МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
НАЦІОНАЛЬНИЙ ЦЕНТР «МАЛА АКАДЕМІЯ НАУК УКРАЇНИ»

НАУКОЄМНІ ТЕХНОЛОГІЇ ОПТИМІЗАЦІЇ ТА КЕРУВАННЯ В ІНФОКОМУНІКАЦІЙНИХ МЕРЕЖАХ

Колективна монографія

Під загальною редакцією
В.М. Безрука, Л.С. Глоби, О.Є. Стрижака

Київ

2019

УДК 621.391+004.7
Н34

Рекомендовано до друку Вченою радою Національного центру «Мала академія наук України» (протокол № 1 від 15 січня 2019року), науково-технічною радою Харківського національного університету радіоелектроніки (*протокол № 2 від 25.06.2018 р.)*

Рецензенти:

Бараннік В.В., д.т.н., проф., начальник кафедри бойового застосування та експлуатації АСУ Харківського Національного університету Повітряних Сил імені Івана Кожедуба; Климаш М.М., д.т.н., проф., завідувач кафедри телекомунікацій Національного університету «Львівська політехніка».

Н34 **Наукоємні технології оптимізації та керування в інфокомунікаційних мережах** : монографія / Під загальною редакцією В.М. Безрука, Л.С. Глоби, О.Є Стрижака. – К.: Інститут обдарованої дитини НАПН України, 2019. – 194 с.
ISBN 978-617-7734-02-3

В монографії викладені сучасні результати досліджень ряду авторів з наукоємних технологій оптимізації та управління в інфокомунікаційних мережах. Розглядаються особливості методів багатокритеріальної оптимізації систем з урахуванням сукупності показників якості, а також особливості та результати вирішення задач управління при передачі та обробці інформації для різних типів інфокомунікаційних мереж.

УДК 621.391+004.7

ISBN 978-617-7734-02-3

ЗМІСТ

ВСТУП

Інфокомунікації – сучасна концепція конвергентної взаємодії телекомунікацій та інформаційних технологій з метою забезпечення своєчасної якісної передачі та обробки інформації.

Із впровадженням у світі технологій 4G LTE та розвитком тенденції переходу до технологій 5G постає питання збільшення швидкості та ефективності передачі та обробки даних. Все більшого поширення набувають сервіси Internet of Things (Інтернет речей) та сервіси міжмашинної взаємодії (M2M). Ці сервіси продукують великі масиви даних у режимі реального часу, що тягне за собою появу концепції BigData. Поряд із цим у світі гостро стоїть проблема економії електроенергії, що наразі споживається центрами обробки даних у величезних обсягах.

Такі виклики сучасного світу інфокомунікацій спричиняють необхідність пошуку нових наукоємних технологій оптимізації та управління при вирішенні задач планування та проектування інфокомунікаційних мереж. В монографії викладені деякі сучасні результати досліджень ряду авторів з питань оптимізації та управління в інфокомунікаційних мережах з урахуванням сукупності техніко-економічних вимог, включаючи швидкість передачі та обробки даних, енергоефективність, оптимальне використання ресурсів та інше. Це визначає актуальність опублікування даної колективної монографії, в якій викладені деякі результати досліджень авторів по вирішенню задач оптимізації та управління в інфокомунікаційних мережах з використанням сучасного математичного апарату та високоефективних обчислювальних засобів. Кожен розділ монографії являє собою завершене викладення результатів досліджень, отриманих авторами цього розділу.

Матеріали 1 розділу підготував Безрук В.М. У цьому розділі розглядаються деякі особливості методів багатокритеріальної оптимізації, що використовуються при розв'язанні різних типів задач оптимізації проектних рішень у мережах зв'язку з урахуванням сукупності показників якості. Наведено методи вирішення багатокритеріальних задач оптимізації при застосуванні безумовного критерію переваги, що приводить до отримання підмножини Парето-оптимальних рішень на множині допустимих варіантів. Для звуження підмножини Парето до єдиного переважного проектного варіанту застосовується умовний критерій переваги. Розглянуто різні методи формування умовного критерію переваги з урахуванням додаткової інформацію від експертів, що дають можливість звуження підмножини Парето до єдиного переважного рішення. У наступних розділах 2 – 4 наводяться деякі приклади, що ілюструють практичні особливості використання розглянутих методів при багатокритеріальному аналізі і виборі оптимальних проектних варіантів для різних засобів зв'язку.

Матеріали 2 розділу підготували Чеботарьова Д.В., Безрук В.М. У цьому розділі розглянуто особливості застосування методів багатокритеріальної

оптимізації на етапі номінального планування радіомережі системи стільникового мобільного зв’язку (СМЗ) 2-го та 3-го покоління. Вирішена задача оптимізації радіомережі СМЗ як задача дискретного вибору Парето-оптимальних проектних варіантів при врахуванні сукупності показників якості. Розглянуто також особливості застосування методів багатокритеріальної оптимизації при номінальному плануванні транспортної мережі СМЗ. Наведені результати вибору оптимальної топології транспортної мережі СМЗ з урахуванням сукупності показників якості.

Матеріали 3 розділу підготували Безрук В.М., Скорик Ю.В., Власова В.О. У даному розділі розглянуто особливості вибору переважного проектного варіанту з урахуванням сукупності показників якості при застосуванні умовного критерія переваги, що заснований на використанні методу аналізу ієрархій (МАІ). Цей метод полягає в отриманні суджень експертів по парним порівнянь різних елементів проблеми вибору. В результаті обробки цих даних за строго формалізованими процедурами обчислюється вектор глобальних пріоритетів, компоненти якого визначають пріоритетність вибору переважного варіанту системи. Досліджено практичні особливості застосування МАІ для вибору переважного варіанту на прикладі різних типів засобів зв'язку, зокрема, мовних кодеків для IP телефонії, системи масового обслуговування заявок у вузлах комутації мережі, проектних варіантів систем мобільного зв'язку 3-го покоління, технологій мобільного зв'язку 4-го покоління, типів мобільних телефонів, протоколів маршрутизації в ad-hoc мережах, модуляції в цифрових системах зв'язку, систем телевізійного мовлення стандартів DVB-T і ATSC, протоколів маршрутизації в бездротовій сенсорно-актуаторній мережі.

Матеріали 4 розділу підготували Безрук В.М., Гальченко К.Р. У даному розділі розглядаються особливості математичного формулювання та вирішення задач оптимальної маршрутизації у мережах зв’язку. Оскільки, практичні умови роботи мереж зв’язку вимагають врахування декількох показників якості (метрик) при виборі оптимальних маршрутів, тому розглянуто багатокритеріальний підхід до вирішення задач оптимальної маршрутизації у мережах зв’язку. Наведено метод формування підмножини Парето-оптимальних варіантів маршрутизації, що дають можливість організації багатошляхової маршрутизації у мережі зв’язку. Наводиться приклад вирішення задачі оптимальної маршрутизації у мультисервісній мережі зв’язку з урахуванням сукупності показників якості.

Матеріали 5 розділу підготували Скулиш М.А., Суліма С.В. У даному розділі запропоновано алгоритм оптимального розподілу ресурсів у віртуалізованих мережах. Метою алгоритму надання ресурсів є виділення достатньої їх кількості для мережевих функцій, так що їх SLA можна задовольнити навіть у присутності пікового навантаження. В основі будь-якого алгоритму надання ресурсів лежать два питання: скільки ресурсів надавати і коли. Ці питання глибоко досліджуються авторами розділу.

Матеріали 6 розділу підготували Глоба Л.С., Степурін О.В., Гвоздецька Н.А., Прокопець В.А. У даному розділі запропоновано підхід до підвищення енергоефективності обчислень для серверного кластера як інформаційно-телекомунікаційної одиниці інфраструктури ЦОД, який відрізняється одночасним врахуванням параметрів енергоефективності та продуктивності при розподілі задач. Розроблено алгоритм підвищення енергоефективності обчислень на основі запропонованого підходу, який складається з етапу попередньої індивідуальної атестації серверного кластера та етапу енергоефективного розподілу задач. Врахування індивідуальних залежностей енергоспоживання серверів від їх завантаженості є основною відмінною рисою запропонованого підходу.

Матеріали 7 розділу підготували Глоба Л.С., Прокопець В.А., Гвоздецька Н.А. У даному розділі проаналізовано ефективність методу динамічного розподілу ресурсів у віртуалізованій мережі мобільного оператора України з використанням механізмів підвищення енергоефективності обробки даних. Основною метою методу динамічного розподілу ресурсів є підтримка обсягу ресурсів, достатнього для виконання вимог SLA (Service Level Agreement – Угода про рівень обслуговування) при уникненні надмірності. Це дозволяє більш ефективно використовувати апаратні ресурси для економії електроенергії та зменшення викидів CO_2. Ефективність методу перевірено для віртуалізованого ядра EPC (Evolved Packet Core) мережі мобільного зв'язку України. Для додаткового підвищення енергоефективності обробки даних у дослідженні застосовано підхід PCPB (Power Consumption and Performance Balance), який запропоновано у розділі 6.

