МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«СИСТЕМНИЙ АНАЛІЗ ЗАСТОСУВАННЯ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ ДЛЯ АВТОМАТИЗАЦІЇ ПОБУДОВИ DATA PIPELINE У СЕРЕДОВИЩАХ BIG DATA»

Виконав: студент 2 курсу

рівень: магістр

галузь знань 12 «Інформаційні технології»

спеціальність 124 «Системний аналіз»

Кричун Олександр Сергійович

Керівник: к.пед.н., доцент

Лобода Юлія Геннадіївна

Рецензент: к.т.н., доцент

Гура Володимир Ігорович

Одеса – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет **кібербезпеки та інформаційних технологій**
Кафедра **інформаційних технологій**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **124 Системний аналіз**
Назва освітньої програми: **Аналіз та безпека даних**

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій
доцент _____ Н.І. Логінова
«____ »_____ 20__ року

ЗАВДАННЯ
на кваліфікаційну (магістерську) роботу
здобувачу Кричун Олександру Сергійовичу

1. Тема роботи: «Системний аналіз застосування інтелектуальних агентів для автоматизації побудови Data Pipeline у середовищах Big Data»

Керівник роботи: к.пед.н, доцент Лобода Юлія Геннадіївна
затверджені наказом НУ ОЮА від «*10*» *жовтня 2025* року № *3183-06*

2. Строк подання здобувачем роботи «*25*» *листопада 2025 рік*

3. Дата видачі завдання «*02*» *червня 2025 рік*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми, вивчення літературних джерел та складання плану роботи	09.09.2025	
2.	Підготовка першого розділу роботи та подання його керівнику	01.10.2025	
3.	Підготовка другого розділу роботи та подання його керівнику	14.10.2025	
4.	Підготовка третього розділу роботи та подання його керівнику	07.11.2025	
5.	Підготовка вступу та висновків	10.11.2025	
6.	Доопрацювання роботи з урахуванням зауважень керівника	24.11.2025	
7.	Подання електронного варіанту роботи для перевірки на плагіат	25.11.2025	
8.	Допуск роботи до захисту завідувачем кафедри	28.11.2025	
9.	Захист роботи	01.12.2025	

Здобувач _____ Кричун О.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лобода Ю.Г.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Системний аналіз застосування інтелектуальних агентів для автоматизації побудови Data Pipeline у середовищах Big Data» на здобуття другого (магістерського) рівня вищої освіти за спеціальністю 124 Системний аналіз, освітньою програмою: Аналіз та безпека даних, містить 15 рисунків, 52 літературних джерел за переліком посилань. Робота виконана на 62 сторінках загального тексту і 68 сторінці основного тексту.

Метою дослідження є системний аналіз застосування інтелектуальних агентів для автоматизації побудови Data Pipeline у середовищах Big Data.

Об’єктом дослідження є процеси побудови та керування Data Pipeline у середовищах Big Data.

Предметом дослідження є методи та моделі застосування інтелектуальних агентів для автоматизації та інтелектуального керування Data Pipeline.

У кваліфікаційній роботі розглядаються актуальні підходи до побудови Data Pipeline у Big Data-середовищах, архітектурні моделі інтелектуальних агентів, механізми взаємодії агентів між собою та з компонентами Data Pipeline. Розроблено концептуальну та технологічну архітектуру мультиагентної системи для автоматизації Data Pipeline. Сформовано алгоритми функціонування агентів Monitoring, Scheduler, Reconfiguration та Analytics, описано механізми координації та реалізовано програмну модель системи з використанням сучасних інструментів оркестрації, потокової обробки та моніторингу. Проведено експериментальне оцінювання ефективності роботи мультиагентної системи. Показано, що застосування інтелектуальних агентів забезпечує зменшення часу виконання Data Pipeline, підвищення адаптивності до змін навантаження, скорочення часу відновлення після збоїв та покращення відповідності вимогам SLA. Отримані результати підтверджують доцільність застосування мультиагентних систем у задачах інтелектуального керування потоками даних у Big Data-середовищах.

ІНТЕЛЕКТУАЛЬНІ АГЕНТИ, МУЛЬТИАГЕНТНІ СИСТЕМИ, DATA PIPELINE, BIG DATA, СИСТЕМНИЙ АНАЛІЗ, ОРКЕСТРАЦІЯ ДАНИХ

ABSTRACT

The qualification thesis “System analysis of the application of intelligent agents for automating data pipeline construction in Big Data environments”, submitted for the second (master’s) level of higher education in specialty 124 System Analysis, educational program Data Analysis and Security, contains 15 figures and 52 references. The thesis comprises 62 pages of total text, including 68 pages of main content.

The purpose of the research is to perform a system analysis of the use of intelligent agents for automating the construction of Data Pipelines in Big Data environments.

The object of the research is the processes of building and managing Data Pipelines in Big Data environments.

The subject of the research is the methods and models of applying intelligent agents for automation and intelligent control of Data Pipelines.

The thesis examines modern approaches to building Data Pipelines in Big Data environments, architectural models of intelligent agents, and mechanisms of agent interaction with each other and with the components of the Data Pipeline. A conceptual and technological architecture of a multi-agent system for Data Pipeline automation has been developed. Algorithms of the Monitoring, Scheduler, Reconfiguration, and Analytics agents have been designed, coordination mechanisms have been described, and a software implementation of the system has been developed using modern orchestration, stream processing, and monitoring tools. Experimental evaluation of the multi-agent system was conducted. The results show that the application of intelligent agents reduces Data Pipeline execution time, increases adaptability to workload changes, decreases recovery time after failures, and improves compliance with SLA requirements. The obtained results confirm the feasibility of using multi-agent systems for intelligent control of data flows in Big Data environments.

INTELLIGENT AGENTS, MULTI-AGENT SYSTEMS, DATA PIPELINE, BIG DATA, SYSTEMS ANALYSIS, DATA ORCHESTRATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

BDI – Beliefs-Desires-Intentions

LLM – Large Language Model

DAG – Directed Acyclic Graph

SQL – Structured Query Language

MAC – Мультиагентна система

IoT – Internet of Things

БД – база даних

API – Application programming interface

MySQL – Databases Management System

HDFS – Hadoop Distributed File System

NoSQL – not only SQL

JSON – JavaScript Object Notation

ML – Machine Learning

DAG – Directed Acyclic Graph

BI – Business intelligence

CSV – Comma-Separated Values

FIPA ACL - протоколи обміну повідомленнями

UML – Unified Modeling Language

BPMN – Business Process Model and Notation

IDEF0 – Function Modeling

IDC – International Data Corporation

S3 – Simple Storage Service

CPU – Central Processing Unit

RAM – Random Access Memory

GA – Genetic Algorithm

A* – A star

ВСТУП

Актуальність теми дослідження зумовлена стрімким зростанням обсягів даних та ускладненням процесів їх обробки, що потребує нових підходів до керування Data Pipeline. Традиційні системи оркестрації забезпечують виконання задач, проте залишаються обмеженими у здатності адаптуватися до змін середовища, своєчасно реагувати на збої та оптимально використовувати ресурси. Застосування інтелектуальних агентів дає змогу впровадити автономне, адаптивне та стійке до відмов керування, що підвищує ефективність роботи систем обробки даних у Big Data-середовищах.

Метою дослідження є системний аналіз застосування інтелектуальних агентів для автоматизації побудови Data Pipeline у середовищах Big Data.

Об’єктом дослідження є процеси побудови та керування Data Pipeline у середовищах Big Data.

Предметом дослідження є методи та моделі застосування інтелектуальних агентів для автоматизації та інтелектуального керування Data Pipeline.

Для досягнення поставленої мети в кваліфікаційній роботі необхідно вирішити наступні **завдання**:

– дослідити предметну область з точки зору системного аналізу;

– проаналізувати існуючі підходи, моделі та архітектури інтелектуальних агентів;

– визначити вимоги, обмеження та критерії ефективності агентного керування Data Pipeline;

– розробити концептуальну архітектуру мультиагентної системи для автоматизації Data Pipeline;

– сформувати алгоритми функціонування та взаємодії агентів на основі системного аналізу;

– реалізувати програмну модель системи та провести експериментальне оцінювання її ефективності.

В кваліфікаційній роботі використовувалися загальнонаукові та спеціальні методи наукового дослідження таки, як системний аналіз, синтез, порівняння, статистичний аналіз, методи машинного навчання, візуалізації даних тощо.

Наукова новизна дослідження полягає у розробленні та обґрунтуванні мультиагентної моделі інтелектуального керування Data Pipeline, яка забезпечує адаптивну реконфігурацію, скорочення затримок та підвищення стійкості конвеєрів обробки даних у Big Data-середовищах.

Теоретична значущість проведеного дослідження полягає у формуванні системної моделі взаємодії інтелектуальних агентів із структурними рівнями Data Pipeline та узагальненні критеріїв ефективності агентного керування. Отримані результати поглиблюють наукові уявлення про архітектури та методи автономної оптимізації конвеєрів даних у високодинамічних Big Data-середовищах.

Практичне значення одержаних результатів полягає у створенні робочої архітектури мультиагентної системи, що забезпечує зменшення часу виконання Data Pipeline, підвищення SLA-відповідності та відмовостійкості за рахунок прогнозного аналізу, адаптивного планування та автоматизованої реконфігурації. Розроблені алгоритми можуть бути інтегровані у реальні системи обробки даних та слугувати основою для подальшого розвитку інтелектуальних механізмів керування.

Результати кваліфікаційного дослідження **апробовані** на VI Міжнародної науково-практичної інтернет конференції «Інформаційні технології: моделі, алгоритми, системи» (ITMAS – 2025), (16-17 листопада 2025 р. Україна, Миколаїв) [45].

1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інтелектуальні агенти: поняття, архітектури та типологія

Стрімке зростання складності сучасних інформаційних систем, їхня динамічність, масштабованість процесів обробки даних та вимоги до автономного прийняття рішень зумовили розвиток нового класу технологій – інтелектуальних агентів. Агентний підхід виник як відповідь на обмеження централізованих систем, для яких із зростанням масштабу різко збільшується складність координації та керування [1]. Інтелектуальні агенти є автономними програмними сутностями, що здатні сприймати середовище, приймати рішення на основі власних знань та цілей та виконувати дії для досягнення визначених результатів [2].

Характерні властивості інтелектуальних агентів :

– автономність – здатність діяти без постійного контролю користувача, самостійно приймаючи рішення та підтримуючи власний стан;

– реактивність – своєчасне реагування на зміни середовища завдяки постійному моніторингу подій у режимі реального часу;

– проактивність – ініціювання дій для досягнення цілей, планування майбутніх кроків, прогнозування можливих ситуацій [3];

– адаптивність – здатність змінювати поведінку на основі попереднього досвіду або алгоритмів машинного навчання;

– соціальність – здатність до взаємодії з іншими агентами, обміну даними, узгодження дій та кооперації [4];

– раціональність – вибір дій, що максимізують корисність або наближають систему до досягнення цілей.

Сукупність цих характеристик забезпечує здатність інтелектуальних агентів функціонувати в умовах невизначеності, динамічних змін та розподіленого середовища.

Поведінка інтелектуального агента формується через послідовність (рис. 1.1) сприйняття (sense) – міркування (reason) – дія (act). Саме це циклічне управління забезпечує адаптацію до змін, стабільність поведінки та здатність до довготривалої роботи.

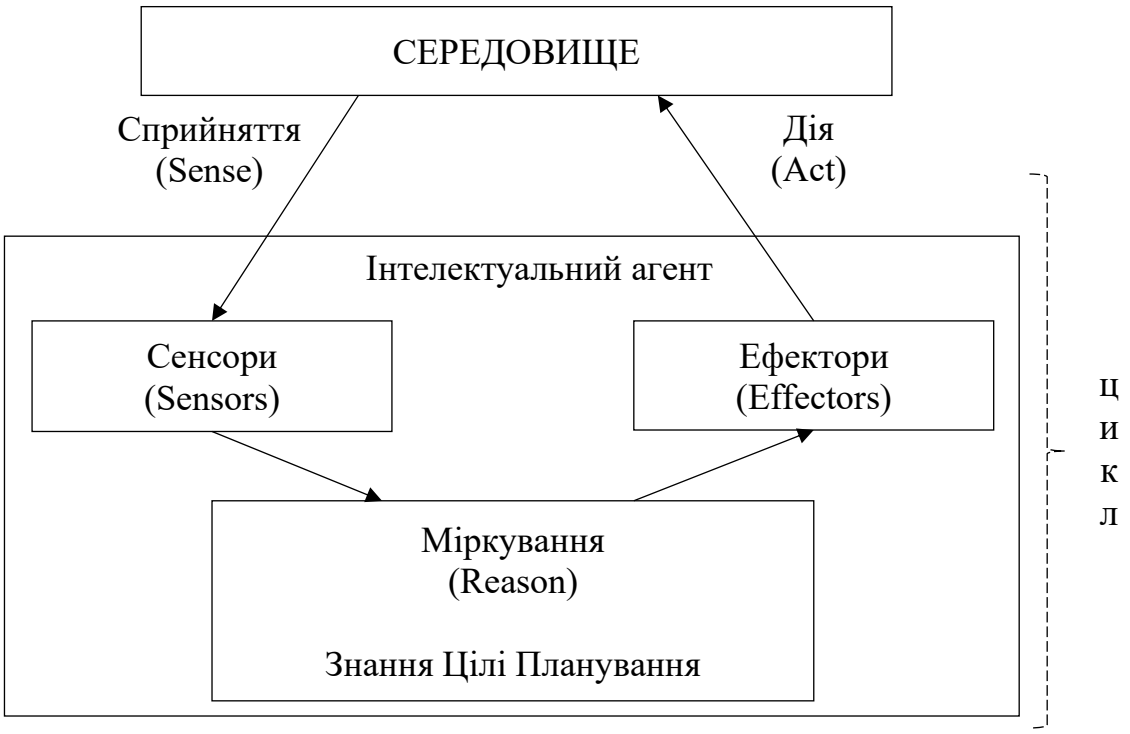

Рисунок 1.1 – Цикл функціонування інтелектуального агента

Постійно отримуючи інформацію із зовнішнього середовища, агент аналізує її, приймає рішення відповідно до внутрішніх моделей або правил та виконує дії, спрямовані на досягнення мети [5].

Різноманітність завдань, у яких застосовуються інтелектуальні агенти, призвела до формування кількох архітектурних підходів.

1. Реактивні агенти.

Працюють за принципом "стимул-реакція" без формування внутрішньої моделі середовища. Реактивні агенти характеризуються високою швидкістю реакції та мінімальними обчислювальними витратами, проте обмежені у здатності до стратегічного планування. Класичний приклад, subsumption-архітектура Брукса, де

поведінкові модулі організовано ієрархічно, і модулі верхніх рівнів можуть пригнічувати (subsume) поведінку нижчих рівнів [6].

2. Когнітивні агенти.

Мають уявлення про стан навколишнього середовища, володіють знаннями та використовують логічні механізми для прийняття рішень. Можуть формувати плани, передбачати наслідки, працювати з неповною інформацією. До таких належать архітектури Soar, ACT-R, експертні та гібридні інтелектуальні системи.

Архітектура BDI (Beliefs-Desires-Intentions) найвідоміша когнітивна модель агентів [7]. Передбачає три рівні опису: переконання (beliefs) – знання про світ, які оновлюються через сприйняття; бажання (desires) – множина бажаних станів або цілей; намір (intentions) – підмножина цілей, які прийняті до виконання [8]. BDI-агент забезпечує поєднання реактивності та стратегічного планування.

