Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 3 (25)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

УДК 004.82
DOI: https://doi.org/10.30837/ITSSI.2023.25.027

А. Козирєв, І. Шубін

МЕТОД ПЛАНУВАННЯ ЗАВДАНЬ ОБРОБЛЕННЯ ДАНИХ У РОЗПОДІЛЕНИХ СИСТЕМАХ З ОБМЕЖЕНОЮ ІНФОРМАЦІЄЮ ПРО ДОСТУПНІ РЕСУРСИ

У сучасному цифровому ландшафті розподілені системи оброблення даних (РСОД) стають усе більш критично важливими для забезпечення ефективного оброблення, аналізу та управління великими обсягами інформації. Ці системи часто використовуються в комерційних, наукових і соціальних доменах для оброблення комплексних даних в реальному часі або пакетному режимі. Одним із ключових складників таких систем є планування завдань, що є надзвичайно складним процесом, зокрема коли інформація про ресурсні потреби не є повною або точною. **Предметом дослідження** є алгоритми, методи та підходи, які використовуються для планування завдань між вузлами в розподілених системах. **Мета роботи** – створення оптимізованого методу планування завдань у РСОД з обмеженою інформацією про доступні ресурси. **Завдання дослідження**: проаналізувати недоліки сучасних методів для планування завдань у розподілених системах оброблення даних; оптимізувати метод планування завдань на основі метаданих між вузлами РСОД, який ґрунтується на методології пошуку найближчих сусідів із використанням методу локалізованого хешування та алгебри скінченних предикатів; розробити архітектуру програмного рішення та його реалізацію на основі оптимізованого методу; протестувати алгоритм на прикладі завдання декодування відео. Застосовано такі **методи**: статистичні алгоритми та техніки, зокрема класифікація та кластерний аналіз, використані для прогнозування потреб у ресурсах; візуалізаційні методи допомогли в аналізі та інтерпретації результатів. **Результати роботи**: проаналізовано недоліки сучасних методів для розподілу завдань у розподілених системах оброблення даних; створено оптимізований метод планування завдань на основі метаданих у РСОД, який ґрунтується на методології пошуку найближчих сусідів із використанням методу локалізованого хешування та алгебри скінченних предикатів; деталізовано процеси в модифікованому алгоритмі пошуку найближчих сусідів; розроблено архітектуру програмного рішення, що інтегрує оптимізований метод планування завдань на основі метаданих та алокації ресурсів; за допомогою практичного сценарію здійснено валідацію програмного рішення – використання створеного алгоритму в задачі планування для декодування відеоінформації. **Висновки.** Запропонований метод, що ґрунтується на методології локалізованого хешування та на застосуванні алгебри скінченних предикатів, є ефективним навіть у разі недостатньої або обмеженої інформації про ресурсні потреби. Це підтверджує можливість використання динамічних стратегій планування для адаптації до мінливих умов навантаження та доступності ресурсів.

Ключові слова: програмна інженерія; розподілені системи; оброблення даних; база даних; планування завдань; алгебра скінченних предикатів.

Вступ

У сучасній цифровій екосистемі величезні масиви даних є результатом активності низки джерел, зокрема IoT-пристроїв, систем машинного навчання, соціальних медіа, платформ електронної комерції та інших технологічних засобів. Відповідно до цього стрімкого приросту інформаційних обсягів, сучасні методи зберігання та оброблення даних мають утілювати найновітніші технологічні розробки. У цьому контексті методи планування завдань для оброблення даних у розподілених системах оброблення даних (далі РСОД) посідають особливо важливе місце.

Розподілені системи, що можуть функціювати на кластерах серверів або в хмарних обчислювальних середовищах, є особливим викликом з погляду планування завдань. Оскільки дані та обчислювальні ресурси розподілені між різними фізичними або віртуальними вузлами, необхідно розробляти алгоритми, що можуть динамічно виділяти завдання, мінімізувати час очікування й оптимізувати використання ресурсів.

Планування завдань у розподілених системах передбачає розподіл обчислювальних задач між мережею взаємопов’язаних вузлів, часто з метою оптимізації продуктивності, використання ресурсів і пропускної здатності системи. У цьому контексті "завдання" означає обчислювальну задачу, яка може варіюватися від запитів до бази даних до алгоритмів машинного навчання, тоді як "розподілена система оброблення даних" охоплює мережну архітектуру взаємопов’язаних програмних засобів та баз даних, що можуть обробляти,

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2023. No. 3 (25)

зберігати й отримувати дані в децентралізований спосіб.

Незважаючи на те, що за умови взаємодії з окремими обчислювальними вузлами в межах розподілених систем можуть бути прийняті локально оптимальні рішення, це не гарантує глобальної оптимальності відображення цілої множини завдань на множину доступних вузлів. Тобто, якщо кожен вузол незалежно вирішує, яке завдання він виконає наступним, можливо, відсутність координації призведе до неефективного розподілу ресурсів на більш високому, системному рівні.