Матеріали 8 розділу підготували Кірюшкін Р.О., Курдеча В.В., Глоба Л.С. У даному розділі проведено огляд основних методів вертикального хендовера та зроблений висновок про неефективність існуючих рішень у бездротових гетерогенних мережах з багатокласовими груповими викликами мобільних вузлів. Описано модифікацію методу вертикального хендовера за рахунок зміни алгоритму прийняття рішень для різних типів викликів мобільного вузла що дозволило підвищити ефективність зв’язку у гетерогенних безпроводових мережах. Розглянуто створену математичну модель алгоритму прийняття рішення, який застосовує методи TOPSIS і MULTIMOORA. Проведено імітаційне моделювання методу прийняття рішення, який базується на запропонованій математичній моделі та доведено його ефективність у гетерогенних бездротових мережах.

6. ОПТИМІЗАЦІЯ ОБЧИСЛЕНЬ У ДАТА–ЦЕНТРІ ЗА КРИТЕРІЄМ ЕНЕРГОЕФЕКТИВНОСТІ

Гвоздецька Н.А., Глоба Л.С., Прокопець В.А., Степурін О.В.

Технології 4G LTE (4th generation of mobile communications Long Term Evolution) мобільного зв'язку 4 покоління на сьогоднішній день широко впроваджуються у світі. Оператори зв'язку та експерти висловлюють своє бачення концепції 5G (5th generation of mobile communications). Вона включає в себе розвиток та поширення технології Інтернету речей (Internet of Things, IoT) та міжмашинної взаємодії M2M. Ці сервіси продукують великі масиви даних, що потребують високої швидкості обробки. Високої швидкості передачі та обробки потребують також сервіси високоякісного відео у форматі 4K та 8K, сервіси віртуальної реальності.

Водночас, обробка даних потребує значних енергозатрат. Згідно джерел [1, 2] кількість енергії, що була спожита центрами обробки даних (серверами, сховищами, апаратурою зв'язку та охолодження) по всьому світу в останні роки мала такий розподіл: становила близько 70,8 ТВт у 2000 році; 152,5 ТВт – у 2005 році (спостерігалось зростання споживання на 115%); зросла до 271,8 ТВт у останні роки. При цьому частка цієї потужності у загальній кількості спожитої у світі становить в середньому 1,5%.

У зв'язку з необхідністю підвищення енергоефективності та зростанням вимог до швидкості обробки інформації гостро постає проблема розробки методів енергоефективної обробки даних у ЦОД, які б водночас позитивно впливали на продуктивність обчислень.

6.1. Аналіз існуючих методів підвищення енергоефективності обчислень у дата–центрі

Підвищення енергоефективності обробки даних є надзвичайно актуальною проблемою сьогодення. Тому, на сьогоднішній день вже існує багато підходів до підвищення енергоефективності обчислень у ЦОД.

6.1.1. Класифікація існуючих підходів до підвищення енергоефективності обробки даних у ЦОД

Існуючі підходи можна класифікувати за рівнем їх застосування в системі, динамічністю прийняття рішення, сферою використання, тощо. Одним із можливих доцільних варіантів класифікації є варіант, запропонований у роботі [3]. Дану класифікацію наведено на рис. 6.1.

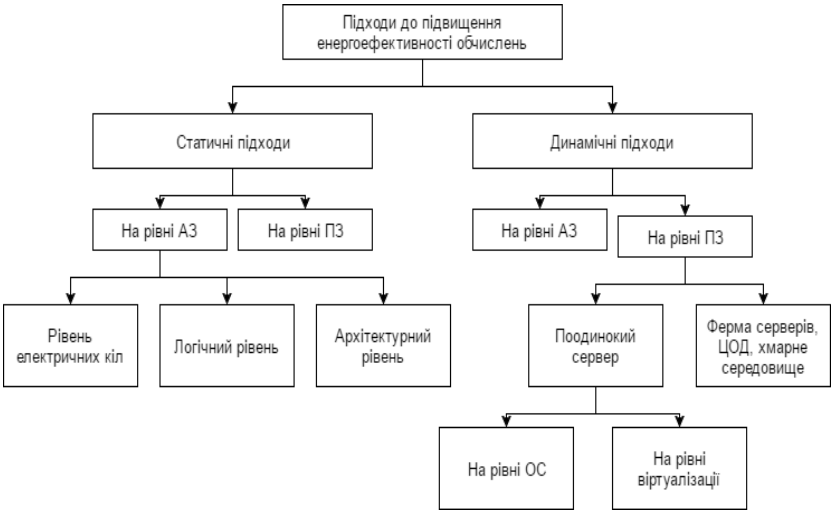


Рис. 6.1. Класифікація підходів до вирішення проблеми енергоефективності обробки даних

За наведеною класифікацією, існуючі підходи до підвищення енергоефективності обробки даних можна розділити на статичні та динамічні за характером прийняття рішень. Аналіз існуючих рішень обох підходів дає змогу зрозуміти, що динамічні рішення мають вищу ефективність.

Як статичні, так і динамічні підходи можуть бути застосовані як на рівні апаратного забезпечення (АЗ), так і на рівні програмного забезпечення (ПЗ). При цьому, дані підходи класифікують також за областю застосування (хмарне середовище, поодинокий сервер, тощо).

Крім того, рішення, націлені на підвищення енергоефективності, можуть бути застосовані на різних рівнях організації обчислювальної системи (логічному, архітектурному, операційної системи (ОС) тощо).

Підходи, що застосовуються на рівні апаратного забезпечення, є більш складними в реалізації та вимагають значних додаткових фінансових вкладень, у той час як програмні підходи можуть бути застосовані за умов обмежених матеріальних та технічних ресурсів.

В обчислювальних системах ефективність обробки даних залежить у тому числі від характеру вхідного навантаження на вузол обробки, поточного стану обчислювальної системи та інших факторів, що змінюються динамічно під час роботи системи. Тому, застосування динамічних підходів здатне забезпечити більшу продуктивність у загальній енергоефективності системи, ніж статичних.

Динамічні підходи на рівні програмного забезпечення можуть бути застосовані як до поодинокого сервера (на рівні його операційної системи, наприклад), так і для набору серверів – серверного кластера. У даній роботі розглядається варіант застосування охарактеризованого підходу до серверного кластера, як частини інформаційно-телекомунікаційної інфраструктури ЦОД.

Серверний або обчислювальний кластер — це декілька незалежних обчислювальних машин, що використовуються спільно, і працюють як одна система для вирішення різноманітних завдань, наприклад, для підвищення продуктивності, забезпечення надійності, спрощення адміністрування тощо. Обчислювальний кластер потрібен для збільшення швидкості розрахунків за допомогою паралельних обчислень [4].

Рішення для серверного кластера може бути масштабоване на великі ЦОД. Серверний кластер може бути гомогенним або гетерогенним. Ця властивість кластера залежить від фізичних характеристик вузлів, що входять до нього. Гетерогенний серверний кластер складається з вузлів, що мають різні фізичні параметри. Для таких кластерів постає проблема вибору найбільш доцільного вузла обробки серед множини можливих для розміщення певної конкретної задачі, тобто проблема балансування або планування навантаження (*англ.*scheduling).

6.1.2. Оптимізація процесу планування задач як можливий шлях підвищення енергоефективності обчислень у ЦОД

Для підвищення енергоефективності та продуктивності серверного кластера необхідно перш за все розподіляти навантаження між вузлами серверного кластера так, щоб досягти балансу між дотриманням вимог до енергоефективності та продуктивності обробки даних.

Проблема розподілу задач (балансування навантаження) вирішується за допомогою планувальника завдань (брокера). У даній роботі запропоновано підхід до підвищення енергоефективності обробки даних, що застосовується саме на етапі планування завдань та представляє собою алгоритм роботи брокера-планувальника завдань.

Серед існуючих алгоритмів балансування навантаження найбільш простим та розповсюдженим є алгоритм Round Robin. Згідно цього алгоритму всі завдання розподіляються по всім активним серверам за циклічним принципом. Цей алгоритм швидкий, адже не потребує знання про поточний розподіл ресурсів серед серверів. Але, оскільки завдання та вузли обробки (сервери) за даним алгоритмом не мають жодного пріоритету, алгоритм не здатний призначити завдання на обробку найбільш відповідному для цього серверу. Це є основним недоліком Round Robin [5].

Однією з успішних модифікацій цього алгоритму є Round Robin зі зваженими коефіцієнтами (WeightedRoundRobin) [5]. На відміну від простого алгоритму Round Robin, дана модифікація використовує знання про обчислювальні потужності вузлів обробки і, відповідно, до значень обчислювальних потужностей, присвоює вагові коефіцієнти серверам, що в подальшому враховується при розподілі завдань між ними.

