронні дані. – 2011. – Режим доступу до ресурсу: http://blog.cloudera.com/blog/2011/12/apache-flume-architecture-of-flume-ng-2/ (дата звернення 15.04.2018) – Назва з екрана.

51. Apache Flume – Hortonworks [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://hortonworks.com/apache/flume/ (дата звернення 15.04.2018) – Назва з екрана.

52. HBase Architecture Analysis Part1(Logical Architecture) [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://www.cyanny.com/2014/03/13/hbase-architecture-analysis-part1-logical-architecture/ (дата звернення 15.04.2018) – Назва з екрана.

53. Leberknight S. HBase architecture [Електронний ресурс] : [Веб-сайт]. / Leberknight. – Електронні дані. – 2013. – Режим доступу до ресурсу: http://www.sleberknight.com/blog/sleberkn/resource/hbase-architecture.png (дата звернення 16.04.2018) – Назва з екрана.

54. Apache HBase Reference Guide [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://hbase.apache.org/book.html (дата звернення 17.04.2018) – Назва з екрана.

55. Рижко Б. Розроблення системи автоматизованого збору, оброблення та аналізу даних на основі технологій Big Data [Текст] / Б. Рижко, П. Катін // Інфокомунікаційні системи та технології. – 2018. – № 2(2). – С. 37-41.

56. adduser(8) - Linux man page [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://linux.die.net/man/8/adduser (дата звернення 20.04.2018) – Назва з екрана.

57. Apache Hadoop 2.5.2 - Hadoop Map Reduce Next Generation-2.5.2 - Cluster Setup [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: http://hadoop.apache.org/docs/r2.5.2/hadoop-project-dist/hadoop-common/ClusterSetup.html (дата звернення 24.04.2018) – Назва з екрана.

58. Flume 1.6.0 User Guide – Apache Flume [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://flume.apache.org/FlumeUserGuide.html (дата звернення 24.04.2018) – Назва з екрана.

Додаток А. Результати розробки архітектури системи комплексного аналізу

Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
debian-master (192.168.1.104:50010)	2	In Service	9.17 GB	12.05 MB	5.56 GB	3.59 GB	84	12.05 MB (0.13%)	0	2.5.1
debian-slave1 (192.168.1.105:50010)	0	In Service	9.17 GB	5.79 MB	4.45 GB	4.72 GB	45	5.79 MB (0.06%)	0	2.5.1
debian-slave2 (192.168.1.103:50010)	1	In Service	9.17 GB	2.74 MB	4.19 GB	4.97 GB	38	2.74 MB (0.03%)	0	2.5.1

Рисунок А.1 – Інформація про вузли IPO

Region Servers

Base Stats | Memory | Requests | Storefiles | Compactions

ServerName	Start time	Requests Per Second	Num. Regions
debian-master,16020,1465203066742	Mon Feb 03 10:36:06 EEST 2018	0	1
debian-slave1,16020,1465203064016	Mon Feb 03 10:36:04 EEST 2018	0	2
debian-slave2,16020,1465203060875	Mon Feb 03 10:36:00 EEST 2018	0	1
Total:3		0	4

Рисунок А.2 – Інформація про вузли RegionServer HBase

Tables

User Tables | System Tables | Snapshots

2 table(s) in set. [Details]

Namespace	Table Name	Online Regions	Offline Regions	Failed Regions	Split Regions	Other Regions	Description
default	indexed	1	0	0	0	0	'indexed', {NAME => 'I'}
default	news	1	0	0	0	0	'news', {NAME => 'news_data'}

Рисунок А.3 – Інформація про наявні таблиці у СКБД HBase

Додаток Б. Лістинги програм

Лістинг Б.1 – Модуль збору інформації

```
import feedparser;
import time;
import calendar;
from dateutil import parser
import dateutil.tz;
import json;
from datetime import datetime;

from BeautifulSoup import BeautifulSoup;

urls = {"top_news":"http://feeds.reuters.com/reuters/topNews", "health": "http://feeds.reut
ers.com/reuters/healthNews", \
        "healthcare":"http://feeds.reuters.com/reuters/UShealthcareNews", \
        "science":"http://feeds.reuters.com/reuters/scienceNews"}

past = datetime.utcnow();
first_get = True;
while True:
    for k, v in urls.items():
        d = feedparser.parse(v)
        for e in d.entries:
            dt = parser.parse(e.published, parserinfo=None).astimezone(dateutil.tz.tzutc()).repl
ace(tzinfo=None)
            if dt < past and first_get == False:
```

```
            continue
        doc = json.dumps({"category":k,
                    "title":e.title.strip(),
                    "summary":BeautifulSoup(e.summary).text.replace('\n', ' ').replace('*'
,'').strip(),
                    "link":e.link,
                    "published":str(calendar.timegm(dt.utctimetuple()))})
        print doc
    if first_get == True:
        first_get = False
    past = datetime.utcnow();
    time.sleep(120);
```

Лістинг Б.2 – Модуль обробки подій

```
package org.apache.flume.sink.hbase;

import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Set;
import java.util.concurrent.atomic.AtomicInteger;

import org.apache.commons.lang.RandomStringUtils;
import org.apache.flume.Context;
import org.apache.flume.Event;
import org.apache.flume.FlumeException;
import org.apache.flume.conf.ComponentConfiguration;
import org.apache.hadoop.hbase.client.Increment;
```

```
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Row;

import org.json.JSONObject;
/**
 *
 * @author boris
 */
public class JsonHbaseEventSerializer implements HbaseEventSerializer {

    protected static final AtomicInteger nonce = new AtomicInteger(0);
    protected static String randomKey = RandomStringUtils.randomAlphanumeric(10);

    protected byte[] columnFamily;
    private byte[] payload;

    @Override
    public void initialize(Event event, byte[] columnFamily) {

        this.payload = event.getBody();
        this.columnFamily = columnFamily;
    }

    @Override
    public List<Row> getActions() throws FlumeException {

        List<Row> actions = new ArrayList<>();
        JSONObject o = new JSONObject(new String(payload));
        Set<String> keySet = o.keySet();
        byte[] rowKey;
```

```
    try {
      rowKey = getRowKey();
      Put put = new Put(rowKey);
      for(String key: keySet) {
        put.addColumn(columnFamily,
            key.getBytes(StandardCharsets.UTF_8),
            o.getString(key).getBytes(StandardCharsets.UTF_8));
      }
      actions.add(put);
    }
    catch(Exception e) {
      throw new FlumeException("Error!", e);
    }

    return actions;
}

@Override
public List<Increment> getIncrements() {
    return new ArrayList<>();
}

@Override
public void close() {
}

@Override
public void configure(Context context) {
}
```

```
    @Override
    public void configure(ComponentConfiguration conf) {
    }

    protected byte[] getRowKey(Calendar cal) {

        String rowKey = String.format("%s-%s-%s", cal.getTimeInMillis(),
                randomKey,
                nonce.getAndIncrement());

        return rowKey.getBytes(StandardCharsets.UTF_8);
    }

    protected byte[] getRowKey() {
        return getRowKey(Calendar.getInstance());
    }
}
```