Такий підхід, імовірно, спричинить низку проблем, зокрема затримки в обробленні, незбалансоване використання ресурсів та взаємні блокування завдань. Відтак, щоб досягти оптимальної ефективності, процедури планування завдань та алокації вузлів мають бути інтегровані в одному процесі планування. Це означає, що замість того, щоб розглядати кожне завдання або вузол ізольовано, система має зважати на глобальний стан, а саме: на доступність ресурсів, поточну завантаженість вузлів та залежності між завданнями – для прийняття більш інформованих та оптимізованих рішень.

Метою дослідження є розроблення методу планування завдань у розподілених системах оброблення даних в умовах неповноти інформації про ресурсні вимоги.

Аналіз останніх досліджень і публікацій

Упродовж останнього десятиліття спостерігається відчутне зростання потреби в розподілених обчислювальних ресурсах у різноманітних сферах застосування [1]. Серед таких завдань можна виокремити електронну комерцію, фінансову аналітику, аналіз даних у соціальних мережах, а також оброблення та дистрибуцію мультимедійного контенту [2].

У широкому використанні зараз перебувають системи паралельного та розподіленого оброблення даних, зокрема обчислювальні кластери [3], ґрід-системи [4] та хмарні платформи [5, 6]. Використання перелічених систем у наукових дослідженнях та для вирішення конкретних завдань призводить до подальшого збільшення навантаження на апаратні ресурси. Це зумовлено постійним зростанням складності завдань, які вимагають все більше обчислювальної та просторової потужності. Необхідно зазначити, що ці завдання часто визначаються нерівномірними вимогами до обчислювальних ресурсів.

Завдання є сутністю, яка надходить на вхід системи планування й містить набір завдань [7]. Потік завдань надходить у вхідну чергу, завдання є атомарною одиницею планування, й у межах одного завдання вони можуть бути незалежними або організованими в дерево залежностей [8]. Відповідно до цих параметрів підзавдання обробляються або паралельно в разі відсутності залежностей, або в заданому порядку, якщо такі залежності існують. Залежності слугують керівними принципами для послідовності виконання та перенесення даних між підзавданнями.

Низка наукових досліджень [5, 9–11] вказує на те, що сучасні завдання у сфері оброблення даних демонструють значну динамічність. З одного боку, можна спостерігати численні "легкі" завдання, що визначаються мінімальними вимогами до ресурсів, наприклад, замовлення на оброблення невеликих текстових файлів або зображень. З іншого боку, існують "важкі" завдання, такі як оброблення великих обсягів даних або виконання складних математичних розрахунків, що потребують значно більших обчислювальних можливостей.

У такому різноманітному обчислювальному середовищі інтелектуальне планування розподілу завдань між доступними обчислювальними вузлами набуває критичної важливості. Це питання актуальне не тільки для забезпечення ефективності роботи конкретного вузла, але й для оптимального використання всієї обчислювальної інфраструктури загалом.

Методи планування можуть бути категоризовані за різними критеріями. Наприклад, існують методи, що ґрунтуються на апріорному знанні про ресурсні потреби кожного задвдання. Тут можуть використовуватися алгоритми оптимізації, які попередньо аналізують завдання та призначають їх вузлам на основі детального ресурсного профілю [12]. З іншого боку, є підходи, де завдання розподіляються динамічно, без попереднього аналізу [13]. У такому разі може застосовуватися загальний принцип оброблення, де завдання просто надходять у чергу й обслуговуються за принципом "першим прийшов – першим обслуговано" [14], або за деякими іншими евристичними правилами.

Основна дилема в плануванні завдань полягає в розподілі обчислювальних ресурсів та встановленні порядку, за яким завдання з вхідної черги будуть

Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 3 (25)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

реалізовані на цих ресурсах [7]. Системи, де ресурси керуються в централізований спосіб, відомі як системи управління ресурсами, і ці системи зазвичай тісно взаємодіють з механізмами планування завдань.

Комбінація системи управління ресурсами та планувальника завдань формує проміжний програмний модуль. З одного боку, цей модуль відповідає за управління ресурсами на базовому рівні, а з іншого – адаптується до високорівневих вимог прикладних завдань. Для досягнення цієї мети система постійно аналізує актуальний стан використання ресурсів та далі ділить їх між заявленими завданнями [15].

У системі управління ресурсами часто інтегрований інформаційний сервіс, який відповідальний за моніторинг стану завдань і ресурсів. В окремих ситуаціях цю роль виконує зовнішня система моніторингу. Планувальник оперує на основі даних, які надає цей інформаційний сервіс, і використовує конкретні методики для управління завданнями та ресурсами.

