Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА

## Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель мультиагентного керування розподілом завдань на обчислювальні ресурси

(тема)


Виконав:

студент II курсу, групи СПм-20-1

Ведмеденко І.О.
(прізвище, ініціали)


Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Філімончук Т.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

# ЗАВДАННЯ

## НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Ведмеденко Івану Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель мультиагентного керування розподілом завдань на обчислювальні ресурси

затверджена наказом по університету від “ 05 ” листопада 2021 р. № 1657 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2021 р.

3. Вхідні дані до роботи планувальники GridWay та DAGMan; перелік основних складових мультиагентних систем; показники роботи мультиагентних систем: якість обслуговування черги завдань, час та надійність їх виконання; балансування навантаження обчислювальних ресурсів в розподілених обчислювальних системах

4. Перелік питань, що потрібно опрацювати у роботі
1. Аналіз існуючих мультиагентних систем
2. Огляд систем управління потоками завдань в розподіленому обчислювальному середовищі
3. Структура мультиагентної системи
4. Балансування навантаження обчислювальних ресурсів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Огляд систем управління потоками завдань в | 09.11.21-12.11.21 | |
| | розподіленому обчислювальному середовищі | | |
| 2 | Огляд складових мультиагентної системи | 13.11.21-18.11.21 | |
| 3 | Розробка моделі поведінки агентів | 19.11.21-22.11.21 | |
| 4 | Розробка алгоритмів функціонування агентів | 23.11.21-29.11.21 | |
| | та перерозподілу ресурсів середовища | | |
| 5 | Проведення експерементів | 30.11.21-03.12.21 | |
| 6 | Оформлення матеріалів кваліфікаційної роботи | 04.12.21-07.12.21 | |
| 7 | Подання кваліфікаційної роботи керівникові | 08.12.21-09.12.21 | |
| | та її попередній захист | | |
| 8 | Подання кваліфікаційної роботи на рецензу- | 10.12.21-11.12.21 | |
| | вання | | |

Дата видачі завдання 08 листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Філімончук Т.В.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 73 с., 21 рис., 10 табл., 1 дод., 53 джерела.

МУЛЬТИАГЕНТНА СИСТЕМА, РОЗПОДІЛЕНА ОБЧИСЛЮВАЛЬНА СИСТЕМА, ВІРТУАЛЬНА СПІЛЬНОТА, ІМІТАЦІЙНА МОДЕЛЬ, АГЕНТ, ДІЯ, ЗАПИТ, ЗАВДАННЯ, ОБЧИСЛЮВАЛЬНИЙ РЕСУРС.

Мета кваліфікаційної роботи полягає в розробці нових інструментальних засобів, які забезпечують зниження трудовитрат при побудові мультиагентної системи в порівнянні з існуючими інструментами, а також моделей та алгоритмів роботи агентів, що дозволять поліпшити показники (якість обслуговування черги завдань, час та надійність їх виконання, балансування завантаження обчислювальних ресурсів) управління потоками обчислювальних завдань в різнорідних розподілених обчислювальних системах в порівнянні з відомими планувальниками.

# ABSTRACT

Master’s thesis: 73 pages, 21 figures, 10 tables, 1 appendice, 53 sources.

MULTIAGENT SYSTEM, DISTRIBUTED COMPUTER SYSTEM, VIRTUAL COMMUNITY, SIMULATION MODEL, AGENT, ACTION, REQUEST, TASKS, COMPUTING RESOURCE.

The purpose of the qualification work is to develop new tools that reduce labor costs in building a multi-agent system compared to existing tools, as well as models and algorithms of agents that will improve performance (quality of service queue, time and reliability, load balancing computing resources) management of flows of computing tasks in heterogeneous distributed computing systems in comparison with known planners.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ВС – віртуальна спільнота

МАС – мультиагентна система

РВС – розподілена обчислювальна система

ACL – стандарт взаємодії агентів (англ., Agent Communication Language)

AppLes – мультиагентна система (англ., Application Level Scheduling)

ARCOL – мова для роботи в системі ARTIMIS (англ., ARTIMIS COmmunication Language)

ARTIMIS – інформаційна система керування розподілом завдань (англ., The Advanced Regional Traffic Interactive Management and Information System)

FIPA – стандарт взаємодії агентів (англ., Foundation for Intelligent Physical Agents)

KQML – мультиагентна система (англ., Knowledge Query and Manipulation Language)

MAAG – мультиагентна система (англ., Multi-Agent Architecture for Grid Environment)

MAGE – мультиагентна система (англ., Mobile Agent-based Grid Environment)

# ВСТУП

На даний час ведуться активні дослідження, які пов’язані із створенням та застосуванням високопродуктивних обчислювальних систем різного призначення на основі парадигм Grid та хмарних обчислень. Результати цих досліджень представлені в роботах українських та зарубіжних вчених. У зв’язку з високою інтенсивністю потоків обчислювальних завдань, які надходять на вхід системи, необхідно ефективне та гнучке управління ними на метарівні.

Ефективність управління полягає в забезпеченні високої якості обслуговування черг завдань, підвищенні надійності виконання завдань, мінімізації часу вирішення задач, підтримки рівномірного балансування завантаження обчислювальних ресурсів та досягнення інших заданих показників. Гнучкість управління полягає в раціональному надання ресурсів, які необхідні для виконання завдань, в системах, що динамічно змінюються.

У процесі управління потоками таких завдань потрібен детальний облік та узгодження критеріїв користувачів, які визначаються специфікою завдань, що вирішуються, і переваг власників обчислювальних ресурсів, що випливають з їх характеристик. Як правило, розподілена обчислювальна система (РОС), яка інтегрує моделі Grid та хмарні обчислення, має низку властивостей, які в свою чергу істотно ускладнюють уніфікацію процесу управління завданнями. До властивостей такого роду відносяться, наприклад, архітектурно-функціональна різнорідність, неповнота опису та динамічність ресурсів, широта спектру завдань, що вирішуються за допомогою цих обчислювальних ресурсів, наявність різних категорій користувачів, які мають свої цілі та завдання експлуатації обчислювального середовища.

Відомі на даний час моделі, методи, алгоритми та програмні засоби централізованого управління потоками завдань не вирішують перераховані вище проблеми повністю. Як правило, це обумовлено наступними причинами [1]:

- обмеженим контролем централізованої системи над її розподіленими ресурсами, що не дозволяє в повній мірі оцінити стан цих ресурсів та впливати на нього в повній мірі;

- відсутністю необхідної інформації про різнорідні ресурси (ступеня їх надійності, показниках продуктивності, обчислювальної історії виконання конкретних завдань, дисциплінах обслуговування черг, адміністративних політик та інших важливих характеристиках) або істотним збільшенням накладних витрат при отриманні таких відомостей, які в свою чергу призводить до зниження ефективності функціонування системи управління;

- недостатньою надійністю самої системи управління, так як при відмові центрального вузла все обчислювальне середовище стає непрацездатним;

- різким зниженням продуктивності системи при значному збільшенні числа завдань потоку.

У зв’язку з цим виникає необхідність розробки нової ефективної та гнучкої системи управління потоками завдань. Для вирішення даної проблеми доцільно використання принципів організації розподіленого групового управління. Якість управління, яке здійснюється окремими компонентами, забезпечується наявністю в них більш повних локальних знань про ресурси, що використовуються ними, в порівнянні з централізованою системою.

Групове управління потоками завдань в процесі вирішення складної прикладної задачі в РОС здійснюється за допомогою передачі повідомлень в комунікаційному середовищі між компонентами розподіленої керуючої системи. З огляду на динамічну природу РОС, доцільно застосовувати адаптивне надання ресурсів в процесі управління потоками завдань. В цьому випадку перспективним підходом є використання мультиагентних технологій. В рамках такого підходу окремі ресурси представляються спеціалізованими додатками (агентами), що утворюють в сукупності мультиагентну систему (МАС) управління.

Аналіз результатів досліджень в області самоорганізації обчислювальних систем показує, що ефективне управління потоками завдань за допомогою МАС досягається за рахунок застосування алгоритмів роботи агентів, що адаптуються в процесі їх виконання до поточних умов та стану функціонування середовища відповідно до заданої агентам мети, визначеними критеріями якості рішення задачі та використання ресурсів, а також знаннями про особливості предметних областей завдань, що вирішуються.

Побудова проблемно-орієнтованої самоорганізуючої МАС породжує ряд проблем, які пов’язані з автоматизацією розробки агентів та агентних платформ, реалізації алгоритмів функціонування агентів, а також накопичення та застосування предметних знань агентами. Існуючі в даний час інструменти для створення МАС не дозволяють вирішити проблеми, що були перераховані вище, в повній мірі [2].

Мета кваліфікаційної роботи полягає в розробці нових інструментальних засобів, які забезпечують зниження трудовитрат при побудові МАС в порівнянні з існуючими інструментами, а також моделей та алгоритмів роботи агентів, що дозволять поліпшити показники (якість обслуговування черги завдань, час та надійність їх виконання, балансування завантаження обчислювальних ресурсів) управління потоками обчислювальних завдань в різнорідної РОС в порівнянні з відомими планувальниками (брокерами) GridWay [3] та Condor Directed Acyclic Graph Manager (DAGMan) [4].

Основні задачі для досягнення поставленої мети:

- проведення порівняльного аналізу відомих методів та засобів організації МАС, а також відомих методів та засобів мультиагентного управління потоками обчислювальних завдань;
- розробка моделі поведінки агентів;
- розробка алгоритмів функціонування агентів;
- розробка інструментального комплексу побудови МАС для

управління потоками обчислювальних завдань;

- створення МАС управління потоками завдань в експериментальній РОС;

- оцінка надійності та ефективності управління потоками завдань за допомогою розробленої МАС, а також трудомісткості її побудови.

Об’єктом дослідження є процес управління потоками обчислювальних завдань в РОС. Предметом дослідження виступають мультиагентні моделі, алгоритми та система управління потоками завдань в різнорідній РОС, а також інструментальні засоби побудови МАС. Методами дослідження виступають методи концептуального, імітаційного та автоматного програмування, організації розподілених обчислень та мультиагентних технологій, а також методи машинного навчання.

# 1 АГЕНТНО-ОРІЄНТОВАНИЙ ПІДХІД ДО УПРАВЛІННЯ ПОТОКАМИ ЗАВДАНЬ

## 1.1 Мультиагентна система

Під агентом розуміється апаратна або програмна сутність, яка здатна діяти в інтересах досягнення цілей, що були поставлені перед ним власником і/або користувачем. Агент здатний приймати самостійні рішення, навчатися, знаходити інших агентів для взаємодії та об’єднання з ними в співтоваристві. Ці властивості агентів дозволяють реалізувати складний механізм, завдяки якому вони (агенти) можуть автономно існувати в своєму агентному середовищі та виконувати свої обов’язки відповідно до ролей, які їм було призначено.

Агент може мати наступні основні властивості [5-7]:

- автономність – здатність функціонувати без втручання людини, здійснюючи самоконтроль над своїми діями та внутрішнім станом;
- громадську поведінку – здатність функціонувати в співтоваристві агентів, обмінюючись повідомленнями за допомогою деякої мови комунікацій;
- реактивність – здатність сприймати стан середовища та своєчасно реагувати на його зміни;
- наявність знань – володіти інформацією про себе, про середовище та про інших агентів, а також про правила застосування рішень на основі цієї інформації;
- переконання – володіти знаннями про недостовірне зовнішнє середовище та про інших агентів, які можуть змінюватися в часі та ставати невірними;
- бажання – стани, досягнення яких є для агента бажаним. Вони можуть бути суперечливими, але агент може обирати тільки їх несуперечливу підмножину;

- наміри – тим, що агент зобов’язаний зробити у відповідності зі своїм вибором або в силу своїх зобов’язань по відношенню до інших агентів;

- мету – конкретну множину кінцевих та проміжних станів, досягнення яких агент вважає реалізацією своїх намірів;

- зобов’язання – тими завданнями, які агент бере на себе на прохання або за дорученням інших агентів в рамках кооперативних цілей;

- проактивність – здатністю діяти в попереджувальної манері, зокрема, генерувати нові цілі та діяти раціонально для їх досягнення, а не тільки реагувати на події;

- мобільність – здатність агента мігрувати по мережі Інтернет в пошуках необхідної інформації або сервісів;

- доброзичливість – готовність агентів допомагати один одному, що передбачає відсутність у агентів конфліктуючих цілей;

- правдивість – властивість агента маніпулювати інформацією, про яку йому завідомо відомо, що вона є помилковою;

- раціональність – властивість агента діяти так, щоб досягти своїх цілей, а не уникати їх досягнення в рамках своїх знань та переконань.

Класифікація агентів в залежності від їх властивостей [8, 9] наведена в таблиці 1.1. Слід зазначити, що в даний час агенти, які використовуються на практиці, як правило, не перевищують рівня кмітливих агентів [8, 10].

МАС є мережею слабо пов’язаних вирішувачів приватних проблем (агентів), які існують в загальному середовищі та взаємодіють між собою для досягнення тих чи інших цілей системи [11].

Як правило, МАС має такі властивості [12-15]:

- кожен агент має неповну інформацію про середовище, має власну модель зовнішнього середовища та обмежені можливості щодо вирішення «своєї» проблеми;

- глобальне управління агентами обмежена або відсутня;

- дані, які використовуються агентами, децентралізовані та частина їх може бути «власністю» окремих агентів;

- агенти функціонують в асинхронному режимі;

- агенти координують свої дії шляхом обміну повідомленнями на мові високого рівня.

Таблиця 1.1 – Характеристики основних типів агентів

| Характеристики | Simple | Smart | Intelligent | Truly intelligent |
|---|---|---|---|---|
| Автономне виконання | + | + | + | + |
| Взаємодія з іншими агентами і/або користувачами | + | + | + | + |
| Стеження за оточенням (реактивність) | + | + | + | + |
| Здатність використання абстракцій | | + | + | + |
| Здатність використання предметних знань | | + | + | + |
| Здатність адаптивної поведінки для досягнення мети | | | + | + |
| Навчання з оточення | | | + | + |
| Толерантність до помилок і/або невірним вхідним сигналам | | | | + |
| Виконання в реальному часі | | | | + |
| Взаємодія з користувачами на природних мовах | | | | + |

Розглянемо ключові відмінні властивості агентів МАС в порівнянні з традиційними об’єктно-орієнтованими системами [13].

Об’єкт є пасивною програмної сутністю, його методи викликаються ззовні, а об’єкт виконує запит безумовно (детермінована реакція). Якщо об’єкт каже «ні» (тобто не реагує на запит), то це виняткова ситуація, що вимагає спеціальної обробки. У той же час звернення до агента – це «прохання», яке він може і не виконувати на основі прийнятих в МАС угод про взаємні зобов’язання в залежності від свого внутрішнього стану та ін., тобто це вирішується агентом (потенційним виконавцем), що в свою чергу є

недетермінованою реакцією. Тому агент є активною сутністю. Крім того, об’єкти не можуть ініціювати взаємодію та мігрувати по мережі, на відміну від агентів. Для взаємодії з об’єктом є тільки один метод на кожен тип вхідного повідомлення. Формат повідомлення повинен строго дотримуватися та не підтримується асинхронний режим. Зміст вхідного повідомлення агента має більш вільну форму у вигляді рядка на мові високого рівня.

Комунікації агента зазвичай асинхронні та підтримують паралельну обробку. Агент може підтримувати відразу кілька діалогів, вирішуючи самостійно питання про черговість участі в них (в залежності від свого внутрішнього стану). Він сам вирішує, як чергувати діалоги з вирішенням завдань (коли переходити від «зовнішньої» поведінки до «внутрішньої»).

Важливою властивістю є здатність до навчання. Об’єкт створюється класом і його здатності далі не змінюються, при цьому агент може бути забезпечений здатністю до навчання.

З точки зору характеру поведінки можна розглянути наступні відмітні ознаки: об’єкт не може реагувати сам на події зовнішнього світу. Слід зауважити, що останні версії UML [16] та Java [17] забезпечують можливість «слухати» зовнішнє середовище. Поведінка об’єкта строго визначена, агент же може не тільки неоднозначно реагувати на специфічні запити, представлені у вхідному повідомленні, але також і самостійно реагувати на події в зовнішньому світі використовуючи механізм «підписки» і/або інформацію від сенсорів, а також приймати рішення про вибір тієї чи іншої поведінки.