Крім того, для розподілу завдань використовуються такі прості алгоритми як FIFO (FirstInFirstOut, FirstAvailable або перший доступний) та LeastConnection (LC) [6].

Алгоритм LC розподіляє кожне наступне вхідне завдання на обробку на сервер, що має найменшу кількість поточних з’єднань. Цей алгоритм

застосовується здебільшого у кластерах, де всі вузли мають однакову продуктивність. Це динамічний алгоритм планування, оскільки він бере до уваги параметр поточної завантаженості вузла.

Алгоритм FIFO розподіляє кожне наступне вхідне завдання на перший доступний сервер, тобто такий, що має достатньо ресурсів для обробки даного завдання. Цей алгоритм не враховує можливу різницю у параметрах вузлів обробки, не враховуючи, в тому числі, до уваги параметр енергоефективності. Основною перевагою даного підходу є його простота і легкість впровадження.

Усі вищеописані алгоритми не беруть до уваги параметр енергоспоживання кластерного вузла.

6.1.3. Аналіз енергоефективних алгоритмів планування задач

Серед енергоефективних підходів до планування навантаження у серверному кластері найбільш близькими до запропонованого у роботі підходу є такі: CTES [7]; EDRP [8]; Min_C [9]; The Most-Efficient-Server First Scheme [10]; «Алгоритм Р» [11].

Серед відомих алгоритмів балансування, що враховують параметр енергоспоживання, слід відмітити алгоритм CTES – Cooperative Two-Tier Energy-Aware Scheduling [7]. Автори розглядають дворівневий підхід до планування завдань із регулюванням швидкості їх виконання, з метою досягнення оптимального використання процесору, замість міграції завдань на інші вузли. Використовуються декілька стратегій планування з прогнозом виконання завдань для оптимального призначення їх на доступні машини. Результати моделювання, представлені в роботі, показують, що цей підхід зменшує загальне споживання енергії у серверному кластері.

Недоліком даного алгоритму є те, що автори вважали модель енергоспоживання сервера (функцію залежності енергоспоживання від навантаження центрального обчислювального вузла) лінійною. Насправді, така модель не точно відображає характер залежності спожитої потужності від завантаженості сервера. Натомість у даній роботі пропонується проведення попередньої атестації кожного сервера кластеру для отримання реальної залежності потужності споживання від завантаженості для кожного вузла кластеру.

Іншим прикладом алгоритму, що має на меті підвищення енергоефективності обробки задач у хмарних системах, EDRP – Energy and Deadline aware Resource Provisioning [8]. Автори зосередились на проблемі мінімізації затрат на хмарні системи, підвищуючи ефективність використання енергії, але гарантуючи терміни виконання задач користувача, визначених в умовах щодо якості обслуговування (SLA), беручи до уваги два типи завдань, незалежні пакетні передачі і завдання із залежностями.

Модель розрахунку споживання електроенергії у момент часу t включає статичне $P_{xstatic}(t)$ і динамічне $P_{xdynamic}(t)$ енергоспоживання. Обидві характеристики розраховуються на основі відсотку завантаження процесору $Util_x(t)$, в якому враховуються тільки параметри використовуваної машини Q-t. Недоліком запропонованого алгоритму є те, що автори не враховують

відмінностей між машинами, на яких виконуються завдання, і машинами, що знаходяться в режимі очікування.

У роботі [9] описано стратегію розподілу завдань, яка має назву Min_C. Дана стратегія враховує різноманітність завдань, що приходять на обробку та відповідну різноманітність ресурсів, яких вони потребують. Недоліком описаного у статті [9] алгоритму є те, що енергетична модель, яку використали автори, хоч і має нелінійний характер, близький до дійсності, проте використана модель має єдиний вигляд для всіх машин в незалежності від їхніх характеристик. Натомість, у даній роботі пропонується індивідуально визначати модель енергоспоживання для кожної машини окремо.

У роботі [10] описано алгоритм планування навантаження The Most-Efficient-Server First Scheme (MES-first). Згідно даного підходу, центральний планувальник задач сортує вузли кластера, базуючись на їх енергоспоживанні. Завдання для обробки розподіляються спершу на найбільш енергоефективні сервери, згодом на менш енергоефективні, згідно позиції сервера у відсортованому списку. Розподіл завдань припиняється, якщо не лишається завдань у черзі на обробку, або якщо черги до всіх серверів заповнені. Для ЦОД, у яких параметри всіх вузлів є однаковими, цей алгоритм має на меті розподілити якомога більше завдань на сервери, що знаходяться в активному стані, до моменту досягнення точки насичення, а вже потім залучати сервери, що знаходяться у режимі сну.

Вузол, на якому розміщується центральний планувальник, містить сортований список доступних серверів та їхні енергетичні профілі. У такому списку найбільш енергоефективні сервери розміщені у верхній частині списку. В момент прибуття завдання, воно призначається на обробку на сервер із вершини списку. Сервер, що отримує завдання на обробку, оновлює свій енергетичний профіль у центральному вузлі із планувальником. Після заповнення найбільш енергоефективного сервера завдання розподілятимуться на сервер, що займає наступну позицію у списку.

Недоліком запропонованого у роботі [10] підходу є те, що автори не враховують параметра продуктивності обробки завдань при їхньому розподілі. Може виникнути ситуація, за якої найбільш енергоефективні сервери виявляться найменш продуктивними. При розподілі задач виключно за параметром енергоефективності існує ймовірність значного програшу у швидкості обробки завдань та недотримання, відповідно, прийнятих SLA (*англ.*Service Level Agreement – домовленість про якість сервісу).

У роботі [11] описано підхід до енергоефективного розподілу завдань, що має назву «Алгоритм Р». Даний алгоритм автори пропонують застосовувати при розподілі завдань між кількома незалежними серверними кластерами. Згідно запропонованого авторами підходу, вхідним параметром для «Алгоритму Р» є значення H_{diff}, що визначає допустиму різницю між мінімальним та максимальним часом очікування завдання у черзі до кластера, за якого алгоритм буде давати економію електроенергії. Алгоритм вибиратиме кластер, у якому завдання буде оброблено із найменшими затратами енергії. Для кожного кластера планувальник навантаження C_j оцінює вартість електроенергії, що буде затрачена на обробку завдання, і параметр $h_j = (\sum_{k=1}^{N} S_k)/W_j$ – відношення загальної площі завдання до ширини кластера (цей параметр визначає оцінку

часу перебування завдання у черзі). Якщо $max(h_j) - min(h_j) > H_{diff}$, то завдання розподіляється на обробку на кластер з мінімальним значенням h_j. Інакше, завдання розподіляється на кластер із мінімальною вартістю енергії.

Автори роботи [11] розглядали у якості оцінюваних параметрів час очікування завдання у черзі та вартість електроенергії, вважаючи її різною для різних кластерів. У випадку застосування підходу у межах одного серверного кластера, оцінюваними параметрами можуть виступати час знаходження завдання у черзі до кожного із серверів та вартість обробки завдання на кожному окремому сервері.

Серед недоліків вищеописаного підходу можна виділити те, що автори не розглядають параметр продуктивності кожного окремого сервера як оцінюваний. Вибір вузла обробки здійснюється на основі поточного стану кластера та завантаженості його вузлів. Такий підхід може негативно вплинути на продуктивність обчислювальної системи в цілому.

6.2. Підхід до підвищення енергоефективності обчислень у центрі обробки даних

У процесі аналізу існуючих підходів до підвищення енергоефективності обробки задач у ЦОД було визначено, що для підвищення загальної енергоефективності та продуктивності обробки даних, найбільш доцільним є внесення змін у процес планування завдань. Крім того, для опису енергоефективного підходу до планування завдань, доцільно на початковому етапі розглянути його застосування у серверному кластері, як одиниці інформаційної інфраструктури ЦОД.

6.2.1. Введення основних термінів та понять

У даній роботі використані такі терміни та поняття:

Обчислювальний кластер або серверний кластер – набір комп'ютерів, що працюють разом і можуть розглядатись як єдина система.

Вузол серверного кластера – одиниця обчислювального кластера, представлена фізично однією ЕОМ:

$$ComputerCluster = \{N_j\},$$

де N_j - j-й вузол обчислювального кластера.

FLOPS – одиниця вимірювання продуктивності комп'ютера (кількість операцій з плаваючою комою, що можуть бути виконані за секунду) [12].

Завдання – набір елементарних операцій, що мають бути обробленими неподільно в обчислювальному кластері:

$$Task = \{task_i\}$$

Робота – завдання разом з вимогами до її виконання:

$$Jobs = \{job_i\} \rightarrow \left\{task_i,\ \left\{V_{req_i},\ k_{core_{req_i}},\ t_{i\_max}\right\}\right\},$$

де V_{req_i} – об'єм оперативної пам'яті для виконання завдання; $k_{core_{req_i}}$ – кількість ядер процесора, що вимагає завдання $task_i$ для свого

виконання;

t_{i_max} – максимальний час, за який завдання $task_i$ має бути оброблено.