Політика планування завдань (*task scheduling policy*) – це комплекс правил, за якими визначається момент та спосіб вибору нового завдання для оброблення [16]. Основний механізм вибору завдань ґрунтується на процедурі відбору завдань щодо їх пріоритетів.

Зазвичай для планування завдань використовується одна з дисциплін обслуговування. Ці дисципліни можуть або базуватися на детальних ресурсних вимогах завдань, або оперувати без попереднього знання, тобто обробляти всі завдання універсальним методом.

Визначення не вирішених раніше частин загальної проблеми

Недоліки методів розподілення запланованих завдань без попереднього знання полягають у тому, що вони не адаптуються до особливих ресурсних потреб, зосереджуються на виборі конкретних етапів для оброблення на основі інших критеріїв, припускають однорідність і використовують однотипні обчислювальні вузли. Це може призвести до неефективного використання ресурсів, особливо коли вузли та завдання мають різні характеристики або коли один вузол може паралельно обробляти декілька завдань.

Використання цих методів планування ресурсів часто не є оптимальним у таких сценаріях:

– коли обчислювальні вузли в розподілених системах оброблення даних мають різну потужність, що вказує на гетерогенність РСОД;

– коли завдання відрізняються за обчислювальною та просторовою (за пам’яттю) складністю, що вимагає різних ресурсних потреб;

– коли вузли РСОД можуть обробляти декілька завдань паралельно.

Додаткова характеристика згаданих методів планування завдань полягає в можливості зберігання стану виконуваних завдань. Це дає змогу призупинити поточне завдання й перейти до іншого, що може бути більш пріоритетним. Методи планування, зокрема *Time Sharing, Least Attained Service* та *Shortest Remaining Processing Time*, належать до цього класу алгоритмів, завдяки чому на одному обчислювальному вузлі може одночасно виконуватися декілька завдань на різних етапах оброблення.

Збереження стану завдань є критичним компонентом у системах планування процесів, що активно використовуються в сучасних операційних системах. Ця методика дає змогу декільком процесам поділити доступ до обмеженого ресурсу процесорного часу, симулюючи цим паралельне виконання завдань. Поділ процесорного часу здійснюється способом присвоєння кожному процесу короткого кванта часу, після якого процес може бути призупинений, його стан збережений, ресурс переданий іншому процесу.

Однак цей підхід має декілька недоліків. По-перше, він передбачає додаткові оперативні витрати, наприклад, на збереження та відновлення стану процесу, а також на переключення контексту між різними процесами. Ці дії можуть стати затратними з погляду обчислювальної ефективності, особливо в системах із значною кількістю активних процесів. По-друге, для завдань, що вимагають високої обчислювальної інтенсивності та синхронізації, переривання для збереження стану може призвести до суттєвого падіння продуктивності. У таких ситуаціях необхідність у перериванні та збереженні стану може знизити загальну пропускну здатність системи та збільшити латентність виконання завдань.

Далі в процесі планування для кожного обраного завдання застосовується політика розміщення на вузлах. Ця політика обирає оптимальний комплекс ресурсів для конкретного завдання оброблення

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2023. No. 3 (25)

даних. У системі ресурсного управління беруться до уваги характеристики кожного вузла або ресурсу в розподіленій інфраструктурі, а також задовольняються попередньо встановлені критерії від застосунків.

Методики призначення ресурсів на завдання можуть відрізнятися, зокрема, залежно від наявності попередньої інформації. Це може бути вибір на менш завантаженому або недавно використаному вузлі, випадковий вибір вузла, вибір на основі метрик черги, вибір вузла з найкоротшим прогнозованим часом виконання завдання або вибір вузла, що найкраще відповідає прогнозуванню споживання ресурсів.

Однак основна проблема наявних методів планування полягає в тому, що вибір завдання та вибір вузла для його виконання розглядаються як два незалежних етапи. Спершу визначається завдання з найвищим пріоритетом, а потім для нього підбирається найбільш підхожий вузол.

На основі проведеного огляду можна визначити необхідність створення інноваційного методу планування завдань у розподілених системах оброблення даних, особливо в разі обмеженої інформації про потреби в ресурсах. Це має потенціал зменшити часові затрати на реалізацію конкретних завдань оброблення інформації.

Для досягнення цієї мети необхідно вивчити взаємозв'язок між атрибутами (метаданими) завдань та метричними показниками використання обчислювальних ресурсів. У дослідженні увага зосереджена на методах планування непов'язаних завдань, що не передбачають підтримки збереження стану виконання, та розробляються в контексті обмеженої доступності інформації про ресурсні потреби завдань.

Матеріали й методи

Стратегія планування завдань, що ґрунтується на метаданих і метричних показниках ресурсів, передбачає пошук найближчих сусідів для даних, що підлягають обробленню. Оскільки ці дані мають як чисельні, так і категоріальні атрибути, традиційні метрики відстані, як-от евклідова або хемінгова відстань, не можна застосувати безпосередньо. Замість цього використовуються гетерогенні функції відстані, що дають змогу порівняти такі різнорідні об'єкти даних. Відстані для чисельних і категоріальних атрибутів обчислюються окремо. Після цього здійснюється нормалізація та зважування отриманих значень для мінімізації впливу шуму та розбіжностей у шкалах.