Агент може запустити процес за власною ініціативою на відміну від об’єкту. Об’єкт не може оголошувати свої інтерфейси, об’єктно-орієнтовані мови дозволяють тільки запитати об’єкт про його інтерфейси. Агент може оголошувати свої сервіси та їх інтерфейси, використовуючи механізм сервісів білих та жовтих сторінок. У зв’язку з цим МАС добре підходять для реалізації посередницьких послуг в процесі обробки потоків завдань, коли агенти наділяються правами та повноваженнями власників ресурсів та їх

користувачів, які вирішують свої завдання за допомогою даних обчислювальних ресурсів. Здатність МАС до самоорганізації робить процес управління більш ефективним та гнучким [13].

Самоорганізація МАС – це динамічний адаптивний процес, що приводить до виникнення та підтримки структур агентів та їх локальних взаємодій без зовнішнього втручання [18]. Основними відмітними рисами самоорганізованих процесів (агентів) мультиагентного управління є наступні ознаки [19, 20]:

- автономність – бути в стані взаємодії зі світом, керувати власною поведінкою, яка спрямована на досягнення своїх локальних цілей без втручання ззовні;

- бути здатними сприймати зовнішнє середовище та локально впливати на нього;

- мати програмне середовище для розподіленої взаємодії з іншими агентами;

- глобальний порядок виникає в системі тільки завдяки локальній взаємодій її компонент;

- наявність емерджентних властивостей, які відсутні на рівні окремих агентів та виявляються тільки на рівні системи в цілому в процесі взаємодії агентів;

- нелінійна динаміка взаємодії агентів, нестійкість та чутливість до варіацій початкового стану та малим варіаціям параметрів системи;

- множинність стійких станів системи;

- надмірність забезпечує нечутливість до відмов або руйнувань окремих елементів;

- адаптивність – це здатність системи, що самоорганізується, змінювати свою поведінку та переходити в новий стійкий стан зі зміною організаційної структури;

- складність системи обумовлена великим числом компонентів, що взаємодіють;

- простота правил, які використовуються компонентами системи в процесах взаємодії, що сприяють складній поведінки системи в цілому;

- ієрархічна структура системи описується на двох рівнях, а саме, на рівні локальних компонент системи і на метарівні, де проявляються емерджентні властивості системи.

Технологія MAC задовольняє базовим вимогам для програмної реалізації систем, що самоорганізуються [21], при цьому вона єдиною технологією, яка має всі засоби для реалізації таких систем. З цієї причини теорія та практика самоорганізації зараз є областю досліджень та розробок, головним чином в області MAC [13].

## 1.2 Мультиагентні системи управління потоками завдань в розподіленому обчислювальному середовищі

На даний час існує широкий спектр MAC, що застосовуються на практиці для обробки потоків завдань в РОС. У їх числі такі системи: GridSolve [22], AppLes [23], MAGE [24], Singh Framework [25], MAAG [26], Condor-G [27].

Система GridSolve призначена для об’єднання розподілених обчислювальних ресурсів за допомогою локальної мережі, в тому числі персональних комп’ютерів. GridSolve грає роль клієнта, сервера, агента та здатний за допомогою віддаленого виклику процедур звертатися до апаратних та програмних компонентів. Тобто він в свою чергу є проміжним програмним забезпеченням. З 2008 року ця система на жаль не підтримується.

AppLeS – це адаптивна система планування на рівні додатків, яка може застосовуватися до Grid-середовищі. Кожна програма, що представлена в Grid-середовищі, може мати ряд власних додатків. Філософія розробки AppLeS полягає в тому, що всі аспекти продуктивності та використання даної системи розглядаються з точки зору програми, яка використовує обрану

систему. Для досягнення продуктивності додатка AppLeS вимірює продуктивність програми на конкретному ресурсі сайту та використовує цю інформацію для подальшого прийняття рішень по вибору обчислювальних ресурсів та планування. Отже, проект AppLeS надає методологію, прикладне програмне забезпечення та програмне середовище для адаптивної диспетчеризації та поширенню додатків в багатокористувацькому різнорідному Grid-середовищі.

MAGE – це система моніторингу Grid-середовища для надання динамічної багаторівневої архітектури. Система MAGE є легковажним засобом організації мобільних агентів для отримання незалежних типів обміну повідомленнями та інших актуальних завдань. Кожен елемент програми може бути замінений новим завданням без втручання в інші завдання, так як агенти представлені сервісами в MAGE.

Основним призначенням платформи Singh Framework є побудова MAC для семантичних веб-сервісів. Вона об’єднує агентів, що працюють на різних рівнях MAC, та розширює можливості по їх взаємодії. Рівні MAC характеризуються різним набором знань агентів, їх можливостями та цілями функціонування. Як правило, перерахованим вище системам характерна наявність агентів, що вбудовані, моніторинг середовища та розподіл ресурсів. Тільки в систему MAGE вбудована можливість навчання агентів.

Архітектура MAAG створювалася з урахуванням властивостей різнорідності обчислювальних ресурсів та додатків в Grid-середовищах. MAAG включає п’ять типів агентів, які здійснюють спільне управління розподіленими ресурсами в Grid-середовищі та підтримують гібридні потоки завдань для забезпечення встановлених критеріїв якості обслуговування додатків. У MAAG кожен сервіс контролюється певним агентом. Але активна розробка даної системи припинена з 2009 року.

Проект Condor-G розробляється групою вчених університету Вісконсіна в Мадісоне та є набором відкритого програмного забезпечення для організації великомасштабних обчислень з інтенсивним потоком завдань.

Цей проект застосовується для управління завантаженням кластерів виділених вузлів, а також для запуску обчислень на персональних комп'ютерах в момент їх простою. Клієнт Condor-G працює під управлінням операційних систем на базі Unix, Linux, FreeBSD, Mac OS X та Microsoft Windows. Condor-G здатний об'єднувати в єдине обчислювальне середовище як виділені вузли, так і невиділені вузли.

Проведений аналіз існуючих МАС дозволяє зробити ряд висновків. Підтримка управління в рамках як локальної системи (наприклад, обчислювального кластера), так і глобальної системи (наприклад, Grid-системи), реалізована лише в системі MAAG. Централізований та децентралізований алгоритми взаємодії агентів, управління на рівні додатків з метою оптимізації роботи із завданнями для конкретного додатка представлені лише в системах Condor-G та GridSolve. На жаль, ряд функціональних можливостей, таких як забезпечення підвищеної надійності обміну повідомленнями між агентами за рахунок використання системи логічного часу, можливість виконання декількох ролей одним агентом, навчання агентів, застосування агентами економічних механізмів управління та ін., які необхідні для забезпечення ефективного та гнучкого управління потоками завдань, часто відсутня в подібних системах.

## 1.3 Аналіз методів та засобів організації мультиагентних систем

Використання загальноприйнятого набору правил взаємодії агентів є важливим аспектом при розробці МАС [28], що забезпечує відкритість, надійність та гнучкість комунікаційних процесів [29]. На даний час основними стандартами взаємодії агентів виступають KQML [30], FIPA-ACL [31], ARCOL [28], COOL [32].

Самим широко застосовуваним мовою взаємодії агентів є KQML. У ньому використовуються вирази-дії, такі як відповісти, сказати, скасувати, сказати зворотне та ін. Кожне повідомлення KQML містить вираз-дію та

додаткові відомості, що записані в різні поля. Безлічі виразів-дій в KQML та їх слотів цілком достатньо, щоб дозволити агентам взаємодіяти між собою. Недоліком виступає те, що відомі проблеми з семантикою виразів-дій та з різною інтерпретацією цих виразів агентами.

Стандарт FIPA-ACL виник багато в чому під впливом ARCOL. FIPA-ACL, ARCOL та KQML спільно встановили перед-стандарт мови взаємодії агентів. Синтаксис та семантика FIPA-ACL дуже схожі на синтаксис та семантику KQML.

Для системи ARTIMIS була розроблена спеціальна мова ARCOL, яка володіє меншим набором комунікаційних примітивів, ніж мова KQML.

Мова COOL заснована на взаємодії на основі мовних актів, що націлені на чітке уявлення та застосування знань в мультиагентних системах. Дана мова фокусується на управлінні переговорами на основі низки правил (правила переговорів, правила помилок, правила продовження).

Загальною проблемою при розробці МАС є вибір конкретного набору програмних засобів [33]. На даний момент відомий широкий спектр агентних платформ та інструментальних засобів розробки агентів та МАС [28]. Більшість з них розробляється на мові програмування Java. Однак тільки окремі системи затребувані для вирішення практичних завдань. Аналіз цієї області говорить що розробка багатьох систем припинена, інші системи мають вузьку спеціалізацію та відповідно обмежену сферу застосування, а деякі системи не відповідають стандартам (наприклад, FIPA, KQML). Важливим фактором затребуваності інструментальних засобів є їх вільне поширення, наявність документації до них, а також підтримка спільнотою розробників (наприклад, форум розробників) [34].

Для кожної з наведених систем розглянемо наступні властивості:

- сумісні ОС;
- ліцензія та вартість застосування;
- відповідність стандартам роботи та обміну повідомленнями;
- мобільність агентів, що розроблюються;

- складність освоєння та використання;

- масштабованість та продуктивність;

- мови програмування, що підтримуються.

Розглянемо ряд найбільш популярних інструментаріїв.

Agent Factory [35] – це середовище з відкритим набором засобів, платформ та мовних засобів підтримки розробки та поширення МАС. Даний засіб умовно поділено на дві сфери розповсюдження агентів – ноутбуки, комп'ютери, сервери та окремо – мобільні пристрої. Основне призначення середовища – це побудова агентних систем загального призначення. Розробником цього середовища заявлений University College Dublin (Дублін, Ірландія). Популярність середовища середня, масштабованість хороша.

AgentBuilder [36] є інтегрованим інструментальним засобом швидкої розробки інтелектуальних агентів та агентно-орієнтованих додатків. AgentBuilder дозволяє зменшити час та вартість розробки агентів за рахунок наявності великої кількості шаблонів агентів та бібліотек, що вбудовані. Основним призначенням засобу розробки виступає побудова агентних систем загального призначення. Розробником заявлений Acronymics Inc (Альма, США). Популярність AgentBuilder середня, масштабованість хороша. Однак на даний момент часу проект завершений та не розвивається.

AgentScape [37] являє собою набір програмних засобів розробки моделей агентно-орієнтованих систем загального призначення та засобів візуалізації роботи моделей. Розробка моделей із застосуванням AgentScape не вимагає написання великих обсягів програмного коду. Проект здебільшого дослідний, стандарти та API схильні до частих змін. Розробляється інструментарій на мові Java під відкритою ліцензією. Основне призначення даного проекту – це масштабовані розподілені агентні системи. Розроблявся даний продукт компанією Delft University of Technology (Делфт, Нідерланди), користувався низькою популярністю, але гарною масштабністю. Даний проект припинив свій розвиток з 2010 року.

Архітектура агентів Cougaar [38] розроблена під відкритою ліцензією

та включає базові сервіси та необхідні елементи інфраструктури функціонування агентів. Робота агентів управління обчисленнями заснована на декомпозиції великих завдань. Агенти керують поведінкою додатків, а середовище відповідає за адаптацію системи. Агенти та середовище можуть розроблятися незалежно один від одного, але виконуються завжди разом. Для розробників доступні засоби написання власних плагінів та додаткових модулів для розширення функціональних можливостей агентів Cougaar. Основне призначення цього продукту – це комбіновані, масштабовані розподілені додатки. Розробниками заявлені Raytheon Bolt, Beranek and Newman Technologies (Вьенна, США). Популярність даного інструментарію низька, але масштабованість висока.

Проект CybelePro [39] надає користувачам надійну високопродуктивну інфраструктуру для швидкої розробки та розгортання великомасштабних, високопродуктивних МАС, та в свою чергу є комерційним продуктом. Інфраструктура агентів CybelePro широко застосовується в уряді, промисловості та науковій сфері для військової логістики, моделювання, управління наземним та повітряним транспортом, комунікаційними мережами та в розробці відкритих систем. Основне призначення даного інструментарію – це масштабовані розподілені агентні системи. Розробником заявлений Intelligent Automation Inc (Роквілл, США). Популярність CybelePro низька, але масштабованість висока.

EMERALD [40] – це програмна платформа для семантичних веб-сервісів, яка орієнтована на знання. Дана платформа призначена для уніфікації процесу спілкування агентів та обміну знаннями в різнорідних інформаційних системах, що представлені у вигляді веб-сервісів. Одним з додатків EMERALD є вивчення процесу навчання агентів під час проведення торгів від імені користувачів. У EMERALD агенти можуть обмінюватися між собою наборами правил без необхідності узгодження формату представлення даних правил з іншими агентами. Агент, що приймає, може використовувати зовнішні сервіси для аналізу семантичного набору правил. У EMERALD

вбудована підтримка таких мов, як XML та Prolog. Область призначення даної платформи – це розподілені додатки, що складаються з автономних сутностей. Розробниками виступають Logic Programming and Intelligent Systems Group, Aristotle University of Thessaloniki (Салоніки, Греція). EMERALD користується низькою популярністю, але має високу масштабованість.

Система GAMA [41] призначена для моделювання та розробки імітаційного середовища для побудови агентних моделей за допомогою спеціальної високорівневої мови GAML. GAMA є системою загального призначення, може знайти застосування в широкому спектрі предметних областей. Але її основне місце – це масштабне розподілене просторово-точне агентне моделювання. Розробниками заявлені Institute of Research for Development, Mathematical and Computer Modelling of Dynamical Systems (Ханой, В'єтнам). Інструментарій не користується високою популярністю, проте має гарну масштабованість.

JADE [42] – це програмне середовище розробки МАС та додатків, яке написано на мові Java та виконується на віртуальній машині Java (JVM). Дане середовище спрощує розробку МАС у вигляді проміжного програмного забезпечення, повністю сумісного з FIPA. У JADE вбудовано набір графічних інструментів, які необхідні при налагодженні та розгортанні агентів. Розроблені за допомогою JADE системи можуть мігрувати між платформами МАС, що розміщені на різних вузлах. JADE включає в себе набір стандартних агентів для управління агентами МАС, що конструюються, та бібліотеку стандартних класів для Java-програм. Деякі інструментарії використовують засоби JADE. Основне призначення даного середовища – це розподілені додатки, що складаються з автономних сутностей. Розробником заявлений Telecom Italia (Рим, Італія). Програмне середовище користується найвищою популярністю та високою масштабованість.

JADEx [43] – це модель переконань, бажань та намірів, вона є добре відомою агентной архітектурою, яка полегшує опис поведінки, цілей, планів

та винагород агентів. Суть ідеї в тому, щоб забезпечити явне розходження між метою та тим, як ця мета досягнута. Такий поділ допомагає по ряду причин: з одного боку, дозволяє розробляти поведінку агентів в зручному та зрозумілому вигляді, а з іншого боку – поведінка стає більш явною після того, як виконано цілі та досягнуто плани. Основне призначення даної моделі – це розподілені додатки, що складаються з автономних сутностей в рамках моделі переконань, бажань та намірів. Розробником заявлений Hamburg University (Гамбург, Німеччина). Дана модель користується високою популярністю і має високу масштабованість.

MaDKit [44] – це мультиагентне середовище розробки, яке написано на мові Java, що дозволяє легко проектувати розподілені додатки та проводити моделювання з застосуванням мультиагентного підходу. Однією з ключових особливостей є застосування моделі «Агент-Група-Роль», яка не прив’язана ні до однієї визначеної моделі, тобто агент відіграє свою роль всередині групи, утворюючи віртуальні спільноти. Крім того, MaDKit не втручається у внутрішню структуру агентів, що дозволяє розробнику вільно створювати власну архітектуру агентів. Основним призначенням MaDKit є мультиагентні системи з агентним моделюванням. Розробником заявлений Institut universitaire de technologie (Монпельє, Франція). MaDKit користується середньою популярністю та має хорошу масштабованість.

Виходячи з порівняльних результатів, наведених у таблиці 1.2, можна зробити висновок, що найбільш придатними для організації МАС EMERALD, GAMA, JADE, JADEx. Вони знаходяться в активній розробці, відповідають стандартам, є кросплатформними, мають відкритий вихідний код, можуть бути застосовними до широкого спектру завдань. Незважаючи на те, що GAMA задовольняє більшості вимог розробників та має зручний інтерфейс користувача, вона має значний недолік: для розробки агентной системи необхідно освоювати мову GAML, що ускладнює застосування цієї платформи спільно з популярними мовами програмування. Ця система більше підходить для агентного моделювання.