6.2.2. Постановка завдання енергоефективного планування в математичному вигляді

Обчислювальний кластер складається з n вузлів $\{N_j\}$. Кожен вузол N_j характеризується:

V_j – об'ємом доступної оперативної пам'яті;

$flops_j$ – продуктивністю вузла N_j, що має k_{core_j} обчислювальних ядер.

$P_j = f_j(CPU_j)$ – функцією енергоспоживання від завантаженості процесора $f_j(CPU_j)$, що експериментально визначена для кожного вузла N_j.

Нове завдання $task_i$ приходить до системи в момент τ.

Кожне завдання потребує певних значень вищеназваних параметрів: $\{V_{req},\ k_{core_{req}},\ t_{max}\}$

$job_i \rightarrow task_i, \{V_{req_i},\ k_{core_{req_i}},\ t_{i_{max}}\}$

Необхідно розробити алгоритм планування завдань такий, що

$P_\Sigma \rightarrow min, t_{task\_i} \rightarrow min,$

де P_Σ – сумарна потужність спожита усім кластером,

t_Σ – час виконання набору із m задач.

6.2.3. Опис запропонованого алгоритму енергоефективного планування завдань

Запропонований підхід до енергоефективного планування завдань складається з двох основних етапів:

1. Етап попередньої атестації;
2. Етап планування завдань.

Етап попередньої атестації проводиться у серверному кластері періодично, при його налаштуванні. Цей етап передбачає попереднє визначення функцій *P = f(CPU)* індивідуально для кожного вузла кластера та формування описів вузлів, що розміщено на центральному вузлі із планувальником (брокером). Отримання залежності $P_j = f_j(CPU_j)$ детально описано у роботі [13]. Індивідуальне визначення функцій *P = f(CPU)* для кожного вузла обробки та їх подальше використання у процесі планування завдань є ключовою особливістю запропонованого підходу.

Другий етап передбачає вирішення оптимізаційного завдання за критеріями енергоефективності та продуктивності обробки завдань з використанням індивідуально визначених енергетичних моделей вузлів. Результатом цього процесу є розміщення кожного поточного завдання для обробки на сервер із оптимальними параметрами.

Запропонований підхід представлено у вигляді алгоритму планувальника завдань, що складається з таких кроків:

Крок 0. Попередня атестація кластера

Крок 1. Оцінка стану кластера в момент τ_{k-1}

Крок 2. Виключення із розгляду всіх непідходящих вузлів (за відсутністю ресурсів для обробки)

Крок 3. Знаходження масиву значень сумарного енергоспоживання кластера $P_\Sigma = \{P_{\Sigma_j}\}$ *при розміщенні завдання* $task_i$ *на кожен з вузлів* N_j

Крок 4. Сортування масиву вузлів N_j *за теоретичним вкладом у загальну спожиту потужність:* $N_P = \{N_{P_j}\}$

Крок 5. Сортування масиву вузлів, що лишилися, за продуктивністю (FLOPS): $N_{FLOPS} = \{N_{FLOPS_j}\}$

Крок 6. Призначення вагових коефіцієнтів (балів) за продуктивність та енергоефективність кожному вузлу

Крок 7. Розміщення завдання $task_i$ *на вузол із найбільшим сумарним ваговим коефіцієнтом.*

Детальний опис кожного кроку алгоритму наведено далі.

Крок 0.

В процесі початкового налаштування кластера та його підготовки до роботи, згідно запропонованого підходу, необхідно виміряти залежності спожитої кожним сервером електроенергії від завантаженості його центрального вузла обробки. Ця залежність може бути виміряна із використанням навантажувального тесту (наприклад, представленого у джерелі [14]) та представлена у вигляді функції:$P = f(CPU)$,

де *CPU* – Central Processor Utilization – навантаженість центрального процесора (вузла обробки).

У даній роботі запропоновано проводити подальшу інтерполяцію функцій $P = f(CPU)$ поліномами ступеня n, що докладно описується у далі.

Після визначення залежностей $P = f(CPU)$, згідно запропонованого підходу, відбувається формування опису кожного вузла. До опису входять такі дані:

- Функція енергоспоживання сервера від *CPU*;
- Загальна продуктивність сервера;
- Кількість ядер центрального процесора сервера;
- Об'єм оперативної пам'яті сервера.

Приклад сформованого опису вузла наведено на рис. 6.2 (функція $P = f(CPU)$ подана у вигляді коефіцієнтів полінома ступеня 4).

Node #1	
P = f(CPU)	0 2,1 -16 35,1
FLOPS	35,7*10^9
k_cores	4
V_RAM	4*10^9

Рис. 6.2. Приклад опису вузла сформований у результаті виконання кроку 0

Крок 1.

У момент часу τ нове завдання (робота) job_i надходить до кластера. У момент часу τ_{k-1} (попередня оцінка) необхідно виміряти такі параметри кожного вузла N_j:

1. $\Delta V_{j_{avail}} = V_j - V_{j\_used} -$ доступний об'єм оперативної пам'яті $j-$го вузла;

де V_{j_used} – оперативна пам'ять, зайнята на момент часу τ_{k-1};

2. $\Delta k_{core_{j_{avail}}} = k_{core_j} - k_{core_{j_{used}}} -$ кількість ядер процесора, доступних на вузлі N_j;

де $k_{core_{j_{used}}}$ – кількість ядер вузла N_j, що зайняті іншими завданнями на момент часу τ_{k-1};

3. $P_\Sigma = \sum_j P_j\,|\tau_{k-1} -$ сумарна потужність, що споживається кластером в момент часу τ_{k-1}

Крок 2.

Якщо доступний об'єм оперативної пам'яті менший за той, що вимагається для виконання завдання $task_{i,}$ $(\Delta V_{j_{avail}} \le V_{req_i})$, вузол N_j кластера виключається із розгляду доступних для обробки даного завдання вузлів. Аналогічно, якщо кількість незадіяних у роботі ядер вузла N_j менша, ніж цього вимагає завдання $task_{i,}$ вузол N_j виключається із розгляду доступних для обробки даного завдання вузлів.

У результаті таких операцій отримуємо масив $\{N_{avail_j}\}$ теоретично доступних для розміщення завдання $task_i$ вузлів такий, що $\{N_{avail_j}\} \subset \{N_j\}$.

Крок 3.

Для кожного j-го вузла обчислювального кластера залежність потужності споживання від навантаженості процесора є відомою функцією: $P_j = f_j(CPU_j)$. Цю залежність було виміряно і збережено при проведенні попередньої атестації кластера.

Припустивши, що робота job_i була призначена для виконання вузлу N_j, та знаючи залежність $P_j = f_j(CPU_j)$ для цього вузла, обчислимо теоретичне значення сумарної спожитої кластером потужності у разі розміщення завдання на обробку на j-й вузол:

$P_{\Sigma_l}|\tau_k \& N_j = P_\Sigma|\tau_{k-1} + \Delta P_{j_i}|\tau_k$

Проведемо такі обчислення для кожного вузла кластера.

Таким чином, маємо масив значень $P_\Sigma = \{P_{\Sigma_j}\}$, отриманий шляхом теоретичного розміщення роботи job_i на кожний з вузлів N_j.

Крок 4.

Проведемо сортування масиву доступних для розміщення завдання $task_i$ вузлів $\{N_{avail_j}\}$ за їх теоретично розрахованим на кроці 3 вкладом у загальне сумарне енергоспоживання всього обчислювального кластера $\Delta P_{j_i}|\tau_k$. Нехай сортування буде також проведене в порядку спадання. В результаті отримаємо масив N_{avail_p}:

$N_{avail_p} = \{N_{p_{j_{max}}}, \dots N_{p_j min}\}$

Крок 5.

Проведемо сортування масиву доступних для розміщення завдання $task_i$ вузлів $\{N_{avail_j}\}$ за їх доступною продуктивністю $flops_j$ у порядку спадання. Доступна продуктивність умовно визначається як:

$$flops_j = FLOPS(1 - \frac{CPU(\%)}{100}),$$

де $FLOPS$ – загальна продуктивність вузла обробки.

В результаті отримаємо масив $N_{flops_{avail}}$:

$N_{avail_{flops}} = \{N_{flops_{j_{max}}}, \dots N_{flops_{jmin}}\}$

Крок 6.

В залежності від позиції вузла із номером j у сортованому масиві $N_{avail_{flops}}$ та N_{avail_p}, він отримує два вагових коефіцієнта (дві оцінки): $mark_p$ і $mark_{flops}$ відповідно. При чому $mark_p$, $mark_{flops}$ = $\overline{1,n}$, де n – кількість вузлів у кластері. $mark_p$ дорівнює позиції j-го вузла у сортованому масиві N_{avail_p}, $mark_{flops}$ – позиції цього вузла у сортованому масиві $N_{avail_{flops}}$ відповідно.