Лістинг Б.3 – Модуль аналізу інформації

```
package org.apache.hadoop.hbase.mapreduce;

import java.io.IOException;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
import java.util.regex.Pattern;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
```

```
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class InvertedIndex extends Configured implements Tool {

   /**
    * Internal Mapper to be run by Hadoop.
    */
   public static class Map
         extends Mapper<ImmutableBytesWritable, Result, ImmutableBytesWritable, Put>
{

      public static final String[] STOP_WORDS_ENGLISH = {
         "i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours",
 "yourself", "yourselves", "he", "him", "his", "himself",
         "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "t
hemselves", "what", "which", "who", "whom",
         "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "bein
g", "have", "has", "had", "having", "do", "does",
         "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "w
hile", "of", "at", "by", "for", "with", "about",
         "against", "between", "into", "through", "during", "before", "after", "above", "belo
```

```
w", "to", "from", "up", "down", "in", "out", "on", "off",
        "over", "under", "again", "further", "then", "once", "here", "there", "when", "where
", "why", "how", "all", "any", "both", "each", "few",
        "more", "most", "other", "some", "such", "no", "nor", "not", "only", "own", "same",
 "so", "than", "too", "very", "s", "t", "can", "will",
        "just", "don't", "should", "now", "i'm", "i'll", "you're", "you'll", "it's", "we're", "th
ey're", "can't", "doesn't", "didn't", "oh", "ah",
        "i'd", "couldn't", "wouldn't", "he's", "she's", "i've", "you've", "he'll", "she'll", "w
e'll", "they'll", "it'll", "there's", "here's",
        "won't", "shouldn't", "haven't", "hasn't", "hadn't", "there're", "here're"
    };

    private static final Pattern WORD_PATTERN = Pattern.compile("\\s*\\b\\s*");
    private static final long TIMESTAMP = 555; // just an example

    private byte[] sourceTableCFamily;
    private ImmutableBytesWritable indexedTableName;
    private byte[] indexedTableCFamily;
    private byte[] sourceTableName;
    private byte[] columnToIndex;

    private Set<String> stopWords;


    @Override
    protected void map(ImmutableBytesWritable rowKey,
        Result result,
        Mapper.Context context)
        throws IOException, InterruptedException {
```

```
    byte[] value = result.getValue(sourceTableCFamily, columnToIndex);
    String strValue = Bytes.toString(value).toLowerCase();

    for(String word : WORD_PATTERN.split(strValue)){
        if(word.isEmpty() || stopWords.contains(word))
            continue;

        Put put = new Put(Bytes.toBytes(word));

        try {
            byte[] freq = put.
                    get(indexedTableCFamily, rowKey.get())
                    .get(0).getValueArray();

            int ifreq = Bytes.toInt(freq);
            ifreq++;
            put.addColumn(indexedTableCFamily,
                    rowKey.get(),
                    TIMESTAMP,
                    Bytes.toBytes(ifreq));
        } catch (IndexOutOfBoundsException e) {
            put.addColumn(indexedTableCFamily,
                    rowKey.get(),
                    TIMESTAMP,
                    Bytes.toBytes(1));
        }
        context.write(indexedTableName, put);
    }
}
```

```
    @Override
    protected void setup(Mapper.Context context)
            throws IOException, InterruptedException {
        Configuration configuration = context.getConfiguration();

        String indexedTableName = configuration
                .get("invertedindex.indexedtablename");
        String indexedColumnFamily = configuration
                .get("invertedindex.indexedtable_columnfamily");
        String sourceTableName = configuration
                .get("invertedindex.sourcetablename");
        String sourceColumnFamily = configuration
                .get("invertedindex.sourcetable_columnfamily");
        String columnToIndex = configuration
                .get("invertedindex.column_to_index");

        this.sourceTableName = Bytes.toBytes(sourceTableName);
        this.sourceTableCFamily = Bytes.toBytes(sourceColumnFamily);
        this.indexedTableName = new ImmutableBytesWritable(
                Bytes.toBytes(indexedTableName));
        this.indexedTableCFamily = Bytes.toBytes(indexedColumnFamily);
        this.columnToIndex = Bytes.toBytes(columnToIndex);

        this.stopWords = new HashSet<>(Arrays.asList(STOP_WORDS_ENGLISH));
    }
}

/**
 * Job configuration.
 *
```

```
 * @param conf
 * @param args
 * @return
 * @throws java.io.IOException
 */
public static Job configureJob(Configuration conf, String[] args)
        throws IOException {

    String sourceTableName = args[0];
    String sourceTableColumnFamily = args[1];
    String indexedTableName = args[2];
    String indexedTableColumnFamily = args[3];
    String columnToIndex = args[4];

    System.out.println("============== tablename=" + sourceTableName
            + "; column_family=" + sourceTableColumnFamily
            + "; column_to_index=" + columnToIndex);
    System.out.println("============== indexed_tablename=" + indexedTableName
            + "; indexed_column_family=" + indexedTableColumnFamily);
    conf.set(TableInputFormat.SCAN,
            TableMapReduceUtil.convertScanToString(new Scan()));
    conf.set(TableInputFormat.INPUT_TABLE, sourceTableName);
    conf.set("invertedindex.sourcetablename", sourceTableName);
    conf.set("invertedindex.sourcetable_columnfamily",
            sourceTableColumnFamily);
    conf.set("invertedindex.indexedtablename", indexedTableName);
    conf.set("invertedindex.indexedtable_columnfamily",
            indexedTableColumnFamily);
    conf.set("invertedindex.column_to_index", columnToIndex);
    Job job = Job.getInstance(conf, sourceTableName);
```

```
        job.setJarByClass(InvertedIndex.class);
        job.setMapperClass(Map.class);
        job.setNumReduceTasks(0);
        job.setInputFormatClass(TableInputFormat.class);
        job.setOutputFormatClass(MultiTableOutputFormat.class);
        return job;
    }

    @Override
    public int run(String[] args) throws Exception {
        Configuration conf = HBaseConfiguration.create(getConf());
        if (args.length < 5) {
            System.err.println("Only " + args.length + " arguments supplied, required: 5");
            System.err.println("Usage: IndexBuilder <SOURCE_TB_NAME> <SOURCE_TB
_COLUMN_FAMILY> <INDEXED_TB_NAME> <INDEXED_TB_COLUMN_FAMIL
Y> <COLUMN_TO_INDEX>");
            System.exit(-1);
        }
        Job job = configureJob(conf, args);
        return (job.waitForCompletion(true) ? 0 : 1);
    }

    public static void main(String[] args) throws Exception {
        int result = ToolRunner
                .run(HBaseConfiguration.create(), new InvertedIndex(), args);
        System.exit(result);
    }
}
```