Таблиця 1.2 – Ключові характеристики поширених агентних платформ

| Інструментарій | Сумісні ОС | Платність | Відповідність стандартам | Обмін повідомленнями |
|---|---|---|---|---|
| Agent Factory | будь-яка ОС з JVM | безкоштовна | частково FIPA | HTTP |
| AgentBuilder | Windows, Linux, Sun Solaris | платна | KQML | KQML, TCP/IP |
| AgentScape | будь-яка ОС з JVM | безкоштовна | невідомо | внутрішня мова платформи |
| Cougaar | Windows, Linux | безкоштовна | невідомо | Cougaar Message Transport Service |
| CybelePro | будь-яка ОС з JVM | платна | невідомо | внутрішня мова платформи |
| EMERALD | будь-яка ОС з JVM | безкоштовна | FIPA, Semantic web standards | Асінхрона ACL |
| GAMA | Windows, Linux, Mac OS | безкоштовна | FIPA, Geographic Information Systems | ACL |
| JADE | будь-яка ОС з JVM | безкоштовна | FIPA | Асінхрона ACL, HTTP |
| JADEx | будь-яка ОС з JVM | безкоштовна | FIPA, Web Services Description Language | HTTP |
| MaDKit | будь-яка ОС з JVM | безкоштовна | Unified Modeling Language | P2P |

Системи Agent Factory, EMERALD, JADE та JADEx в цілому близькі за порівнюваним характеристикам, але при цьому популярність у EMERALD досить низька, Agent Factory не в повній мірі відповідає стандартам FIPA та KQML. Це пов’язано з тим, що одним з бажаних критеріїв є можливість роботи агентів на мобільних платформах, таким вимогам задовольняють лише JADE [42, 45] та JADEx [43, 46]. Обидві ці платформи частіше за інших використовуються для розробки мультиагентних систем [47, 48].

Загальним недоліком розглянутих систем є слабка прив’язка до предметної області задачі, що розв’язується, при розробці МАС.

# 2 МОДЕЛЬ МУЛЬТИАГЕНТНОГО КЕРУВАННЯ ПОТОКАМИ ЗАВДАНЬ

## 2.1 Структура мультиагентної системи

Кваліфікаційна робота спрямована на розробку моделі, яка орієнтована на автоматизацію створення та застосування агентів для МАС із заданою організаційною структурою (рисунок 2.1). Її ієрархічна структура може містити два або більше рівнів функціонування агентів. На кожному рівні можуть функціонувати агенти, які грають різні ролі та відповідно виконують різні функції.

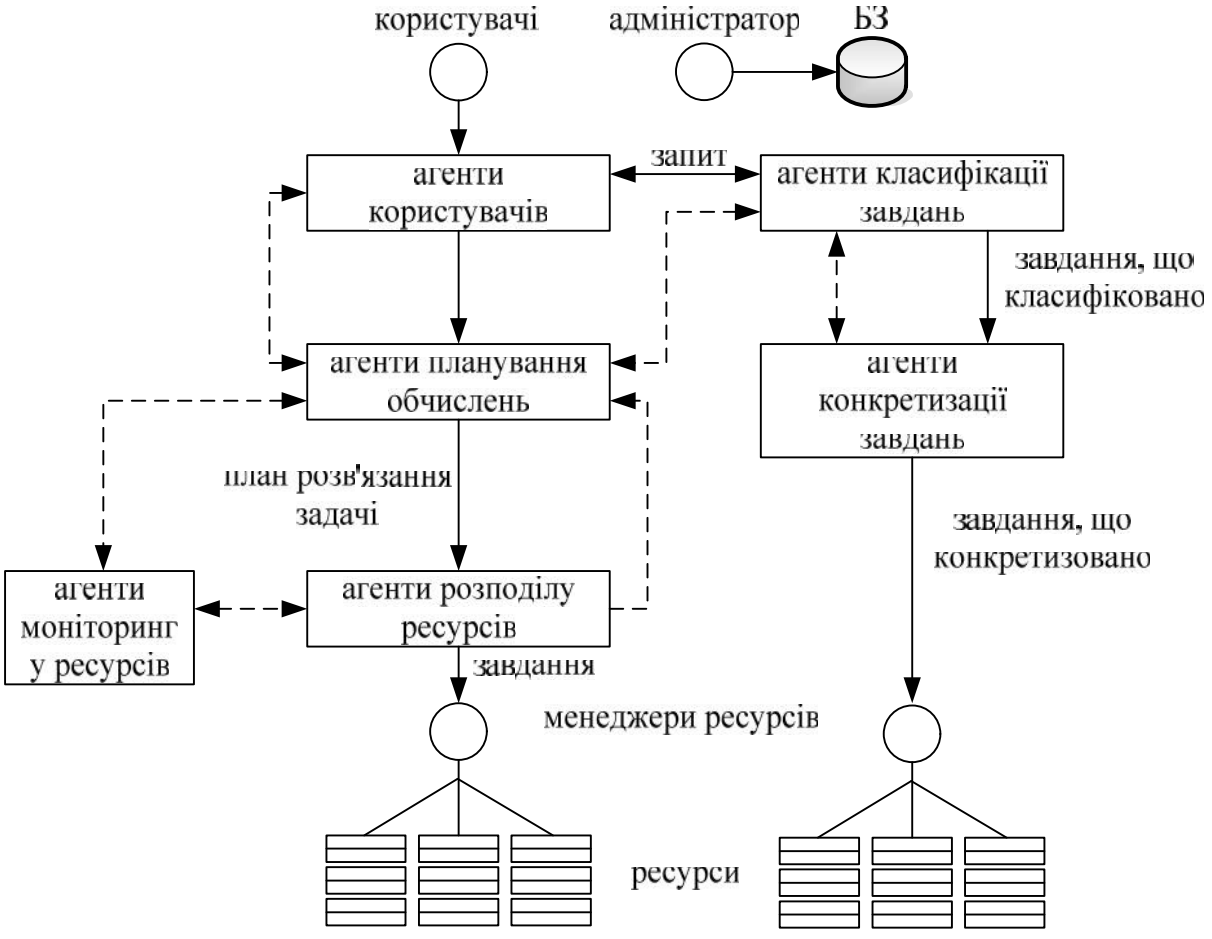


Рисунок 2.1 – Структура МАС

Ролі агентів можуть бути постійними та тимчасовими, що виникають у дискретні моменти часу у зв’язку з необхідністю організації колективної взаємодії. Рівні ієрархії агентів відрізняються обсягом їх знань, агенти

вищого рівня ієрархії мають більший обсяг знань порівняно з агентами нижчого рівня і, крім того, можуть звертатися до агентів будь-якого нижче лежачого рівня із запитом на отримання локальних знань цих агентів. На кожному рівні агенти можуть об’єднуватися у віртуальні спільноти, кооперуватися та конкурувати у їх рамках.

Координація дій агентів здійснюється за допомогою загальних правил групової поведінки. Агенти функціонують відповідно до заданих ролей і для кожної ролі визначено свої правила поведінки у віртуальному співтоваристві агентів. Мультиагентна система включає агентів розподілу ресурсів та керуючого агента. Задачею агентів лише на рівні РОС є отримання такого розподілу завдань, що надходять, який зберігає показники якості функціонування цієї системи у заданих адміністратором межах.

МАС включає агентів постановки завдання, планування обчислень, моніторингу та розподілу ресурсів, класифікації, конкретизації та виконання завдань, а також агентів параметричного налаштування алгоритмів функціонування перерахованих вище агентів на основі імітаційного моделювання.

Схеми баз знань агентів формуються на основі спеціалізованої концептуальної моделі [49], яка забезпечує взаємопов’язане уявлення проблемно-орієнтованого, програмно-апаратного, імітаційного та керуючого шарів знань про РОС. Тим самим було забезпечено проблемна орієнтація МАС. Дані про поточний стан РОС передаються до її бази знань системою моніторингу. Ефективність та надійність функціонування МАС, що розробляється, досліджується за допомогою натурного моделювання.

## 2.2 Концептуальна модель

Концептуальна модель є окремим випадком семантичної мережі, що забезпечує уявлення знань про програмні модулі для вирішення завдань у предметних областях та роботи з об’єктами РОС, схемних знань про

модульну структуру моделі та алгоритмів, продукційних знань для підтримки прийняття рішень щодо вибору оптимальних алгоритмів залежно від стану середовища, а також знань про програмно-апаратні інфраструктури даної системи та адміністративних політиках в її вузлах.

Нехай $Z$, $F$ та $M$ – це множина параметрів, операцій та програмних модулів моделі. Модулі є елементом обчислювальних знань. Параметри, операції та їх взаємозв'язки відбивають схемні знання. Операції з множини $F$ визначають обчислювальні дії на множині параметрів $Z$.

Кожній операції $f_i \in F$ відповідає модуль $m_j \in M$, де $i \in \overline{1, n_f}$, $j \in \overline{1, n_m}$, де $n_f$ – кількість операцій, $n_m$ – число модулів. Один модуль може здійснювати кілька операцій. Специфікація модуля включає мову програмування, тип та семантику вхідних, вихідних та транзитних параметрів, засоби передачі параметрів та обробки нестандартних ситуацій, модуль уявлення, необхідний компілятор та інші відомості.

З кожною операцією $f_i$ пов'язані дві множини параметрів $Z_i^{in}$, $Z_i^{out} \subset Z$. Множина $Z_i^{in}$ визначає параметри, значення яких необхідно задати, щоб отримати значення параметрів, представлених множиною $Z_i^{out}$. Множини $Z_i^{in}$ та $Z_i^{out}$ представляють відповідно множину вхідних та вихідних параметрів модуля $m_j$, що реалізує операцію $f_i$. Розрізняються базові та складові операції. Складові операції можуть містити базові операції, а також конструкції для організації розгалуження та ітерації.

Постановка задачі може формулюватися у повній чи скороченій формі. За сформульованою постановкою задачі будується схема її розв'язання на основі методів статичного, динамічного чи статико-динамічного планування обчислень. Оператор статико-динамічного планування може включатися до складової операції. Загалом у моделі може існувати безліч $S$ еквівалентних схем розв'язання задачі. Схема $s \in S$ визначає, які операції та в якій послідовності повинні бути виконані для вирішення задачі. Повна постановка

задачі $s_f$, що збігається зі схемою $s$, визначається структурою $s_f = \{F_s, X_0, Y_0\}$, де $F_s \subset F$ – це множина операцій, які потрібно виконати для розв'язання задачі, $X_0 \subset Z$ – множина вихідних параметрів, значення яких задані, $Y_0 \subset Z$ – множина цільових параметрів, значення яких слід обчислити.

У множинах параметрів та операцій моделі введено відповідно підмножини системних параметрів, що відображають характеристики об'єктів РОС, та операцій, що представляють алгоритми планування обчислень, моніторингу та розподілу ресурсів, моделювання обчислювальних процесів та інших процесів у цій системі. У процесі експлуатації РОС значення системних параметрів визначаються системою моніторингу.

## 2.3 Модель поведінки агентів

Функціонування агентів здійснюється з використанням парадигми кінцево-автоматного програмування. У кваліфікаційній роботі запропоновано нову модель агента, яка представлена кортежем 2.1

$$M_a = \{sm^p, sm^c_{i,j}, send\}, \quad (2.1)$$

де $sm^p$ – батьківський автомат;

$sm^c_{i,j}$ – дочірні автомати $sm^c_{i,j} : i \in \overline{1, n_{vc}}$, $j \in \overline{1, n_{rol}}$;

$n_{vc}$ – число віртуальних спільнот, в які може бути включено агент;

$n_{rol}$ – число ролей, які може грати агент;

$send$ – множина повідомлень агента.

Основною функцією батьківського автомата $sm^p$ є створення дочірнього автомата $sm^c_{i,j}$ при кожному включенні агента в нову віртуальну спільноту, де $i$ та $j$ – це відповідно номери віртуального співтовариства та ролі агента.

Модель батьківського автомата можливо представити кортежем 2.2

$$sm^p = \{US^p, us_0^p, X_{in}^p, X_{out}^p, g^p, h^p, D^p\}, \quad (2.2)$$

де $US^p$ – кінцева множина керуючих станів батьківського автомата;

$us_0^p \in US^p$ – початковий стан;

$X_{in}^p \subset Z$ – кінцева множина вхідних впливів, що породжуються РОС;

$X_{out}^p \subset Z$ – кінцева множина вихідних впливів;

$g^p : US^p \times X_{in}^p \times D^p \to X_{out}^p$ – функція виходів;

$h^p : US^p \times X_{in}^p \to US^p$ – функція переходів;

$g^p, h^p \in F$, $D^p \subset F$ – кінцева множина дій батьківського автомата.

У множині $X_{in}^p \cup X_{out}^p$ виділяється набір глобальних змінних батьківського автомата, які доступні дочірнім автоматам. У загальному випадку $X_{in}^p \cap X_{out}^p \neq \varnothing$. Схема формування множин $X_{in}^p$ та $X_{out}^p$ наведена на рисунку 2.2.

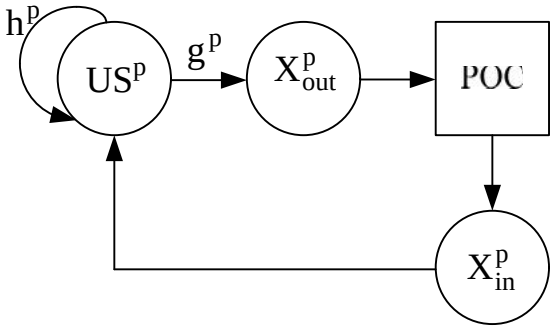


Рисунок 2.2 – Формування вхідних та вихідних впливів батьківського автомату

При створенні агента всі схеми виконання дій, що представляють собою абстрактні програми, генеруються за допомогою мови програмування Java. Модель дочірнього автомата представлена кортежем 2.3

$$sm_{i,j}^c = \{US_{i,j}^c, us_{i,j,0}^c, X_{i,j,in}^c, X_{i,j,out}^c, g_{i,j}^c, h_{i,j}^c, D_{i,j}^c\}, \quad (2.3)$$

де $sm^c_{i,j}$ – кінцева множина керуючих станів дочірнього автомата;

$us^c_{i,j,0} \in us^c_{i,j}$ – початковий стан;

$X^c_{i,j,in} \subset Z$ – кінцева множина вхідних дій, що породжуються РОС;

$X^c_{i,j,out} \subset Z$ – кінцева множина вихідних впливів;

$g^c_{i,j} : US^c_{i,j} \times X^c_{i,j,in} \times D^c_{i,j} \to X^c_{i,j,out}$ – функція виходів;

$h^c_{i,j} : US^c_{i,j} \times X^c_{i,j,in} \to US^c_{i,j}$ – функція переходів;

$h^c_{i,j}, h^c_{i,j} \in F, D^c_{i,j} \subset F$ – кінцева множина дій дочірнього автомата.

До множини $US^c_{i,j}$ входить кінцевий стан $us^c_{i,j,end}$, в якому агент завершує свою роботу. В загальному випадку $X^c_{i,j,in} \cap X^c_{i,j,out} \neq \varnothing$. Схема формування $X^c_{i,j,in}$ та $X^c_{i,j,out}$ представлена на рисунку 2.3.

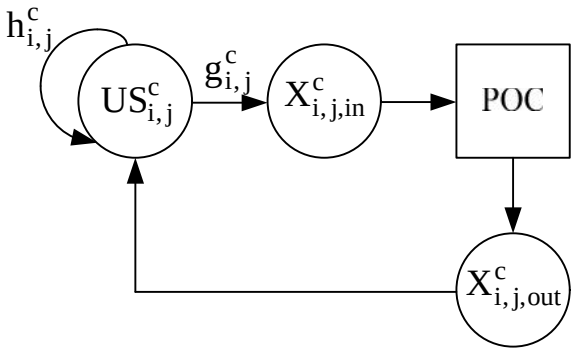


Рисунок 2.3 – Формування вхідних та вихідних впливів дочірнього автомату

Дочірні автомати різних агентів, що входять до однієї віртуальної спільноти, взаємодіють шляхом обміну повідомленнями, які передаються через батьківські автомати. Автомати, що є нащадками одного і того ж батьківського автомата, обмінюються інформацією про використання загальних ресурсів агента через глобальні змінні батьківського автомата та агентну базу знань.