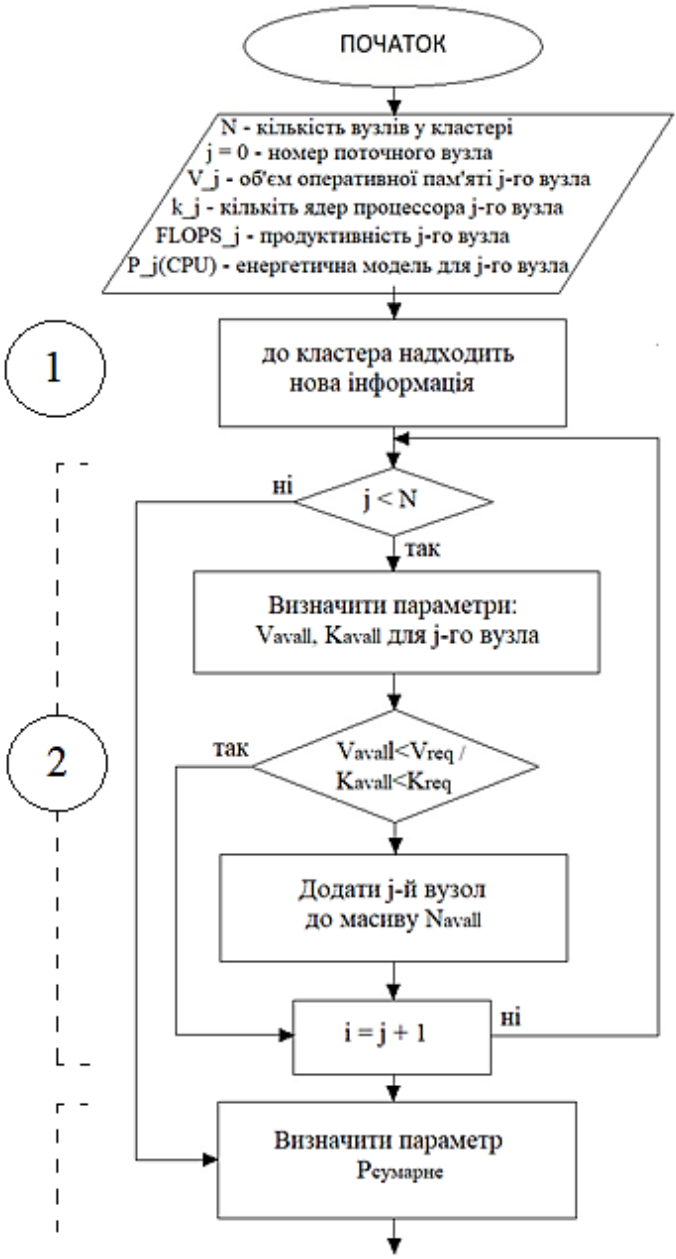
Крок 7.

Сумарна оцінка кожного вузла:

$mark_{\Sigma_j} = mark_{flops_j} + mark_{p_j}$

Вузол, для якого $mark_{\Sigma_j} = \max_j mark_{\sum j}$ вибирається для здійснення обробки завдання $task_i$.

Блок-схема для запропонованого алгоритму наведена на рис. 6.3.


ПОЧАТОК
N - кількість вузлів у кластері
j = 0 - номер поточного вузла
V_j - об'єм оперативної пам'яті j-го вузла
k_j - кількіть ядер процессора j-го вузла
FLOPS_j - продуктивність j-го вузла
P_j(CPU) - енергетична модель для j-го вузла
1
до кластера надходить нова інформація
ні
j < N
так
Визначити параметри: Vavall, Kavall для j-го вузла
2
так
Vavall<Vreq / Kavall<Kreq
Додати j-й вузол до масиву Navall
i = j + 1
ні
Визначити параметр Рсумарне

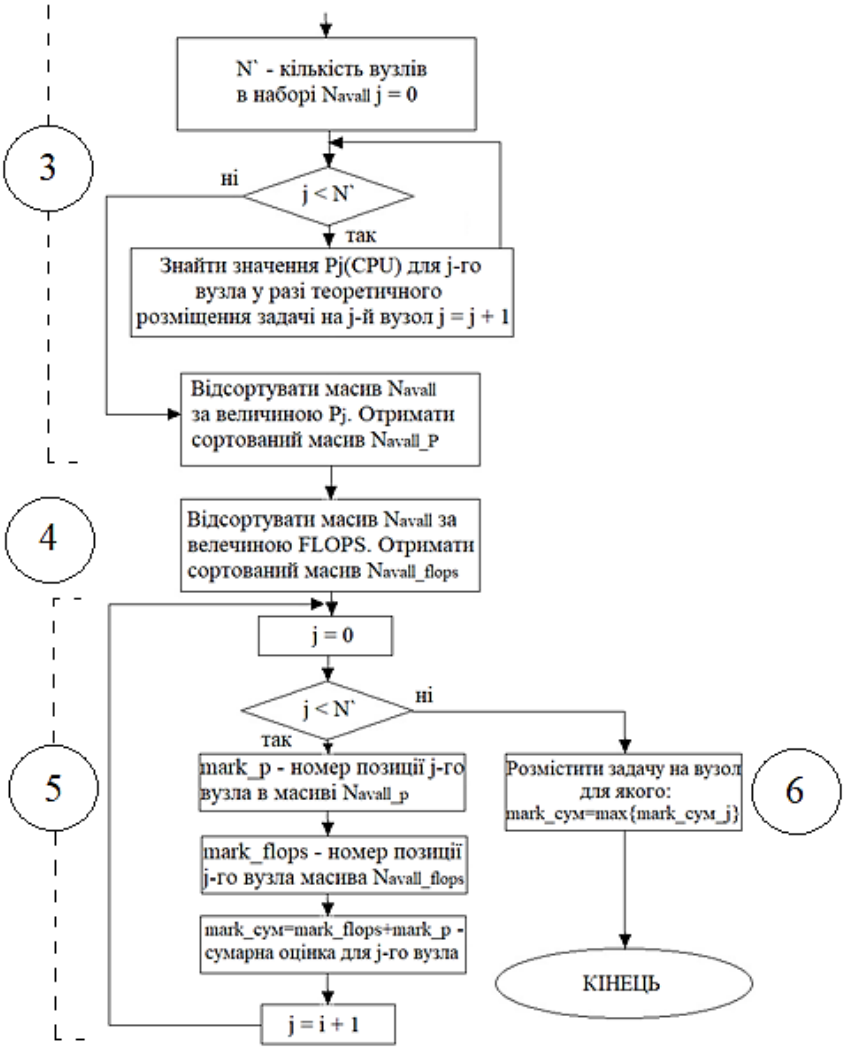


Рис. 6.3. Блок-схема запропонованого алгоритму

На рис. 6.3 пункти зображеної блок-схеми позначають такі дії:

Пункти 1-2 блок-схеми: До кластера надходить нове завдання *task_i*, що повинно бути розміщено на один з вузлів обробки. Вузли, що не мають достатньо ресурсів, виключаються з розгляду.

Пункти 3-4 блок-схеми: Проводиться сортування вузлів за продуктивністю та теоретичним внеском у загальне енергоспоживання кластера.

Пункт 5 блок-схеми: Кожен з вузлів кластера отримує бал (ваговий коефіцієнт) за продуктивність та енергоефективність. Підраховується сумарний бал за обома параметрами.

Пункт 6 блок-схеми: Завдання розміщується на вузол, що має найвищий сумарний бал.

6.3. Експериментальна перевірка роботи запропонованого підходу

Для перевірки роботи запропонованого підходу до підвищення енергоефективності обчислень у ЦОД було проведено натурний експеримент.

Метою експерименту було порівняння запропонованого підходу у формі алгоритму планування задач із існуючим широко використовуваним алгоритмом. Для порівняння було обрано алгоритм RoundRobin (RR). Принцип роботи цього алгоритму описано у пункті 1 розділу. Простота його реалізації, широке використання у реальних умовах ЦОД та наявність відкритого коду планувальника RR стали переважними якостями вибору даного алгоритму для порівняння.

План експерименту:

1. Побудувати гетерогенний серверний кластер.
2. Підготувати пакет завдань для перевірки роботи RR та запропонованого алгоритму (однаковий для обох випадків).
3. Отримати залежності між спожитою кожним вузлом серверного кластера потужністю та завантаженість процесора даного вузла.
4. Перевірити роботу алгоритму RR в умовах експерименту (проведення опорного експерименту).
5. Перевірити роботу запропонованого алгоритму в тих самих умовах.
6. Порівняти роботу алгоритмів за критеріями енергоефективності та продуктивності роботи кластера.