3. Гібридні агенти

Поєднують реактивні механізми нижніх рівнів із когнітивними компонентами верхніх. Нижні рівні забезпечують швидку реактивну поведінку для обробки термінових ситуацій, тоді як верхні рівні відповідають за довгострокове планування та складне міркування [9] Така інтеграція дозволяє досягти балансу між швидкістю реакції та складністю мислення, здатність до багаторівневої поведінки, стійкість у мінливих і непередбачуваних середовищах.

4. Агенти на основі великих мовних моделей (LLM- агенти).

Новий клас агентів, що використовують генеративні мовні моделі для природномовного міркування, побудови планів, динамічного вибору інструментів, автоматично генерувати фрагменти коду, DAG, SQL, адаптивне прийняття рішень, архітектуру типа ReAct (Reason+Act) [10]. Мають обмеження: стохастичну природу, непередбачуваність, потребу у значних обчислювальних ресурсах.

Порівняльний характеристика архітектур інтелектуальних агентів представлено в таблиці 1.1. Вона виявляє різні компроміси між складністю реалізації, обчислювальними вимогами та функціональними можливостями інтелектуальних агентів.

Таблиця 1.1 – Порівняльна характеристика архітектур інтелектуальних агентів

Характеристики архітектури	Реактивна	Когнітивна	Гібридна	LLM-based
Швидкість реакції	Дуже висока	Середня	Висока	Середня
Здатність до планування	Відсутня	Дуже висока	Висока	Дуже висока
Адаптивність	Низька	Середня	Висока	Дуже висока
Складність реалізації	Низька	Висока	Дуже висока	Середня
Обчислювальні вимоги	Низькі	Середні	Високі	Дуже високі
Застосування у Big Data	Моніторинг	Планування	Оркестрація	Генерація коду

Останній рядок таблиці "Застосування у Big Data" вказує на оптимальну відповідність типу архітектури агента до конкретної ролі в системі керування Data Pipeline. Такі відповідності використовуються при проєктуванні агентної системи.

Аналіз архітектур інтелектуальних агентів дозволяє зробити висновок, що вибір конкретного типу архітектури повинен ґрунтуватися на вимогами до адаптивності та складності середовища, потребі в плануванні та прогнозуванні, обчислювальних можливостях. У контексті автоматизації процесів обробки даних у Big Data-середовищах найбільш придатними є когнітивні та гібридні моделі, які забезпечують баланс між реактивною поведінкою та здатністю до стратегічного планування.

Мультиагентна система (МАС) є розподіленою обчислювальною системою, що складається з множини автономних агентів, які взаємодіють для досягнення спільних або індивідуальних цілей [11]. Колективна поведінка МАС часто має емерджентний характер, коли властивості системи виникають із локальних взаємодій агентів і не можуть бути зведені до суми можливостей окремих компонентів. Тобто в результаті взаємодії окремих агентів виникає нова, складніша поведінка, така як самоорганізація, оптимізація або колективне вирішення задач.

Структура МАС може бути централізованою, розподіленою або ієрархічною [12].

У централізованій моделі координатор-агент керує діями інших агентів, що спрощує управління, але створює «єдину точку відмови». Розподілена архітектура

забезпечує рівноправність агентів і peer-to-peer взаємодію, що підвищує відмовостійкість та масштабованість. Ієрархічні системи формують декілька рівнів управління, де агенти верхніх рівнів делегують підлеглим виконання часткових завдань.

Взаємодія агентів реалізується за допомогою мов комунікації агентів, зокрема стандарту FIPA ACL, який визначає типи повідомлень: inform, request, propose, accept-proposal, reject-proposal, query тощо [13]. Наявність чіткої комунікативної семантики забезпечує узгодженість поведінки агентів і можливість реалізації складних механізмів координації. Для розподілу завдань між агентами застосовуються протоколи Contract Net Protocol, де агент-ініціатор оголошує тендер (call for proposals), а інші агенти надсилають пропозиції. Також використовуються аукціонні механізми (англійські, голландські, Vickrey аукціони) та алгоритми колективного прийняття рішень (голосування, консенсус).

У контексті Big Data МАС є природною моделлю організації розподілених обчислень. Різні агенти можуть виконувати спеціалізовані ролі:

– агенти-монітори (Monitoring Agents) контролюють продуктивність і якість даних, виявляють аномалії в режимі реального часу;

– агенти-планувальники (Scheduler Agents) формують структуру Data Pipeline, оптимізують розклад виконання задач;

– агенти реконфігурації (Reconfiguration Agents) адаптують структуру конвеєрів, виконують відновлення після збоїв;

– агенти-аналітики (Analytics Agents) досліджують метадані, прогнозують навантаження, оптимізують параметри системи.

Завдяки децентралізації МАС здатні адаптуватися до змін інфраструктури, відмов окремих вузлів та нерівномірного навантаження, забезпечуючи стійкість і масштабованість системи. Таким чином, концептуальні основи інтелектуальних агентів і мультиагентних систем формують теоретичний фундамент для побудови адаптивних та децентралізованих рішень автоматизації, здатних ефективно функціонувати у складних динамічних Big Data-середовищах.

1.2 Екосистема Big Data як середовище функціонування агентів

Екосистема Big Data сформувалася як відповідь на потребу обробляти великомасштабні, швидкоплинні та різнорідні потоки інформації, що надходять із численних цифрових джерел [14]. На відміну від традиційних інформаційних систем, Big Data середовище поєднує високий рівень розподіленості, паралелізму та гетерогенності, що створює складні умови для узгодженого управління процесами обробки даних. Саме тому Big Data виступає природним середовищем для застосування інтелектуальних агентів, здатних забезпечувати автономну координацію, адаптивність та оптимізацію обчислювальних процесів у масштабованих конвеєрах обробки даних.

Екосистема Big Data являє собою складну ієрархічну систему, що інтегрує різнорідні програмно-апаратні компоненти для забезпечення повного життєвого циклу даних від моменту їх збору та отримання до зберігання, обробки, аналізу й фінальної візуалізації результатів. Архітектура типової Big Data-екосистеми включає кілька взаємопов'язаних функціональних рівнів, кожен з яких виконує специфічні завдання в рамках загального процесу обробки інформації [15].

Функціональні рівні архітектури Big Data-екосистеми:

1. Рівень збору даних та отримання даних.

На цьому рівні дані надходять із сенсорів IoT, журналів подій, корпоративних БД, API вебсервісів та потокових платформ. Рівень характеризується значною варіативністю швидкості та структури, дані можуть надходити нерівномірними потоками, формувати пікові навантаження або різко знижуватися, що вимагає адаптивного масштабування та механізмів буферизації.

Інструменти цього рівня поділяються на дві групи: пакетні та потокові.

Пакетне отримання (batch ingestion):

– Apache Sqoop використовується для масового перенесення структурованих даних з реляційних баз (MySQL, PostgreSQL, Oracle) до Hadoop або об’єктних сховищ. Apache Sqoop [16] підтримує інкрементальні завантаження (append-only, lastmodified),

що дозволяє передавати лише нові або змінені записи, зменшуючи навантаження на інфраструктуру.

Потокове надходження подій (streaming ingestion):

– Apache Flume – інструмент збору логів із розподілених систем;

– Logstash – компонент Elastic Stack для потокового збору, нормалізації та маршрутизації логів і метрик;

– Apache Kafka, Amazon Kinesis, Apache Pulsar – розподілені платформи для високошвидкісного приймання та доставки подій, які забезпечують стійкість до відмов, горизонтальне масштабування та гарантовану доставку [17].

Для агентних систем цей рівень є джерелом «сирих сигналів», що дозволяють агентам-моніторам реагувати на пікові навантаження, а агентам-аналізаторам — виявляти аномалії потоків у реальному часі.

2. Рівень зберігання.

Рівень забезпечує стійку, масштабовану і відмовостійку персистентність структурованих, напівструктурованих та неструктурованих даних.

Ключові технології цього рівня включають:

– Hadoop HDFS – розподілена файлова система, що зберігає дані блоками з багаторівневою реплікацією, забезпечуючи стійкість до відмов вузлів та стабільну пропускну здатність при великих послідовних операціях читання [18];

– об’єктні сховища (Amazon S3, MinIO, Ceph) забезпечують практично необмежену масштабованість, вбудоване версіонування об’єктів, доступ за REST API та інтеграцію з хмарними сервісами;

– NoSQL-сховища, оптимізовані під різні патерни доступу: колонкові (Apache Cassandra, HBase) для високошвидкісних записів і читань; документоорієнтовані (MongoDB) для гнучких нереляційних структур JSON; графові бази (Neo4j) для представлення та аналізу складних мережевих структур [19].

Для інтелектуальних агентів цей рівень є джерелом даних про стан системи, обсяг даних, зміни у схемах, збільшення затримок або аномальні зростання логів можуть

слугувати тригерами для агентів-моніторів або агентів-оптимізаторів.

3. Рівень обробки та обчислень.

Рівень реалізує основну логіку трансформації даних, включаючи фільтрацію, агрегацію, трансформацію, розподілені обчислення та моделі машинного навчання. Саме тут формуються графи обчислювальних завдань, що виконуються на кластерних обчислювальних платформах.

Основні інструменти:

– Apache Spark – уніфікована розподілена обчислювальна платформа, що забезпечує пакетну обробку, потокову обробку (Structured Streaming), роботу з SQL (Spark SQL), графовими структурами (GraphX) та моделями ML (MLlib). Завдяки оптимізатору Catalyst Spark ефективно будує плани виконання обчислювальних графів [20];

– Apache Flink – система потокової обробки в режимі реального часу з низькими затримками та гарантією exactly-once;

– Apache Storm – система потокової обробки подій з акцентом на наднизькі затримки [21].

Через те, що обчислювальні навантаження нерівномірні, на цьому рівні часто виникають ситуації перевантаження або недовикористання ресурсів. Саме тут інтелектуальні агенти можуть виконувати ролі планувальників, оптимізаторів ресурсів або детекторів аномалій у виконанні задач.

4. Рівень оркестрації та управління робочим процесом (workflow).

Оркестраційний рівень координує виконання складних багатокрокових процесів обробки даних, забезпечуючи впорядкованість, контроль станів, відновлення після помилок і узгодження залежностей між задачами.

Найпоширеніші системи оркестрації:

– Apache Airflow – реалізує робочі процеси у вигляді DAG (Directed Acyclic Graph), де кожна задача має залежності, розклад виконання, політику повторних запусків та механізми логування;

– Prefect – сучасний оркестратор, який пропонує декларативний опис потоків, автоматичне відстеження станів, кращу типізацію та ізоляцію виконання;

– Dagster – платформа, що фокусується на даних як на першорядних об’єктах, підтримуючи версіонування артефактів, типізовані pipeline та модульність [22].

Рівень особливо важливий для агентних систем, оскільки агенти можуть автоматично модифікувати DAG, адаптувати графи під змінні ресурси, виявляти помилки вузлів, ініціювати перезапуски задач або перерозподіляти навантаження між обчислювальними платформами.

5. Рівень метаданих, керування схемами та якості даних

Рівень відповідає за централізоване управління метаданими, що визначають структуру даних, їхню якість, залежності між наборами даних та походження (data lineage). Без цього рівня масштабні Data Pipeline стають непрозорими і важко керованими.

Основні інструменти:

– Apache Atlas – система керування метаданими, яка забезпечує побудову повного ланцюжка походження даних, тобто визначає шлях трансформації даних від джерела до кінцевого набору.

– Hive Metastore – каталог схем таблиць, партицій та форматів зберігання.

– DataHub, Amundsen – сучасні Data Catalog системи, що додають можливості пошуку наборів даних, документування, оцінки якості, а також інтеграцію з ML-процесами.

На цьому рівні інтелектуальні агенти можуть виконувати валідацію схем, визначати порушення сумісності (schema drift), контролювати якість даних, відслідковувати походження та позначати потенційні ризики.

6. Рівень аналітики та візуалізації.

Фінальний рівень екосистеми Big Data забезпечує доступ до результатів обробки даних для аналітиків, дослідників, бізнес-користувачів та систем підтримки прийняття рішень.

Основні інструменти:

– SQL-движки (Presto, Trino, Apache Impala) – забезпечують інтерактивні запити до розподілених джерел даних з мілісекундними затримками;

– BI-платформи (Apache Superset, Tableau, Metabase) – дозволяють візуалізувати результати, будувати дашборди, створювати інтерактивні панелі управління [23].

У контексті агентних систем цей рівень може бути точкою зворотного зв'язку: агенти-аналітики можуть генерувати автоматичні звіти, оцінювати тренди, передбачати навантаження або пропонувати рекомендації щодо зміни структури конвеєрів даних.

З позицій системного аналізу екосистема Big Data – це складна розподілена система, що характеризується високою динамічністю, мінливістю потоків навантаження та різнорідністю програмних компонентів. У таких умовах зростає важливість автономних механізмів координації, прийняття рішень та адаптації, які можуть бути реалізовані через інтелектуальних агентів.

Одним із ключових структурних елементів Big Data є Data Pipeline (конвеєр обробки даних) – послідовність процесів переміщення та трансформації даних, яка в системному аналізі розглядається як мережа взаємодіючих компонентів, що мають власні стани, ресурси та залежності [24].

Структурна складність Data Pipeline проявляється у багаторівневих залежностях між задачами. Кожна трансформація може мати умови запуску, вимоги до ресурсів, часові обмеження або залежати від результатів попередніх кроків. Сукупність таких залежностей формує Directed Acyclic Graph (DAG) – модель кінцевої системи обробки даних, у якій кожен вузол виконує окрему функцію, а зміна одного елемента може впливати на весь ланцюг [25].

Другою фундаментальною характеристикою є розподіленість. Обробка великих обсягів даних виконується на десятках або сотнях вузлів, що працюють паралельно. Звідси виникають задачі координації, балансування навантаження, обробки відмов і

забезпечення узгодженості. У традиційних підходах це вимагає ручного налаштування, складної конфігурації та постійного втручання інженера.

Гетерогенність Data Pipeline полягає у великій різноманітності джерел даних, форматів, протоколів доступу та обчислювальних платформ, що беруть участь у його роботі. У межах одного Pipeline можуть одночасно використовуватися реляційні бази даних, NoSQL-системи, файлові сховища, потокові платформи, API вебсервісів, а також різні формати даних (JSON, Avro, Parquet, CSV).

Через це кожен етап конвеєра потребує перетворення, узгодження схем, конвертації форматів та адаптації семантики даних. Така гетерогенність ускладнює інтеграцію компонентів, підвищує ризик несумісності та значно збільшує обсяг ручної роботи з підтримки Data Pipeline [24].

На практиці побудова та підтримка Data Pipeline супроводжується низкою проблем. Згідно зі звітами Wakefield Researc [26] та IBM CDO [27], до 65% часу інженери витрачають на виправлення помилок схем, налаштування з’єднань, синхронізацію форматів та усунення збоїв. Близько 30–40% Data Pipeline втрачають працездатність після зміни структури джерел даних (schema drift). У багатьох організаціях понад 50% інцидентів пов’язані з ручним втручанням у конфігурацію або логіку Data Pipeline [28]. Такі факти підкреслюють масштаб проблеми та необхідність автономних систем керування.