Додаток В. Результати тестування побудованої програмної системи на основі архітектури системи комплексного аналізу

Retired Jobs

Show 20 entries Search:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
2018.02.18 15:21:19 EEST	2018.02.18 15:21:24 EEST	2018.02.18 15:21:32 EEST	job_1463563953474_0004	news	hduser	default	SUCCEEDED	1	1	0	0

Showing 1 to 1 of 1 entries First Previous 1 Next Last

Рисунок В.1 – Інформація про виконані завдання

```
hduser@debian-master: ~
File  Edit  View  Search  Terminal  Help
viewpoints              column=i:1463143642166-oiJmUxNGn8-131, timestamp=555, valu
                        e=\x00\x00\x00\x01
viewpoints              column=i:1463143645405-oiJmUxNGn8-161, timestamp=555, valu
                        e=\x00\x00\x00\x01
violence                column=i:1463143670436-oiJmUxNGn8-190, timestamp=555, valu
                        e=\x00\x00\x00\x01
virus                   column=i:1463143642154-oiJmUxNGn8-105, timestamp=555, valu
                        e=\x00\x00\x00\x01
virus                   column=i:1463143672602-oiJmUxNGn8-196, timestamp=555, valu
                        e=\x00\x00\x00\x01
vital                   column=i:1463143642154-oiJmUxNGn8-104, timestamp=555, valu
                        e=\x00\x00\x00\x01
vocal                   column=i:1463143641656-oiJmUxNGn8-30, timestamp=555, value
                        =\x00\x00\x00\x01
voice                   column=i:1463143645410-oiJmUxNGn8-172, timestamp=555, valu
                        e=\x00\x00\x00\x01
volcanic                column=i:1463143670431-oiJmUxNGn8-178, timestamp=555, valu
                        e=\x00\x00\x00\x01
volcano                 column=i:1463143641664-oiJmUxNGn8-53, timestamp=555, value
                        =\x00\x00\x00\x01
vote                    column=i:1463143641627-oiJmUxNGn8-0, timestamp=555, value=
                        \x00\x00\x00\x01
vote                    column=i:1463143641644-oiJmUxNGn8-3, timestamp=555, value=
                        \x00\x00\x00\x01
vote                    column=i:1463143641647-oiJmUxNGn8-9, timestamp=555, value=
```

Рисунок В.2 – Побудована таблиця інвертованого індексу на основі таблиці новин

Додаток Г. Тези доповіді «Розробка архітектури системи автоматизованого збору, обробки та аналізу даних на основі технології Big Data» на V Міжнародній науково-практичній конференції «Winter InfoCom Advanced Solutions 2017»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНСТИТУТ МОДЕРНІЗАЦІЇ ЗМІСТУ ОСВІТИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

WINTER INFOCOM ADVANCED SOLUTIONS 2017

МАТЕРІАЛИ

V МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

CONFERENCE PROCEEDINGS

5th SCIENTIFIC AND PRACTICAL CONFERENCE

КИЇВ, УКРАЇНА

1-2 ГРУДНЯ 2017 РОКУ

ЗМІСТ/CONTENTS

Розробка архітектури системи автоматизованого збору, обробки та аналізу даних на основі технології Big Data

Рижко Борис Володимирович
КПІ ім. Ігоря Сікорського
Київ, Україна
r.borya@gmail.com

Смолинець Остап Тарасович
КПІ ім. Ігоря Сікорського
Київ, Україна
o.smolynets@ukr.net

Анотація. **Для зменшення витрат на побудову високопродуктивних кластерних систем (ВПКС) пропонується використовувати відкрите програмне забезпечення, що базується на технології Big Data. Описаний метод побудови архітектури систем автоматизованого збору, обробки та аналізу даних (САЗ).**

Ключові слова: обробка даних, великі дані.

Вступ

На сьогоднішній день актуальною стала проблема збору, обробки і аналізу інформації. Прикладом є аналіз продаж і витрат у розподілених системах, управління трафіком через розподілені автоматизовані системи, обробка великої кількості відеоінформації, тощо.

Для рішення цих завдань активно використовують високопродуктивні кластерні системи [1]. Вони побудовані за архітектурою кластерних обчислювальних систем з розподіленою пам'яттю.

Одним із завдань, які потрібно вирішити при проектуванні ВПКС, є формалізація та моделювання програмного забезпечення і роботи паралельних систем. Ці задачі вирішувалися з використанням теорії мереж Петрі [2,3].

Проблемою при розробці ВПКС стає дуже висока вартість запуску і експлуатації подібних систем. Для зменшення витрат пропонується низка відкритих програмних комплексів для реалізації ВПКС.

Завдання даної системи дещо відрізняються від типових ВПКС. Окрім традиційної для ВПКС функції аналізу даних, до САЗ включаються також функції збору та обробки великих наборів даних (наприклад, погодних даних, геолокаційних даних, даних зібраних з «розумних» пристроїв та ін.)

Загальна шкала великих наборів даних постійно змінюється, залежить від часу і може істотно відрізнятися в різних організаціях експлуатантах, що спонукає до розробки такого класу систем.

Таким чином основною метою є розробка архітектури САЗ з використанням відкритих програмних систем.

Для реалізації поставленої мети потрібно:

– сформувати структуру САЗ на основі ВПКС, програмною базою якої має бути відкритий програмний комплекс;

– провести вибір та обгрунтування елементів структури САЗ;

– побудувати програмну і апаратну архітектури САЗ на основі ВПКС.

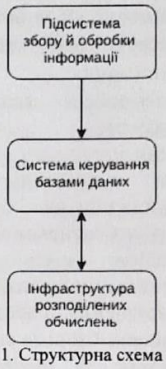


Рис. 1. Структурна схема САЗ

Вимоги до САЗ

САЗ, побудовані з використанням технологій «Big Data» можуть бути використані в будь-яких сферах бізнесу.

Такі системи повинні мати необхідний інструментарій для обробки як структурованих, так і неструктурованих даних дуже великих обсягів.

В заданих умовах необхідно забезпечити можливості для швидкого збору дуже великої кількості інформації, швидкої обробки й аналізу цього масиву даних.

Таким чином, до САЗ можна встановити наступні вимоги:

– масштабованість;

– можливість збирати структуровані та неструктуровані дані;

– швидка обробка великих обсягів даних;

– можливість зберігати дані у кластерній файловій системі;

– відмовостійкість;

– представлення результатів аналізу у вигляді, зручному для подальшого відображення.

Продуктивність інфраструктури розподілених обчислень (ІРО) цілком залежить від технічного та програмного забезпечення.

Структура системи

Виходячи з зазначених вимог до САЗ, було визначено наступні структурні елементи:

інфраструктура розподілених обчислень (ІРО), підсистема збору та обробки інформації, система керування базами даних (СКБД).