Розглянуті вище батьківський та дочірній автомати є окремими випадками кінцевого керуючого автомата. Їх важливою відмінністю є те, що

в якості станів автомата використовуються стани-дії. В рамках одного стану агент може виконувати одну чи кілька дій. У кожному стані батьківський (дочірній) автомат змінює значення параметрів функцій переходів $h^p$ ($h^c_{i,j}$) за допомогою функції $g^p$ ($g^c_{i,j}$), а потім здійснює перехід у новий стан.

При вступі агента в і-е віртуальне співтовариство для відповідного дочірнього автомата створюється система логічного часу $SLT_i$, яка визначається кортежем

$$SLT_i = \{T, T_m, g_t, g_m, g_r\}, \quad (2.4)$$

де $T$ – область значень логічного часу;

$T_m$ – область значень тимчасових маркерів датування повідомлень ($T_m \subseteq T$);

$g_t$, $g_m$ та $g_r$ – функції датування подій автомата, маркування повідомлень та порівняння значень логічного часу $\forall i \in \overline{1, n_{vc}}$.

Система логічного часу використовує векторний годинник, в якому число компонент вектора часу дорівнює числу агентів віртуального співтовариства, і забезпечує відношення часткового порядку на безлічі подій віртуального співтовариства з з урахуванням їхньої обумовленості. Застосування функції датування повідомлень забезпечує їх обробку у встановленій логічній послідовності, а не у довільному порядку надходження їх у загальний пул.

## 2.4 Розширення концептуальної моделі

З метою підтримки побудови та застосування запропонованої рольової моделі поведінки агентів, концептуальна модель була розширена новими об’єктами: агентами, їх ролями, віртуальними спільнотами та відносинами між ними, а також станами, функціями та графами переходів автоматів. В

якості стану агента будемо використовувати стан-дію – послідовність системних операцій над полем системних параметрів моделі.

Фрагмент такої моделі, що описує об’єкти, необхідні для побудови графів переходів агентів, представлений на рисунку 2.4. Тут G – множина графів переходів, A – множина агентів, VC – множина віртуальних спільнот агентів, R – множина ролей агентів та US – множина стану автоматів. На рисунку 2.4 відносини між об’єктами позначені $в_1 - в_8$. Відношення $в_4$ представляє взаємозв’язок станів з операціями, що реалізують функції переходів. Відношення $в_5$ представляє взаємозв’язок станів з іншими системними операціями. База знань агента створюється на основі фрагмента моделі POC, який було розглянуто вище.

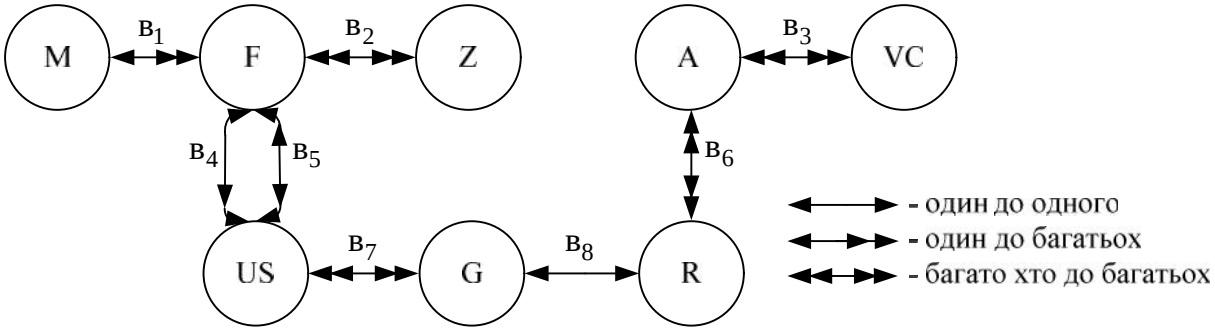


Рисунок 2.4 – Фрагмент опису розширеної концептуальної моделі

## 2.5 Алгоритми функціонування агентів

Вибір вузлів обчислювального кластера, в яких виконуватиметься завдання, здійснюється за допомогою алгоритму багаторівневого конкретизуючого планування завдань із заданими критеріями якості (показниками надійності, часу та вартості) їх виконання. Процес планування здійснюється агентами, які представляють вузли обчислювального кластера. Процес планування включає формування всієї множини доступних вузлів; конкретизацію сформованої множини шляхом виключення з неї перевантажених вузлів (щодо поточного середнього завантаження вузлів з урахуванням наявних черг завдань); побудова плану виконання завдання у

вузлах; вилучення із побудованого плану спеціалізованого плану, що задовольняє заданим критеріям якості виконання завдання з урахуванням поточного стану РОС та призначення вузлів обчислювального кластера для виконання цього завдання. Побудова спеціалізованого плану складається на основі економічного механізму регулювання попиту та пропозиції обчислювальних ресурсів. У таблицях 2.1-2.3 наведено стани-дії агентів, функції переходів $f_1 - f_{36} \in F$ та параметри $q_1 - q_{36} \in Z$ цих функцій.

Таблиця 2.1 – Стан-дія автоматів

| Стан | Дія |
|---|---|
| 1 | 2 |
| $S_{start}$ | Очікування готовності |
| $S_{end}$ | Завершення роботи |
| $S_1$ | Перевірка середовища, ініціалізація |
| $S_2$ | Обробка черги повідомлень |
| $S_3$ | Формулювання постановки задачі |
| $S_4$ | Обробка збійних постановок завдань |
| $S_5$ | Побудова плану розв’язання задачі |
| $S_6$ | Обробка збійних планів розв’язання задач |
| $S_7$ | Передача завдань агенту класифікатору |
| $S_8$ | Передача завдання, що було класифіковано, агенту формування і-ої ВС |
| $S_9$ | Обслуговування дочірніх автоматів і-ої ВС |
| $S_{10}$ | Перевірка відповідності області допустимих значень характеристик завдання характеристикам наявних класів обчислювальних ресурсів |
| $S_{11}$ | Формування множини допустимих для розподілу ресурсів |
| $S_{12}$ | Передача завдання, яке було класифіковано, агенту формулювання постановки задачі та побудови плану її вирішення |
| $S_{13}$ | Підготовка повідомлення про формування нової ВС, топології ВС та класифікованого завдання агентам, які представляють ресурси |

Продовження таблиці 2.1

| 1 | 2 |
|---|---|
| $S_{14}$ | Обробка отриманих ставок та визначення агента координатора і-ої ВС |
| $S_{15}$ | Підготовка повідомлення про завершення торгів та розподіл модулів завдання між агентами |
| $S_{16}$ | Ініціалізація роботи і-ої ВС, розсилання модулів на виконання ресурсним агентам |
| $S_{17}$ | Опитування стану агентів і-ої ВС, отримання інформації від агента моніторингу |
| $S_{18}$ | Обробка відмови, вибір сценарію |
| $S_{19}$ | Формування та відправлення залишкової схеми агенту постановки завдань |
| $S_{20}$ | Обробка завдань, що завершено, та надсилання повідомлення батьківському автомату про завершення виконання модулів у і-ій ВС |
| $S_{21}$ | Обробка результатів обчислення, розформування і-ої ВС, завершення роботи координатора і-ої ВС |
| $S_{22}$ | Запуск дочірнього автомата агента, що виконує модулі завдання в і-ій ВС |
| $S_{23}$ | Передача модулів менеджеру ресурсів на виконання |
| $S_{24}$ | Відновлення зв’язку з координатором |
| $S_{25}$ | Обробка завдань, що було завершено |
| $S_{26}$ | Організація перевиборів координатора |
| $S_{27}$ | Завершення роботи дочірнього автомата агента, що виконує модулі завдання в і-ій ВС |
| $S_{28}$ | Вступ до і-ої ВС, формування ставки за модулі, формування пропозиції щодо вибору координатора |
| $S_{29}$ | Надсилання повідомлень агенту координатору про завершення виконання модулів |
| $S_{30}$ | Запуск дочірнього автомата агента координатора і-ої ВС |

Таблиця 2.2 – Функції переходів агентів

| Функція | Умова переходу | Функція | Умова переходу |
|---|---|---|---|
| $f_1$ | $\overline{q_1} \wedge q_2$ | $f_{19}$ | $\overline{q_5} \wedge q_{21}$ |
| $f_2$ | $q_1$ | $f_{20}$ | $q_{16}$ |
| $f_3$ | $\overline{q_1} \wedge \overline{q_2} \wedge q_3$ | $f_{21}$ | $q_{13}$ |
| $f_4$ | $\overline{q_3}$ | $f_{22}$ | $q_{14}$ |
| $f_5$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3}$ | $f_{23}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{18}$ |
| $f_6$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{18}}$ | $f_{24}$ | $q_{19}$ |
| $f_7$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{11}}$ | $f_{25}$ | $q_{21}$ |
| $f_8$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{25}} \wedge \overline{q_8}$ | $f_{26}$ | $q_{22}$ |
| $f_9$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_1} \wedge q_8$ | $f_{27}$ | $q_{24}$ |
| $f_{10}$ | $q_4$ | $f_{28}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{25}$ |
| $f_{11}$ | $q_6$ | $f_{29}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{25}} \wedge q_8$ |
| $f_{12}$ | $q_7$ | $f_{30}$ | $q_{28}$ |
| $f_{13}$ | $q_9$ | $f_{31}$ | $q_5$ |
| $f_{14}$ | $q_{10}$ | $f_{32}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{17}} \wedge q_{15}$ |
| $f_{15}$ | 1 | $f_{33}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{17}$ |
| $f_{16}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{11}$ | $f_{34}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{17}} \wedge \overline{q_{15}}$ |
| $f_{17}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{12}} \wedge q_{23}$ | $f_{35}$ | $q_{20}$ |
| $f_{18}$ | $\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{12}$ | $f_{36}$ | $\overline{q_5} \wedge \overline{q_{21}}$ |

Таблиця 2.3 – Параметри функцій переходів

| Параметр | Опис |
|---|---|
| 1 | 2 |
| $q_1$ | Прапор виходу |
| $q_2$ | Таймер переходу не готовий |
| $q_3$ | Система не ініціалізована |

Продовження таблиці 2.3

| 1 | 2 |
|---|---|
| $q_4$ | Наявність необроблених завдань |
| $q_5$ | Наявність пропозиції вступити до і-ої ВС |
| $q_6$ | Успішно оброблені нові завдання |
| $q_7$ | Сформульовано постановку завдання |
| $q_8$ | Наявність завершених модулів агентів і-ої ВС |
| $q_9$ | Успішна побудова планів для всіх нових завдань |
| $q_{10}$ | Побудовано план розв’язання задачі |
| $q_{11}$ | Наявність нових завдань, що було класифіковано |
| $q_{12}$ | Готовність таймера перевірки дочірніх автоматів |
| $q_{13}$ | Наявність нових завдань, що було класифіковано |
| $q_{14}$ | Сформовано нове повідомлення для розсилки агентам і-ої ВС |
| $q_{15}$ | Наявність завершених модулів даного агента |
| $q_{16}$ | Завершено виконання всіх модулів даного агента |
| $q_{17}$ | Втрачено зв’язок з координатором |
| $q_{18}$ | Наявність нових повідомлень від агентів і-ої ВС |
| $q_{19}$ | Вибори агента координатора і-ої ВС завершені успішно |
| $q_{20}$ | Сформовано нове повідомлення про завершення торгів та розподіл модулів завдання між агентами |
| $q_{21}$ | Отримано повідомлення про призначення ролі координатора і-ої ВС |
| $q_{22}$ | і-а ВС успішно ініціалізована |
| $q_{23}$ | Завершено всі модулі і-го ВС |
| $q_{24}$ | Сформовано нове повідомлення для агентів і-ої ВС |
| $q_{25}$ | Виявлено ознаки відмов |

На різних етапах функціонування агентів між ними відбувається обмін повідомленнями у форматі JavaScript Object Notation (JSON). Повідомлення містять наступну основну інформацію:

- ідентифікатор контейнера JADE, у якому працює агент;
- ідентифікатор агента;
- сервісна інформація;
- тіло повідомлення.

Властивості формату JSON дозволяють у текстовому форматі передавати будь-які дані як «ключ»: «значення», зокрема і масиви. Ця якість використовується для формування та передачі ставок агентів, передачі списку параметрів, що виконуються модулями. Кожен агент містить необхідні відомості для інтерпретації повідомлень.

Графи переходів основних розглянутих агентів представлені на рисунках 2.5-2.10.

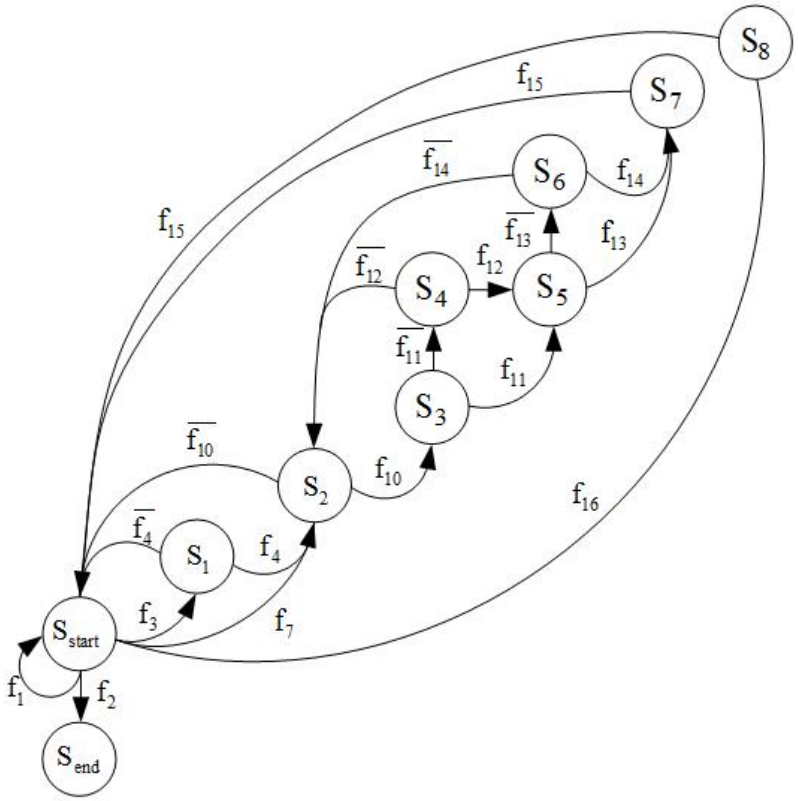


Рисунок 2.5 – Граф переходів батьківського автомата для агента формулювання постановки задачі та побудови плану її вирішення

Кожна вершина графа представлена станом-дією, а умовою переходу в цей стан є задоволення умов відповідних функцій переходу, поданих у таблиці 2.2. Для виконання модулів чергового завдання користувача

формується ВС, індекс якого в таблицях представлений через і.