Динамічність середовища Big Data зумовлює непередбачуваність поведінки системи. Обсяги даних можуть змінюватися в десятки разів, структури джерел еволюціонувати, а обчислювальна інфраструктура зазнавати тимчасових відмов. Data Pipeline повинен адаптуватися до таких змін без втрати продуктивності та цілісності даних, що вимагає механізмів самоорганізації, самовідновлення та прогнозування.

Висуваються також підвищені вимоги до достовірності та якості даних. Необхідно гарантувати відсутність дублікатів, правильність трансформацій, відповідність схемам, відповідність бізнес-правилам та стабільність затримки. Помилка в одному кроці може поширитися на десятки систем і призвести до некоректних управлінських рішень.

Таким чином, екосистема Big Data створює середовище з високою складністю, мінливістю та гетерогенністю, у якому традиційні централізовані методи керування процесами є недостатніми. Інтелектуальні агенти здатні забезпечити автономне прийняття рішень, оптимізацію ресурсів, адаптивну координацію та високий рівень стійкості Data Pipeline, що робить їх застосування в Big Data одним із найбільш перспективних напрямів розвитку сучасних інформаційних систем.

1.3 Методи аналізу процесів автоматизації Data Pipeline

Автоматизація побудови Data Pipeline у середовищах Big Data є складним завданням, оскільки конвеєри обробки даних включають десятки взаємопов’язаних процесів, що виконуються у розподіленому та динамічному середовищі. Data Pipeline має багаторівневу структуру, тому для ефективного проєктування таких систем потрібні спеціалізовані методи аналізу, які дозволяють формально описати його архітектуру, виявити залежності між етапами, оцінити властивості виконання та визначити критерії ефективності.

На відміну від базових підходів, де конвеєр даних розглядається як послідовність окремих операцій, аналіз у контексті Big Data має враховувати розподіленість інфраструктури, динамічність потоків даних, гетерогенність технологічного середовища та мінливість навантаження, що зумовлює необхідність застосування як структурних, так і поведінкових методів моделювання, а також порівняльного аналізу сучасних інструментів автоматизації.

У процесі аналізу важливо враховувати й таку властивість, як оркестрація – координація, планування та управління виконанням усіх етапів Data Pipeline, включно з контролем залежностей, відновленням після збоїв та забезпеченням узгодженого виконання. Саме оркестрація визначає поведінку Data Pipeline як цілісної системи.

Ключові методи аналізу підходів, які використовуються під час проєктування та оцінювання процесів автоматизації Data Pipeline, включають аналіз архітектурних підходів, визначення критеріїв ефективності та огляд існуючих рішень.

Проєктування Data Pipeline ґрунтується на визначенні структури обробки даних, послідовності трансформацій, механізмів взаємодії компонентів та способів забезпечення продуктивності й відмовостійкості. Серед найпоширеніших підходів:

– підхід на основі статичних DAG (Directed Acyclic Graph) є класичною моделью, у якій процеси представлено як вузли графа, а залежності між задачами як ребра графа. Такий підхід реалізовано у Apache Airflow, Azkaban, Luigi [29].

Переваги: прозорість залежностей, керованість, відтворюваність.

Обмеження: статичність структури, складність адаптації до змін.

– підхід динамічного формування робочого процесу (runtime-oriented execution) реалізовано у Prefect. Граф формується в момент виконання, що дозволяє адаптувати структуру Data Pipeline у відповідь на зміну кількості вхідних файлів, появу нових джерел даних, зміну стану обчислювальної інфраструктури.

– дані-орієнтований підхід (asset-based) реалізовано у Dagster. Основною сутністю виступає актив (asset) – набір даних або модель, а задачі є функціями над ними. Підхід дозволяє моделювати траєкторію формування даних у процесі їх обробки, відстежувати фундаментальні залежності та забезпечувати контроль якості.

– подієво-орієнтований підхід (event-driven) характерний для потокових-систем (Kafka Streams, Flink). Такі підходи не мають статичної структури, Data Pipeline формується як реакція на події.

Сучасні підходи до проєктування Data Pipeline задовольняють вимоги до масштабованості та продуктивності, проте не забезпечують автономності, самовідновлення та проактивної оптимізації, що обумовлює потребу в агентному підході.

Система інтелектуальних агентів розглядається як цілісне утворення, що має:

– мету функціонування: мінімізація затримки, підвищення пропускної здатності, забезпечення якості даних;

– структуру: типи агентів, канали комунікації, протоколи взаємодії, ресурси;

– функції та процеси: збір метрик, прогнозування навантаження, керування виконанням Pipeline, оптимізація ресурсів;

– середовище: інфраструктура Big Data, потоки даних, кластерні ресурси, зовнішні системи;

– обмеження: часові, обчислювальні, логічні, ресурсні;

– критерії ефективності: стабільність, відмовостійкість, швидкість реакції, адаптивність [30].

Такий підхід дозволяє побачити система інтелектуальних агентів як багаторівневу взаємодію, де кожен агент впливає на цілісні властивості Data Pipeline.

Системний аналіз таких систем охоплює кілька взаємопов’язаних рівнів:

Рівень середовища (environment level) – описує зовнішнє середовище, в якому функціонує інфраструктура Big Data; джерела даних; кластерні ресурси; потоки подій та навантаження. На цьому рівні досліджуються властивості мінливості, гетерогенності, розподіленості та обмежень інфраструктури.

Рівень агентів (agent level). Охоплює: типи агентів, їхні стани, локальні цілі, політики поведінки, алгоритми прийняття рішень. Для Data Pipeline це можуть бути агенти-монітори, агенти-планувальники, агенти-виконавці, агенти-оптимізатори.

Рівень взаємодій (interaction level). Описує механізми координації: протоколи обміну повідомленнями (FIPA ACL), механізми узгодження рішень, розподіл задач, спільне планування. Для Big Data важливими є синхронізація, відмовостійкість та відновлення.

Рівень системної поведінки (system-level behaviour). Аналізує: емерджентні властивості, стійкість системи, здатність до самоорганізації та самовідновлення, глобальну ефективність. Саме тут визначається, наскільки система як ціле дозволяє зменшувати затримки, стабілізувати перфоманс і оптимізувати обчислення.

Для дослідження автоматизації Data Pipeline застосовуються п'ять груп методів аналізу, кожна з яких фокусується на певному аспекті системи.

Групи методів аналізу Data Pipeline:

1. Структурні методи.

Використовують для дослідження архітектури Data Pipeline, взаємозв’язків компонентів, залежностей між задачами та потоками даних. Застосовуються нотації UML (діаграми діяльності, компонентів), BPMN, IDEF0, а також графові моделі залежностей. Такі методи дозволяють формально описати структуру Data Pipeline, визначити критичні маршрути та потенційні вузькі місця.

2. Поведінкові методи.

Досліджують динаміку виконання процесів, реакцію системи на зміни навантаження, затримки та збої. Застосовуються логічні моделі виконання, аналіз подій, моніторинг runtime-метрик, профілювання потоків. Методи дозволяють виявляти латентні проблеми, які не видно зі статичних моделей.

3. Аналіз гетерогенного середовища.

Оскільки Data Pipeline зазвичай працює поверх різнорідних платформ (Spark, Kafka, NoSQL, Data Lake), необхідно враховувати вплив технологічної неоднорідності на узгодженість, продуктивність та відмовостійкість. Методи аналізу включають оцінювання формату даних, механізмів доставки, політик consistency та обмежень інфраструктури.

4. Порівняльний аналіз інструментів автоматизації.

Для вибору оптимального підходу до побудови Data Pipeline використовують порівняльний аналіз оркестраторів (Apache Airflow, Prefect, Dagster, Luigi). Аналізу підлягають їхні властивості, моделі виконання, здатність до масштабування, підтримка lineage, fault tolerance та динамічності.

5. Аналітичні методи оцінювання ефективності.

Дозволяють оцінити end-to-end затримки, пропускну здатність, надійність, стійкість до збоїв та ресурсну ефективність.

Застосування наведених методів дозволяє комплексно дослідити процеси автоматизації Data Pipeline, виявити обмеження традиційних інструментів та обґрунтувати доцільність використання інтелектуальних агентів, що забезпечують перехід від пасивної оркестрації до автономного, адаптивного управління Data Pipeline у розподілених середовищах Big Data.

Висновки до розділу 1

У першому розділі проведено комплексний аналіз ключових складових, необхідних для розроблення агентної системи автоматизації Data Pipeline у середовищах Big Data. Розглянуто поняття інтелектуальних агентів, їх властивості та основні архітектурні підходи. Проаналізовано особливості функціонування мультиагентних систем, механізми їх взаємодії та координації, включаючи протоколи комунікації та колективне розв’язання задач. Показано, що властивості MAC є природною основою для побудови розподілених адаптивних систем управління конвеєрами обробки даних.

Детально проаналізовано архітектуру Big Data-екосистеми та функціональні рівні Data Pipeline. Виявлено характеристики Data Pipeline як складної розподіленої та динамічної системи, що працює в умовах мінливого навантаження, різнорідності форматів даних та високих вимог до продуктивності, узгодженості та відмовостійкості.

Здійснено аналіз методів дослідження та проєктування Data Pipeline. Розглянуто підходи до побудови конвеєрів. Показано їх переваги й обмеження у контексті автоматизації. Проаналізовано групи методів, які застосовуються для опису та оцінювання процесів.

Проведений огляд показав, що сучасні підходи та інструменти автоматизації Data Pipeline мають низку системних обмежень, які формують передумови для застосування агентного підходу. На відміну від традиційних рішень, агентні системи забезпечують децентралізоване автономне управління, здатність до адаптації, самоорганізації та проактивних дій у Big Data-середовищах.

2 СИСТЕМНИЙ АНАЛІЗ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ ДЛЯ АВТОМАТИЗАЦІЇ DATA PIPELINE

2.1 Системний аналіз процесів керування Data Pipeline

Сучасні Big Data-середовища характеризуються стрімким зростанням обсягів, швидкості та різноманітності даних, що підтверджується аналітичними звітами Gartner, IDC і Seagate [31, 32]. За прогнозами, глобальний обсяг створюваних даних зріс із 1,2 зетабайт у 2010 році до 120 зетабайт у 2023 році, а до 2027 років очікується збільшення до понад 200 зетабайт [33]. Такий масштабний ріст суттєво підвищує вимоги до гнучкості, адаптивності та стійкості систем обробки даних. У цих умовах Data Pipeline виступає фундаментальною інфраструктурною одиницею, яка забезпечує безперервний потік даних від джерел до аналітичних платформ та систем підтримки прийняття рішень [34, 35].

Data Pipeline розглядається як складна динамічна система, що включає послідовні процеси: отримання даних (data ingestion), очищення й попередня підготовка (preprocessing), збереження у розподілених сховищах (storage), трансформацію (transformation), аналітику (analytics), доставку результатів та візуалізацію (visualization).

Кожен етап функціонує як підсистема зі своїми вхідними та вихідними параметрами, набором операцій і правилами взаємодії [36]. У реальних умовах між етапами існують численні зворотні зв’язки, що ускладнює процеси управління, моніторингу та реагування на події (рис. 2.1).

З позицій системного аналізу Data Pipeline має такі властивості:

– цілісність – усі етапи формують єдину функціональну систему;

– модульність – компоненти можуть масштабуватися та оновлюватися незалежно;

– ієрархічність – процеси організовані за рівнями обробки, що відповідають архітектурі Big Data;

– адаптивність – система реагує на зміни обсягів, структури й швидкості надходження даних [37];

– дистрибутивність – виконання задач відбувається на багатьох обчислювальних вузлах;

– варіативність середовища – джерела, формати, обчислювальні платформи й ресурси можуть змінюватись у режимі реальному часі [38].

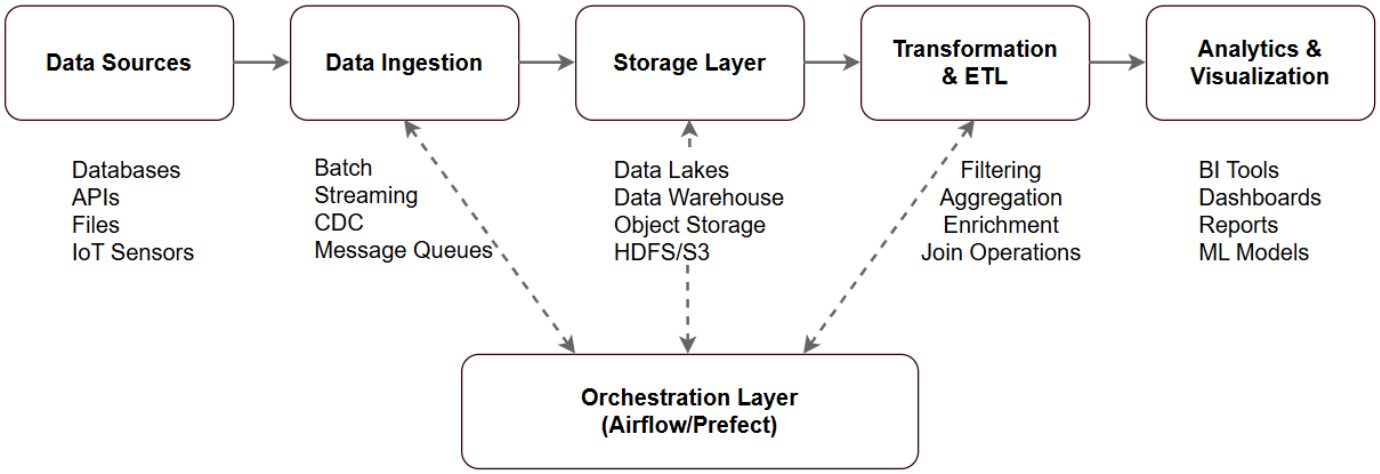

Рисунок 2.1 – Типова архітектура Data Pipeline

У практиці застосовуються три базові архітектурні підходи:

1. Монолітна архітектура – усі етапи інтегровані в одному застосунку (Informatica PowerCenter, IBM DataStage). Переваги: інтегрованість, надійність. Недоліки: низка масштабованість, обмежена гнучкість, складність оновлення [39, 40].

2. Мікросервісна архітектура – кожен етап реалізується як окремий сервіс із власним API та контейнеризацією (Docker, Kubernetes) [41, 42]. Переваги: ізоляція збоїв, адаптивне масштабування. Недоліки: складність оркестрації та забезпечення консистентності.

3. Оркестраційна архітектура – побудована на фреймворках керування потоками даних Apache Airflow, Prefect, Dagster. DAG-структури (Directed Acyclic Graph) визначають залежності та порядок виконання задач [43]. Оркестровані системи

виступають керуючим рівнем над середовищами обчислень (Apache Spark, Apache Flink, Hadoop, Beam). Вони забезпечують узгодженість потоків даних та контроль станів. Недолік: статичність DAG та недостатня здатність до адаптації при зміні стану системи або характеристик даних [44].

На основі аналізу існуючих підходів виділено чотири рівні функціонування Data Pipeline (рисунок 2.2).