Структурна схема САЗ наведена на рис. 1.

Визначені елементи дозволяють розділити функції всієї САЗ, що значно полегшує налагодження системи і закладає принципи модульності. Така структура також визначає взаємозамінність кожної з підсистем у випадку необхідності.

В свою чергу, кожна з підсистем повинна також мати модульний принцип побудови, що є одним з критеріїв їх вибору.

Компоненти системи

Для вибору окремих компонентів було встановлено ряд вимог, що повинні задовольнятися, для реалізації поставленої мети. Таким чином, ІРО повинна відповідати наступним вимогам:

– горизонтальна масштабованість;
– наявність кластерної файлової системи (ФС);
– обробка великих даних;
– наявність інтерфейсів доступу до ФС та обчислювальних модулів;
– відмовостійкість.

З огляду на поставлені критерії, було проаналізовано існуючі рішення і обрано ІРО Hadoop.

Проект складається з чотирьох модулів [4]:

– Hadoop Common (набір інфраструктурних програмних бібліотек і утиліт, використовуваних для інших модулів і споріднених проектів);
– HDFS (розподілена файлова система);
– YARN (система для планування завдань і управління кластером);
– Hadoop MapReduce (платформа програмування і виконання розподілених MapReduce-обчислень).

При реалізації обчислень в парадигмі MapReduce, вони приймають набір вхідних пар ключ-значення і створюють набір вихідних пар ключ-значення. Користувач бібліотеки MapReduce виражає розрахунок як дві функції: Map та Reduce.

Map, написаний користувачем, приймає пару вхідних даних і видає набір проміжних пар ключ-значення. Бібліотека MapReduce об'єднує всі проміжні значення, пов'язані з тим самим проміжним ключем, і передає їх в функцію Reduce.

Функція Reduce, також написана користувачем, приймає проміжний ключ і набір значень для цього ключа. Вона об'єднує разом ці значення, щоб сформувати, якщо можливо, менший набір значень. Зазвичай при виклику Reduce вихідне значення або не повертається, або повертається лише одне. Проміжні значення подаються на функцію Reduce користувача за допомогою ітератора. Це дозволяє обробляти списки значень, які занадто великі для розміщення в пам'яті.

Вибір підсистеми збору та обробки інформації базується на наступних критеріях:

– модульний дизайн;
– масштабованість;
– сумісність з базою даних;
– сумісність з ІРО;
– гнучкі налаштування;
– розподіленість виконання задач;
– багатопоточність;
– відкрите програмне забезпечення.

В якості підсистеми збору та обробки інформації було обрано відкрите програмне забезпечення Apache Flume.

Flume є розподіленим та надійним сервісом збору, обробки та переміщень великих обсягів даних. Він має просту та гнучку архітектуру, що базується на потоках даних [5].

Критерії вибору СКБД наступні:

– масштабованість;
– сумісність з ІРО;
– сумісність з підсистемою збору та обробки інформації;
– можливість використання інтерфейсу доступу до СКБД зовнішніми модулями;
– швидкий доступ до великих обсягів даних;
– надійність;
– відкрите програмне забезпечення.

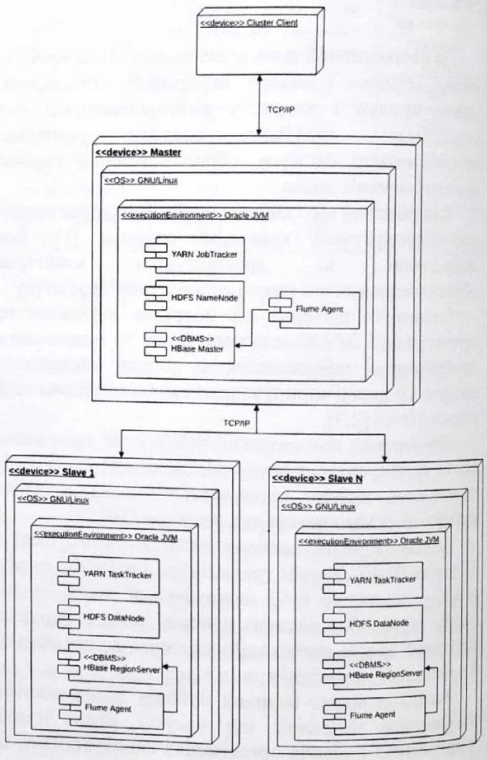


Рис. 2. Архітектура САЗ

Заданим вимогам відповідає обрана СКБД Apache HBase. Це стовпчико-орієнтована NoSQL СКБД. NoSQL означає, що СКБД відмінна від традиційного представлення даних у реляційних СКБД. Дані у HBase зберігаються у вигляді структурованих та напівструктурованих пар ключ-значення. Дана СКБД надає наступні можливості [6]:

– запис-зчитування у режимі реального часу;
– інтерфейс управління СКБД;

– конфігурація розподіленої СКБД між вузлами;

– додаткові інтерфейси для доступу зі сторонніх програм.

Архітектура САЗ

Розглянемо архітектуру САЗ, що побудована на основі структурної схеми, в нотації UML [7] (рис. 2).

В розробленій архітектурі були враховані усі вимоги, що пред'являються структурною схемою та окремим програмним забезпеченням. Отримана архітектура системи відповідає типу master-slave.

Master-вузол керує операціями простору імен файлової системи (такими як відкриття, закриття та перейменування файлів і каталогів), регулює доступ клієнтів до файлів, є точкою взаємодії між користувачами та системою MapReduce; виконує призначення регіону, контролює операції DDL (створення, видалення таблиць) бази даних HBase.

Відповідає за:

– додаткові інтерфейси для доступу зі сторонніх програм;

– координацію серверів регіону;

– призначення регіонів при запуску;

– перепризначення областей для відновлення або балансування навантаження;

– моніторинг усіх примірників RegionServer у кластері (прослуховує сповіщення від Zookeeper).

Крім того, на master-вузлі розміщено приймач даних від усіх агентів (slave-вузлів), що збирають та обробляють інформацію з різних джерел.

Slave-вузли несуть відповідальність за обслуговування запитів на читання та запис клієнтів файлової системи разом із виконанням створення, видалення та реплікації блоків за інструкцією master-вузла; виконують обчислювальні задачі MapReduce, що надаються master-вузлом; виконують задачі збору, первинної обробки інформації, запису до бази даних.

Висновки

В результаті аналізу різних ІРО було виявлено, що на сьогоднішній день, за співвідношенням вартість-ефективність для розробки автоматизованої системи збору, обробки та аналізу інформації доцільно використовувати кластери, побудовані з використанням Hadoop, для загального користування.

Розроблена архітектура і САЗ в цілому забезпечує: масштабованість, модульність та гнучкість у налаштуванні всіх параметрів.