6.3.1. Постановка експерименту

Для проведення експерименту було використано 5 машин (серверів). Серед них 3 належали до одного типу (мали подібні фізичні характеристики), а 2 інші – до другого типу. Характеристики машин кожного типу наведені у табл. 6.1.

Таблиця 6.1. Фізичні характеристики серверів, використаних у експерименті

	Сервери типу 1 (x3)	**Сервери типу 2 (x2)**
Об’єм оперативної пам’яті	4 Gb	8 Gb
Тип та частота процесора	CPU Intel Core 2 Quad Q9400 2.66GHz	CPU Intel Core i5-43 3GHz

	Сервери типу 1 (x3)	Сервери типу 2 (x2)
Продуктивність	35.7GFLOPS	40-50GFLOPS

На всі машини була встановлена операційна система Cent OS. З цих 5 машин було утворено серверний кластер. Один вузол був обраний головним у кластері, саме на ньому розміщувався брокер (планувальник) задач. Інші підлеглі вузли були рівноправними між собою та мали прямий зв'язок із головним. Топологію кластера зображено на рис. 6.4.

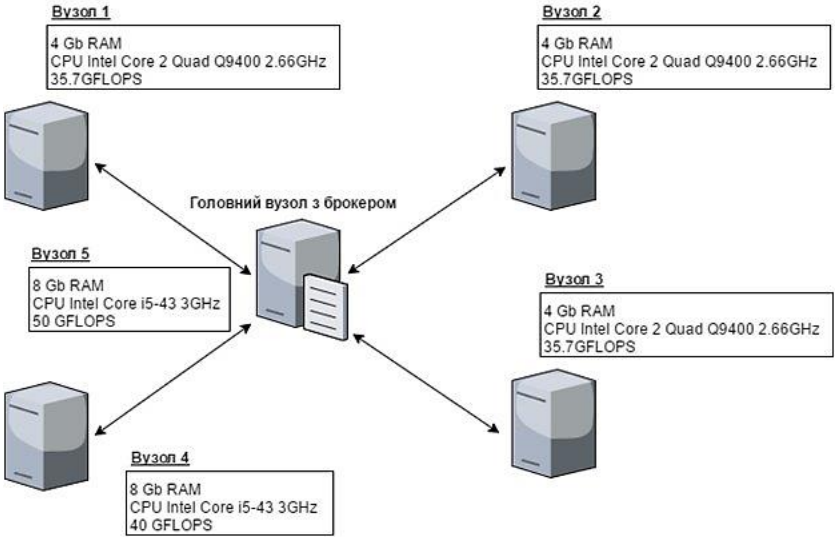


Рис. 6.4. Топологія кластера використаного в експерименті

Для зняття характеристик $P_j = f_j(CPU_j)$ кожного вузла, було використано цифровий мультиметр Yokogawa WT210, що здатний вимірювати струм, напругу та миттєву потужність безпосередньо між джерелом живлення (розеткою) та навантаженням.

Для моделювання реальної роботи серверного кластера було обрано різні за характеристиками завдання, при цьому завдання були об'єднані у пакет завдань.

Для отримання характеристик $P_j = f_j(CPU_j)$ було використано навантажувальний тест (stress-test)[14]. Визначення характеристик проводилось шляхом послідовного підключення мультиметра у коло живлення кожного сервера. Вимірювання потужності споживання проводилось для 0, 25, 50, 75 та 100% завантаженості *CPU*.

При перевірці роботи алгоритму RR і запропонованого алгоритму були використані ідентичні пакети завдань. При цьому, в рамках пакету завдання були різними та являли собою підрахунок числа π з точністю до 1000, 5000, 10000,

25000, 50000, 75000, 100000 знаків після коми. Підрахунок виконувався 5 різними методами. Кількість знаків після коми, що виступала параметром завдання, була випадковою величиною, розподіленою за нормальним законом.

Для порівняння алгоритмів було використано параметри продуктивності та енергоефективності. Для оцінки продуктивності було проаналізовано середній час обробки одного завдання. Ця величина є обернено пропорційною продуктивності кластера (формула 6.1):

$$P = m/t_{\sum}, \tag{6.1}$$

де m – кількість завдань, що підлягають обробці;
$t_{\sum}$ – час роботи кластера над обробкою набору з m завдань.
Тоді, середній час виконання одного завдання з набору:

$$t_{task\_i} = t_{\sum}/m \tag{6.2}$$

Виходячи з цього, можливо оперувати параметром «середній час обробки завдання» як обернено пропорційним до параметру продуктивності. Мінімізація параметру часу приведе до максимізації параметра продуктивності.

Для оцінки енергоефективності обробки завдання було обрано параметр загального сумарного енергоспоживання серверного кластера за час проведення експерименту у Ваттах.

6.3.2. Хід експерименту та обробка результатів

На початковому етапі експерименту, згідно запропонованого підходу, було проведено попередню атестацію серверів кластера. Експериментально виміряні залежності $P_j = f_j(CPU_j)$ були представлені у табличному та графічному вигляді (рис. 6.5).

Для формування опису вузлів та розміщення описів на головному вузлі із планувальником необхідним є представлення функцій $P_j = f_j(CPU_j)$ в аналітичному вигляді. Для цього було обрано метод інтерполяції функцій поліномом ступеня *n*.

Інтерполяція має як практичне, так і теоретичне значення. На практиці часто виникає завдання про відновлення неперервної функції по її табличних значеннях, отриманих, наприклад, в ході будь–якого експерименту. Для обчислення багатьох функцій виявляється ефективним наближення їх поліномами або дрібно-раціональними функціями [15].

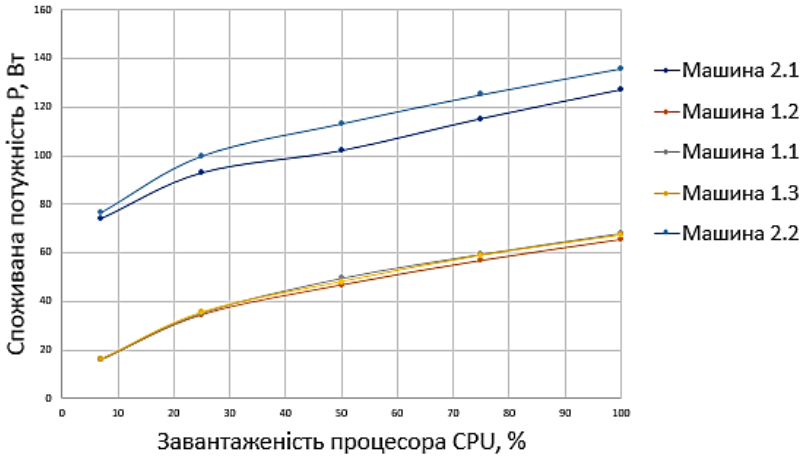


Рис. 6.5. Експериментально отримані залежності $P_j = f_j(CPU_j)$ для всіх серверів кластера

Процес інтерполяції було автоматизовано завдяки використанню інструментів середовища MATLAB. Ступінь інтерполяції $n = 4$ було обрано емпіричним шляхом – використання нижчого ступеня не дає потрібної точності (графіки функцій отриманих експериментальним шляхом та інтерполяцією та не співпадають), а використання вищих ступенів є надлишковим і не дає помітного наближення до початкового вигляду функцій.

В результаті проведення етапу попередньої атестації для серверів кластера було отримано їх описи (рис. 6.6).

Наступним етапом проведення експерименту стала перевірка роботи алгоритму RR. Для цього код планувальника RRбуло завантажено на центральний вузол серверного кластера. Отримані результати представлено у табл. 6.2.

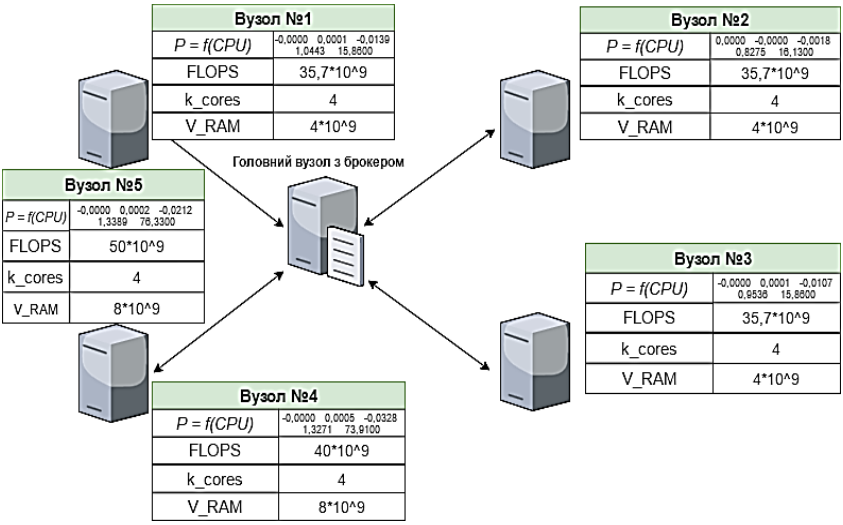


Рис. 6.6. Описи вузлів отримані в ході проведення етапу попередньої атестації

Після завершення перевірки роботи алгоритму RR на центральний вузол кластера було завантажено код планувальника, реалізований згідно запропонованого алгоритму. Алгоритм було реалізовано на мові програмування C, згідно блок-схеми (див. Додаток А). Результати експериментальної перевірки запропонованого підходу та алгоритму RR наведені у табл. 6.2.