Основний недолік такого типу пошуку полягає в його великій розмірності. Для кожного заданого об'єкта (завдання) потрібно визначити відстань до всіх n об'єктів з історії виконань, що призводить до лінійної асимптотичної складності одного запиту, або $O(n)$.

Під час планування набору завдань потрібно виміряти ресурсну складність для всіх можливих пар (вузол-процесор, завдання). Це призводить до загальної асимптотичної складності, що формулюється як

$$O(h-v-n)=h-v-O(n)=O(n), \qquad (1)$$

де h – кількість доступних для планувальника вузлів-оброблювачів, v – кількість завдань, що очікують розроблення.

Хоча теоретична складність є лінійною, у практичному застосуванні константи h і v також суттєво впливають на обчислювальну вартість процесу планування. Наприклад, у разі наявності 33 обчислювальних вузлів і мінімум 33 завдань у черзі очікування загальна кількість запитів становитиме 1089. Кожний із цих запитів має лінійну складність, що корелює з розміром історичних даних.

Зменшення розмірності в завданні пошуку найближчих сусідів (*Nearest Neighbor Search, NNS*) є критичною проблемою, особливо у великих датасетах із високою розмірністю. Алгебра скінченних предикатів, що є математичною теорією, може бути використана для вирішення цієї проблеми. Сутність полягає у формалізації атрибутів або властивостей елементів набору даних за допомогою логічних виразів або предикатів.

Одним із способів реалізації може бути застосування алгебри скінченних предикатів. Основна ідея полягає в тому, щоб використовувати логічні оператори для розбиття високовимірного простору на підпростори, що спрощує пошук.

Візьмемо декартовий простір $\mathbb{R}^d$, де d – розмірність простору. Цільова функція відстані між двома точками $x, y \in \mathbb{R}^d$ може бути евклідовою відстанню:

$$D(x,y)=\sqrt{\sum_{i=1}^{d}(x_i-y_i)^2}\,. \qquad (2)$$

За допомогою алгебри скінченних предикатів можна створити систему логічних умов, які розділять простір на менші підпростори [17]. Наприклад,

Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 3 (25)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

можна ввести предикати $P_1, P_2, \dots P_n$, що розділять простір на n підпросторів:

$$P_i(x): \mathbb{R}^d \to \{0,1\}. \quad (3)$$

Тепер, замість того, щоб шукати найближчого сусіда в усьому просторі $\mathbb{R}^d$, можна обмежитися пошуком у підпросторах, у яких є точка запиту, тобто тільки в тих підпросторах, для яких $P_i(x) = 1$.

Застосування алгебри скінченних предикатів уможливить створення більш гнучких та оптимізованих алгоритмів планування. Ці математичні інструменти можуть бути використані для моделювання обмежень і вимог до системи, які традиційно важко виразити алгебраїчно.

Розглянемо систему з H обчислювальними вузлами та V завданнями, що потрібно запланувати. Для кожного вузла h та завдання v можна ввести предикат $P_{h,v}$, який оцінює, чи може завдання v бути ефективно виконаним на вузлі h:

$$P_{h,v}(x): \{h, v\} \to \{0,1\}. \quad (4)$$

Щоб зменшити розмірність задачі знаходження k найближчих сусідів для об'єктів із різноманітними характеристиками, рекомендовано використовувати алгоритм, оснований на методі локалізованого хешування (*Locality-Sensitive Hashing, LSH*). Такий підхід виявляється масштабованим і дає змогу виконати деяку частину пошуку в константний час $O(1)$.

Задача пошуку найближчих сусідів у високовимірних просторах часто виявляється вимогливою щодо обчислювальних ресурсів. Один із способів оптимізації цієї задачі полягає у використанні алгоритмів локалізованого хешування, які намагаються згрупувати близькі точки в одному або декількох хеш-бакетах, забезпечуючи швидкий пошук.

Алгебра скінченних предикатів може бути інтегрована в алгоритм *LSH* для додаткової оптимізації. Предикати можуть бути використані для вибору оптимальних функцій хешування або для динамічної адаптації алгоритму до структури даних.