Модульність архітектури дозволяє підключати додаткові модулі до складу робочих програмних систем або окремо від них з використанням програмних інтерфейсів цих систем.

Перевагами системи можна назвати:

– поєднання ресурсів: об'єднання наявного простору для зберігання даних є чіткою перевагою, але процесор та об'єднання пам'яті також надзвичайно важливі; обробка великих наборів даних вимагає великої кількості всіх трьох цих ресурсів;

– висока доступність: кластери можуть забезпечувати різний рівень стійкості до відмов та наявності гарантій, щоб запобігти апаратному або програмному збою від доступу до даних та обробки; це стає все більш важливим, оскільки ми продовжуємо підкреслювати важливість аналітики в реальному часі;

– легка масштабованість: кластери полегшують масштабування горизонтально, додавши до групи додаткові машини. Це означає, що система може реагувати на зміни вимог до ресурсів без розширення фізичних ресурсів на машині.

За співвідношенням вартості до ефективності розроблена система має переваги перед іншими аналогічними комерційними рішеннями.

Література

1. November 2015 | TOP500 Supercomputer Sites [Електронний ресурс] – Режим доступу до ресурсу: http://www.top500.org/lists/2015/11/ (дата звернення 30.05.2017). — Назва з екрана.

2. Катін П. Ю. , Досенко Є. В., Іванів Р. Б. Моделювання програм з паралельними обчисленнями за допомогою мереж Петрі / П. Ю. Катін, Є. В. Досенко, Р. Б. Іванів // Економіка і управління. – 2013. – №4(60). – С.124-132

3. Катін П. Ю. Моделювання процесів синхронізації багато поточних програм з використанням мереж Петрі / П. Ю. Катін // Автоматика-2014: Матеріали 21-ї Міжнародної конференції з автоматичного управління, м. Київ, 23-27 вересня 2014 р. – К.: Вид-во НТУУ "КПІ" ВІПІ ВПК "Політехніка", 2014.–С. 320-321.

4. White T. Hadoop: The Definitive Guide / O'Reilly Media, 2010. – 528 с.

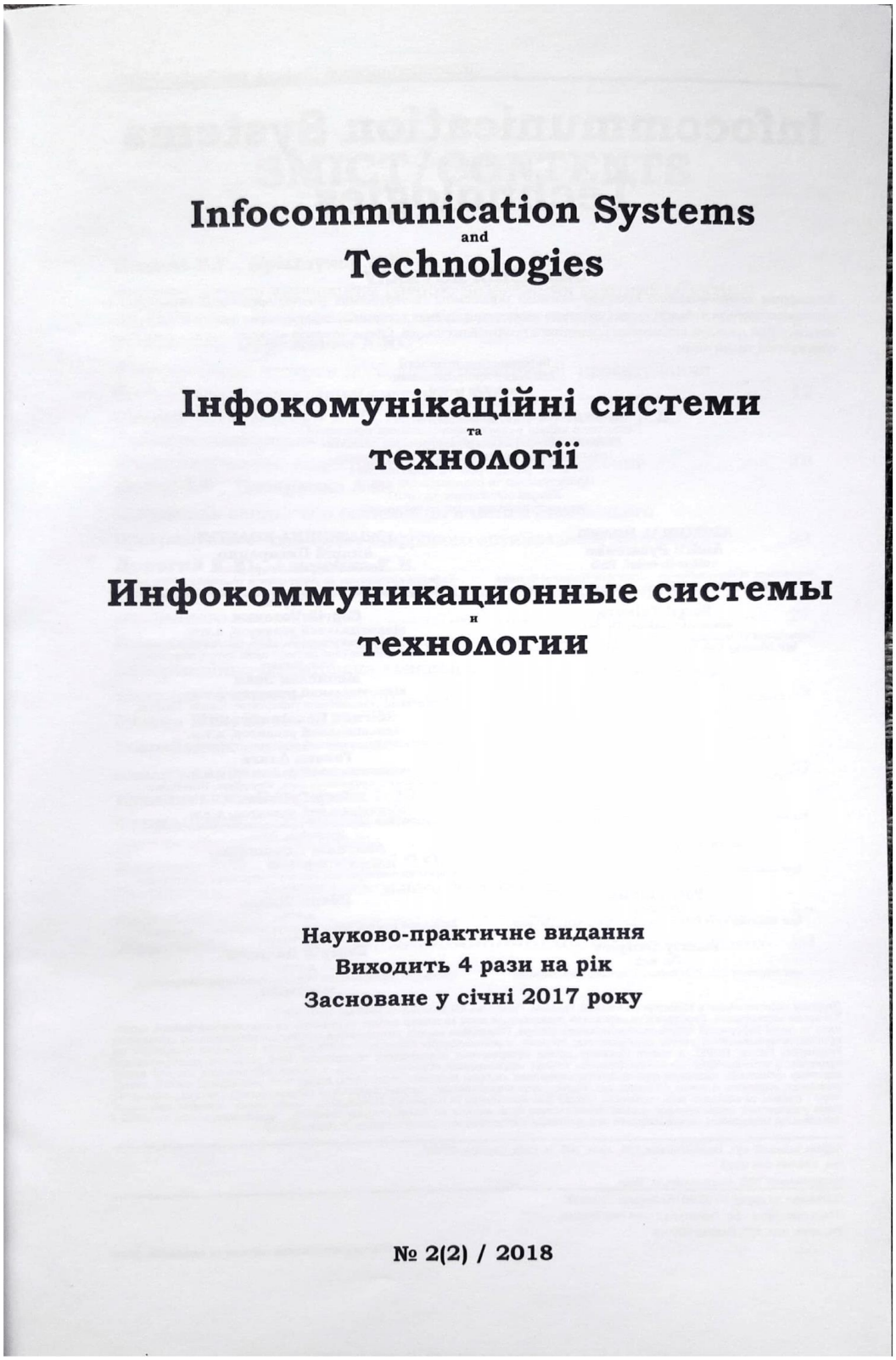
5. Welcome to Apache Flume — Apache Flume [Електронний ресурс] – Режим доступу до ресурсу: https://flume.apache.org/ (дата звернення 30.05.2017). — Назва з екрана.

6. Apache HBase Reference Guide [Електронний ресурс] – Режим доступу до ресурсу: https://hbase.apache.org/book.html (дата звернення 31.05.2017). — Назва з екрану.

7. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. Язык UML. Руководство пользователя. – М., СПб.: ДМК Пресс, Питер, 2004. – 432 с.