Таблиця 6.2. Результати експериментального порівняння запропонованого підходу та алгоритму Round Robin для кластеру із 5 вузлів

Алгоритм	**Середній час обробки завдання,** *с*	**Сумарна спожита потужність,** *Вт*	**Виграш у порівнянні із Round Robin,** *%*	
			За часом	**За спожитою енергією**
Round Robin	14.835	376.88	-	-
Запропоно-ваний підхід	13.324	365.8	10,2%	3%

6.4. Аналіз отриманих результатів

Згідно результатів, наведених у табл. 6.2, запропонований алгоритм дозволяє отримати виграш у порівнянні із алгоритмом RR за параметром продуктивності – 10,2%, за параметром енергоефективності – 3%.

Отриманий виграш є незначним. Виграш за параметром енергоефективності майже підпадає під класифікацію статистичної помилки (при визначеній похибці у 2%).

Такий незначний виграш пояснюється тим, що вузли серверного кластера, використаного в експерименті, за своїми характеристиками належали до 2–х груп, причому в рамках кожної групи характеристики були однаковими. Крім того, залежності $P = f(CPU)$ для досліджуваного кластера також були близькими. Отже, можна припустити, що гетерогенність такого кластера була надто низькою, що й призвело до низької ефективності запропонованого підходу. Для перевірки даної гіпотези необхідно здійснити перевірку роботи запропонованого підходу для серверних кластерів, що містять більше вузлів, які є водночас більш різноманітними за своїми характеристиками.

6.5. Оцінка ефективності алгоритму шляхом моделювання

Для перевірки ефективності підходу у масштабному гетерогенному кластері було використано підхід імітаційного моделювання. Першим кроком було доведено адекватність моделі до об’єкту дослідження шляхом моделювання кластера з 5 вузлів. Після чого було створено модель з 20 вузлів, що має топологію, зображену на рис. 6.7. кластера з 5 вузлів. Після чого було створено модель з 20 вузлів, що має топологію, зображену на рис. 6.7.

При цьому, ефективність запропонованого підходу було порівняно із широким колом широко використовуваних алгоритмів розподілу задач.

Алгоритми, використані в ході моделювання, та їх короткі характеристики наведені у табл. 6.3.

Моделювання імітує роботу серверного кластера протягом доби (із середньостатистичним розподілом навантаження щогодини). Було використано набір завдань, аналогічний до випадку моделювання кластера з 5 вузлів. Інтервали часу між надходженням завдань – випадкова величина розподілена за нормальним законом. Результати моделювання зображено у табл.6.4.

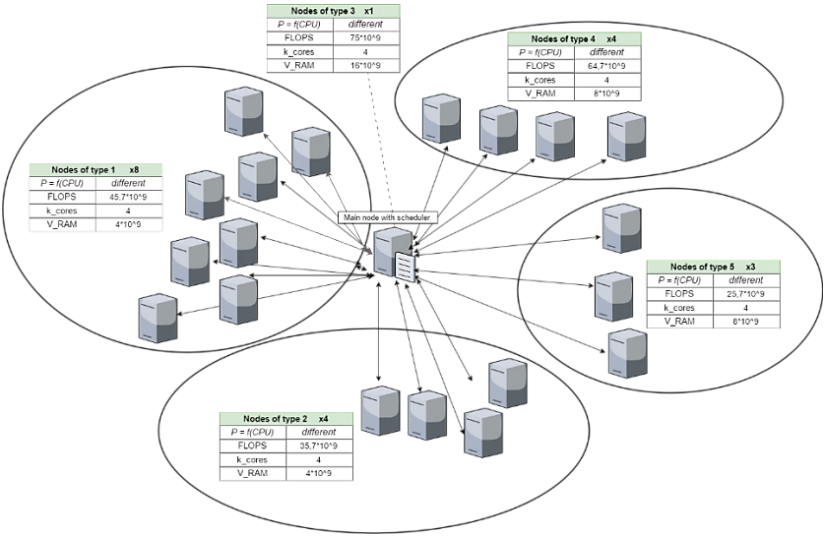


Рис. 6.7. Топологія кластера з 20 вузлів, використана у процесі моделювання

Таблиця 6.3**.** Алгоритми, що були використані в ході моделювання для порівняння із запропонованим

Алгоритм	**Характеристика**	**Головна ідея**	**Посилання**
RoundRobin (RR)	Простий, широко застосовний	Розміщує завдання на вузли по черзі	https://en.wikipedia.org/wiki/Round-robin_scheduling
FIFO(First Available)	Простий	Розміщує завдання на перший доступний вузол	https://en.wikipedia.org/wiki/First-come,_first-served
Least Connections	Широко застосовний	Розміщує завдання на вузол, що завантажений найменше	https://www.citrix.com/blogs/2010/09/02/load-balancing-least-connections/
Weighted RR	Покращений RR, широко застосовний	Деякі вузли більш пріоритетні для завантаження, тому вони мають більший ваговий коефіцієнт. Завдання розміщуються пропорційно	http://www.brocade.com/content/html/en/configuration-guide/nos-700-l2guide/GUID-B75BD370-B251-45A3-B3FD-87F54E35DC6E.html

Алгоритм	Характеристика	Головна ідея	Посилання
		коефіцієнтам, але почергово, як і для RR	
«Алгоритм Р»	Енерго-ефективний	Розміщує завдання на вузол з найменшою чергою. Якщо на цьому вузлі недостатньо ресурсів – на найбільш енергофективний вузол	http://cyberleninka.ru/article/n/energoeffektivnye-vychisleniya-dlya-gruppy-klasterov.pdf
The Most-Efficient-Server first (MES1)	Енерго-ефективний	Розміщує завдання на найбільш енергоефективні вузли. Якщо ці вузли зайняті – використовується RR для розміщення на інші вузли почергово	http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6679892&tag=1

Таблиця 6.4. Результати моделювання роботи кластера з 20 вузлів протягом доби

Алгоритм	**Середній час обробки завдання,** *с*	**Сумарна спожита потужність за годину,** *Вт*	**Програш у порівнянні із запропонованим алгоритмом,** *%*	
			За часом	**За спожитою енергією**
Запропонований підхід	10,880	1281,167	-	-
FIFO	10,996	1324,836	1,06	3,41
RR	16,221	1396,948	49,09	9,04
RR Weighted	14,019	1359,887	28,85	6,14
Least Connections	13,474	1357,448	23,84	5,95
MES1	13,857	1309,055	27,36	2,18
«Алгоритм Р»	15,470	1381,166	42,19	7,81

Таким чином, при дослідженні роботи кластера протягом доби із реальним розподілом навантаження було виявлено, що запропонований підхід дає виграш за обома параметрами у порівнянні із усіма алгоритмами. Найбільший виграш у 49,03% дає підхід за параметром продуктивності у порівнянні із алгоритмом RR. Виграш запропонованого експерименту за енергоефективністю близький до існуючих енергоефективних алгоритмів MES1 та «Алгоритм Р», проте, для випадку добової роботи, виграш за параметром продуктивності був значно більшим, ніж у зазначених алгоритмів.

Отримані результати дають право вважати, що ефективність запропонованого підходу зростає зі збільшенням кількості вузлів у кластері, їх гетерогенності та часу моделювання. Це дає підстави стверджувати, що запропонований підхід доцільно застосовувати у середніх та великих гетерогенних кластерах при тривалому часі їх роботи.

6.6. Огляд можливих модифікацій для підвищення ефективності запропонованого підходу та рекомендації щодо застосування підходу у обчислювальному середовищі

Для підвищення ефективності алгоритму запропоновано 2 основних напрями його модифікації. Перший полягає у використанні підходів масштабування додатково до застосування запропонованого алгоритму, другий – у врахуванні типів та характеристик вхідних завдань при їх розподілі.

6.6.1. Модифікація з використанням підходів масштабування

Підхід масштабування передбачає тимчасове переведення частини вузлів кластера у режим сну у випадку, якщо кількість вхідних завдань незначна і може бути оброблена без участі даних вузлів. Використання даного підходу дозволить отримати значний виграш за параметром енергоефективності, адже споживання вузлів серверного кластера у режимі сну мінімальне.