У межах алгебри скінченних предикатів можна додати ряд предикатів $P_1, P_2, \dots P_n$, що слугують критеріями для хешування:

$$P_i(x, y): \mathbb{R}^d \times \mathbb{R}^d \to \{0,1\}. \quad (5)$$

Ці предикати можуть вимірювати різні аспекти подібності між точками x та y. Застосування предикатів до пар точок може бути використано для динамічного вибору функцій хешування в алгоритмі *LSH*. Наприклад, для пар точок, для яких $P_i(x, y) = 1$, можна застосовувати функцію хешування h_i, що є оптимальною для цього типу подібності.

$$h(x) = \begin{cases} h_1(x), \text{ if } P_1(x, y) = 1 \\ h_2(x), \text{ if } P_2(x, y) = 1 \\ \vdots \\ h_k(x), \text{ if } P_k(x, y) = 1. \end{cases} \quad (6)$$

Цей підхід забезпечує більшу гнучкість у виборі функцій хешування та, відповідно, може підвищити ефективність пошуку найближчих сусідів у розподілених системах.

Ключовим аспектом методу локалізованого хешування є обережний вибір атрибутів даних, які будуть використовуватися для розрахунку хеш-значення. Обрані атрибути визначають, який сегмент пошукового процесу може бути оптимізований до константного часу, а який залишиться з лінійною складністю. Ці атрибути мають відображати подібність між цими об'єктами, та їх вибір має ґрунтуватися на особливостях конкретного прикладного домену.

Варто зазначити, що атрибути зазвичай об'єднують обмежену кількість сутностей для подання характеристик, які корелюють із значною кількістю інших сутностей. У такий спосіб атрибути можуть слугувати як фокусні точки й природно лягати в основу для класифікації за принципом еквівалентності. Сучасні методи дослідження часто не роблять чіткого розмежування між атрибутами сутності та їх взаємозв'язками [18].

Сутність e пов'язана з іншою сутністю e' через скінченний двійковий предикат p:

$$p_1(x_0, x_1) \wedge \dots \wedge p_n(x_{n-1}, x_n) \wedge a_1(x_{l1}) \wedge \dots \wedge a_m(x_{lm}) \to p(x_0, x_n). \quad (7)$$

Атрибути часто корелюють із значною кількістю об'єктів, формуючи цим особливі "концентратори" взаємодії. Ідентифікацію таких концентраторів рекомендується проводити на основі метрики щільності їх з'єднань [18]. Щільність концентратора можна визначити як

$$den_{in}(e, p) = \frac{\#e' : p(e', e)}{\#(e', e") : p(e', e")}. \quad (8)$$

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2023. No. 3 (25)

На основі поданих методів у цій роботі запропонована модифікована версія алгоритму знаходження найближчих сусідів, що ґрунтується на методі локалізованого хешування з використанням алгебри скінченних предикатів. Ця модифікація дає змогу здійснити деяку частину пошуку в константний час $O(1)$, а іншу частину – в сублінійний час $O(t)$.

Алгоритм знаходження найближчих сусідів, оснований на методі локалізованого хешування, передбачає такі етапи:

Крок 1. Селекція атрибутів для хеш-функції, а саме:

– згрупувати атрибути на числові та категоріальні типи;

– оцінити значущість кожного атрибуту за допомогою навчального датасету;

– обрати n найбільш ресурс-інтенсивних категоріальних атрибутів та n числових атрибутів із найвищою значущістю.

Крок 2. Конструювання хеш-таблиці, основаної на тренувальному датасеті:

– розрахувати хеш-значення для вибраних категоріальних та числових атрибутів;

– використовувати згенерований агрегований хеш як індекс для розміщення об’єктів у відповідному "кошику" хеш-таблиці;

– зберегти унікальний ідентифікатор об’єкта в хеш-таблиці за відповідною адресою.

Крок 3. Пошук найближчих сусідів для нової данини:

– обчислити хеш-значення для нового об’єкта на основі атрибутів, обраних на першому кроці;

– використати згенерований агрегований хеш для доступу до відповідного "кошика" в хеш-таблиці;

– виміряти відстань до всіх об’єктів, чиї ідентифікатори зберігаються в цьому "кошику";

– упорядкувати згенерований набір відстаней;

– обрати k об’єктів із найменшою виміряною відстанню.

Така модифікація алгоритму спрямована на оптимізацію пошуку найближчих сусідів і водночас забезпечує більш ефективне використання ресурсів і швидкість пошуку.

На рис. 1 зображено фінальний етап роботи алгоритму, а саме методику знаходження найближчих сусідів, основану на вибраних атрибутах.