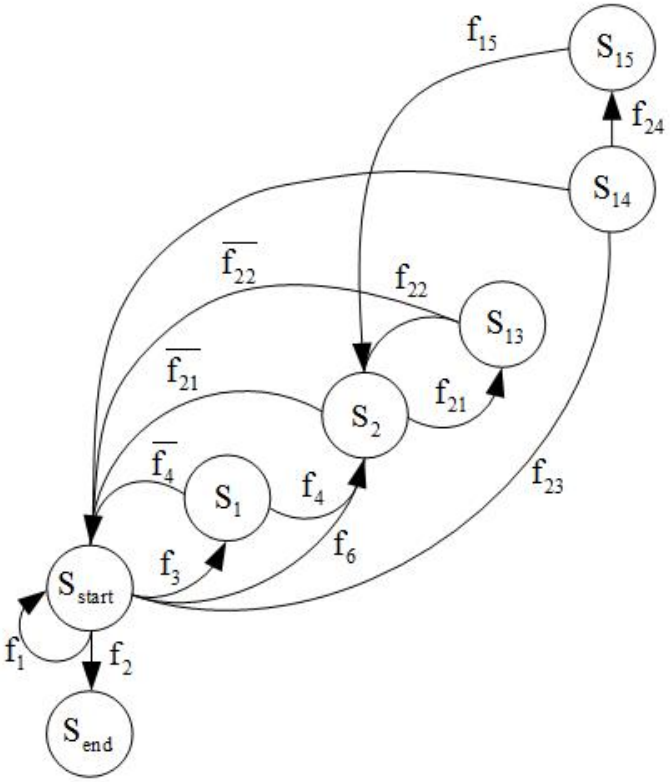


Рисунок 2.6 – Граф переходів батьківського автомата для агента організації ВС

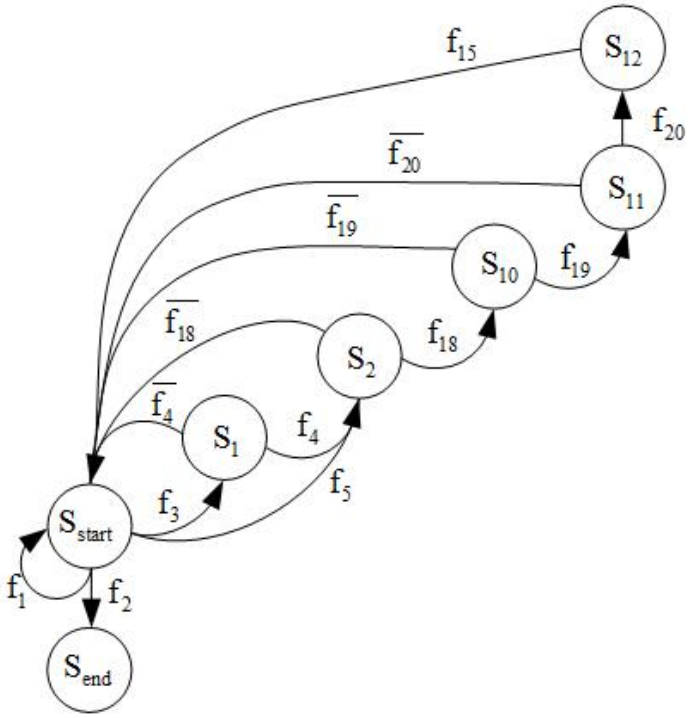


Рисунок 2.7 – Граф переходів батьківського автомата агента класифікації завдань

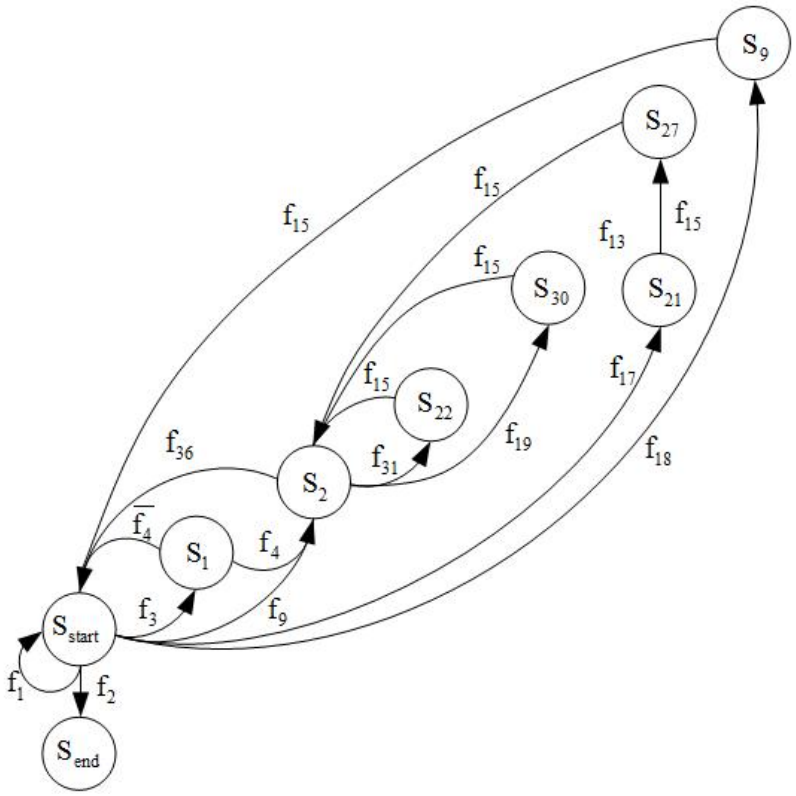


Рисунок 2.8 – Граф переходів батьківського автомата агента, що представляє ресурси середовища

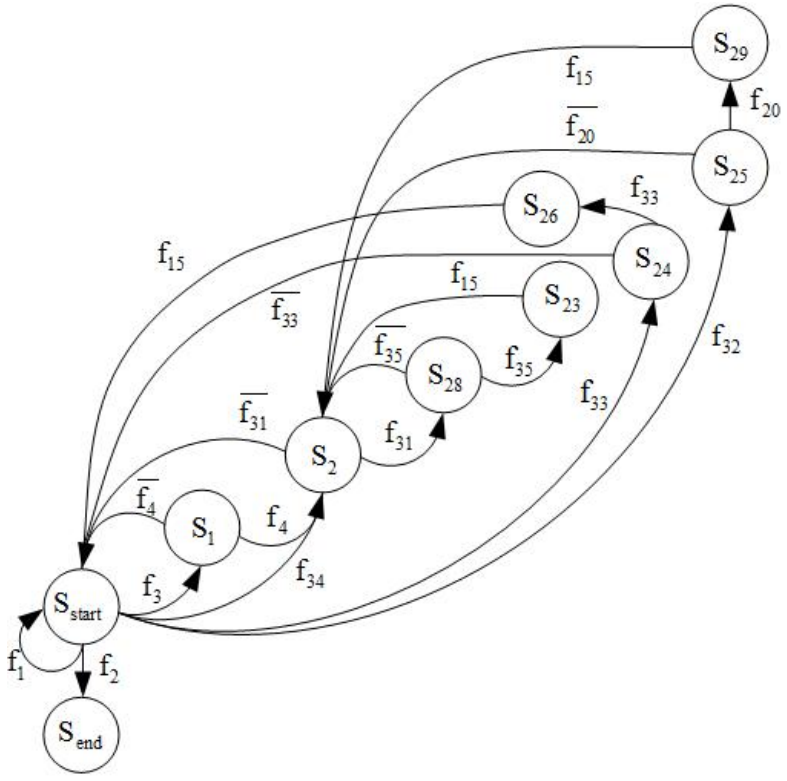


Рисунок 2.9 – Граф переходів дочірнього автомата для агента, що представляє ресурси середовища та функціонуючого в ролі ресурсного агента

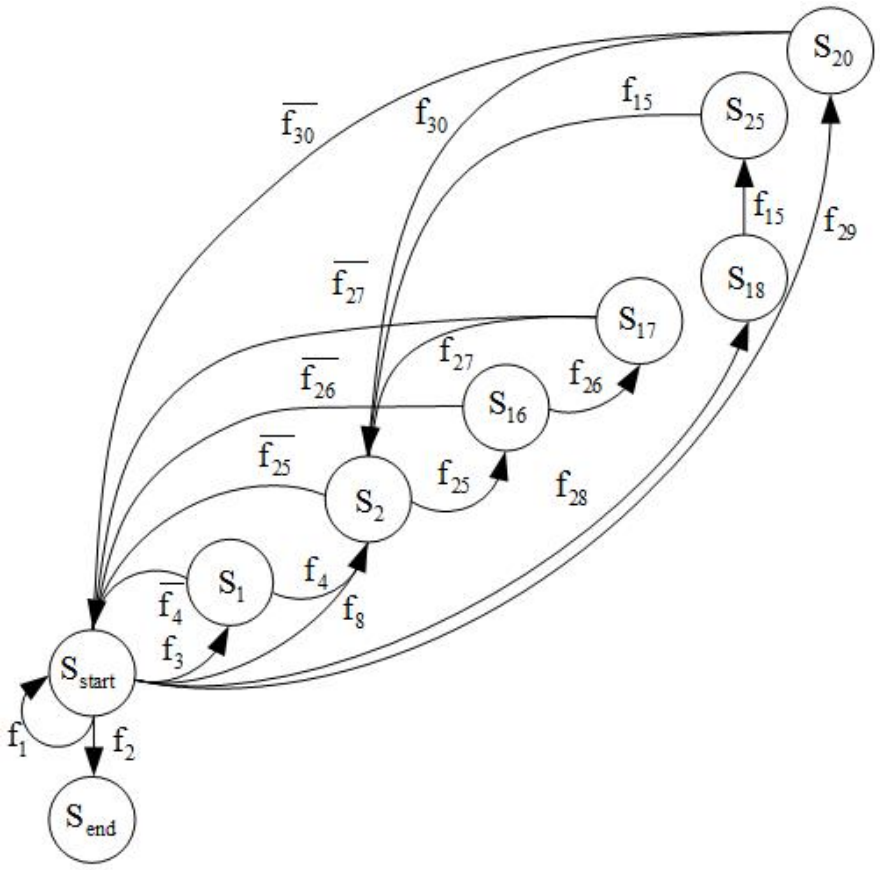


Рисунок 2.10 – Граф переходів дочірнього автомата для агента, що представляє ресурси середовища та функціонує в ролі координатора ВС

## 2.6 Алгоритм перерозподілу ресурсів середовища

На даний час забезпечення відмовостійкості обчислювального процесу в РОС, наприклад, у Grid-системах або хмарних платформах, як і раніше залишається актуальною проблемою. У зв’язку з цим у роботі розроблено новий мультиагентний алгоритм для перерозподілу обчислювальних ресурсів РОС у разі відмови процесу розв’язання задачі. Залишкова схема розв’язання задачі формується з використанням методів конкретизуючого програмування На відміну від відомих алгоритмів подібного призначення запропонований алгоритм реалізує адаптивне мультисценарне вирішення цієї задачі і тим самим підвищує ступінь відмовостійкості обчислювального процесу.

Проблема відновлення обчислювального процесу (завдання) внаслідок відмови програмно-апаратного забезпечення актуальна при вирішенні

великих фундаментальних та прикладних завдань у різнорідних РОС. Існують різні підходи до вирішення цієї проблеми [50, 51].

Зокрема, підходом який широко застосовується на практиці для перезапуску завдань є механізм контрольних точок [52]. Проте формування таких контрольних точок у вузлі обчислювальної системи не дозволяє зробити рестарт у разі відмови самого вузла. Створення системних контрольних точок на рівні ОС, що забезпечує перенесення завдання до інших вузлів, спричиняє суттєві накладні витрати та підтримується далеко не всіма засобами управління обчисленнями.

МАС використовує алгоритм перерозподілу обчислювальних ресурсів РОС для поліпшення стійкості до відмови процесу вирішення задачі. Алгоритм включає наступні основні етапи роботи:

- виявлення ті ідентифікація відмови;

- обробка відмови;

- вибір сценарію для забезпечення відказостійкості виконання схеми вирішення задачі;

- побудова залишкової схеми розв’язання задачі.

На першому етапі виявлення та ідентифікація збою здійснюється засобами системи моніторингу. Далі агенти цієї системи запускають процедуру обробки збою, що включає часткове усунення наслідків та реконфігурацію множини доступних ресурсів. Крім того, вони збирають та передають необхідну інформацію агентам віртуального співтовариства.

На основі отриманих даних агенти обирають сценарій, що забезпечує стійкість до відмови схеми вирішення задачі. Коли вузол агента виходить з ладу під час виконання модуля, агент може ухвалити незалежне рішення виконати цей модуль одному з резервних вузлів. В інших випадках агенти обирають сценарій колективно.

На останньому етапі агенти у процесі торгів борються за право виконати модулі залишкової схеми розв’язання задачі відповідно до обраного сценарію. Формування схеми складається основі методів редукції

надлишкових обчислень.

Рисунок 2.11 ілюструє сценарії обробки зазначених вище відмов. Склад віртуального співтовариства агентів, які виконують рішення завдання, оновлюється після призначення нових агентів для виконання модулів.

Виділення ресурсів агентами здійснюється з урахуванням відмінностей обчислювальних характеристик вузлів РОС при реалізації багаторівневого паралелізму алгоритму розв’язання задачі.

Модуль може бути виконаний іншим агентом у разі всіх трьох відмов. Крім того, при відмові вузла модуль може бути виконаний у резервному вузлі. Додатково, при відмові модуля схеми розв’язання задачі, може бути виконано модуль еквівалентної схеми рішення задачі.

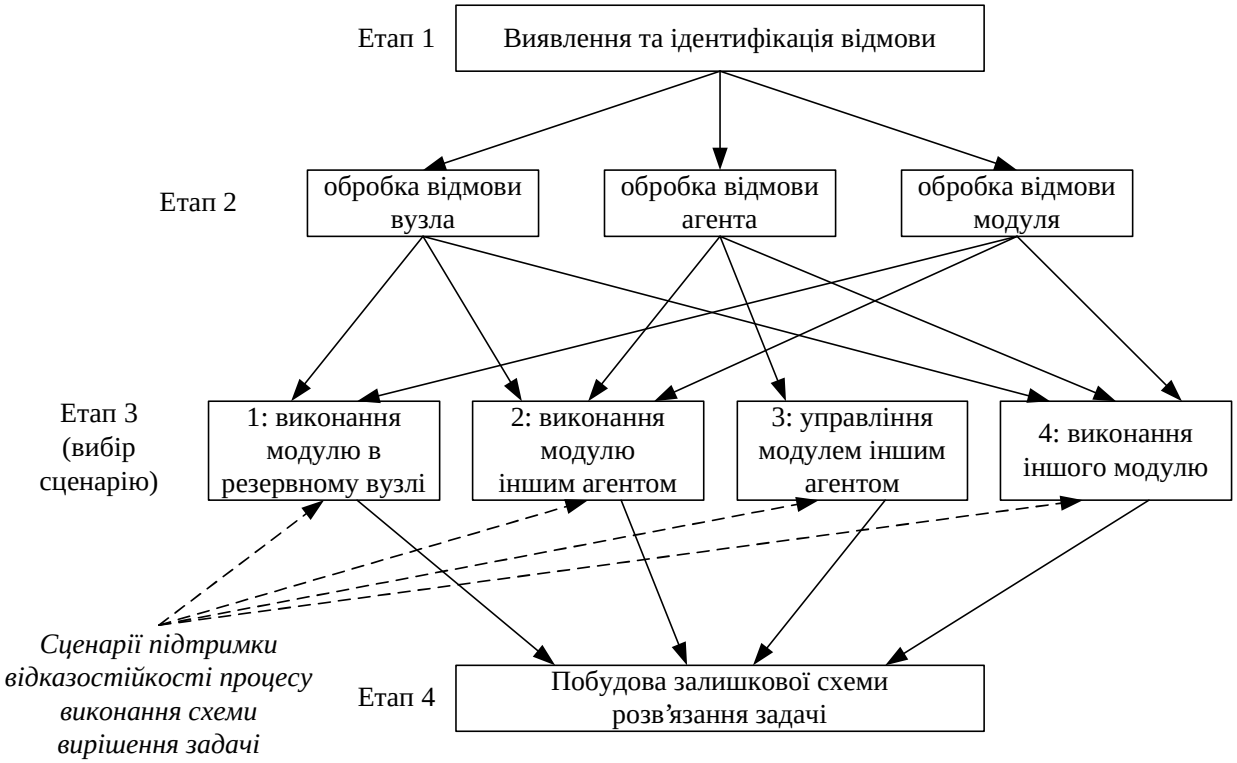


Рисунок 2.11 – Сценарії обробки відмов

Є три можливі варіанти призначення агента для виконання модуля:

- централізоване призначення модуля іншому агенту, який має можливість виконати цей модуль, координатором виконання схеми розв’язання задачі;

- децентралізоване призначення модуля шляхом проведення торгів між агентами, які можуть виконати цей модуль;

- децентралізоване призначення модулів залишкової схеми розв’язання задачі шляхом проведення торгів між агентами, які можуть брати участь у виконанні модулів залишкової схеми.

## 2.7 Методи та засоби навчання агентів

Процес навчання агентів будується на комплексному використанні методів концептуального моделювання, класифікації завдань та параметричного налаштування системи управління.

Робота системи класифікації завдань ґрунтується на використанні концептуальної моделі РОС та є її доповненням. У системі класифікації завдань адміністратори вузлів на основі своїх експертних знань визначають множину $H = \{h_1, h_2, ..., h_m\}$ можливих характеристик завдань (час вирішення завдання, розміри оперативної та дискової пам’яті, кількість вузлів, процесорів та ядер, режими виконання модулів тощо), а також області їх допустимих значень. Потім вони формують класи завдань $C = \{c_1, c_2, ..., c_n\}$, що мають набори характеристик із певної множини. Область допустимих значень конкретної характеристики при її включенні до того чи іншого класу може бути конкретизовано.

Сформованим класам зіставляються найбільш підходящі обчислювальні ресурси до виконання завдань, які відносяться до цих класів (рисунок 2.12).