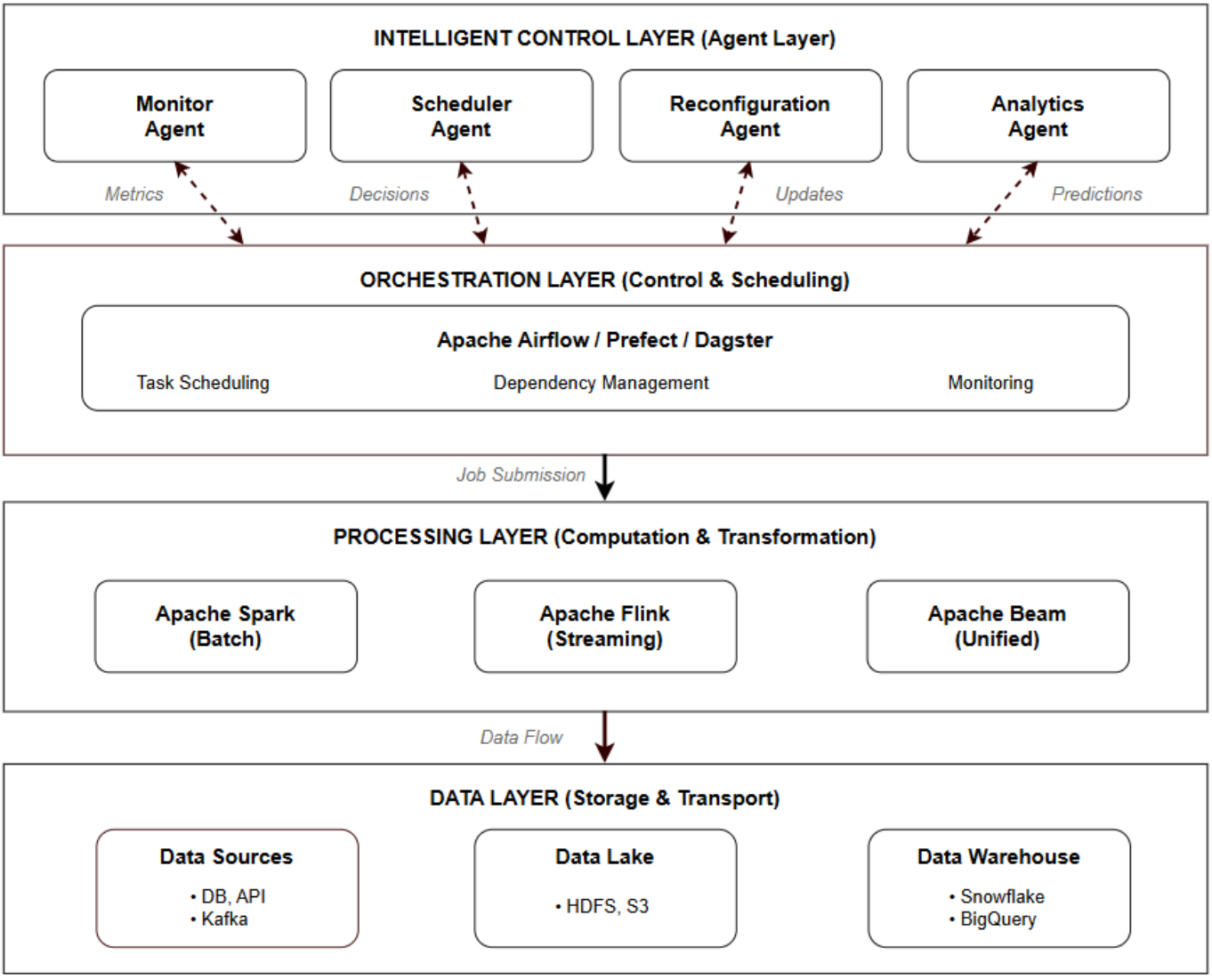

Рисунок 2.2 – Чотирирівнева архітектура Data Pipeline з інтелектуальним рівнем керування

1. Data Layer (Рівень даних) – включає джерела (реляційні бази даних, NoSQL-сховища, потокові платформи Kafka), проміжні сховища (data lakes на базі HDFS або S3, data warehouses типу Snowflake/BigQuery), черги повідомлень і транспортні протоколи (REST API, gRPC). Рівень відповідає за фізичне зберігання та транспортування даних.

2. Processing Layer (Рівень обробки) – реалізує трансформації та аналітичні операції з використанням розподілених фреймворків: Apache Spark, Apache Flink, Apache Beam. На цьому рівні виконуються операції фільтрації, агрегації, збагачення та трансформації даних.

3. Orchestration Layer (Рівень оркестрації) – керує життєвим циклом задач через спрямовані ациклічні графи (DAG). Системи оркестрації (Apache Airflow, Prefect, Dagster) обробляють залежності між задачами, здійснюють півтори при збоях, збирають метрики та логи. Рівень є критичним для забезпечення надійності всього конвеєра.

4. Intelligent Control Layer (Рівень інтелектуального керування) – новий рівень, де функціонує мультиагентна система для автономного, адаптивного керування. Саме цій рівень є ключовим для інтелектуального керування Data Pipeline, на ньому впроваджуються автономні агенти, які виконують завдання моніторингу, планування, реконфігурації та аналітики [45].

Запропонована чотирирівнева архітектура Data Pipeline повністю узгоджується з шістьма функціональними рівнями Big Data-екосистеми і формує схему Data Pipeline.

Відповідність подано в таблиці 2.1.

Таблиця 2.1 – Відповідність чотирирівневої архітектури Data Pipeline та шестирівневої архітектури Big Data-екосистеми

Чотирирівнева архітектура Data Pipeline	Шестирівнева архітектура Big Data
1. Data Layer (Рівень даних)	Рівень 1: Збору даних та отримання даних (Kafka, Flume, Sqoop)
	Рівень 2: Зберігання (HDFS, S3, NoSQL)
2. Processing Layer (Рівень обробки)	Рівень 3: Обробки та обчислень (Spark, Flink, Storm)

Продовження табл.2.1

3. Orchestration Layer (Рівень оркестрації)	Рівень 4: Оркестрації та управління робочим процесом (Airflow, Prefect, Dagster) Рівень 6: Аналітики та візуалізації (частково – моніторинг виконання)
4. Intelligent Control Layer (Рівень інтелектуального керування) Забезпечує автономне керування всіма рівнями через спеціалізовані агенти	НОВИЙ РІВЕНЬ (відсутній у традиційній Big Data-архітектурі) Реалізується через агентну систему: – Monitoring Agent (MA) – Scheduler Agent (SA) – Reconfiguration Agent (RA) – Analytics Agent (AA)

Додатково рівень 5 традиційної Big Data-архітектури (метадані, керування схемами та якість даних) підтримує Knowledge Base агентної системи, яка зберігає історичні дані, політики керування та моделі прогнозування.

Типи агентів відповідають своїм функціональним вимогам:

– реактивні агенти для моніторингу;

– когнітивні BDI-агенти для планування;

– гібридні агенти для реконфігурації;

– когнітивні агенти з ML для аналітики.

Агенти координують дії через Contract Net Protocol, формуючи адаптивну систему, здатну до самовідновлення (self-healing), самооптимізації (self-optimization) та самоадаптації (self-adaptation).

Таким чином, інтелектуальний рівень керування виступає необхідною надбудовою над традиційною Big Data-архітектурою, забезпечуючи перехід від статичних конвеєрів до проактивних, самонавчальних і стійких систем обробки даних.

2.2 Функціональна та концептуальна модель системи

Побудова інтелектуального рівня керування (Intelligent Control Layer) у структурі Data Pipeline потребує формування узгодженої функціональної моделі, здатної забезпечити автономне, адаптивне та стійке управління процесами обробки даних у середовищах Big Data. У таких середовищах традиційні підходи до оркестрації робочих процесів демонструють обмежену здатність до самостійного реагування на зміни навантаження, збої інфраструктури або еволюцію джерел даних, що зумовлює необхідність впровадження інтелектуальних механізмів керування.

У цьому контексті мультиагентний підхід розглядається не як альтернатива існуючим засобам оркестрації, а як надбудова над ними, що забезпечує інтелектуальне керування життєвим циклом Data Pipeline. Мультиагентна система виконує роль функціональної моделі керування, у межах якої автономні агенти спеціалізуються на виконанні окремих управлінських функцій та координують свої дії на основі спільного контексту.

Функціональна модель базується на принципі поділу відповідальності (separation of concerns), відповідно до якого кожен агент відповідає за окремий аспект керування Data Pipeline. Такий підхід забезпечує модульність системи, спрощує масштабування та підвищує стійкість до відмов, оскільки збій або деградація одного агента не призводить до втрати працездатності всієї системи [46].

У межах запропонованої концептуальної моделі виділяються чотири основні типи агентів: Monitoring Agent, Scheduler Agent, Reconfiguration Agent та Analytics Agent. Вибір саме цих типів агентів обумовлений аналізом архітектур інтелектуальних агентів, наведеним у розділі 1, а також специфікою задач керування Data Pipeline у Big Data-середовищах.

1. Monitoring Agent (MA) виконує функції безперервного спостереження за станом системи та формування оперативного уявлення про її поточну поведінку.

Архітектурно даний агент належить до реактивного типу, що забезпечує мінімальний час реагування на події та зміни параметрів середовища.

Основними функціями Monitoring Agent є :

– збір метрик продуктивності обчислювальних ресурсів;
– контроль тривалості виконання задач;
– виявлення простоїв та деградації продуктивності;
– детектування аномальних ситуацій на основі статистичних методів та моделей машинного навчання.

Зібрана інформація використовується для формування подій, попереджень та оновлення спільного контексту системи [47].

2. Scheduler Agent (SA) відповідає за інтелектуальне планування виконання задач у межах Data Pipeline. Даний агент реалізує когнітивну архітектуру типу BDI (Beliefs–Desires–Intentions), що дозволяє формувати уявлення про стан системи, визначати цільові стани та обирати оптимальні дії з урахуванням багатокритеріальних обмежень.

Функції SA:

– аналіз поточного навантаження та доступності ресурсів;
– формування оптимального розкладу виконання задач;
– врахування обмежень CPU/RAM/disk I/O та SLA;
– застосування алгоритмів евристичного пошуку;
– багатокритеріальна оптимізація (мінімізація makespan, latency, cost);
– динамічна адаптація розкладу при зміні середовища;
– оптимізація DAG з урахуванням пріоритетів задач.

На основі цієї інформації агент формує оптимальний розклад виконання задач, здійснює балансування навантаження та адаптує порядок виконання у разі зміни умов середовища.

3. Reconfiguration Agent (RA) забезпечує адаптивність системи шляхом динамічної зміни структури виконання Data Pipeline. Даний агент реалізує гібридну

архітектуру, що поєднує реактивні механізми швидкого реагування з когнітивними компонентами стратегічного планування.

Функції RA:

– стану конвеєра після виникнення збоїв або зміни характеристик джерел даних;
– визначення критичних шляхів та вузьких місць;
– перенесення задач між вузлами кластера;
– реконфігурація топології виконання;
– реалізація механізмів відновлення (retry, fallback, circuit breaker);
– застосування теорії графів для вибору оптимальних сценаріїв відновлення.

Агент може здійснювати реконфігурацію топології виконання, міграцію задач між вузлами та застосування політик відмовостійкості.

4. Analytics Agent (AA) виконує функції прогнозної аналітики та стратегічного управління. Даний агент реалізує когнітивну архітектуру з інтеграцією моделей машинного навчання, що дозволяє виявляти патерни у поведінці Data Pipeline та прогнозувати майбутні стани системи.

Функції AA:

– прогнозування майбутніх навантажень (LSTM, GRU);
– оцінка ризиків порушення SLA;
– виявлення трендів та патернів у поведінці Data Pipeline;
– прогнозування аномальних ситуацій;
– рекомендації щодо оптимальних моментів масштабування;
– класифікація типів збоїв та кластеризація моделей навантаження.

Взаємодія між агентами здійснюється на основі асинхронного обміну повідомленнями та спільного використання контекстної інформації. Такий підхід забезпечує децентралізований характер керування, знижує залежність від централізованих компонентів та підвищує стійкість системи до часткових відмов.

Агенти обмінюються узагальненими подіями, рекомендаціями та результатами аналізу, що дозволяє узгоджувати управлінські рішення без жорсткої синхронізації [48].

Життєвий цикл агентів у системі керування Data Pipeline описується моделлю "сприйняття – міркування – дія – навчання", Perception–Reasoning–Action–Learning (PRAL), що забезпечує безперервне: сприйняття сигналів середовища; аналіз і прийняття рішень; виконання дій; самооновлення та навчання.

Модель відображає безперервний цикл адаптації та навчання агента.

Фаза 1. Сприйняття (Perception)

– Monitor Agent збирає метрики з Prometheus API та Grafana;
– Scheduler Agent читає чергу задач з Airflow API або Prefect;
– Reconfiguration Agent отримує події про збої з системних логів;
– Analytics Agent завантажує історичні дані з Knowledge Base.

Фаза 2. Міркування (Reasoning)

– виявляє проблеми або можливості для оптимізації;
– оцінює поточний стан відносно цільових метрик (SLA, resource utilization);
– визначає стратегію реагування на основі політик у Knowledge Base;
– використовує rule-based системи (if-then правила) для простих випадків;
– застосовує моделі машинного навчання для складних сценаріїв [49];
– розраховує очікувану корисність різних варіантів дій.

Фаза 3. Дія (Action)

– Scheduler Agent надсилає команди оркестратору для створення/модифікації задач;
– Reconfiguration Agent змінює структуру DAG або мігрує задачі;
– Monitor Agent генерує оповіщення при виявленні критичних проблем;
– Analytics Agent оновлює прогнози та рекомендації.

Дії виконуються з урахуванням обмежень на ресурси та пріоритетів задач. Агент забезпечує транзакційність операцій та можливість отката у разі помилок.

Фаза 4. Навчання (Learning)

– порівнює фактичні результати з очікуваними;

– оновлює базу знань новими спостереженнями;

– адаптує політики поведінки на основі отриманого досвіду;

– використовує Reinforcement Learning [50], де агент отримує винагороду за досягнення цільових показників;

– коригує вагові коефіцієнти у utility-функції;

– оновлює моделі прогнозування з урахуванням нових даних.

Навчання відбувається як online (у реальному часі), так і batch (періодичне перенавчання моделей на історичних даних).

На етапі сприйняття агенти отримують інформацію про стан середовища та системи. На етапі міркування відбувається аналіз отриманих даних, оцінювання відповідності поточного стану цільовим показникам та вибір стратегії дій. На етапі дії агенти ініціюють управлінські впливи на систему, спрямовані на оптимізацію її роботи. Етап навчання забезпечує оновлення внутрішніх моделей та політик керування на основі отриманого досвіду, що підвищує адаптивність системи у довгостроковій перспективі.

Інформаційну підтримку агентної системи забезпечує спільний контекст знань, який використовується для зберігання узагальненої інформації про стан системи, історію виконання Data Pipeline, керувальні політики та результати аналітичних моделей. Наявність такого спільного контексту дозволяє агентам приймати узгоджені рішення та враховувати як поточні, так і історичні характеристики функціонування системи.

Структура бази знань представлена на рис. 2.3 та охоплює чотири компоненти.