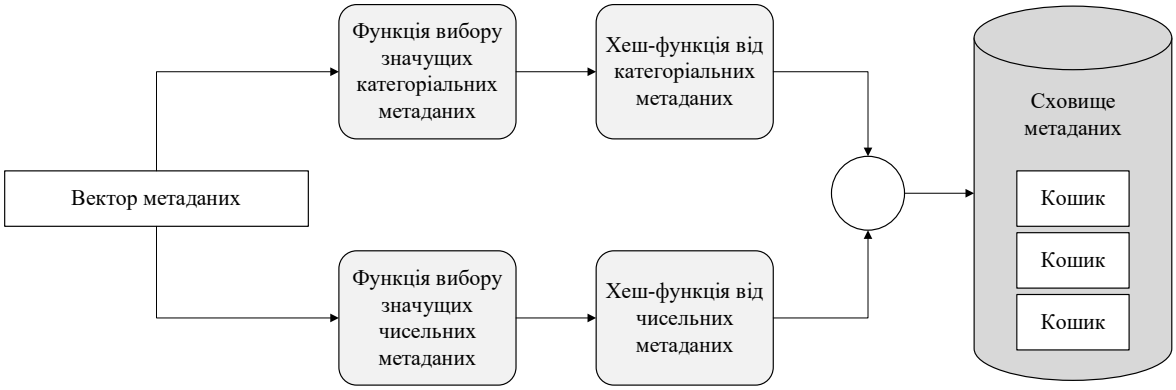

Рис. 1. Процес пошуку найближчих сусідів

Запропонована програмна архітектура застосовує зазначений метод, що передбачає детальний аналіз використовуваних атрибутів даних, показників застосування обчислювальних ресурсів, а також методики оцінювання ресурсних витрат на основі попередніх досліджень.

Планувальник завдань і ресурсів відповідає за організацію оброблення даних і розподіл обчислювальних ресурсів, інтегруючи модулі для оцінювання ресурсних потреб і розподілу ресурсів. Модуль оцінювання ресурсних потреб аналізує історичні експериментальні дані та реалізує функціонал fust.2 з пропонованого методу планування. Це передбачає обчислення коефіцієнтів ресурсних витрат і створення набору кортежів вигляду $\{t,k,c\} \in Ghist$ на основі отриманих метрик Mhist. Цей модуль також слугує для оцінювання ресурсних потреб нових завдань із використанням попередньо побудованої моделі для знаходження схожих завдань Thist.nearest, чиї ресурсні потреби усереднюються для нового завдання. У підсумку модуль генерує матрицю ресурсних потреб C.

Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 3 (25)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Модуль розподілу ресурсів приймає цю матрицю ресурсних потреб C від модуля оцінювання та вирішує проблеми розподілу, шукаючи найефективніший спосіб розподілу наявних завдань на доступні обчислювальні вузли. Результатом є матриця розподілу ресурсів W .

Система управління ресурсами та модуль для збирання ресурсних метрик функціонують як додаткові компоненти. Система управління ресурсами отримує матрицю розподілу W від планувальника завдань, а потім видає інструкції для запуску конкретних завдань на відповідних обчислювальних вузлах. Додатково ця система має функціонал для сповіщення про завершення раніше ініціалізованих завдань і локацію, де зберігаються результати оброблення даних.

Модуль збирання метрик надає дані про використання системних ресурсів, таких як пам’ять, процесор, графічний процесор, системи введення-виведення тощо, на обчислювальних вузлах. Цей модуль також видає інформацію про часові характеристики виконання завдань і дає змогу отримувати деталі конфігурації кожного обчислювального вузла в системі. Він відіграє роль інформаційного сервісу в контексті сучасних систем управління ресурсами.

Модуль оброблення метаданих відповідає за операції з витягу, перетворення та збереження метаданих.

У межах засобу планування завдань існує об’єкт класу *metadata mapper*, який здійснює конвертацію метаданих до формату, що підходить для виконання пошуку найближчих сусідів згідно з рекомендованим алгоритмом. Також тут розміщується об’єкт класу *ModelBuilder*, який відповідає за побудову моделі прогнозування.

Для експериментального вивчення методів планування завдань додано клас *Task Generator*, що може створювати набори завдань із різними режимами інтенсивності: низький, середній та високий. Він також здатний моделювати різні рівні трудомісткості завдань за допомогою чотирьох типів розподілу: константний, рівномірний, випадковий та зважений дискретний.

Як приклад застосування модифікованого методу планування завдань розглянемо завдання декодування відео для мобільних пристроїв. Традиційні відеохостинги використовують заздалегідь оброблені профілі для декодування відео, але *Amazon Elastic Transcoder* дозволяє динамічну оренду ресурсів для цієї мети. Це забезпечує баланс між користувацькими потребами та оптимізацією ресурсів. Застосовується метод планування на основі метаданих і метрик для ефективного розподілу завдань.

У розподіленому виконавчому середовищі функціонує 33 обчислювальних вузли, конфігурація яких описана в табл. 1.

Таблиця 1. *Конфігурація РСОД*

Тип вузла	Одиниць	Cores	Sockets	RAM
transcoding.node. a	7	1	1	1024
transcoding.node.b	8	2	1	1024
transcoding.node.c	8	2	2	1024
transcoding.node.d	5	3	2	1024
transcoding.node.e	5	4	2	1024

Для різних груп метаданих існують окремі життєві цикли, які керують їх збиранням, перетворенням, збереженням і використанням. Властивості відеопотоку визначаються наперед, коли завантажуються нові вихідні відеофайли, тоді як властивості користувацького обладнання та мережі стають відомими тільки в разі надходження завдань. Метадані відео збираються за допомогою утиліти *MediaInfo* та зберігаються в централізованій базі даних. Щодо обладнання та мереж, то система планується з урахуванням їх типових характеристик, для яких задаються декодувальні профілі на основі утиліти *Handbrake*.