Рецензент: к.т.н., доц. каф. АУТС, КПІ ім. Ігоря Сікорського, П.Ю. Катін

Додаток Д. Стаття «Розроблення системи автоматизованого збору, оброблення та аналізу даних на основі технологій Big Data» у журналі Інфокомунікаційні системи та технології № 2(2)



Infocommunication Systems
and
Technologies

Інфокомунікаційні системи
та
технології

Инфокоммуникационные системы
и
технологии

Науково-практичне видання
Виходить 4 рази на рік
Засноване у січні 2017 року

№ 2(2) / 2018

ЗМІСТ/CONTENTS

Розроблення системи автоматизованого збору, оброблення та аналізу даних на основі технологій Big Data

Рижко Борис
Катін Павло
КПІ ім. Ігоря Сікорського
Київ, Україна
r.borya@gmail.com
p.katin@kpi.ua

Анотація. **Для зменшення витрат на побудову високопродуктивних кластерних систем (ВПКС) пропонується використовувати відкрите програмне забезпечення, що базується на технології Big Data. У даній статті описаний метод побудови систем автоматизованого збору, обробки та аналізу даних (САЗ) на основі системного методу. Крім того, показані результати тестування і дослідної експлуатації САЗ на прикладі автоматизованого агрегатору новин.**

Ключові слова: обробка даних, великі дані, агрегатор.

Вступ

На сьогоднішній день актуальною стала проблема збору, обробки і аналізу інформації. Прикладом є аналіз продаж і витрат у розподілених системах, управління трафіком через розподілені автоматизовані системи, обробка великої кількості відеоінформації, тощо.

Для рішення цих завдань активно використовують високопродуктивні кластерні системи [1]. Вони побудовані за архітектурою кластерних обчислювальних систем з розподіленою пам'яттю.

Одним із завдань, які потрібно вирішити при проектуванні ВПКС, є формалізація та моделювання програмного забезпечення і роботи паралельних систем. Ці задачі вирішувалися з використанням теорії мереж Петрі [2,3].

Проблемою при розробці ВПКС стає дуже висока вартість запуску і експлуатації подібних систем. Для зменшення витрат пропонується низка відкритих програмних комплексів для реалізації ВПКС.

Завдання даної системи дещо відрізняються від типових ВПКС. Окрім традиційної для ВПКС функції аналізу даних, до САЗ включаються також функції збору та обробки великих наборів даних (наприклад, погодних даних, геолокаційних даних, даних зібраних з «розумних» пристроїв і т.д.).

Загальна шкала великих наборів даних постійно змінюється, залежить від часу і може істотно відрізнятися в різних організаціях експлуатантах, що спонукає до розробки такого класу систем.

Таким чином основною метою є розробка САЗ з використанням відкритих програмних систем.

Для реалізації поставленої мети, а саме розробки САЗ на основі відкритої системи, потрібно:

– сформувати структуру САЗ на основі ВПКС, програмною базою якої відкритий програмний комплекс;

– провести ретельний аналіз та порівняння існуючих рішень ВПКС;

– провести вибір та обгрунтування елементів структури САЗ;

– побудувати програмну і апаратну архітектури САЗ на основі ВПКС;

– провести моделювання системи на основі побудованої архітектури, реалізувати окремі модулі для підтвердження працездатності САЗ на прикладі агрегатору новин;

– розробити базу даних і адаптувати її для рішення задачі на прикладі агрегатору новин.

Огляд існуючих рішень

Комп'ютерний кластер складається з множини зв'язаних між собою комп'ютерів, що працюють в такому режимі, що їх можна розглядати як єдину систему. Комп'ютерні кластери з'явилися як результат наявності дешевих процесорів, швидкісних мереж та відповідного програмного забезпечення для розподілених обчислень.

Вони мають широке коло використання: від невеликих кластерів для малого бізнесу до найпродуктивніших суперкомп'ютерів у світі.

Можна виділити наступні типи комп'ютерних кластерів [4]:

– високої доступності;

– обчислювальні;

– балансуючі навантаження;

– сховища.

Для вирішення поставленої задачі комп'ютерний кластер повинен поєднувати властивості обчислювальних кластерів та кластерів-сховищ.

За даними критеріями можна виділити наступні види інфраструктури розподілених обчислень:

– Hadoop;

– Dryad;

– Message Passing Interface.

Hadoop - проект фонду Apache Software Foundation, вільно розповсюджуваний набір утиліт, бібліотек і фреймворк для розробки і виконання розподілених програм, що працюють на кластерах з сотень і тисяч вузлів. Використовується для реалізації пошукових

механізмів багатьох високонавантажених веб-сайтів, в тому числі, для Yahoo! і Facebook. Розроблено на Java в рамках обчислювальної парадигми MapReduce, згідно з якою додаток поділяється на велику кількість однакових елементарних завдань, що виконуються на вузлах кластера.

Функціонуючі кластери розміром в тисячі вузлів підтверджують дієздатність і економічну ефективність таких систем. Так, станом на 2011 рік, відомо про великі кластери Hadoop в Yahoo (більше 4 тис. вузлів з сумарним об'ємом для зберігання 15 Пбайт), Facebook (близько 2 тис. вузлів на 21 Пбайт) і Ebay (700 вузлів на 16 Пбайт). Проте, вважається, що горизонтальна масштабованість в Hadoop-системах обмежена. Для Hadoop до версії 2.0 максимально можлива кількість вузлів оцінювалася в 4 тис. при використанні 10 MapReduce завдань на вузол. Багато в чому даному обмеженню сприяла концентрація в модулі MapReduce функцій з контролю за життєвим циклом завдань. Вважається, що з виносом її в модуль YARN в Hadoop 2.0 і децентралізацією – розподілом частини функцій з моніторингу на вузли обробки – горизонтальна масштабовність підвищилася.

Масштабовність Hadoop-систем в значній мірі залежить від характеристик оброблюваних даних. Перш за все, залежить від їх внутрішньої структури, особливостей вилучення необхідної інформації і складності обробки. Це, в свою чергу, диктує організацію циклів обробки, обчислювальну інтенсивність атомарних операцій, і, в кінцевому рахунку, рівень паралелізму і завантаженість кластера. У конфігураціях Hadoop (перших версіях, раніше 2.0) вказувалося, що прийнятним рівнем паралелізму є використання 10-100 примірників базових оброблювачів на вузол кластера, а для задач, які не потребують значних витрат процесорного часу — до 300; для згорток вважалося оптимальним використання їх за кількістю вузлів, помноженому на коефіцієнт з діапазону від 0,95 до 1,75 і константу mapred.tasktracker.reduce.tasks.maximum. З великим значенням коефіцієнта найбільш швидкі вузли, закінчивши перший раунд обробки, раніше отримають другу порцію проміжних пар для обробки, таким чином, збільшення коефіцієнта надлишково завантажує кластер, але при цьому забезпечує більш ефективне балансування навантаження. У YARN, натомість, використовуються конфігураційні константи, що визначають значення доступної оперативної пам'яті і віртуальних процесорних ядер, доступних для планувальника ресурсів, на підставі яких і визначається рівень паралелізму [5].