Адміністратор РОС задає значення параметрів, які визначають наміри ресурсних агентів щодо виконання завдань різних класів, нижні та верхні межі допустимого відхилення від середнього навантаження ресурсів агентів віртуальних співтовариств, величини штрафів за відхилення від середнього навантаження (для агента параметричного налаштування); склад та періодичність збору інформації, формати даних, системи моніторингу та

контрольно-вимірювальні прилади, межі зміни величин, що вимірюються, а також керуючі впливи, що застосовуються при їх досягненні (для агента моніторингу); допустимі квоти за кількістю завдань, часом їх виконання та числом вузлів, що використовуються.

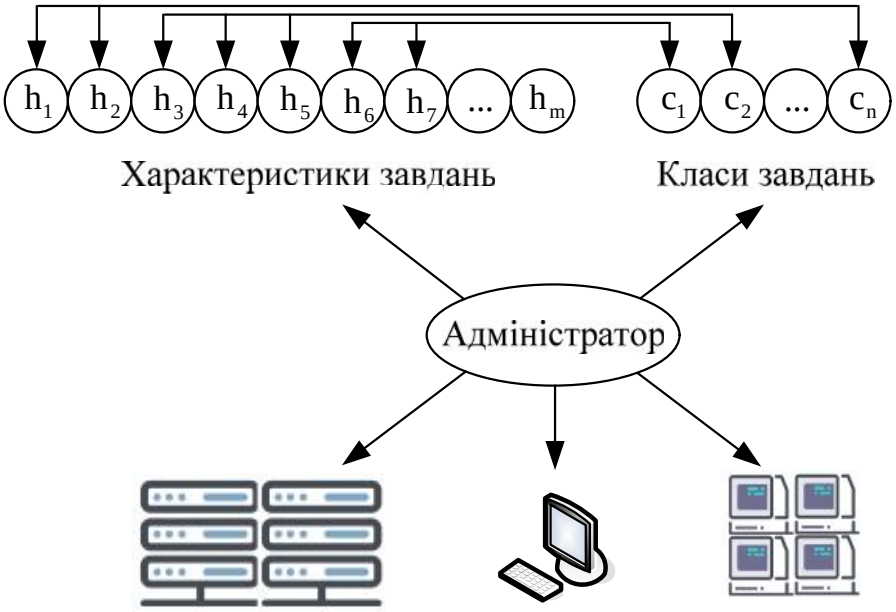


Рисунок 2.12 – Схема формування класів завдань та зіставлення їм обчислювальних ресурсів

Для підтримки адаптивного навчання агентів розроблено нову технологію параметричного налаштування мультиагентних алгоритмів управління РОС з метою оптимізації розподілу її ресурсів при виконанні завдань користувачів. Схема параметричного налаштування алгоритмів роботи агентів розподілу ресурсів наведена на рисунку 2.13. Агент-класифікатор ідентифікує класи завдань, з урахуванням яких формується віртуальне співтовариство ресурсних агентів. Ці агенти розподіляють ресурси на основі тендеру обчислювальних робіт, і алгоритми їх роботи визначаються вектором параметрів, що забезпечує агентам вибір оптимальної стратегії поведінки. Ці параметри керування регулюються агентом параметричного налаштування та відображають значення вектора змінних

імітаційної моделі РОС. Знаходження значень параметрів, що управляють, проводиться за допомогою багатокритеріальних правил дискретного вибору.

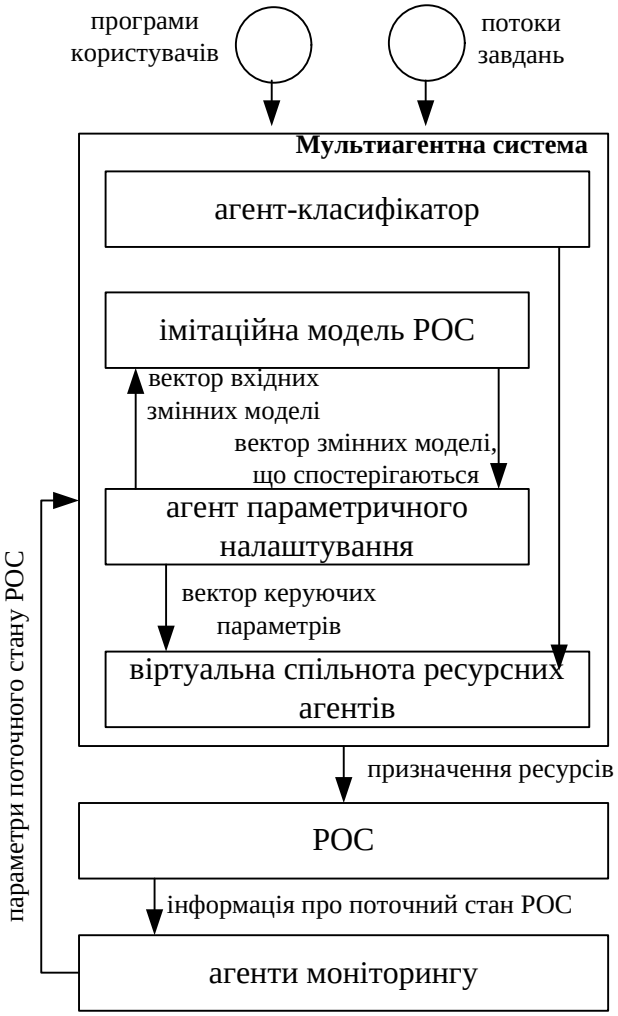


Рисунок 2.13 – Схема параметричного налаштування

Агент моніторингу призначений для забезпечення суб’єктів РОС актуальною інформацією щодо завантаження її ресурсів, фізичному стані обчислювального обладнання та пристроїв інженерної інфраструктури. Важливою особливістю застосовуваних агентів моніторингу на відміну від інших систем є здатність агентів аналізувати та приймати керуючі впливу безпосередньо на обчислювальному вузлі, на якому функціонує агент. Дані, які було зібрано, уніфікуються, агрегуються та передаються експертній системі для їх аналізу. У разі виявлення критичних подій виконуються необхідні функції виконавчої системи з метою застосування керуючих

впливів для автоматичного усунення несправностей. При цьому адміністратором може проводитися попереднє навчання агентів моніторингу з урахуванням призначення обчислювальних вузлів та завдань, які на них виконуються.

Оцінка дій ресурсних агентів регулюється системою штрафів відхилення від середнього навантаження ресурсів у їхній віртуальній спільноті. На основі аналізу результатів розподілу ресурсів агенти можуть змінювати наміри щодо виконання завдань різних класів. Рисунок 2.14 показує результати моделювання середнього завантаження процесорів ресурсів потоком завдань. Ці результати показують, що параметричне налаштування агентів може суттєво покращити балансування завантаження процесорів з урахуванням заданих критеріїв ефективності завдань.

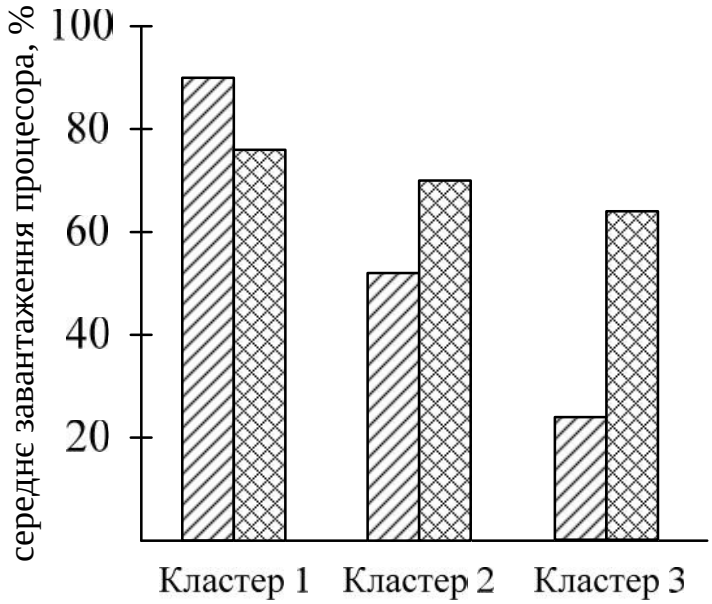


Рисунок 2.14 – Балансування завантаження

У системі навчання агентів використовується нова гібридна модель вилучення та застосування проблемно-орієнтованих знань агентами в процесі керування потоками завдань. Розробники додатків здійснюють концептуальне моделювання проблемної галузі та описують специфіку завдань, що розв’язуються. Адміністратори формують знання шляхом

додавання описів системних об’єктів РОС та правил їх роботи в концептуальній моделі, а також за допомогою визначення конфігураційних налаштувань окремих агентів та мультиагентної системи загалом. Знання розробників додатків та адміністраторів утворюють базу знань РОС, на основі якої створюються локальні бази знань додатків та агентів, що функціонують у рамках РОС з урахуванням локальних взаємодій між собою.

На рисунку 2.15 наведено схему передачі знань між суб’єктами, які генерують ці знання, агентами та програмними комплексами.

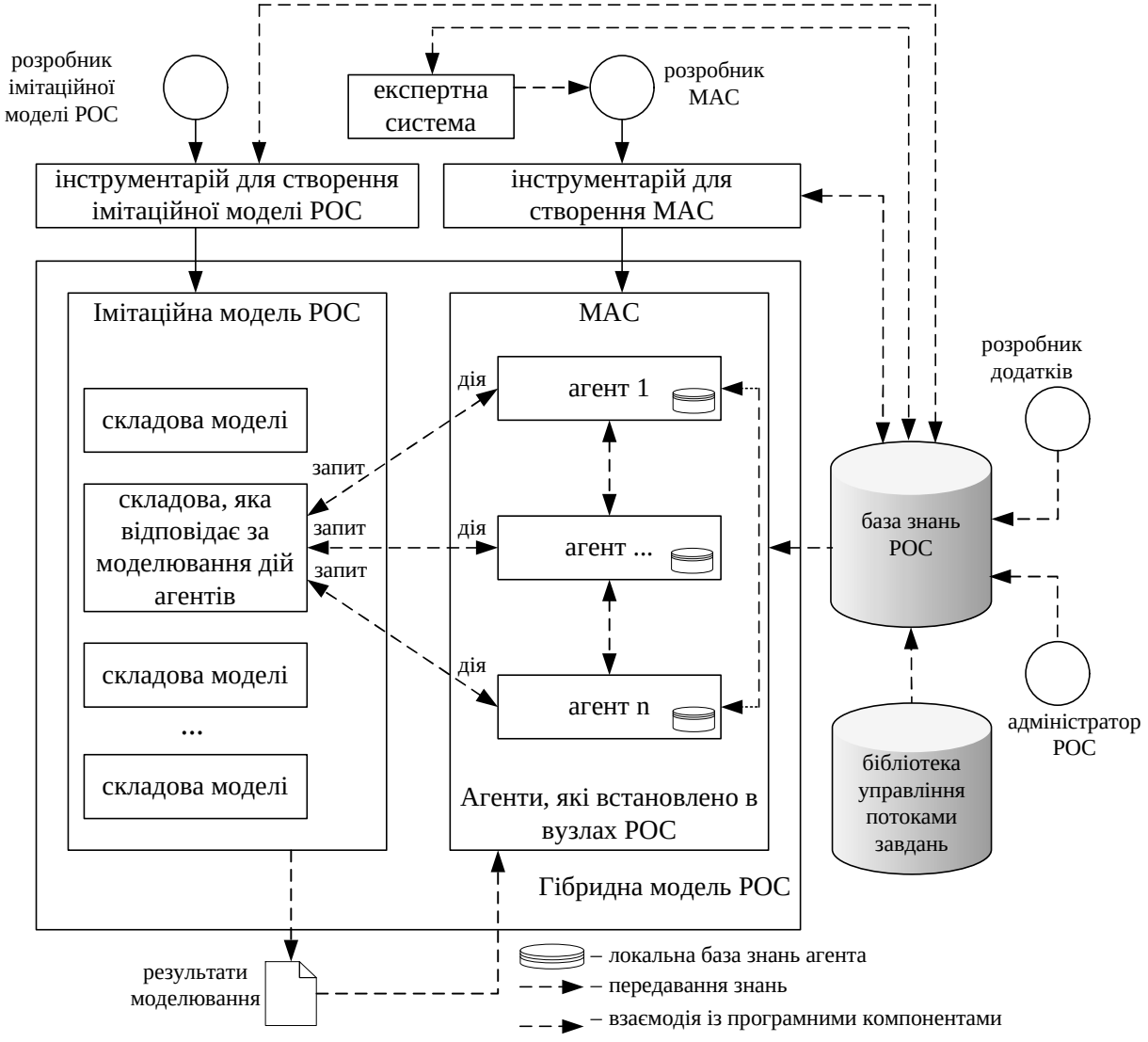


Рисунок 2.15 – Схема передачі знань

Локальні бази знань агентів формуються розробниками МАС. У разі створення агентів на рівні програми його розробник може виступати і як

розробник МАС. Як правило, адміністратор розробляє агентів для управління обчисленнями лише на рівні РОС. Спеціальні агенти витягують додаткові знання шляхом розпізнавання властивостей додатків та їх завдань, моделювання обчислювальних процесів та прогнозування інтенсивності надходження та часу виконання завдань. Додаткові знання передаються агентам розподілу ресурсів для виконання завдань з метою підвищення ефективності управління обчисленнями. Нова гібридна модель РОС організується шляхом включення в імітаційну модель засобів взаємодії з реальними агентами, які розміщено у вузлах РОС, що суттєво підвищує ступінь достовірності результатів моделювання в порівнянні з окремо взятим імітаційним моделювання. Моделі поведінки реальних агентів будуються в процесі їх розробки з експертною підтримкою цього процесу.

# 3 ОБЧИСЛЮВАЛЬНІ ЕКСПЕРИМЕНТИ

У розділі розглянуто обчислювальні експерименти у розподіленій обчислювальній системі, яка була розглянута у попередніх розділах. Дані експерименти відбивають різні показники якості функціонування мультиагентної системи під час управління потоками завдань.

При проведенні експериментів РОС включає три пули обчислювальних ресурсів. Склад пулів обчислювальних ресурсів наведено в таблиці 3.1, яка містить пікову продуктивність $P_{max}$, число вузлів $n_n$, число процесорів $n_p$, число ядер $n_c$, а також ОС для кожного пулу обчислювальних ресурсів. Детальний опис вузлів пулів наведено в таблиці 3.2.

Таблиця 3.1 – Пули обчислювальних ресурсів експериментальної РОС

| Пул | $P_{max}$ | $n_n$ | $n_p$ | $n_c$ | ОС |
|---|---|---|---|---|---|
| Пул 1 | 1.35 | 4 | 8 | 128 | Windows |
| Пул 2 | 1.97 | 6 | 12 | 192 | Windows, Linux |
| Пул 3 | 1.43 | 12 | 16 | 128 | Linux |

До розподіленої обчислювальної системи можуть бути підключені інші різнорідні обчислювальні ресурси: персональні комп’ютери (ПК), високопродуктивні сервери, кластери ПК, системи зберігання даних та ін. Потоки завдань надходять до розподіленої обчислювальної системи через спеціальний шлюз.

Таблиця 3.2 – Вузли пулів обчислювальних ресурсів

| Вузол | Характеристики вузла |
|---|---|
| Вузол пулу 1 | 16 core, 2.3 GHz, 16 MB L3 cache, 64 GB ОЗП DDR3-1600 |
| Вузол пулу 2 | 16 core, 2.3 GHz, 16 MB L3 cache, 64 GB ОЗП DDR3-1600 |
| Вузол пулу 3 | 2 core with hyper-threading, 2.4 GHz, 2 GB ОЗП |

## 3.1 Відмовостійкість РОС

Проведено порівняльний аналіз стійкості до відмови РОС при обробці потоків завдань за допомогою натурного моделювання. З цією метою було розроблено програму, що генерує потік завдань на основі обчислювальної історії. Статистика обчислень зібрана у процесі вирішення типових практичних завдань розподілу завдань на ресурси.

Модулі програми моделюють роботу прикладних програм, які використовуються у процесі вирішення перерахованих вище завдань. Вони виконують у РОС реальне завантаження ресурсів та обмін заданими обсягами даних, що відповідають показникам виконання реальних прикладних програм. Програма включає 8 модулів. Характеристики їх виконання представлені у таблиці 3.3.