Для цілей експерименту завдання щодо оброблення даних генеруються програмним методом за допомогою генератора завдань і далі надсилаються до координаційного вузла (обидва компоненти розміщені на одній фізичній машині). Тут відбувається розподіл завдань на оброблювальні вузли. Генератор завдань випускає набір кортежів, що складаються з типу пристрою, типу мережного з’єднання та ідентифікатора відеофайлу, з фіксованим часовим інтервалом T .

Після надходження таких кортежів у систему відбувається збір метаданих, їх перетворення та безпосереднє оброблення інформації. Запит для оброблення ініціюється користувачем через вибір відеофайлу за допомогою команди *show_video(id)*. Після цього програмний клієнт доповнює запит метаданими, що містять інформацію про тип обладнання та характеристики мережного з’єднання через функцію *add_metadata()*. Зрештою відеодекодувальний сервіс отримує завдання

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2023. No. 3 (25)

у вигляді кортежа, що містить тип пристрою, тип мережного з'єднання та ідентифікатор відеофайлу.

Після отримання завдання сервіс ініціює процес відображення вхідних даних у відповідний набір метаданих, які керують операцією декодування, через функцію *map_metadata()*. Цей процес відбувається в декілька стадій. Насамперед за ідентифікатором відеофайлу завантажуються відповідні метадані вихідного відеопотоку з бази даних. Далі за ідентифікатором типу обладнання з бази даних завантажується профіль декодування, що визначає характеристики вихідного відеопотоку.

Також на основі ідентифікатора типу мережного з'єднання з бази даних завантажується окремий профіль декодування, який ставить обмеження на якість вихідного відеопотоку з огляду на характеристики мережного з'єднання. Якість декодування, зокрема відеобітрейт, визначається відповідно до трьох параметрів: значення властивості у вихідному відеофайлі (*psrc*), характеристика, специфічна для певного типу обладнання (*Pdevice*), і верхнє обмеження, основане на швидкості мережного з'єднання (*Pconnection*).

Отже, складність операції декодування формується на підставі характеристик вихідного відеофайлу, апаратних можливостей типу обладнання для відтворення та пропускної здатності мережного з'єднання. Наприклад, незалежно від того, наскільки потужне обладнання, недоречно надавати відеопотік високої якості в разі обмеженої пропускної здатності мережі, оскільки це може викликати нестабільність у передачі даних, зокрема джитер (*jitter*).

Після конвертації метаданих сервіс інтегрує нове завдання в чергу планувальника за допомогою функції *add_task(data)*. Планувальник, зазвичай відомий як *Scheduler*, забезпечує динамічне розміщення завдань на вузлах-оброблювачах завдяки методу *assign_task(task_data)*. Взамін вузол-оброблювач декодує відеодані, а URL для доступу до вихідного відеопотоку надсилається на клієнтське обладнання.

Найбільш звичайний підхід до блокування завдань між вузлами-оброблювачами ґрунтується на використанні пріоритетної черги *FIFO (First In, First Out)* та алгоритму виділення завдань на найменш завантажений вузол, зазвичай позначеного як *LLF (Least Loaded First)*.

Цей метод, відомий як *FIFO_LLF*, був обраний як базовий для аналітичного порівняння переважно тому, що він, так само як і модифікований метод планування на основі метаданих (*Meta_Sched*), не вимагає попереднього знання про ресурсні потреби. Додатково цей підхід використовується для акумуляції емпіричних даних, що пізніше можна використати для оптимізації методу *Meta_Sched*.

Метод *Meta_Sched* вимагає розроблення моделі, що здатна визначати обчислювальні витрати для нових завдань на основі раніше зібраної статистичної інформації про виконані завдання. У процесі виконання різноманітних завдань, крім отримання результатів декодування, система також збирає метрики, пов'язані з використанням обчислювальних ресурсів, і зберігає їх в базі даних. До кожного запису також додаються метадані про саме завдання й характеристики обчислювального вузла.

Специфічні метрики й параметри обчислювальних вузлів, що збираються, містять середнє завантаження процесорів вузла під час декодування (*avg_cpu_load_percent*), кількість оперативної пам'яті, використаної процесом декодування (*ram_load_kb*), та низку часових штампів, зокрема момент уведення завдання в систему (*submit_timestamp*) та початок декодування (*start_processing_timestamp*). Додатково збирається інформація про час, потрібний для оброблення даних (*elapsed_seconds*), доступний обсяг оперативної пам'яті на вузлі (*node_free_ram*) і кількість вільних процесорів на вузлі (*node_free_processors*).