Для реалізації даної модифікації, постає завдання експериментального або аналітичного визначення порогових кількостей завдань у черзі на обробку, при яких повинно бути здійснено переведення одного вузла кластера у режим сну та навпаки. При цьому у режим сну повинні бути переведені машини із «найгіршими» характеристиками енергоспоживання.

Крім того, постає проблема визначення порогу перенавантаження активних вузлів, коли необхідним стає виведення нового вузла з режиму сну та переведення його в активний режим. Ці проблеми тісно пов’язані з проблемами консолідації віртуальних машин при використанні підходів масштабування.

6.6.2. Модифікація з використанням типізації задач

Для збільшення продуктивності серверного кластера доцільним є використання підходу типізації завдань додатково до запропонованого алгоритму. Цей підхід передбачає розбиття вхідного потоку завдань на умовно «складні» та «прості» в залежності від ресурсів, яких кожне з них потребує.

Після проведення типізації підхід передбачає розподіл «простих» завдань на вузли, що мають меншу продуктивність, а «складних» – відповідно, більшу.

Для реалізації цього підходу необхідно експериментальним або аналітичним шляхом визначити значення параметрів завдань, за якими їх можна віднести до одного із типів («складні» чи «прості»). Для кожного окремого випадку такий поділ має бути налаштований індивідуально в залежності від характеристик завдань.

Для отримання загального уявлення про ефективність запропонованих модифікацій, було проведено імітаційне моделювання у середовищі MATLAB. При цьому пороги для ввімкнення та вимкнення вузлів кластера та характеристики завдання для їх типізації було підібрано методом послідовного наближення.

Для модифікації із застосуванням підходів масштабування, шляхом зміни значення порогів вимкнення та ввімкнення серверів у діапазоні від 5 завдань у черзі до 30 завдань у черзі із кроком 5, було визначено, що для конкретних умов моделі (20 вузлів кластера із характеристиками, зазначеними на рис. 6.7), оптимальним є значення 10 і менше завдань у черзі для переведення у режим сну найменш енергоефективного сервера, та 25 і більше завдань у черзі для виведення одного сервера із режиму сну. Для модифікації із використанням типізації завдань, аналогічно, було визначено поріг у 350 GFLOPS. Завдання з об'ємом меншим цього порогу, вважалися «простими» в рамках даного випадку моделювання. Більш обчислювально– об'ємні задачі вважалися «складними».

Моделювання проводилось для кластера з 20 вузлів за умов, описаних у пункті 6.3.1. Результати проведеного моделювання представлені у табл. 6.5.

Як видно із табл. 6.5, для даних умов моделювання застосування модифікації із використанням підходу типізації завдань не дає відчутного результату. Дослідження умов, за яких дане доповнення до запропонованого підходу дозволить отримати виграш, буде досліджено у майбутньому. При застосуванні підходу масштабування стає можливим отримання значного виграшу за параметром енергоефективності. Для умов моделювання відносний виграш склав 33,59%.

На основі аналізу отриманих результатів можна зробити висновок, що подальшого дослідження вимагають обидва підходи модифікації. Серед них, більш перспективним є підхід із застосуванням масштабування.

Таблиця 6.5. Результати моделювання роботи запропонованого підходу із можливими модифікаціями

Алгоритм	**Середній час обробки задачі,** *с*	**Сумарна спожита потужність,** *Вт*	**Виграш у порівнянні із запропонованим алгоритмом,** *%*	
			За часом	**За спожитою енергією**
Запропонований підхід	10,880	1281,167	0,00	0,00

Алгоритм	Середній час обробки задачі, *с*	Сумарна спожита потужність, *Вт*	Виграш у порівнянні із запропонованим алгоритмом, *%*	
			За часом	За спожитою енергією
Модифікація 1	10,885	1284,339	0,05	0,25
Модифікація 2	10,893	850,838	0,12	33,59

Висновки

1. Проведено аналіз існуючих підходів до підвищення енергоефективності обчислень у центрі обробки даних, що показав наявність низки методів, що мають на меті зниження енергоспоживання під час обробки даних, але не враховують одночасно параметр продуктивності обчислень.
2. Виділено клас методів підвищення енергоефективності обчислень, а саме енергоефективне планування завдань, у рамках якого доцільним є внесення змін з метою покращення показників енергоефективності та продуктивності обчислень.
3. Запропоновано підхід до підвищення енергоефективності обчислень для серверного кластера як інформаційно-телекомунікаційної одиниці інфраструктури ЦОД, який відрізняється одночасним врахуванням параметрів енергоефективності та продуктивності при розподілі завдань.
4. Розроблено алгоритм підвищення енергоефективності обчислень на основі запропонованого підходу, який складається з етапу попередньої індивідуальної атестації серверного кластера та етапу енергоефективного розподілу завдань. Врахування індивідуальних залежностей енергоспоживання серверів від їх завантаженості є основною відмінною рисою запропонованого підходу.
5. Перевірено роботу запропонованого підходу експериментально та шляхом імітаційного моделювання. Проведено аналіз отриманих результатів, та визначено, що підхід проявляє більшу ефективність – до 49,09% за параметром продуктивності та до 9,04% за параметром енергоефективності – для великих гетерогенних кластерів.

Література

1. Koomey J. G. Worldwide electricity used in data centers / J. G. Koomey. // Environmental Research Letters, vol. 3. – 2008. – №3. – С. 034008.
2. Growth in data center electricity use 2005 to 2010 [Електронний ресурс] // Analyticspress – Режим доступу до ресурсу: http://www.analyticspress.com/datacenters.html.
3. Beloglazov A. Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing / Beloglazov Anton – Melbourne, 2013.

4. Dayd? M. High Performance Computing for Computational Science / M. Dayd?, J. Dongarra., 2005. – 120 с. – (ISBN 3-540-25424-2).

5. Jyoti V. Comparative Study of Load Balancing Algorithms / V. Jyoti, K. J. Anant. // IOSR Journal of Engineering (IOSRJEN). – 2013. – С. 45–50.

6. Choi D. An Improvement on the Weighted Least-Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems / D. Choi, K.S. Chung, J. Shon // Springer, vol.21, Berlin, Heidelberg

7. Hosseinimotlagh S. A Cooperative Two-Tier Energy-Aware Scheduling for Real-Time Tasks in Computing Clouds / S. Hosseinimotlagh, F. Khunjush, S. Hosseinimotlagh. // Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. – 2014. – №22. – С. 178–182.

8. An Energy and Deadline Aware Resource Provisioning, Scheduling and Optimization Framework for Cloud Systems / Y.Gao, Y. Wang, S. K. Gupta, M. Pedram. // IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. – 2013. – №9. – С. 31:1–31:10

9. Армента-Кано Ф. Min_c: стратегія неоднорідної концентрації задач для енерго-заощаджуючих комп'ютерних планувальників / Ф. Армента-Кано, А. Черних, Х. М. Кортес-Мендоза // Труды ИСП РАН. – 2015. – №6. – С. 355.

10. Liu N. Task Scheduling and Server Provisioning for Energy-Efficient Cloud-Computing Data Centers / N. Liu, Z. Dong, R. Rojas-Cessa. // IEEE 33rd International Conference on Distributed Computing Systems Workshops. – 2013. – №33. – С. 226–231.

11. Грушин Д. А. Энергоэффективные вычисления для группы кластеров / Д. А. Грушин, Н. Н. Кузюрин. – Москва, 2013. – 433 с.

12. Dongarra J. Performance of Various Computers Using Standard Linear Equations Software // ACM SIGARCH Computer Architecture News, June, 2014, pp. 22-44.

13. Gvozdetska N.A. Experimental Analysis of PCPB Scheduling Algorithm / N.A. Gvozdetska, O.V. Stepurin, L.S. Globa // CADSM'2017, February, 2017, pp. 244-247.

14. Waterland A. Workload generator for POSIX systems [Електронний ресурс] / Amos Waterland. – 2014. – Режим доступу до ресурсу: https://people.seas.harvard.edu/~apw/stress/.

15. Іванов В. В. Методи обчислень на ЕОМ. Довідник / В. В. Іванов. – Київ: Наукова думка, 1986, 584 с.

16. Alexander Schill, Larysa Globa, Oleksandr Stepurin, Nataliia Gvozdetska, Volodymyr Prokopets «Power Consumption and Performance Balance (PCPB) scheduling algorithm for computer cluster», // UkrMiCo'2017, 11-15 September 2017, Odesa, Ukraine, pp. 1-8.

17. Глоба Л. С. Енергоефективний підхід до розподілу задач у серверному кластері / Л. С. Глоба, Н. А. Гвоздецька, В. А. Прокопець, О. В. Степурін // Вісник Харівського національного університету імені В.Н. Каразіна. – Харків. – 2017. – Том 34. – С. 18-28.