Dryad – програмний фреймворк загального призначення для розподілених обчислень. Dryad є проектом підрозділу Microsoft Research. Центральної концепцією проекту Dryad є побудова орієнтованого ациклічного графу (directed acyclic graph, DAG) для кожного розподіленого завдання. Вершини графа представляють собою операції над даними (по суті, програми), а ребра графа — канали, по яких дані передаються [6].

Абстракція на основі моделі орієнтованого ациклічного графу дає можливість ефективно реалізовувати плани виконання великої кількості паралельних алгоритмів, ітеративних алгоритмів, алгоритмів машинного навчання. Таким чином, єдина (до появи YARN) програмна модель MapReduce, що реалізована в Hadoop, по суті є лише окремим випадком моделі розподілених обчислень, яку надає Dryad.

Проект Dryad складається з 3-ох ключових компонентів:

– Dryad – середовище виконання розподілених обчислень (Dryad Runtime);

– DryadLINQ – високорівнева мова запитів, що базується на програмній моделі .NET Language Integrated Query (LINQ);

– Distributed Storage Catalog (DSC) – розподілена файлова система з надмірністю.

Dryad Runtime виконує наступні функції:

– планування і управління розподіленими завданнями;

– управління ресурсами;

– забезпечення відмовостійкості;

– моніторинг.

Завдання в Dryad Runtime представляє собою орієнтований ациклічний граф, де вершини представляють собою програми, а ребра графа – канали даних. Цей логічний граф розділяється (mapped) середовищем виконання на фізичні ресурси, що знаходяться в кластері. Загалом, кількість вершин в графі перевищує кількість фізичних обчислювальних вузлів в кластері [7].

Message Passing Interface (MPI, інтерфейс передачі повідомлень) – програмний інтерфейс (API) для передачі інформації, який дозволяє обмінюватися повідомленнями між процесами, які виконують одну задачу.

В першу чергу MPI орієнтований на системи з розподіленою пам'яттю, тобто коли витрати на передачу даних великі, в той час як OpenMP орієнтований на системи із загальною пам'яттю (багатоядерні із загальним кешем). Обидві технології можуть використовуватися спільно, щоб оптимально використовувати в кластері багатоядерні системи[8].

Таким чином, виходячи з вищесказаного, можна сказати, що кластери з використанням Hadoop швидко розвиваються і легко масштабуються, орієнтовані на комп'ютери загального призначення.

Розробку програмного фреймворку Dryad припинено на користь Hadoop, а MPI орієнтований на системи з масово-паралельною архітектурою із розподіленою пам'яттю.

Отже, в якості кластерної системи було обрано Hadoop.

Вимоги до САЗ

САЗ, побудовані з використанням технологій «Big Data» можуть бути використані в будь-яких сферах бізнесу.

Такі системи повинні мати необхідний інструментарій для обробки як структурованих, так і неструктурованих даних дуже великих обсягів.

В заданих умовах необхідно забезпечити можливості для швидкого збору дуже великої

кількості інформації, швидкої обробки й аналізу цього масиву даних.

Таким чином, до системи можна встановити наступні вимоги:

– масштабованість;

– можливість збирати структуровані та неструктуровані дані;

– швидка обробка великих обсягів даних;

– можливість зберігати дані у кластерній файловій системі;

– відмовостійкість;

– представлення результатів аналізу у вигляді, зручному для подальшого відображення.

Продуктивність інфраструктури розподілених обчислень (ІРО) цілком залежить від технічного та програмного забезпечення.

Сценарії використання

Розглянемо систему у термінах і підходах UML[9].

У системі передбачено два типи користувачів: оператор та адміністратор системи.

Адміністратор системи повинен мати права змінювати конфігурації підсистем, а також можливості конфігурації бази даних (рис. 1).

Оскільки САЗ є кластерною, то, за необхідності збільшення її продуктивності, можна підключити додаткові вузли. Оскільки кожна з підсистем є масштабованою, то одним з необхідних прав адміністратора є додавання нових вузлів у конфігурації. Таким чином, запущена підсистема має змогу підключити усі наявні в системі вузли для максимальної швидкодії та збільшення обсягів файлової системи.

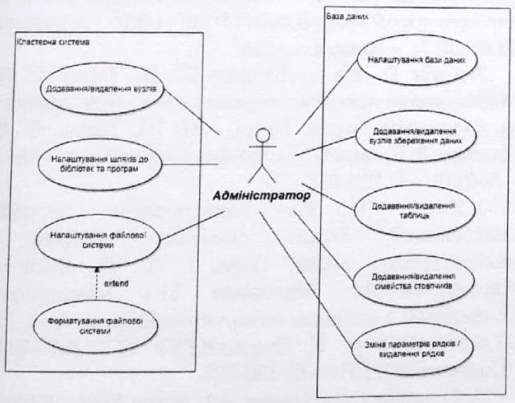


Рис. 1. Діаграма прецедентів адміністратора

Можливість налаштування шляхів до бібліотек та програм дозволяє адміністратору відокремити бібліотеки та виконавчі файли в окремих папках. Це призводить до зменшення розмірів програм (деякі програми можуть використовувати однакові бібліотеки, але кожна з них буде включати дану бібліотеку на етапі компіляції) та спрощує запуск і відлагодження системи.

Таким чином, адміністратор має виконувати наступні завдання у процесі обслуговування системи та її налаштування:

– налаштування мережі кластерної системи;

– налаштування часу і серверу синхронізації;

– підтримка працездатності сервісів.

Перед запуском ІРО є необхідність виконати операцію створення нової файлової системи з ідентифікатором, що однаковий на всіх вузлах. Це означає, що створена кластерна файлова система і вона використовує все вільне наявне місце на кожному з вузлів для власних потреб. Після даної операції можна зберігати, видаляти файли або папки у файловій системі або завантажувати файли або папки з неї.

Оператор системи повинен мати можливість конфігурувати підсистему збору та обробки інформації, таким чином він має змогу визначити наступні складові конфігурації:

– джерела інформації;

– канали передачі інформації;

– сховища даних.

Для кожного з вузлів системи можна визначити конфігурацію і розподілити завдання збору й обробки інформації.

Для сховища даних, як правило, використовується розподілена база даних, що розгорнута поверх ІРО. Тому окрім визначення параметрів модулів-агентів збору й обробки інформації, він повинен мати змогу додавати, змінювати або видаляти таблиці з бази даних, що використовуються підсистемами. Він також може визначати сімейство стовпчиків. За необхідності оператор може змінювати окремі рядки в базі даних,

Оператор також має змогу запускати модулі для аналізу зібраних даних окремо або встановлення розкладу запусків модулів.

Надані оператору функції можна побачити на діаграмі прецедентів, що зображена на рис. 2.