1. Метрики системного стану. Містять поточні значення ключових операційних метрик:

– ресурси вузлів: CPU utilization (%), Memory usage (GB), Disk I/O (MB/s), Network throughput (Mbps);

– стан задач: execution status (running/failed/pending), queue depth, task duration;

– продуктивність: latency (ms), throughput (records/sec), error rate (%);

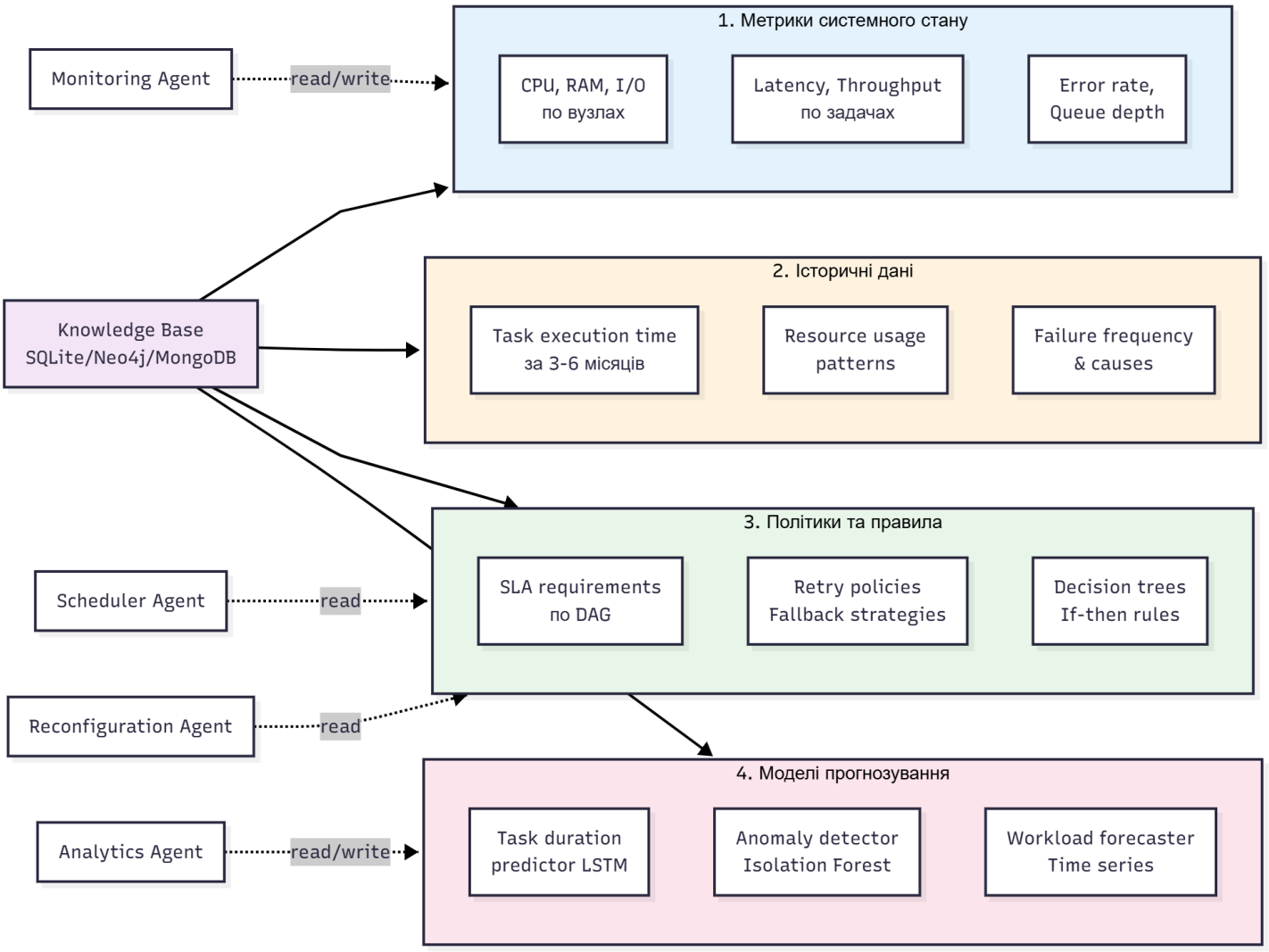

Рисунок 2.3 – Структура бази знань

– SLA метрики: compliance level (%), time to SLA breach, priority violations.

Метрики оновлюються з частотою від 10 секунд (критичні показники) до 5 хвилин (агрегатні статистики). Такий підхід забезпечує прозорість моніторингу та оперативної діагностики стану системи.

1. Історичні дані. Архів виконання пайплайнів за визначений період (зазвичай 3–6 місяців), який включає:

– показники тривалості задач: task execution time, pipeline end-to-end latency;

– використання ресурсів: peak/average CPU, memory footprint, I/O patterns;

– частоту та причини виникнення збоїв: failure frequency, root cause analysis, correlation with external factors;

– кореляції між окремими метриками та появою проблем, залежності від часу доби/дня тижня.

Аналіз історичних даних дає змогу виявляти закономірності, сезонні патерни та оптимізувати управління системою. Використовуються методи time series analysis, seasonal decomposition та кластеризації.

2. Політики та правила. Містять формалізовані правила керування системою:

– SLA-вимоги до різних типів пайплайнів: максимальна тривалість виконання, допустима затримка, пріоритети пайплайнів;

– стратегії відновлення після збоїв: retry policies (max attempts, backoff strategy), fallback mechanisms, circuit breaker patterns;

– правила балансування навантаження: load distribution algorithms, resource allocation policies;

– алгоритми прийняття рішень, представлені у вигляді конструкцій if–then або дерев рішень.

Політики представлені у вигляді конструкцій if-then або дерев рішень, що забезпечує єдину базу для автоматизованого reasoning'у агентів.

Приклад правила:

```
IF task_duration > 1.5 * historical_average AND cpu_usage < 50%
THEN trigger_scheduler_agent WITH priority=HIGH
```

3. Моделі прогнозування. Охоплюють натреновані моделі машинного навчання для :

– передбачення тривалості задач,

– класифікації типів збоїв,

– виявлення аномальних сценаріїв функціонування

– кластеризації поведінки пайплайнів.

Інтеграція моделей у базу знань підвищує рівень автономності та адаптивності

агентів. Моделі періодично перенавчуються (кожні 1-2 тижні) для врахування змін у поведінці системи.

Реалізація бази знань може здійснюватися на основі графових баз даних Neo4j або документо-орієнтованих рішень MongoDB, що забезпечує гнучкість у зберіганні різнорідних даних. Доступ до бази знань здійснюється через визначені інтерфейси REST API або GraphQL, що дозволяє агентам отримувати лише релевантну інформацію відповідно до їхніх задач, а також ефективно управляти навантаженням на мережу та інфраструктуру системи.

Запропонована функціональна та концептуальна модель мультиагентної системи формує основу інтелектуального рівня керування Data Pipeline. Вона забезпечує автономність, адаптивність, прогнозування та відмовостійкість системи в умовах високої динамічності Big Data-середовищ і відповідає сучасним вимогам до побудови масштабованих та інтелектуальних інфраструктур обробки даних.

2.3 Архітектура та механізми взаємодії агентів

Архітектура інтелектуального рівня керування Data Pipeline спирається на децентралізовану модель, у якій агенти взаємодіють між собою через подієву інфраструктуру, спільні інформаційні ресурси та протоколи координації.

Взаємодія агентів реалізується через три ключові компоненти:

1. Подієва шина.

Подієва шина забезпечує асинхронний обмін повідомленнями між агентами за принципом publish–subscribe. Через подієва шину надходять операційні події, попередження про збої, сигнали про перевищення порогових значень метрик, запити на реконфігурацію,прогнози нав антаження.

2. База знань.

Використовується агентами як єдине джерело правил, історичних даних, статистики та прогнозних моделей. Усі агенти мають доступ до бази знань через

REST/GraphQL API.

3. Оркестратор (Airflow / Prefect / Dagster).

Оркестратор виконує команди агентів і забезпечує оновлення DAG, запуск, призупинення та перенесення задач.

База знань та подієва шина виступають як комунікаційна інфраструктура, через яку реалізуються розповсюдження подій, передача сигналів про аномалії, запити на планування чи реконфігурацію, обмін прогнозами та метриками [51]. Архітектура взаємодії є децентралізованою, немає головного агента, рішення приймаються колективно або через протоколи конкуренції.

Для координації використовується Contract Net Protocol (CNP) – стандартний механізм децентралізованого вибору виконавця задачі (рис. 2.4).

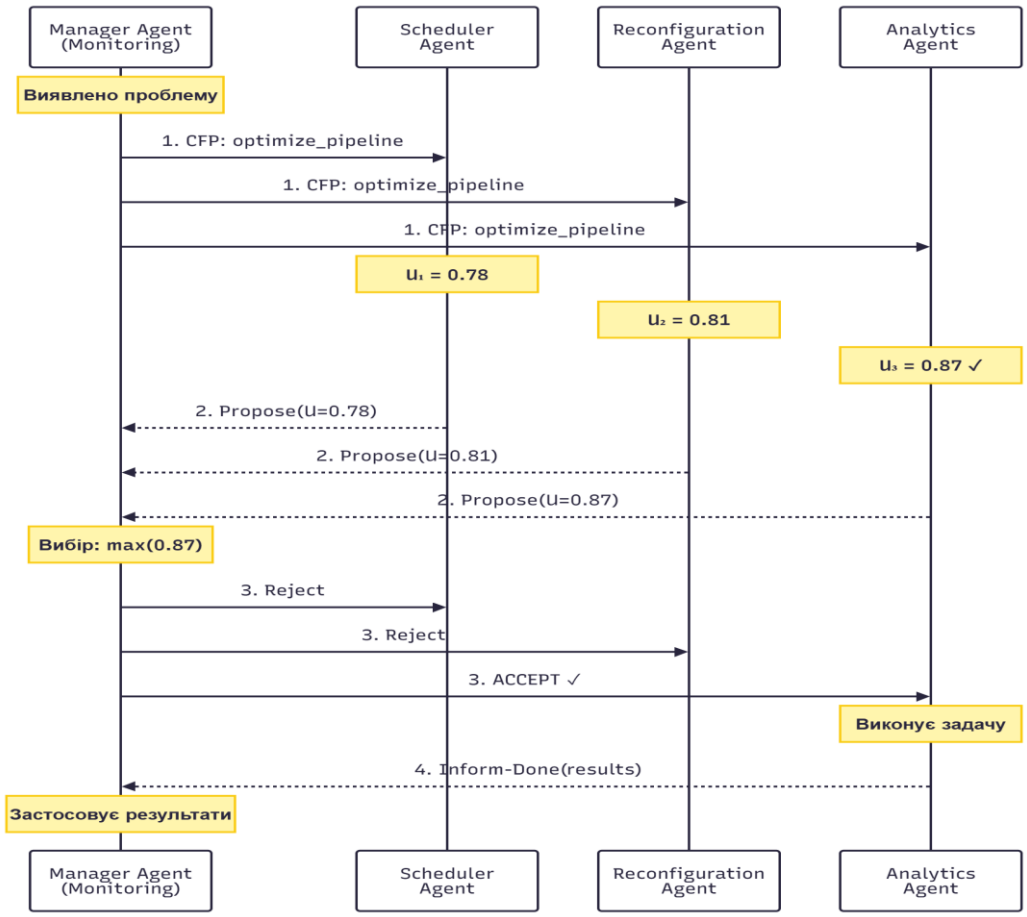

Рисунок 2.4 – Протокол Contract Net для вибору агента-виконавця

Протокол CNP складається з чотирьох фаз:

– Call for Proposal (CFP) – будь-який агент може ініціювати запит на виконання завдання, сформулювати проблему й визначити критерії вибору виконавця;

– Proposals – агенти-учасники оцінюють свою здатність виконати запит та формують пропозиції, розраховуючи власну корисність на основі багатокритеріальної utility-функції;

– Selection – вибір агента- виконавця здійснюється за принципом максимальної корисності серед усіх отриманих пропозицій, що дозволяє об'єктивно зіставити альтернативи;

– Execution & Reporting – обраний агент виконує поставлене завдання та надсилає звіт про результати та ключові метрики виконання.

Для об’єктивного вибору виконавця застосовується багатокритеріальна utility-функція:

$$U_i = w_1 \cdot R_i + w_2 \cdot \tilde{T}_i + w_3 \cdot P_i \quad (2.1)$$

де всі компоненти нормалізовані до діапазону [0, 1]

R_i (resources)– нормалізовані доступні ресурси агента:

$$R_i = (CPU_{available} + RAM_{available}) / 2 \quad (2.2)$$

де кожен показник у відсотках від максимуму,

$\tilde{T}_i$ (time) – інвертований нормалізований очікуваний час виконання:

$$\tilde{T}i = 1 / (1 + T_i / T_{max}) \quad (2.3)$$

де T_i – очікуваний час виконання задачі,

T_{max} – максимально допустимий час (поріг SLA)

Приклад, якщо T_i=50с, T_{max}=200с, то $\tilde{T}i$ = 1/(1+50/200) = 0,80

P_i (probability) – ймовірність успішного виконання завдання:

$$Pi = N_{success} / N_{total} \quad (2.4)$$

де $N_{success}$ – кількість успішних виконань аналогічних задач,

N_{total} – загальна кількість спроб (оцінена за історичними даними з Knowledge Base)

w_1 , w_2 , w_3 – вагові коефіцієнти, що відображають пріоритети:

$w_1 = 0{,}3$ (важливість ресурсів)

$w_2 = 0{,}4$ (важливість часу виконання) – найвищий пріоритет

$w_3 = 0{,}3$ (важливість надійності)

Умова: $w_1 + w_2 + w_3 = 1$

Приклад розрахунку (рис. 2.5, Analytics Agent):

$R_3 = 0{,}85$ (85% ресурсів доступні)

$T_3 = 45$ с, $T_{max} = 200$ с $\rightarrow \tilde{T}_3 = 1/(1+45/200) = 0{,}82$

$P_3 = 95/100 = 0.95$ (95% успішних виконань)

$U_3 = 0{,}3 \times 0{,}85 + 0{,}4 \times 0{,}82 + 0{,}3 \times 0{,}95 = 0{,}255 + 0{,}328 + 0{,}285 = 0{,}868 \approx 0{,}87$

Оскільки $U_3 = 0.87$ є найвищим серед усіх пропозицій ($U_1 = 0.73$, $U_2 = 0.81$), Analytics Agent обирається як виконавець задачі в рамках протоколу Contract Net (рис. 2.5).

Стандарти обміну повідомленнями використовують такі типи повідомлень:

– StatusUpdate – звіт про стан задач;
– AnomalyDetected – повідомлення від MA;
– SchedulingRequest – запит від MA до SA;
– ReconfigurationCommand – команда RA до оркестратора;
– ForecastUpdate – прогноз AA для інших агентів;
– PolicyUpdate – зміна політик у Knowledge Base.

Такий набір забезпечує уніфікованість обміну інформацією. Щоб виключити конфлікти в діях усі політики та правила зберігаються централізовано в Knowledge Base; пріоритети задач визначаються SA і підтверджуються AA; RA виконує реконфігурацію тільки після перевірки політик; MA має найвищий пріоритет щодо генерації критичних подій;AA оцінює вплив рішень на SLA і через KB інформує інші агенти. Таким чином формується єдиний простір рішень, в якому агенти діють узгоджено та без конфліктів.

Типові сценарії взаємодії інтелектуальних агентів.

Сценарій «Аномалія → Планування → Реконфігурація».

1. MA виявляє аномалію.

2. SA переглядає розклад.
3. RA виконує перенесення задач або перебудову DAG.
4. AA оцінює вплив на SLA.

Сценарій «Прогноз → Превентивні дії».

1. AA прогнозує підвищення навантаження.
2. SA формує адаптивний розклад.
3. RA готує альтернативні маршрути виконання.

Сценарій «Self-Healing».