Таблиця 3.3 – Характеристики виконання модулів програми

| Задача | Модуль | Середній час виконання, сек | Середній обсяг даних, МБ | |
|---|---|---|---|---|
| | | | вхідні | вихідні |
| Задача 1 | $m_1$ | 141.43 | 61.54 | 65.43 |
| | $m_2$ | 183.54 | 333.85 | 386.69 |
| | $m_3$ | 21.49 | 87.51 | 93.48 |
| | $m_4$ | 32,46 | 11.58 | 12.39 |
| Задача 2 | $m_5$ | 24.15 | 12.85 | 14.63 |
| | $m_6$ | 22.68 | 18.48 | 23.65 |
| | $m_7$ | 36.25 | 22.67 | 68.21 |
| | $m_8$ | 32.57 | 95.11 | 21.78 |

Потік завдань, який було згенеровано, по черзі виконано під управлінням трьох систем: метапланувальників GridWay та CondorDAGMan, а також наведеної в кваліфікаційній роботі МАС з навчанням агентів та без нього. У процесі роботи систем моделювалися типові відмови їх компонентів

та вузлів різнорідної РОС. Інтенсивність відмов становила близько одного збою на годину. Така кількість відмов перевищує їхню реальну інтенсивність для обчислювальних кластерів. Однак така частота відмов дозволяє наочніше продемонструвати переваги та недоліки кожної із систем.

Рисунок 3.1 показує успішну обробку відмов вузлів усіма трьома системами. Число вузлів, в яких відмови були не усунені, дорівнює 0 у всіх чотирьох випадках.

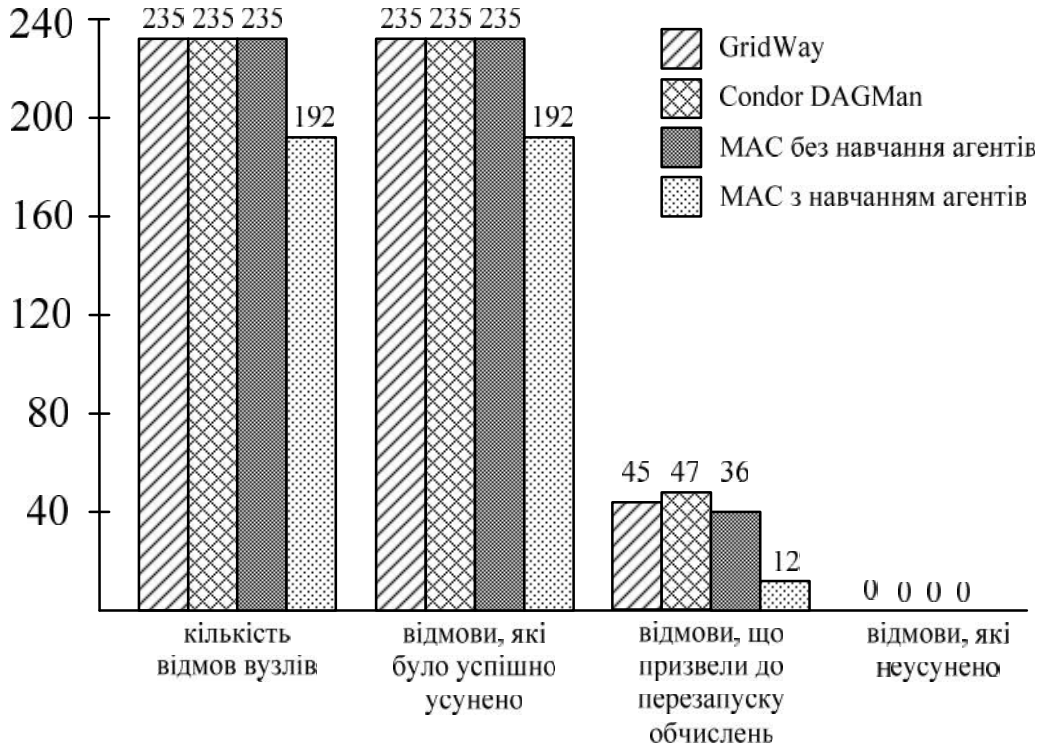


Рисунок 3.1 – Усунення відмов вузлів

Однак можна відзначити низку переваг мультиагентного управління обчисленнями. По-перше, воно забезпечило зниження числа рестартів у порівнянні з традиційними метапланувальниками GridWay та Condor DAGMan. Таке зниження стає особливо очевидним під час використання системи навчання агентів. По-друге, навчання агентів дозволило знизити кількість відмов вузлів за рахунок призначення агентами надійніших вузлів для виконання завдань. Результати, представлені на рисунку 3.1, демонструють істотне зниження кількості відмов вузлів під керівництвом

MAC з навчанням агентів більш, ніж 17% проти трьох інших випадків.

Переваги MAC щодо обробки відмов компонентів системи управління відображені на рисунку 3.2. У випадках виникнення таких відмов агент-лідер брав на себе управління вузлом агента, що відмовив, і продовжував виконання обчислень в даному вузлі аж до їх завершення або моменту відновлення агента, що відмовив.

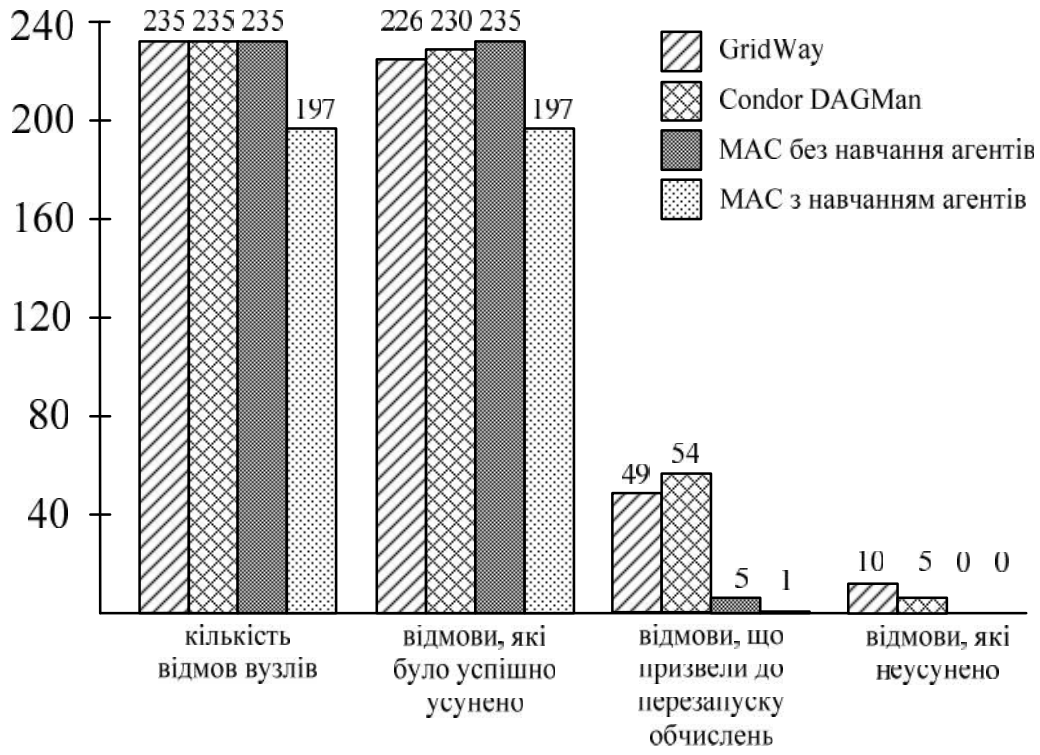


Рисунок 3.2 – Усунення відмов компонентів системи управління

У той же час GridWay та CondorDAGMan очікували відновлення своїх компонентів, що відмовили, і в деяких випадках перезапускали обчислювальні процеси. Навчання агентів дозволило покращити результати, отримані MAC, що функціонує без застосування такого навчання.

У багатьох випадках відмови компонентів керуючих систем явно чи опосередковано зумовлені відмовами обчислювальних вузлів. Результати, представлені рисунку 3.2, демонструють істотне зниження кількості відмов компонентів MAC з навчанням агентів більш, ніж 15% проти трьома іншими випадками.

Число відмов, що призвели до перезапуску обчислень, становить 1 та 5 відповідно для МАС з навчанням агентів та без нього. У той же час кількість таких відмов для GridWay та CondorDAGMan значно вища (49 та 54 відмови відповідно).

У роботі МАС як із навчанням агентів, так і без нього немає неусунених відмов компонентів систем. У процесі функціонування GridWay та Condor DAGMan можна бачити 10 та 5 неусунених відмов відповідно.

У кваліфікаційній роботі ефективність використання ресурсів характеризується середнім завантаженням процесорів вузлів при обробці потоків завдань різними керуючими системами (рисунок 3.3). Очевидно, що навчання агентів дозволяє їм виводити ненадійні вузли з РОС, підвищуючи тим самим корисне обчислювальне навантаження на вузли, що залишилися, і збільшуючи середнє завантаження процесора.

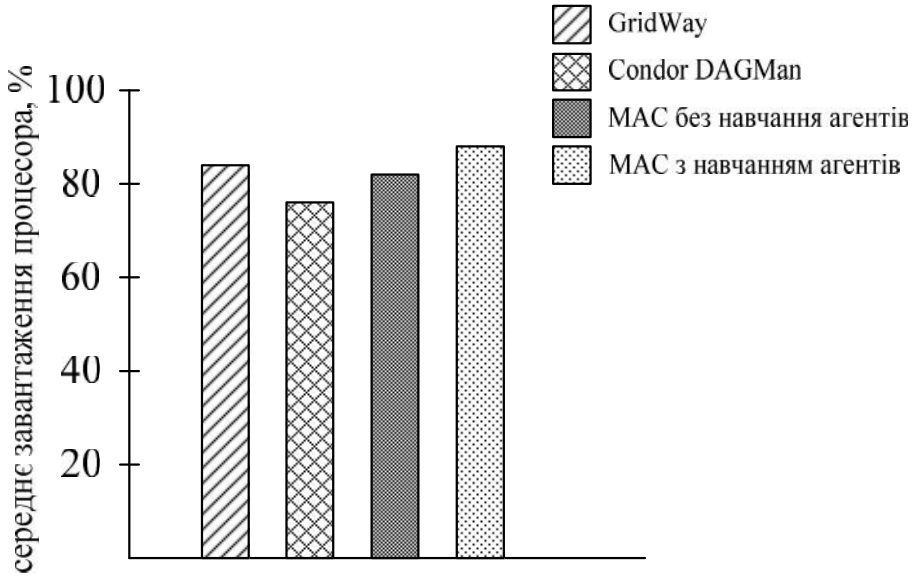


Рисунок 3.3 – Середнє завантаження процесора

Середнє завантаження процесора вузлів експериментальної РВС під управлінням систем GridWay та Condor DAGMan становить відповідно 81% та 77%. Застосування МАС з навчанням агентів і без нього дозволило підвищити цей показник до 87% та 82% відповідно.

## 3.2 Балансування завантаження обчислювальних ресурсів

У межах однорідної обчислювальної системи, наприклад, на кластері досить ефективне планування обчислень та розподіл ресурсів може бути побудований на базі системи керування завданнями. Зазвичай, системи управління завданнями підтримують кілька черг завдань. Завдання розподіляються по чергах залежно від категорії користувача, пріоритету завдання, класу завдання та його вимог до ресурсів.

Для ефективного використання обчислювальних ресурсів необхідне балансування їхнього навантаження. Використання ресурсів не є ефективним, якщо одна частина вузлів перевантажена, а інша частина вузлів простоює. Підтримка міграції завдань між вузлами дає змогу збалансувати навантаження ресурсів. Міграція забезпечує можливість переривання, переміщення в інші вузли та повторного запуску завдання. Вона може бути обумовлена перевантаженням одних вузлів та простоєм інших. Проте організація міграції завдань між обчислювальними кластерами з різними системами управління завданнями є нетривіальною проблемою різнорідної РОС.

Застосування МАС дозволяє частково вирішити цю нетривіальну проблему. Як приклад розглянуто обробку реального та синтетичного потоку завдань в експериментальній РОС. Характеристики завдань синтетичного потоку (час надходження, число ядер, розмір оперативної та дискової пам’яті, час виконання та ін.) повністю відповідають характеристикам завдань реального потоку.

Рисунки 3.4 та 3.5 ілюструють значення розподілу завдань реального потоку в пулах 1-3 протягом доби щогодини за допомогою традиційних засобів HTCondor та PBS Torque, а також із застосуванням додаткового рівня управління, що реалізується за допомогою МАС для синтетичного потоку завдань. Ці значення більш детально викладено у таблицях 3.4 та 3.5.

Таблиця 3.4 – Значення навантажень, які отримано за допомогою традиційних засобів HTCondor та PBS Torque

| час доби | кількість завдань | | | час доби | кількість завдань | | |
|---|---|---|---|---|---|---|---|
| | пул 1 | пул 2 | пул 3 | | пул 1 | пул 2 | пул 3 |
| 1 | 903 | 27 | 23 | 13 | 1001 | 125 | 89 |
| 2 | 6900 | 3949 | 3957 | 14 | 1488 | 25 | 42 |
| 3 | 7114 | 786 | 796 | 15 | 1867 | 325 | 21 |
| 4 | 3947 | 13 | 11 | 16 | 2036 | 78 | 96 |
| 5 | 1986 | 7 | 4 | 17 | 421 | 100 | 98 |
| 6 | 328 | 5 | 2 | 18 | 749 | 7 | 3 |
| 7 | 2093 | 26 | 29 | 19 | 756 | 138 | 53 |
| 8 | 2852 | 6 | 3 | 20 | 631 | 86 | 53 |
| 9 | 1549 | 8 | 6 | 21 | 1294 | 86 | 84 |
| 10 | 985 | 989 | 981 | 22 | 2253 | 632 | 548 |
| 11 | 312 | 7 | 5 | 23 | 1752 | 2122 | 864 |
| 12 | 53 | 4 | 2 | 24 | 116 | 53 | 64 |

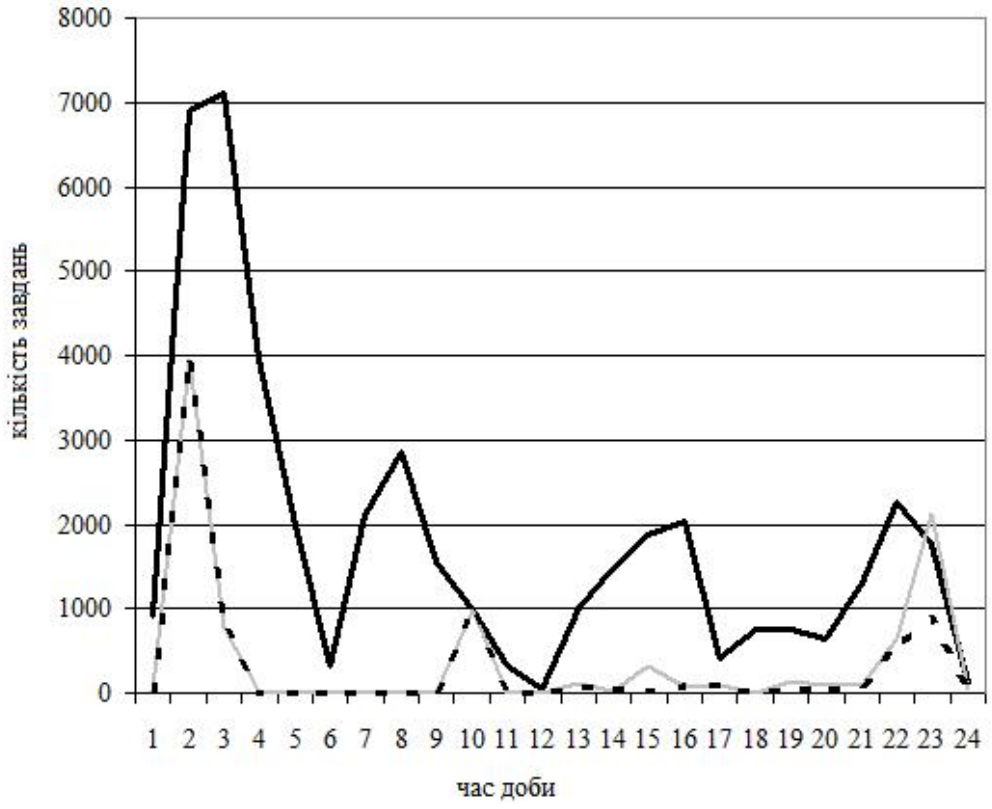


Рисунок 3.4 – Балансування завантаження за допомогою традиційних засобів HTCondor та PBS Torque, які використовуються в пулах ресурсів РОС