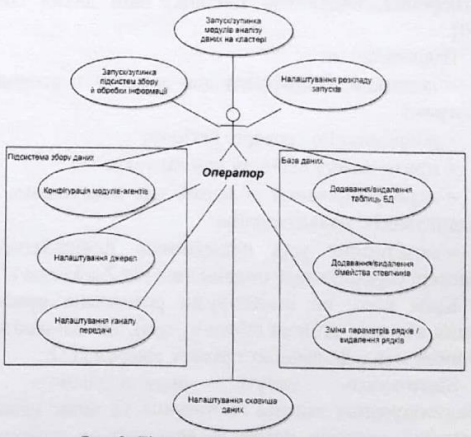


Рис. 2. Діаграма прецедентів оператора

Архітектура САЗ

Побудована архітектура САЗ на основі структурної схеми зображена на рис. 3.

В розробленій архітектурі були враховані усі вимоги, що пред'являються структурною схемою та окремим програмним забезпеченням. Отримана архітектура системи відповідає типу master-slave.

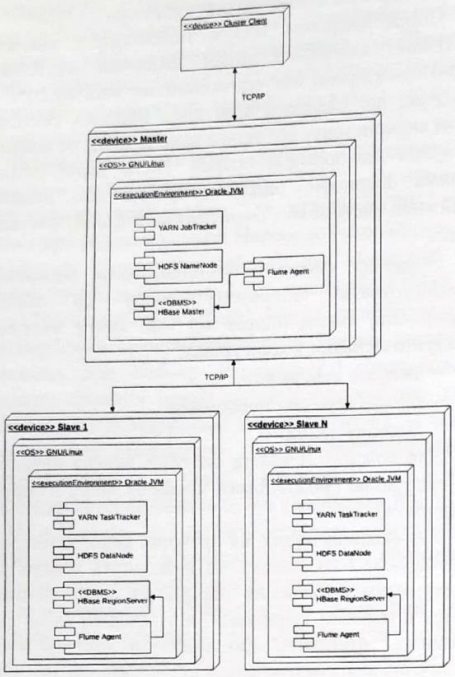


Рис. 3. Архітектура САЗ

Master-вузол керує операціями простору імен файлової системи (такими як відкриття, закриття та перейменування файлів і каталогів), регулює доступ клієнтів до файлів, є точкою взаємодії між користувачами та системою MapReduce; виконує призначення регіону, контролює операції DDL (створення, видалення таблиць) бази даних HBase [10].

Відповідає за:

– додаткові інтерфейси для доступу зі сторонніх програм;

– координацію серверів регіону;

– призначення регіонів при запуску;

– перепризначення областей для відновлення або балансування навантаження;

– моніторинг усіх примірників RegionServer у кластері (прослуховує сповіщення від Zookeeper).

Крім того, на master-вузлі розміщено приймач даних від усіх агентів (slave-вузлів), що збирають та обробляють інформацію з різних джерел [11].

Slave-вузли несуть відповідальність за обслуговування запитів на читання та запис клієнтів файлової системи разом із виконанням створення, видалення та реплікації блоків за інструкцією master-вузла; виконують обчислювальні задачі MapReduce, що надаються master-вузлом; виконують задачі збору, первинної обробки інформації, запису до бази даних.

Висновки

В результаті аналізу різних ІРО було виявлено, що на сьогоднішній день, за співвідношенням вартість-ефективність для розробки автоматизованої системи збору, обробки та аналізу інформації доцільно використовувати кластери, побудовані з використанням Hadoop, для загального користування.

Розроблена САЗ забезпечує: масштабованість, модульність та гнучкість у налаштуванні всіх параметрів. Перевагами системи можна назвати:

– поєднання ресурсів: об'єднання наявного простору для зберігання даних є чіткою перевагою, але процесор та об'єднання пам'яті також надзвичайно важливі; обробка великих наборів даних вимагає великої кількості всіх трьох цих ресурсів;

– висока доступність: кластери можуть забезпечувати різний рівень стійкості до відмов та наявності гарантій, щоб запобігти апаратному або програмному збою від доступу до даних та обробки; це стає все більш важливим, оскільки ми продовжуємо підкреслювати важливість аналітики в реальному часі;

– легка масштабованість: кластери полегшують масштабування горизонтально, додавши до групи додаткові машини. Це означає, що система може реагувати на зміни вимог до ресурсів без розширення фізичних ресурсів на машині.

За співвідношенням вартості до ефективності розроблена система має переваги перед іншими аналогічними комерційними рішеннями.

Для тестування архітектури САЗ були розроблені тестові модулі для збору, обробки та аналізу новин з відкритих джерел. В ході практичного тестування були отримані позитивні результати, висока надійність і, відповідно, продуктивність системи.

Література

1. November 2015 | TOP500 Supercomputer Sites [Електронний ресурс] – Режим доступу до ресурсу: http://www.top500.org/lists/2015/11/ (дата звернення 30.05.2017). – Назва з екрана.

2. Катін П. Ю. , Досенко Є. В., Іванів Р. Б. Моделювання програм з паралельними обчисленнями за допомогою мереж Петрі / П. Ю. Катін, Є. В. Досенко, Р. Б. Іванів // Економіка і управління. – 2013. – №4(60). – С.124-132

3. Катін П. Ю. Моделювання процесів синхронізації багато поточних програм з використанням мереж Петрі / П. Ю. Катін // Автоматика-2014: Матеріали 21-ї Міжнародної конференції з автоматичного управління, м. Київ, 23-27 вересня 2014 р. – К.: Вид-во НТУУ "КПІ" ВІПІ ВІК "Політехніка", 2014.–С. 320-321.

4. Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design) 4th – San Francisco, CA, USA: Morgan Kaufmann Publishers, 2008. – 912 c.

5. White T. Hadoop: The Definitive Guide / O'Reilly Media, 2010. – 528 c.

6. Dryad: distributed data-parallel programs from sequential building blocks / [M. Isard, M. Budiu, Y. Yu та ін.]. // Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007 (EuroSys '07). – 2007. – С. 59–72.

7. Machine Learning, Cloud Computing Big Data: Dryad vs Hadoop [Електронний ресурс] – Режим доступу до ресурсу: http://www.codeinstinct.pro/

2013/05/dryad-vs-hadoop.html (дата звернення 30.05.2017). – Назва з екрана.

8. Gropp W. Using Mpi: Portable Parallel Programming With the Message-Passing Interface / W. Gropp, E. L. Lusk, A. Skjellum. – Mit Press, 1994.

9. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. Язык UML. Руководство пользователя. – М., СПб.: ДМК Пресс, Питер, 2004. – 432 с.

10. Apache HBase Reference Guide [Електронний ресурс] – Режим доступу до ресурсу: https://hbase.apache.org/book.html (дата звернення 31.05.2017). – Назва з екрану.

11. Welcome to Apache Flume – Apache Flume [Електронний ресурс] – Режим доступу до ресурсу: https://flume.apache.org/ (дата звернення 30.05.2017). – Назва з екрана.