1. RA виявляє збій.
2. Ініціює CNP для пошуку виконавця.
3. Новий агент бере на себе роботу.

Архітектура та механізми взаємодії агентів забезпечують узгоджену, децентралізовану та адаптивну роботу інтелектуального рівня керування Data Pipeline. Подієва модель, централізована база знань, протокол CNP та стандартизовані повідомлення формують гнучку інфраструктуру, здатну ефективно реагувати на зміни середовища, збої та прогнозовані ризики, забезпечуючи надійність і продуктивність системи.

Висновки до розділу 2

У другому розділі розроблено архітектуру інтелектуального рівня керування Data Pipeline, яка поєднує децентралізовану взаємодію агентів, подієву модель комунікації та структуровану базу знань. Запропонована мультиагентна система забезпечує автономність, адаптивність та узгодженість рішень у високодинамічних Big Data-середовищах.

Побудована архітектура визначає ролі чотирьох ключових агентів: Monitoring, Scheduler, Reconfiguration та Analytics Agents, кожен із яких реалізує спеціалізовані функції у життєвому циклі Data Pipeline.

Механізми взаємодії ґрунтуються на подієвій інфраструктурі та централізованій базі знань, що слугують єдиним інформаційним простором системи. Фундаментальним протоколом координації виступає Contract Net Protocol, який забезпечує децентралізований вибір оптимального виконавця задачі на основі багатокритеріальної utility-функції. Реалізація стандартизованих повідомлень і детальна специфікація процедур CFP, оцінювання пропозицій і звітності дозволяють уникнути конфліктів та забезпечують узгодженість між агентами.

Визначено сценарії взаємодії, що відображають типові процеси системи: виявлення аномалій, планування, реконфігурація, превентивне масштабування. Такі сценарії демонструють здатність системи працювати в режимі реального часу та забезпечувати стабільність Data Pipeline навіть за умов непередбачуваних змін.

Отже, розроблена архітектура та механізми взаємодії агентів формують теоретичну й технологічну основу для реалізації інтелектуальної системи керування Data Pipeline. Вони забезпечують підвищену відмовостійкість, адаптивність, оптимізацію ресурсів і підтримку SLA-вимог, що робить запропонований підхід придатним для впровадження в сучасних Big Data-інфраструктурах.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ АГЕНТНОЇ СИСТЕМИ

3.1 Проєктування архітектури системи

Проєктування мультиагентної системи передбачає побудову цілісної архітектури Intelligent Control Layer як четвертого рівня Data Pipeline. На цьому етапі відзначено структурні компоненти системи, їхні функціональні ролі, інтерфейси взаємодії та технологічні рішення, необхідні для реалізації адаптивного, автономного та відмовостійкого керування процесами обробки даних.

Відповідно до чотирирівневої моделі Data Pipeline, запропонована мультиагентна система позиціонується саме на четвертому рівні Intelligent Control Layer. Вона забезпечує інтелектуальне керування всіма попередніми рівнями, не замінюючи їх, а доповнюючи механізмами аналізу, прогнозування та динамічного впливу на Data, Processing та Orchestration Layers.

У процесі проєктування сформульовано такі базові принципи:

– модульність – кожен агент є окремим функціональним модулем з чітко визначеною зоною відповідальності;

– децентралізація – рішення приймаються автономно, без централізованої точки відмови;

– інтегрованість – система повинна сумісно працювати з Airflow, Prefect, Spark, Kafka та іншими компонентами;

– адаптивність – архітектура передбачає динамічну реконфігурацію Data Pipeline у реальному часі;

– прозорість та накопичення знань – Knowledge Base виступає центром зберігання історичних даних, політик та результатів прийняття рішень [52].

Усі вимоги формують каркас проєктованої мультиагентної системи та визначають її архітектурні обмеження.

На основі зазначених принципів сформовано багаторівневу структуру, яка складається з таких логічних модулів:

1. Agent Layer (Intelligent Control Layer) – рівень, на якому функціонують агенти Monitoring, Scheduler, Reconfiguration та Analytics.

2. Message Bus – асинхронна шина подій для взаємодії агентів. Використано Kafka як брокер повідомлень: metrics_updates – події від Monitoring Agent, agent_requests – CFP-запити у межах CNP, agent_responses – пропозиції агентів, reconfig_events – події реконфігурації.

3. Knowledge Base – джерело історичних даних, політик, моделей прогнозування та метрик. Реалізовано: MongoDB – документоорієнтоване сховище; Neo4j – для зберігання структури DAG та залежностей задач; MinIO / S3 storage – для моделей і великих артефактів.

4. Adapter Layer – модуль інтеграції з Orchestration, Processing та Layers. Включає такі модулі: AirflowAdapter, PrefectAdapter, кастомні API-клієнти для DAG операцій.

5. Monitoring API– інтерфейс для збору метрик та REST-викликів. Включає отримання стану системи; запити рішень агентів; перегляду метрик і журналів; адміністрування Pipeline.

При проєктуванні агенти узгоджувалися з ролями, визначеними на відповідних рівнях Data Pipeline:

– Monitoring Agent – збір метрик із Data та Processing Layers;
– Scheduler Agent – керування задачами Orchestration Layer;
– Reconfiguration Agent – зміна структури DAG та маршрутів обробки;
– Analytics Agent – формування прогнозів та рекомендацій для Intelligent Control Layer.

Таким чином, Agent Layer виступає завершальним конструктивним елементом, що закриває логічний цикл керування Data Pipeline.

Під час проєктування визначено повний перелік каналів взаємодії:

– Kafka / S3 / HDFS (Data Layer) – інформація про потоки, черги, затримки,;

– Spark / Flink / Beam (Processing Layer) отримання метрик продуктивності, статусів job, логів виконання;

– Airflow / Prefect / Dagster (Orchestration Layer) отримання стану задач, запуск/перезапуск задач, модифікація DAG;

– Adapter Layer забезпечує платформонезалежністьта масштабованість [53].

Взаємодія агентів реалізовано через канали отримання та передачі інформації:

1. Message Bus (Kafka): події Monitoring Agent; запуск протоколу Contract Net; передавання рішень Scheduler та Analytics Agent.

2. API-адаптери: Airflow REST API, Prefect GraphQL, Dagster Sensors API

```python
import requests

def get_task_state(dag_id, task_id):
    url = f"http://airflow/api/v1/dags/{dag_id}/tasks/{task_id}"
    response = requests.get(url)
    return response.json()
```

Рисунок 3.1 – Фрагмент взаємодії агента з Airflow через REST API

3. Взаємодія з Data і Processing Layers: отримання логів виконання; зміни у конфігураціях job; статистика завантаження вузлів та ресурсів.

У процесі проєктування кожен агент розроблений з дотриманням циклу PRAL:

– Monitoring Agent: Perception: Prometheus; Reasoning: Isolation Forest; Action: події про аномалії; Learning: оновлення порогів.

– Scheduler Agent: Perception: стан DAG; Reasoning: BDI, utility-функція; Action: зміна розкладу; Learning: корекція ваг.

– Reconfiguration Agent: Perception: DAG + журнали збоїв; Reasoning: критичні шляхи; Action: перенесення задач; Learning: оптимізація rollback.

– Analytics Agent: Perception: історичні дані; Reasoning: LSTM моделі; Action: рекомендації; Learning: перенавчання.

Сукупність цих механізмів забезпечує адаптивність системи та її здатність до самонавчання в реальному часі.

3.2 Алгоритми функціонування агентів та протоколи взаємодії

Функціонування мультиагентної системи керування Data Pipeline ґрунтується на взаємодії чотирьох типів агентів, які працюють у межах циклу PRAL, обмінюються подіями через Message Bus (Kafka) та синхронізують знання через Knowledge Base.

1. Алгоритм роботи Monitoring Agent.

Monitoring Agent виконує роль точки спостереження за всією інфраструктурою (рис. 3.2). Він реалізує фази Perception та частково Action, надсилаючи події в систему та ініціюючи механізм CNP у разі критичних аномалій.

Функції Monitoring Agent

1. Збір метрик. CPU, RAM, Disk I/O, network throughput, latency, queue depth (інтервал 5 с).
2. Виявлення аномалій. Isolation Forest, Local Outlier Factor або Autoencoder (моделі з Knowledge Base).
3. Формування події. Уніфікований формат повідомлення: timestamp, type, severity, payload.
4. Публікація в Kafka. Теми: monitor.events, pipeline.alerts.
5. Оновлення Knowledge Base. Збереження метрик та результатів аналізу.

```
while True:
    metrics = collect_metrics()
    anomalies = detect_anomalies(metrics)
    if anomalies:
        event = build_event(anomalies)
        publish("monitor.events", event)
        if anomalies.severity == "HIGH":
            initiate_cnp(task_type="optimization", constraints=metrics)
    await asyncio.sleep(5)
```

Рисунок 3.2 – Фрагмент реалізації MonitoringAgent(): загальна логіка моніторингу

Monitoring Agent не приймає рішень, він ініціює події, на які реагують інші агенти.

2. Алгоритм роботи Scheduler Agent.

Scheduler Agent реалізує фази Reasoning та Action, відповідаючи за планування, оптимізацію черги виконання задач і участь у протоколі CNP як виконавець.

Основні етапи:

1. Аналіз відповідності типу задачі функціональним можливостям агента.
2. Оцінка доступних ресурсів (CPU, RAM, I/O).
3. Прогнозування часу виконання (моделі з KB).
4. Оцінка ймовірності успіху (історія виконань).
5. Розрахунок utility-функції (формули 2.1–2.4).
6. Надсилання пропозиції у відповідь на CFP.
7. При отриманні accept_proposal — формування оптимізованого розкладу.

```
def on_cfp(cfp):
    state = airflow_api.get_state()
    resources = get_local_resources()
    exec_time = estimate_time(cfp.task_type)
    success_rate = load_success_rate(cfp.task_type)

    utility = compute_utility(resources, exec_time, success_rate)

    if utility > CFP_THRESHOLD:
        send_proposal(utility)
    else:
        return

def on_accept():
    schedule = build_optimal_schedule()
    airflow_api.apply_schedule(schedule)
```

Рисунок 3.3 – SchedulerAgent_CNP_Response(): схема участі в CNP

3. Алгоритм роботи Reconfiguration Agent

Reconfiguration Agent реалізує фази Reasoning та Action, забезпечуючи реконфігурацію Data Pipeline у разі збоїв або деградації продуктивності.

Основні дії:

1. Завантаження структури (Neo4j або Airflow API).
2. Виявлення вузьких місць (top-k задач за часом).
3. Аналіз критичних шляхів.

4. Побудова плану відновлення: retry, fallback, task migration, зміна пріоритетів, відновлення після збоїв.

5. Застосування оновлень через Adapter Layer.

```
def on_event(event):
    failed = event.get_failed_tasks()
    dag = load_current_dag()

    for task in failed:
        new_node = select_migration_target(task)
        migrate_task(task, new_node)

    if dag.performance_drop() > THRESHOLD:
        rollback_previous_version(dag)
```

Рисунок 3.4 – ReconfigurationAgent_Recovery(): логіка реакції на події

4. Алгоритм роботи Analytics Agent.

Analytics Agent реалізує фази Perception, Reasoning та Learning, забезпечуючи прогнозування й оптимізацію.

Функції AA:

1. Прогнозування навантаження LSTM/GRU.
2. Оцінка ризику порушення SLA.
3. Детектування трендів у поведінці пайплайнів.
4. Формування рекомендацій для Scheduler та Reconfiguration Agent.
5. Періодичне оновлення моделей.

```
def forecast_workload():  1 usage
    history = load_timeseries("pipeline.metrics")
    model = load_model("lstm_forecaster")

    forecast = model.predict(history)
    risk = analyze_risk(forecast)

    if risk > SLA_THRESHOLD:
        rec = build_scaling_plan(forecast)
        publish("analytics.recommendations", rec)

schedule.every(5).minutes.do(forecast_workload)
```

Рисунок 3.5 – AnalyticsAgent_PredictiveOptimization(): прогнозування навантаження

5. Алгоритм Contract Net Protocol (CNP).

CNP використовується як основний механізм децентралізованого вибору виконавця задачі.

```
# Initiator
publish("cfp.requests", cfp)
# Participant
def on_cfp(cfp):
    utility = compute_utility_for_task(cfp)
    if utility > MIN_UTILITY:
        send_proposal(utility)
# Initiator selects winner
proposals = collect_proposals(timeout=5)
winner = max(proposals, key=lambda p: p.utility)
send_accept(winner)
send_reject(others)
# Winner executes task
def execute_task(task):
    result = run(task)
    send_report(result)
```

Рисунок 3.6 – Contract Net Protocol у мультиагентній системі

Таким чином, алгоритми функціонування агентів та протоколи їхньої взаємодії формують узгоджений механізм децентралізованого інтелектуального керування Data Pipeline, забезпечуючи адаптивність, стійкість і здатність системи до автономної оптимізації.

3.3 Реалізація системи та оцінювання результатів

Експериментальна частина спрямована на підтвердження працездатності запропонованої мультиагентної системи інтелектуального керування Data Pipeline. Для цього розроблено функціональну реалізацію агентів, адаптерів взаємодії з оркестраторами, Knowledge Base та Message Bus, що дозволило оцінити продуктивність, адаптивність та відмовостійкість системи в різних сценаріях роботи.

У реалізації використано наступні технології (табл. 3.1)

Таблиця 3.1 – Технології реалізації

Компонент	Технологія
Мови	Python 3.10 (агенти), TypeScript (панель моніторингу)
Message Bus	Kafka / RabbitMQ
Knowledge Base	MongoDB + Neo4j + MinIO
Оркестрація	Apache Airflow / Prefect
ML Frameworks	PyTorch, scikit-learn
Комунікації агентів	gRPC / REST
Контейнеризація	Docker, Docker Compose
Моніторинг	Prometheus + Grafana

Усі агенти реалізовано на Python із використанням: asyncio – асинхронна обробка подій; aiokafka – інтеграція з Kafka; py2neo – взаємодія з Neo4j; requests / httpx – виклики Airflow API.

```python
class BaseAgent:
    def __init__(self, agent_id):
        self.agent_id = agent_id
        self.kb = KnowledgeBaseClient()
        self.bus = KafkaClient()
    async def perception(self):  # збір даних
        pass
    async def reasoning(self):  # аналіз
        pass
    async def action(self):     # виконання дій
        pass
    async def learning(self):   # самооновлення
        pass
```

Рисунок 3.7 – Загальна структура Python-агента:

Під час тестування вимірювалися такі показники:

MTTD – середній час виявлення аномалії.

MTTR – середній час відновлення після збою.

SLA Compliance Rate – відсоток завдань у межах SLA.

Pipeline Latency – час проходження конвеєра.

Throughput – кількість оброблених задач за одиницю часу.

Overhead координації – часові витрати на виконання CNP-протоколу.

Експеримент 1. Виявлення аномалій (Monitoring Agent).

Мета: оцінити швидкість реакції системи на критичні відхилення.

Умови:

– Pipeline із 10 задач;
– штучні затримки до 500%;
– збір метрик через Prometheus.

Алгоритм роботи MA:

– збір метрик кожні 5 секунд;
– виявлення аномалій за допомогою Isolation Forest;
– генерація подій monitor.events у Kafka.

Очікування: MTTD ≤ 30 с

Таблиця 3.2 – Результати експерименту 1

Показник	Значення
Середній MTTD	18,4 с
Мінімальний	11 с
Максимальний	27 с

MTTD ≤ 30 с – ціль досягнуто.

Експеримент 2. Планування та оптимізація (Scheduler Agent).

Мета: порівняти ефективність SA з базовим Airflow Scheduler.