Таблиця 3.5 – Значення навантажень, які отримано із застосуванням додаткового рівня керування, що реалізується за допомогою МАС

| час доби | кількість завдань | | | час доби | кількість завдань | | |
|---|---|---|---|---|---|---|---|
| | пул 1 | пул 2 | пул 3 | | пул 1 | пул 2 | пул 3 |
| 1 | 1001 | 563 | 112 | 13 | 1354 | 812 | 436 |
| 2 | 3526 | 2997 | 2085 | 14 | 1439 | 893 | 258 |
| 3 | 3126 | 3129 | 996 | 15 | 1445 | 845 | 856 |
| 4 | 1968 | 865 | 726 | 16 | 1832 | 453 | 441 |
| 5 | 1129 | 559 | 96 | 17 | 982 | 459 | 429 |
| 6 | 981 | 568 | 572 | 18 | 713 | 721 | 643 |
| 7 | 1006 | 1987 | 396 | 19 | 802 | 725 | 229 |
| 8 | 2201 | 1801 | 432 | 20 | 538 | 731 | 326 |
| 9 | 1009 | 1693 | 24 | 21 | 1136 | 26 | 24 |
| 10 | 1083 | 967 | 972 | 22 | 1835 | 1395 | 1268 |
| 11 | 991 | 567 | 559 | 23 | 1865 | 1973 | 1863 |
| 12 | 1242 | 786 | 775 | 24 | 123 | 258 | 15 |

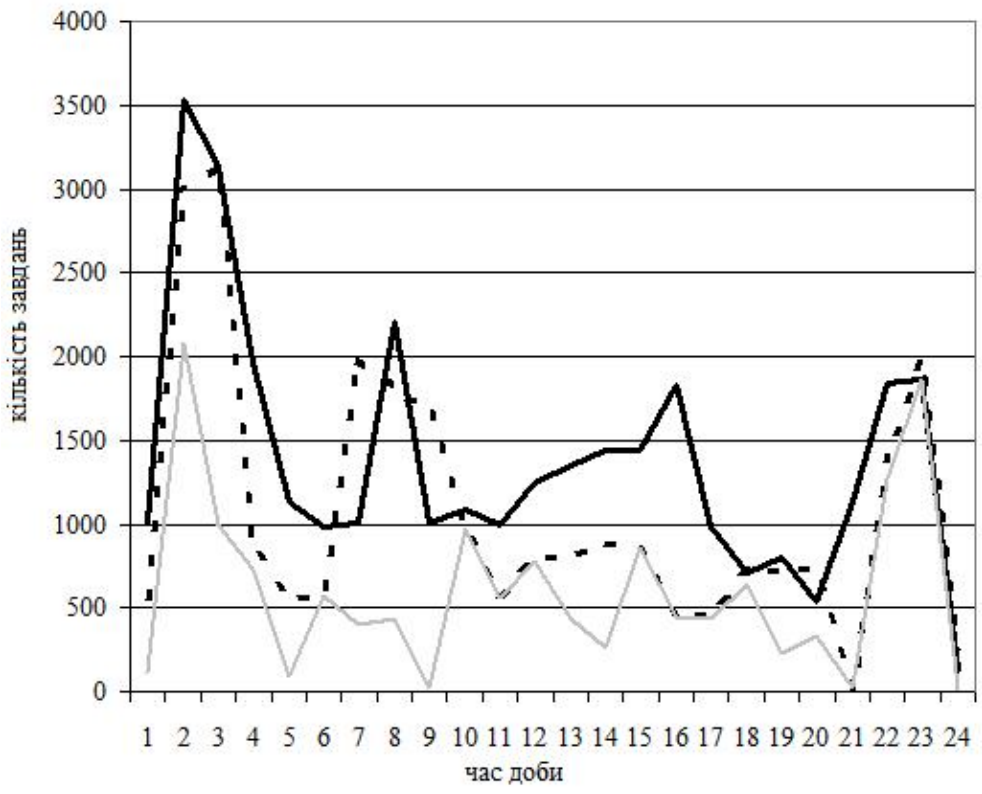


Рисунок 3.5 – Балансування навантаження із застосуванням додаткового рівня керування, що реалізується за допомогою МАС

З метою покращення балансування завантаження ресурсів МАС здійснювала перерозподіл обчислювального навантаження. Завдання можна було призначати на ресурс, що був відмінний від ресурсу, на якому виконувався в процесі обробки реального потоку завдань, тільки в тому випадку, якщо завдання, що перерозподіляється, зберігало належність певному для нього класу завдань.

Рисунок 3.6 демонструє покращення середнього завантаження процесора при керуванні завданнями за допомогою МАС порівняно з традиційними системами керування завданнями, що використовуються на практиці в пулах 1-3. Таке поліпшення отримано за рахунок перерозподілу завдань з невеликим часом виконання та мінімальним числом необхідних ядер з пулу 1 в пули 2 та 3.

Даний перерозподіл завдань забезпечив просування та запуск виконання великих завдань у пулі 3, а також збільшення завантаження вузлів пулів 2 та 3 відповідно. В результаті велика кількість завдань, які залишалися в черзі на практиці в кінці розглянутого періоду часу, були виконані при обробці синтетичного потоку завдань.


Рисунок 3.6 – Середнє завантаження процесора

Загалом перерозподіл завдань дозволив підвищити середнє завантаження процесорів у пулах 1-3 на 8%, 15% та 12% відповідно. При цьому покращення середньоквадратичного відхилення завантаження процесора по годинам від середнього завантаження у разі мультиагентного управління склало більше 58%, 36% та 3% для пулів 1-3 відповідно порівняно з використанням лише систем HTCondor та PBS Torque. У свою чергу покращення балансування завантаження призвело до скорочення часу виконання потоку завдань загалом на 17%.

# ВИСНОВКИ

У першому розділі розглянуто загальні питання, що пов’язані з теоретичною та практичною складовою організації та застосування МАС для управління потоками обчислювальних завдань в РОС. Також розглянуто ряд відомих МАС (EMERALD, GAMA, JADE, JADEx та інші системи), визначені їх переваги та недоліки. На основі цих даних проведено детальний порівняльний аналіз методів та засобів організації таких систем та обґрунтована необхідність розробки нового інструментального комплексу МАС для управління потоками обчислювальних завдань. Далі сформульовані системні та функціональні вимоги до інструментального комплексу що розробляється.

У другому розділі розглянуто питання, пов’язані з розробкою мультиагентних моделей, методів, алгоритмів та інструментальних засобів керування РОС. Наведена структура МАС для керування потоками завдань та формалізовано концептуальну модель такої системи. В наслідок аналізу концептуальної моделі та виявлення деяких незручностей було вирішено розширити її за рахунок введення нової автоматної модель поведінки агентів [53]. Запропоновано алгоритм перерозподілу ресурсів середовища та створено систему машинного навчання агентів, що базується на використання гібридної моделі подання та застосування знань.

Третій розділ присвячено обчислювальним експериментам, які показують суттєве підвищення стійкості до відмови. Застосування МАС дозволяє покращити балансування обчислювальних ресурсів, а також підвищити середнє завантаження процесорів вузлів експериментальної РОС. Підвищення ефективності використання ресурсів зумовлює скорочення часу вирішення наукових та прикладних завдань у РОС, а також інших важливих характеристик з погляду теорії масового обслуговування.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Lewis P.R., Marrow P., Yao X. Resource allocation in decentralised computational systems: an evolutionary market-based approach // Autonomous Agents and Multi-Agent Systems. 2010. Vol. 21, № 2. P. 143-171.

2. Kravari K., Bassiliades N. A survey of agent platforms // Journal of Artificial Societies and Social Simulation. 2015. Vol. 18, № 1.

3. GridWay Metascheduler [Електронний ресурс]. Режим доступу: http://www.gridway.org.

4. Tannenbaum T.A. Distributed Job Scheduler // Beowulf Cluster Computing with Linux. MA: The MIT Press, 2002. 350 p.

5. Wooldridge M. Agent-based software engineering // IEEE Proceedings Software Engineering. 1997. Vol. 144(1). P. 26-37.

6. Городецкий В.И. Многоагентные системы: современное состояние исследований и перспективы применения // Новости искусственного интеллекта. 1996. С. 44-59.

7. Городецкий В.И. Многоагентные системы: основные свойства и модели координации поведения // Информационные технологии и вычислительные системы. 1998. С. 22-34.

8. Nwana H.S. Software agents: an overview // The Knowledge Engineering Review. 1996. Vol. 11, № 3. P. 205-244.

9. Russell S.J., Norvig P. Artificial Intelligence: A Modern Approach – Pearson Education, 2003. 1132 p.

10. Городецкий В.И., Скобелев П.О., Бухвалов О.Л. Промышленные применения многоагентных систем: прогнозы и реалии // Проблемы управления и моделирования в сложных системах: труды XVIII Междунар. конф. Самара: Изд-во ООО «Офорт», 2016. С. 137-162.

11. Городецкий В.И. и др. Современное состояние и перспективы индустриальных применений многоагентных систем // Управление техническими системами и технологическими процессами. 2017. С. 94-157.

12. Ковтуненко А.С., Масленников В.А. Создание распределенных информационно-управляющих систем на базе агентно-ориентированного подхода // Материалы ВСПУ-2014. – Институт проблем управления им. В.А. Трапезникова РАН, 2014. С. 8984-8994.

13. Городецкий В.И. Самоорганизация и многоагентные системы // Известия РАН. Теория и системы управления. 2012. № 2. С. 92-120.

14. Васильев В.И., Ильясов Б.Г. Интеллектуальные системы управления: теория и практика. Москва: Радиотехника, 2009. 392 с.

15. Массель Л.В., Гальперов В.И. Проектирование и разработка многоагентной системы оценивания состояний электроэнергетических систем // Вестник Иркутского технического университета. 2015. Т. 10, № 105. С. 27-33.

16. Unified Modeling Language (UML) [Електронний ресурс]. Режим доступу: https://www.uml.org.

17. Java [Електронний ресурс]. Режим доступу: https://www.java.com.

18. Camazine S. et al. Self-Organization in Biological Systems. Princeton University Press: USA, 2001. 562 p.

19. Di Marzo Serugendo G., Gleizes M.-P., Karageorgos A. Self-organization in multi-agent systems // The Knowledge Engineering Review. Cambridge University Press, USA, 2005. Vol. 20, № 2. P. 165-189.

20. Ковтуненко А.С., Масленников В.А., Валеев С.С. Агентно-ориентированная программная архитектура распределенной обработки и хранения потоковых данных // Proceedings 2nd International Conference «Intelligent Technology Information Process. Manag. 2014. P. 65-70.

21. Weiss Ed. G. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA, USA: MIT Press, 1999. 648 p.

22. YarKhan A., Dongarra J., Seymour K. GridSolve: The Evolution of a Network Enabled Solver // Grid-based problem solving environment. 2007. Vol. 239. P. 215-224.

23. Laxmi C.V.T.E.V., Somasundaram K. Application Level Scheduling

(AppLeS) in Grid with Quality of Service (QoS) // Journal of Grid Computing. 2014. Vol. 5, № 2. P. 1-10.

24. Shi Z. Advanced Artificial Intelligence. Hackensack: World Scientific Publishing, 2011. 613 p.

25. Singh A., Malhotra M. Agent Based Framework for Scalability in Cloud Computing // International Journal of Computer Science Engineering and Technology. 2012. Vol. 3, № 4. P. 41-45.

26. Rezaee A. A Multi-Agent Architecture for QoS Support in Grid Environment // Journal of Computer Science. 2008. Vol. 4, № 3. P. 225-231.

27. Frey J. Condor-G et al. A Computation Management Agent for Multi-Institutional Grids // Cluster Computing. 2002. Vol. 5. P. 237-246.

28. Unland R., Klusch M., Calisti M. Software Agent-Based Applications, Platforms and Development Kits // Birkhauser Verlag, 2005. 455 p.

29. Filimonchuk T., Volk M., Ruban I., Tkachov V. Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment. Eastern-European Journal of Enterprise Technologies. Information and controlling system. 2016. Vol. 3/9 (81). P.45-53.

30. Finin T. et al. KQML as an agent communication language // Proceedings of the Third International Conference on Information and Knowledge Management. ACM, 1994. P. 456.

31. Labrou Y., Finin T. Semantics and Conversations for an Agent Communication Language // Readings in Agents. 1998. P. 235-242.

32. Badica C., Al E. Software Agents: Languages, Tools, Platforms // ComSIS Consort. 2011. Vol. 8, № 2. P. 255-298.

33. Ruban I., Volk M., Filimonchuk T., Ivanisenko I., Risukhin M., Romanenkov Y. The Method for Ensuring the Survivability of Distributed Computing in Heterogeneous Computer Systems. 5th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, October 9-12, 2018. P. 233-237.

34. Kantamneni A. et al. Survey of multi-agent systems for microgrid

control // Engineering Applications of Artificial Intelligence. 2015. Vol. 45. P. 192-203.

35. Gupta R., Kansal G. A Survey on Comparative Study of Mobile Agent Platforms // International Journal of Computer Science Engineering and Technology. 2011. Vol. 3(3). P. 1943-1948.

36. Acronymics Inc. AgentBuilder [Електронний ресурс]. Режим доступу: https://www.agentbuilder.com.

37. Delft University of Technology. AgentScape [Електронний ресурс]. Режим доступу: http://www.agentscape.de.

38. Raytheon BBN Technologies. Cognitive Agent Architecture [Електронний ресурс]. Режим доступу: http://www.cougaar.world.

39. Bhamra G.S., Verma A.K., Patel R.B. Intelligent Software Agent Technology: An Overview // International Journal of Computer Applications. 2014. Vol. 89, No. 2. P. 19-31.

40. Kravari K., Kontopoulos E., Bassiliades N. EMERALD: A multi-agent system for knowledge-based reasoning interoperability in the semantic web // Lecture Notes in Computer Science. 2010. Vol. 6040. P. 173-182.

41. Grignard A. et al. GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation // The 16th International Conference on Principles and Practices in Multi-Agent Systems. 2013. Vol. 8291. P. 242-258.

42. Telecom Italia. Java Agent DEvelopment Framework [Електронний ресурс]. Режим доступу: https://jade.tilab.com.

43. University H. JADEx [Електронний ресурс]. Режим доступу: https://www.activecomponents.org.

44. Multiagent Development Kit [Електронний ресурс]. Режим доступу: http://www.madkit.net.

45. Bellifemine F. et al. Jade – A Java Agent Development Framework // Multi-Agent Programming. 2005. Vol. 15. P. 125-147.

46. Braubach L., Pokahr A., Lamersdorf W. Jadex: A BDI agent system combining middleware and reasoning // Software Agent-Based Applications,

Platforms and Development Kits. Basel-Boston-Berlin: Birkhauser-Verlag, 2005. P. 143-168.

47. Kravari K., Bassiliades N. A survey of agent platforms // Journal of Artificial Societies and Social Simulation. 2015. Vol. 18, № 1.

48. Филимончук Т.В., Волк М.А., Казмина Д.Р., Ольшанская Т.И., Рисухин М.В. Модифицированная информационная технология распределения заданий на ресурсы для систем облачных вычислений. Сучасний стан наукових досліджень та технологій в промисловості. 2019. №1(7). С. 121-128.

49. Bychkov I. et al. Conceptual Model of Problem-oriented Heterogeneous Distributed Computing Environment with Multi-agent Management // Procedia Computer Science. Elsevier. 2017. Vol. 103. P. 162-167.

50. Bychkov I. et al. Agent-based approach to monitoring and control of distributed computing environment // Lecture Notes in Computer Science. Springer, Cham. 2015. Vol. 9251. P. 253-257.

51. Tel G. Introduction To Distributed Algorithms. Cambridge University Press. 2000. 596 p.

52. Бондаренко А.А., Якобовский М.В. Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью локальных контрольных точек // Вестник ЮУГУ. Сер. Вычислительная математика и информатика. 2014. Т.3, №3. С. 20-36.

53. Філімончук Т.В., Ведмеденко І.О. Модель мультиагентного керування розподілом завдань на обчислювальні ресурси. Проблеми інформатизації: Тези доповідей дев’ятої міжнародної науково-технічної конференції. Том 2, секція 4. Черкаси: ЧДЕУ, Харків: НТУ «ХПІ», Баку: ВА ЗС АР, Бельсько-Бяла: УТіГН, ДП «ПД ПКНДІ АП». 2021. С. 97.