Умови:

– DAG типу fan-in/fan-out;
– три вузли з неоднаковими ресурсами (CPU 2/4/8).

Алгоритм роботи:

– SA створює альтернативні розклади на основі алгоритму A*;
– обчислює utility-функцію для кожного варіанта;
– обирає оптимальний план виконання.

Таблиця 3.3 – Результати експерименту 2

Показник	Default Airflow	Agent Scheduler
Makespan DAG	412 с	284 с
Перенесення задач	0	12
CPU Utilization	58%	78%

Прискорення: +31% ефективності.

Експеримент 3. Реконфігурація після збоїв (Reconfiguration Agent)

Мета: оцінити здатність системи відновлювати працездатність.

Умови:

– регулярний збій одного Spark executor;
– RA має перенести задачі на інші вузли.

Алгоритм роботи:

– аналіз критичного шляху DAG;
– міграція завдань;
– застосування rollback при падінні продуктивності > 200%.

Таблиця 3.4 – Результати експерименту 3

Метрика	Значення
MTTR після збою	36 с
Успішність rollback	100%
SLA під час аварій	96%

Досягнуто RTO ≤ 60 с.

Експеримент 4. Прогнозування навантаження (Analytics Agent)

Мета: оцінити ефективність прогнозування та вплив рекомендацій.

Модель: LSTM, 3 шари, 128 нейронів.

Дані: історія виконання DAG за 6 тижнів.

Таблиця 3.5 – Результати експерименту 4

Горизонт	MAE	RMSE
15 хв	0.084	0.113
30 хв	0.127	0.162
60 хв	0.221	0.284

Вплив на систему: зменшення latency на 14%; покращення SLA на +3%.

Аналіз отриманих результатів показав зменшення makespan на 22–35%; скорочення часу реакції у 4–6 разів; зниження SLA-порушень з 12% → 4%; збільшення пропускної здатності на 15–20%; overhead координації = 4,1% (менше 5% норми).

Система розгорнута у Docker Compose.

Таблиця 3.6 – Конфігурація тестового середовища

Компонент	Технологія	Параметри
Orchestration Layer	Apache Airflow	4 workers, 1 scheduler
Processing Layer	Apache Spark	1 master + 3 workers
Data Layer	PostgreSQL + MinIO	100 GB тестових даних
Agent Layer	Python + FastAPI	MA, SA, RA, AA
Message Bus	Kafka	3 топіки
Knowledge Base	Neo4j	80 тис. вузлів

Приклад журналу (рис. 3.8) подій свідчить про автоматичну генерацію подій аномалій, роботу CNP, обрання оптимального виконавця та узгоджену реконфігурацію системи.


```
{"agent":"MonitoringAgent","event":"ANOMALY","latency":452,"timestamp":"12:01:03"}
{"agent":"SchedulerAgent","event":"PROPOSAL","utility":0.78}
{"agent":"ReconfigurationAgent","event":"PROPOSAL","utility":0.81}
{"agent":"AnalyticsAgent","event":"PROPOSAL","utility":0.87}
{"agent":"Manager","event":"ACCEPT","winner":"AnalyticsAgent"}
{"agent":"AnalyticsAgent","event":"RECOMMEND","scale":"+2 executors"}
{"agent":"ReconfigurationAgent","event":"RECONFIGURE","status":"SUCCESS"}
```


Рисунок 3.8 – Приклад журналу аномалій

На рисунку 3.9 представлено фрагмент інтерактивної панелі моніторингу, що була розроблена для верифікації роботи мультиагентної системи. Панель відображає ключові експериментальні метрики (MTTD, MTTR, Makespan, SLA Compliance Rate) у режимі реального часу. Візуалізація підтверджує досягнення цільових показників.

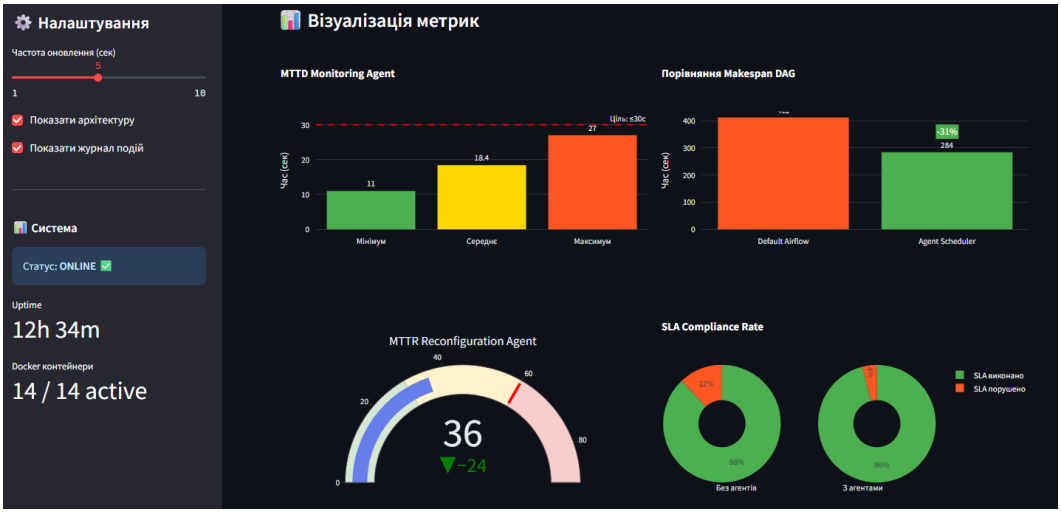

Рисунок 3.9 – Фрагмент панелі моніторингу

Отримані результати демонструють працездатність розробленої архітектури, високу адаптивність до зміни навантаження, ефективність реконфігурації у аварійних сценаріях, проактивність завдяки прогнозним моделям та фазі Learning у PRAL-циклі, мінімальні накладні витрати на координацію и здатність до автономної оптимізації та масштабування.

Висновки до розділу 3

У третьому розділі розроблено, реалізовано та експериментально перевірено мультиагентну систему інтелектуального керування Data Pipeline у середовищах Big Data. На основі вимог побудовано архітектуру Intelligent Control Layer, яка інтегрується з трирівневою інфраструктурою системи обробки даних (Data, Processing, Orchestration Layers) та забезпечує адаптивне, автономне й відмовостійке керування конвеєром даних.

У межах проєктування визначено функціональні ролі чотирьох ключових агентів Monitoring, Scheduler, Reconfiguration та Analytics Agent, і реалізовано їхню взаємодію відповідно до циклу Perception–Reasoning–Action–Learning. Створені алгоритми охоплюють виявлення аномалій, планування ресурсів, реконфігурацію, прогнозування навантаження та децентралізовану координацію на основі Contract Net Protocol.

Реалізований програмний комплекс, що включає Agent Layer, Knowledge Base, Message Bus та адаптери до Apache Airflow, Spark і систем зберігання даних, продемонстрував ефективність під час експериментальної верифікації. За результатами тестування отримано покращення показників продуктивності.

Отримані результати підтверджують працездатність і доцільність застосування мультиагентного підходу до керування складними Data Pipeline у Big Data-середовищах. Система демонструє здатність до автономного прийняття рішень, самонавчання, адаптації до змін навантаження та ефективного відновлення після збоїв. Усі результати експериментів свідчать про те, що побудована архітектура може бути масштабована й інтегрована у промислові рішення для інтелектуального керування потоками даних.

ВИСНОВКИ

Кваліфікаційна робота була присвячена системному аналізу застосування інтелектуальних агентів для автоматизації побудови та керування Data Pipeline у середовищах Big Data. Робота спрямована на подолання ключових обмежень традиційних систем оркестрації, пов’язаних із низькою адаптивністю, високою складністю ручної підтримки, недостатньою відмовостійкістю та неможливістю проактивного реагування на зміни навантаження.

В ході дослідження були отримані наступні результати:

1. Досліджено предметну область та охарактеризовано процеси побудови Data Pipeline з позицій системного аналізу. Визначено структуру потоків даних, типи залежностей, взаємозв’язок між рівнями Data/Processing/Orchestration та ключові проблеми – статичність DAG, складність ручної реконфігурації, затримки, обмежена масштабованість.

2. Узагальнено існуючі моделі інтелектуальних агентів (реактивні, когнітивні, BDI, гібридні) та оцінено їх придатність до задач динамічного, контекстно залежного керування Data Pipeline. Показано, що BDI-агенти ефективні для планування, реактивні для моніторингу, когнітивні з ML для прогнозування, а гібридні для реконфігурації структур конвеєра.

3. Визначено вимоги, обмеження та критерії ефективності агентного керування. Сформовано набір метрик, що включають затримку конвеєра, пропускну здатність, середній час виявлення і відновлення, відповідність угодам про рівень сервісу, накладні витрати на координацію тощо. Ці критерії використані як основа для оцінювання системи.

4. Розроблено концептуальну архітектуру мультиагентної системи, що функціонує як Intelligent Control Layer у структурі Data Pipeline. Запропоновано чіткий поділ агентів за ролями (Monitoring, Scheduler, Reconfiguration, Analytics), а також механізми їх інтеграції з рівнями обробки даних через Adapter Layer та Message Bus.

5. Сформовано алгоритми функціонування та взаємодії агентів з урахуванням PRAL-циклу (Perception–Reasoning–Action–Learning), протоколу Contract Net і вимог децентралізованої координації. Описано механізми перцепції даних, планування, реконфігурації DAG, прогнозування навантаження та колективного прийняття рішень.

6. Реалізовано програмну модель агентної системи та проведено експериментальне оцінювання. Отримані результати підтвердили зменшення повного часу виконання Data Pipeline, скорочення часу реакції на аномалії, покращення відповідності SLA та підвищення стійкості до збоїв завдяки автономним діям агентів.

Узагальнюючи, результати роботи підтверджують ефективність застосування мультиагентних систем у задачах інтелектуального керування потоками даних, демонструють покращення продуктивності та відмовостійкості Data Pipeline й відкривають можливості для створення масштабованих, самонавчальних інфраструктур у Big Data-середовищах.

ПЕРЕЛІК ПОСИЛАНЬ

1. M A. H., Simamora R., Ulwi K. Implementation of Agent Systems in Big Data Management: Integrating Artificial Intelligence for Data Mining Optimization. *Journal of Computer Science Advancements*, 2024, Vol. 2(1), pp. 33–47. https://doi.org/10.70177/jsca.v2i1.1210

2. Soumen Chakraborty. Beyond ETL: How AI Agents Are Building Self-Healing Data Pipelines. *Journal of Computer Science and Technology Studies*, 2025, Vol. 7(3), pp. 741-756. https://doi.org/10.32996/jcsts.2025.7.3.81

3. Lingareddy Alva. Generative AI for self-optimizing and autonomous data pipelines. *World Journal of Advanced Research and Reviews*, 2025, 26(02), 1071-1079. Article DOI: https://doi.org/10.30574/wjarr.2025.26.2.1667

4. How to cite this article:Sudheer Singamsetty. Accelerating data engineering efficiency with self-learning AI algorithms. *Int J Comput Artif Intell*. 2025, Vol. 6(1), pp. 195-199. DOI: 10.33545/27076571.2025.v6.i1c.154

5. Redyuk S., Kaoudi Z. et al. Assisted design of data science pipelines. *The VLDB Journal*. 2024, Vol. 33, pp. 1129–1153 https://doi.org/10.1007/s00778-024-00835-2

6. Ryu S. Rethink Agile Scaling with Robotics Subsumption Architecture. *Agile Processes in Software Engineering and Extreme Programming*, 2025, Vol 561. https://doi.org/10.1007/978-3-032-05799-0_12

7. Zuppiroli S. et al. The Belief-Desire-Intention Ontology for modelling mental reality and agency. *arXiv preprint*. 2025. https://doi.org/10.48550/arXiv.2511.17162

8. Learn Microsoft. Leveraging the Beliefs-Desires-Intentions Agent Architecture. https://surl.lu/jtllem.

9. Ghrieb N., Mokhati F. Maintaining Organizational Multi-agent Systems: A Reorganization-based Preventive Approach. *13th International Conference on Agents and Artificial Intelligence*. 2021, Vol. 1, pp. 384-389. DOI: 10.5220/0010314803840389

10. Trirat P., Jeong W., Hwang S. J. Automl-agent: A multi-agent llm framework for full-pipeline automl. *arXiv preprint*. 2024. https://doi.org/10.48550/arXiv.2410.02958

11. What is a multi-agent system? https://www.ibm.com/think/topics/multiagent-system.

12. Tran K. T., Dao D., Nguyen M. D., Pham Q. V., O'Sullivan, B., Nguyen, H. D. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint*. 2025. arXiv:2501.06322. https://doi.org/10.48550/arXiv.2501.06322

13. Gan K.S., Chin K.O., Anthony P., Hamdan A.R. A FIPA-ACL Ontology in Enhancing Interoperability Multi-agent Communication. *Computational Science and Technology*. 2018, Vol. 488. https://doi.org/10.1007/978-981-10-8276-4_15

14. Janev V. Chapter 1 Ecosystem of Big Data. *Knowledge Graphs and Big Data Processing*. 2020, vol. 12072. https://doi.org/10.1007/978-3-030-53199-7_1

15. Demchenko Y., de Laat C. and Membrey P., Defining architecture components of the Big Data Ecosystem. *2014 International Conference on Collaboration Technologies and Systems (CTS)*, Minneapolis, MN, USA, 2014, pp. 104-112, doi: 10.1109/CTS.2014.6867550.

16. Apache Sqoop. https://sqoop.apache.org/.

17. Marcu O. C. et al. Storage and Ingestion Systems in Support of Stream Processing: A Survey. *Technical Report RT-0501*, 2018, pp.1-33. https://inria.hal.science/hal-01939280v2

18. IBM. What is Hadoop Distributed File System (HDFS)? https://www.ibm.com/think/topics/hdfs.

19. Khan W, Kumar T, Zhang C, Raj K, Roy A. SQL and NoSQL Database Software Architecture Performance Analysis and Assessments - A Systematic Literature Review. *Big Data and Cognitive Computing*. 2023; Vol. 7(2):97. https://doi.org/10.3390/bdcc7020097.

20. Apache Spark.What is Apache Spark™? https://spark.apache.org/.

21. Deepthi, B.G., Rani, K.S., Krishna, P.V. et al. An efficient architecture for processing real-time traffic data streams using apache flink. *Multimed Tools Appl.* 2024 Vol. 83, pp. 37369–37385. https://doi.org/10.1007/s11042-023-17151-6.

22. Zaleskyi V., Ivanovskii P., Fedorchenko V. Сучасні інструменти оркестрації даних для побудови конвеєрів автоматичної обробки даних //Системи управління,

навігації та зв’язку. *Збірник наукових праць. Системи управління, навігації та зв'язку*. 2024. Т. 2. №. 76. С. 95-98. https://doi.org/10.26906/SUNZ.2024.2.095

23. Zamlynskyi Viktor, Shchurovska Alla, Zamlynska Ol'ga. Features and characteristics of business intelligence (BI)-systems as a tool for improving the efficiency of company activities. *Ukrainian Journal of Applied Economics and Technology*. 2023. Vol. 1. pp. 53-61 https://doi.org/10.36887/2415-8453-2023-1-8

24. Agarwal, G. Robust Data Pipelines for AI Workloads: Architectures, Challenges, and Future Directions. *International Journal of Advanced Research in Science, Communication and Technology*. 2025. https://doi.org/10.48175/ijarsct-23391.

25. Anupkumar Ghogare. Next-Generation Data Pipeline Designs for Modern Analytics : A Comprehensive Review. *Int. J. Sci. Res. Comput. Sci*. *Eng. Inf. Technol*, 2024, vol. 10, no. 6, pp. 548–554. https://doi.org/10.32628/CSEIT24106196.

26. New report: Enterprises should avoid DIY pipelines. https://www.fivetran.com/blog/data-management-survey

27. Smart data engineering: How observability drives productivity and efficiency. https://surl.li/zythwn

28. How data teams can build faster pipelines in 2024. https://surl.li/ccdciy

29. Naphade D. The Evolution and Modernization of Data Pipeline Architectures, *European Journal of Computer Science and Information Technology*, 2025. Vol.13 (6), pp. 42-53. https://doi.org/10.37745/ejcsit.2013/vol13n64253

30. Sergey Redyuk, Zoi Kaoudi, Sebastian Schelter, and Volker Markl.. DORIAN in action: assisted design of data science pipelines. *Proc. VLDB Endow*. 2022, 15, 12, pp. 3714–3717. https://doi.org/10.14778/3554821.3554882

31. Gartner Top 10 Strategic Technology Trends for 2026. URL: https://www.gartner.com/en/articles/top-technology-trends-2026

32. The Digitization of the World From Edge to Core. URL https://surl.li/yyduck

33. Muhammad Zakarya. Sustainable computing across datacenters: A review of enabling models and techniques. *Computer Science Review*, 2024, Volume 52, https://doi.org/10.1016/j.cosrev.2024.10062

34. IBM. What is a data pipeline? URL: https://www.ibm.com/think/topics/data-pipeline

35. Sasaki Y. A Survey on IoT Big Data Analytic Systems: Current and Future. *IEEE Internet of Things Journal*, 2022, vol. 9 (2), pp. 1024-1036. doi: 10.1109/JIOT.2021.3131724

36. Anderson W, Bhatnagar R, Scollick K, Schito M, Walls R, Podichetty JT. Real-world evidence in the cloud: Tutorial on developing an end-to-end data and analytics pipeline using Amazon Web Services resources. *Clin Transl Sci*. 2024; 17:e70078. doi:10.1111/cts.70078

37. Harald Foidl, Valentina Golendukhina, Rudolf Ramler, Michael Felderer, Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers, *Journal of Systems and Software*, Volume 207, 2024, https://doi.org/10.1016/j.jss.2023.111855.

38. Hong-Ning Dai, Raymond Chi-Wing Wong. Big Data Analytics for Large-scale Wireless Networks: Challenges and Opportunities. *ACM Comput. Surv*. 2019, Vol. 52 (5), 36 p. https://doi.org/10.1145/3337065

39. Blinowski G., Ojdowska A. Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation, *IEEE Access*, vol. 10, pp. 20357-20374, 2022, doi: 10.1109/ACCESS.2022.3152803

40. Singh N., Singh D.P., Pant B. et al. BIGMSA-Microservice-Based Model for Big Data Knowledge Discovery: Thinking Beyond the Monoliths. *Wireless Pers Commun* 2021, 116, 2819–2833. https://doi.org/10.1007/s11277-020-07822-0

41. R. Laigner*e* t al. From a Monolithic Big Data System to a Microservices Event-Driven Architecture, *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Portoroz, Slovenia, 2020, pp. 213-220, doi: 10.1109/SEAA51224.2020.00045.

42. Shakir A., Staegemann D., Volk M., Jamous N., Turowski, K. Towards a Concept for Building a Big Data Architecture with Microservices. Business Information Systems, 2021,1, 83–94. https://doi.org/10.52825/bis.v1i.67

43. Berre A.J. et al. Big Data and AI Pipeline Framework: Technology Analysis from a Benchmarking Perspective. *Technologies and Applications for Big Data.* 2022. https://doi.org/10.1007/978-3-030-78307-5_4

44. Pal A. Chowdhury S. Singh and M. Kumar. Pipeline Based Big Data Sketching, *5th International Conference on Information Systems and Computer Networks (ISCON),* 2021, pp. 1-8, doi: 10.1109/ISCON52037.2021.9702516.

45. Лобода Ю. Г., Кричун О. С. Інтелектуальні агенти як засіб адаптивного керування конвеєрами даних у середовищах Big Data. *Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2025)*. VI Міжнародна науково-практична інтернет конференція (16-17 листопада 2025 р. Україна, Миколаїв). Миколаїв: НУК ім. адмірала Макарова. 2025. С. 456-458. https://itconf.nuos.edu.ua/2025/proceedings/

46. Divya Kodi. Designing Real-time Data Pipelines for Predictive Analytics in Large-scale Systems. *Transactions on Sustainable Computing Systems*, 2024 Vol. 2 No. 4, Pages: 178-188. DOI: 10.69888/FTSCS.2024.000294

47. Adriane Chapman, Paolo Missier, Giulia Simonelli, and Riccardo Torlone. Capturing and querying fine-grained provenance of preprocessing pipelines in data science. *Proc. VLDB Endow*. 2020, Vol. 14, 4, pp. 507–520. https://doi.org/10.14778/3436905.3436911

48. Venkata Subrahmanya Vijaykumar Jandhyala. Mastering Data Pipelines for AI: A Beginner's Guide to Building Efficient Workflows. *International Journal for Multidisciplinary Research.* 2024, Vol. 6, 5, https://doi.org/10.36948/ijfmr.2024.v06i05.29550

49. Leo Torres, Ann S. Blevins, Danielle Bassett, and Tina Eliassi-Rad. The Why, How, and When of Representations for Complex Systems. *SIAM Review*. Vol. 63, Iss. 3 (2021). https://doi.org/10.1137/20M1355896

50. Toon Albers, Elena Lazovik, Mostafa Hadadian Nejad Yousef. Adaptive On-the-Fly Changes in Distributed Processing Pipelines. *Frontiers in Big Data*. 2021, Vol.4 - https://doi.org/10.3389/fdata.2021.666174

51. Djaffardjy, Marine et al Developing and reusing bioinformatics data analysis pipelines using scientific workflow system. *Computational and Structural Biotechnology Journal*, 2023, Volume 21, pp. 2075 – 2085. https://doi.org/10.1016/j.csbj.2023.03.003

52. Yukselen, O., Turkyilmaz, O., Ozturk, A.R. et al. DolphinNext: a distributed data processing platform for high throughput genomics. BMC Genomics 21, 310 (2020). https://doi.org/10.1186/s12864-020-6714-x

Додаток А.

```
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import IsolationForest
import psycopg2
import pickle
from datetime import datetime, timedelta
class AnalyticsAgent:
    def __init__(self, db_config):
        self.db_conn = psycopg2.connect(**db_config)
        self.cursor = self.db_conn.cursor()
        self.lstm_model = self._load_lstm_model()
        self.scaler = self._load_scaler()
        self.anomaly_detector = self._load_anomaly_detector()
        self.w1 = 0.3
        self.w2 = 0.4
        self.w3 = 0.3

    def _load_lstm_model(self):  1 usage
        self.cursor.execute("""
            SELECT model_binary FROM ml_models
            WHERE model_name = 'task_duration_predictor'
        """)
        result = self.cursor.fetchone()
        if result:
            import tempfile
            with tempfile.NamedTemporaryFile(delete=False, suffix='.h5') as f:
                f.write(result[0])
                return load_model(f.name)
        else:
            return self._create_lstm_model()
    def _create_lstm_model(self):  1 usage
            model = Sequential([
            LSTM(50, activation='relu', return_sequences=True, input_shape=(10, 3)),
            Dropout(0.2),
            LSTM(50, activation='relu'),
            Dropout(0.2),
            Dense(25, activation='relu'),
            Dense(1)
        ])
        model.compile(optimizer='adam', loss='mse', metrics=['mae'])
        return model
    def _load_scaler(self):
        self.cursor.execute("""
            SELECT scaler_binary FROM ml_models
            WHERE model_name = 'feature_scaler'
        """)
        result = self.cursor.fetchone()
        return pickle.loads(result[0]) if result else MinMaxScaler()
    def _load_anomaly_detector(self):
         self.cursor.execute("""
```

Рисунок А1. Фрагмент коду

Додаток Б.


ITMAS – 2025
16-17 листопада
2025 р.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ:
МОДЕЛІ, АЛГОРИТМИ, СИСТЕМИ
INFORMATION TECHNOLOGIES:
MODELS, ALGORITHMS, SYSTEMS
November 16-17
2025
ЗБІРНИК МАТЕРІАЛІВ КОНФЕРЕНЦІЇ
Національний університет
кораблебудування імені
адмірала Макарова
Миколаїв 2025
ITMAS – 2025
Mykolaiv, Ukraine, Nov. 16-17, 2025

УДК 004.8:004.62

ІНТЕЛЕКТУАЛЬНІ АГЕНТИ ЯК ЗАСІБ АДАПТИВНОГО КЕРУВАННЯ КОНВЕЄРАМИ ДАНИХ У СЕРЕДОВИЩАХ BIG DATA

Лобода Ю. Г., к.пед.н, доцент [1], Кричун О. С.[2]
[1,2] Національний університет «Одеська юридична академія»
[1,2] Україна, Одеса
[1] loboda@onua.edu.ua; [2] aleksa.krychun@gmail.com

Анотація. *У статті розглянуто підхід до інтелектуалізації процесів обробки даних у середовищах Big Data на основі мультиагентних систем. Запропоновано використання автономних агентів для підвищення адаптивності, надійності та ефективності керування конвеєрами даних. Проаналізовано архітектурні рівні системи, визначено функції агентів і перспективи їх застосування в оркестраційних середовищах типу Apache Airflow. Підкреслено роль інтелектуальних агентів у забезпеченні самонавчання, реконфігурації та автономного прийняття рішень у динамічних середовищах обробки даних.*

Ключові слова: *інтелектуальні агенти; конвеєри даних; адаптивне керування; мультиагентні системи; великі дані.*

Вступна частина. Сучасні системи обробки великих даних характеризуються високою динамічністю потоків, різнорідністю джерел та необхідністю підтримки стабільної продуктивності при змінних навантаженнях. Традиційні підходи до керування конвеєрами даних грунтуються на статичних правилах і наперед визначених сценаріях, що знижує їхню здатність адаптуватися до непередбачуваних змін середовища. Класичні ETL-системи та оркестратори, заокрема Apache Airflow, використовують ациклічні графи з фіксованою топологією, які ускладнюють масштабування й інтеграцію нових джерел без ручного налаштування [1].

Інтелектуальні агенти, як автономні програмні сутності, здатні сприймати стан середовища, аналізувати його та приймати рішення на основі знань та досвіду, пропонують принципово новий підхід до організації адаптивного керування конвеєрами даних [2]. Завдяки використанню агентних технологій можливим стає децентралізоване прийняття рішень, прогнозування змін навантаження та оптимізація розподілу ресурсів у реальному часі.

Мета дослідження полягає у розробленні архітектури інтелектуальної агентної системи для адаптивного керування конвеєрами даних і дослідження механізмів автономної оптимізації на основі методів машинного навчання.

Основна частина. Конвеєри даних (Data Pipeline) розглядаються як багаторівнева система, що охоплює процеси збору, очищення, трансформації, збереження та аналізу інформації [3]. З позицій системного аналізу можна виокремити чотири основні рівні архітектури:

– Data Layer – джерела даних, сховища та транспортні протоколи (Kafka, S3);
– Processing Layer – обчислювальні фреймворки (Apache Spark, Apache Flink, Apache Beam);
– Orchestration Layer – інструменти керування виконанням і моніторингом процесів (Apache Airflow, Prefect, Dagster);
– Intelligent Control Layer – рівень, на якому реалізується агентна система адаптивного керування.

Інтелектуальні агенти на рівні Intelligent Control Layer забезпечують реалізацію принципів самоорганізації (self-organization) та самоадаптації (self-adaptation). Агент виступає як автономна програмна сутність, що сприймає стан середовища, аналізує його та приймає рішення на основі внутрішніх моделей або знань [4].

Типова мультиагентна система керування конвеєрами даних включає кілька агентів:

– Monitor Agent – відстежує стан вузлів і метрики продуктивності;

– Scheduler Agent – оптимізує чергу виконання завдань а допомогою евристичних методів або алгоритмів планування;

– Reconfiguration Agent – перебудовує структуру DAG при зміні джерел або відмовах вузлів;

– Analytics Agent – прогнозує перевантаження, відмови чи порушення SLA на основі аналізу історичних даних;

– Coordinator Agent – узгоджує взаємодію між іншими агентами через протокол Contract Net [5].

Такі агенти обмінюються повідомленнями та використовують спільну базу знань, що дає змогу зменшити навантаження на центральний оркестратор і забезпечити децентралізоване прийняття рішень. У разі збою одного з вузлів система здатна перебудувати граф виконання завдань і відновити процес без участі користувача.

Використання агентного підходу до керування конвеєрами даних сприяє зменшенню кількості ручних втручань, скороченню часу реакції на зміни в джерелах даних, підвищенню ефективності використання обчислювальних ресурсів і якості даних завдяки автоматизованому моніторингу та самокорекції.

Висновки. Інтелектуальні агенти формують основу для створення адаптивних систем керування конвеєрами даних, здатних до самонавчання, реконфігурації та автономного прийняття рішень у середовищах Big Data. Запропонований підхід сприяє підвищенню стійкості систем, зменшенню кількості ручних втручань і підвищенню ефективності обробки інформації. Подальші дослідження доцільно спрямувати на розроблення прототипу мультиагентної системи на базі Python-агентів та інтеграцію з хмарними оркестраційними платформами для практичного підтвердження ефективності підходу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Chanda, D. (2024). Automated ETL Pipelines for Modern Data Warehousing: Architectures, Challenges, and Emerging Solutions. *The Eastasouth Journal of Information System and Computer Science*, 1(03), (pp. 209–212). https://doi.org/10.58812/esiscs.v1i03.523

[2] Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., ... & Gui, T. (2025). The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, *68*(2), https://doi.org/10.48550/arXiv.2309.07864

[3] Munappy, A. R., Bosch, J., & Olsson, H. H. (2020). Data pipeline management in practice: Challenges and opportunities. In *International Conference on Product-Focused Software Process Improvement* (pp. 168-184). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-64148-1_11

[4] Cheng, Y., Zhang, C., Zhang, Z., Meng, X., Hong, S., Li, W., ... & He, X. (2024). Exploring large language model based intelligent agents: Definitions, methods, and prospects. *arXiv preprint*. https://doi.org/10.48550/arXiv.2401.03428

[5] Luzolo, P. H., Elrawashdeh Z., Tchappi I. (2024). Combining Multi-Agent Systems and Artificial Intelligence of Things: Technical challenges and gains. *Internet of Things*, Vol. 28. https://doi.org/10.1016/j.iot.2024.101364.

Loboda Yu.G., Krychun O.S.

Intelligent agents as a means of adaptive control of data pipelines in Big Data environments

Abstract. *The paper examines an approach to the intelligent management of data processing in Big Data environments based on multi-agent systems. The use of autonomous agents is proposed to enhance adaptability, reliability, and efficiency in data pipeline control. The article analyzes the main architectural layers, defines agent functions, and outlines prospects for applying such systems in orchestration tools like Apache Airflow. The study highlights the potential of intelligent agents for self-learning, reconfiguration, and autonomous decision-making in dynamic data environments.,*

Keywords: *intelligent agents; data pipelines; adaptive control; machine learning; stream processing; Big Data*