Ці показники дають змогу оцінювати не тільки обчислювальні й часові затрати, пов'язані з обробленням інформації, але й затрати часу на організаційні аспекти завдань, а саме: період очікування та тривалість виконання.

Для кожного запису обчислюється нормалізована величина затрат за формулою:

$$cost = W1 \times cpu\_cost\left(avg\_ram\_load\_kb, node\_free\_ram\right) + \\ + W2 \times ram\_cost\left(avg\_cpu\_load, node\_free\_processors\right), \tag{9}$$

де *cpu_cost* – функція оцінювання процесорних затрат; *ram_cost* – функція оцінки затрат пам'яті; $W1$, $W2$ – вагові коефіцієнти значущості обчислювального ресурсу.

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 3 (25) ISSN 2524-2296 (online)

У формулі (9) вагові фактори налаштовуються так, щоб надати перевагу ключовим ресурсам залежно від типу завдання:

– для обчислювально інтенсивних завдань увага переноситься на процесор;

– для завдань, що потребують великих обсягів пам'яті, увага переноситься на пам'ять;

– для підсистеми введення-виведення;

– для графічної компоненти;

– для мережних операцій.

Задача декодування відео належить до категорії обчислювально інтенсивних, тому коєфіцієнт важливості для процесора *W1* дорівнює 1. За паралель, споживання пам'яті є мінімальним, із ваговим коефіцієнтом *W2* рівним 0,3.

Функції *cpu_cost* і *ram_cost* аналізують вартість використання конкретних ресурсів, водночас беручи до уваги ступінь завантаженості або перевантаження певного ресурсу.

Для вимірювання відстаней до всіх потенційних об'єктів з отриманого набору поатрибутні відстані коригуються залежно від важливості кожного числового параметра. На основі оцінки використання ресурсів для всіх задач і оброблювальних вузлів формується матриця ресурсних затрат. Створена матриця ресурсних затрат застосовується для вирішення проблеми розподілу, що приводить до отримання матриці призначень, згідно з якою завдання оброблення даних будуть ділеговані на відповідні вузли.

Для оцінювання ефективності методів планування завдань розподілена обчислювальна система тестується в різних обставинах. Сценарії тестування варіюються за розподілом складності вхідного потоку завдань та за рівнем завантаженості системи.

Результати досліджень та їх обговорення

Для тестування модифікованого методу планування завдань здійснено планування та оброблення завдань з декодування відеоінформації впродовж 10 хв. Після цього на основі агрегованих статистичних показників якість виконання методів планування *FIFO_LLF* та *Meta_Sched* оцінюється за допомогою таких метрик:

– рівень використання процесорних ресурсів обчислювального вузла;

– середній час очікування на оброблення завдання, у секундах;

– загальний час перебування завдання в системі, у секундах (час очікування та час оброблення);

– середній коефіцієнт затримки оброблення завдання, у секундах (час очікування поділено на час оброблення);

– флуктуація загального часу виконання завдань для різних сценаріїв;

– коливання середнього часу очікування на оброблення завдання в різних сценаріях.

Компоненти програмної системи зображені на діаграмі компонентів і діаграмі класів (рис. 2 та 3).

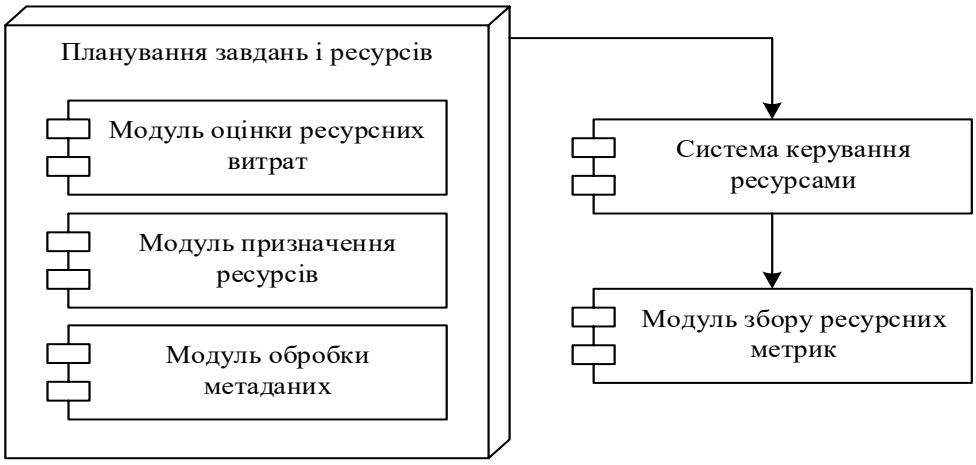

Рис. 2. Діаграма компонентів системи планування завдань і ресурсів РСОД

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2023. No. 3 (25)

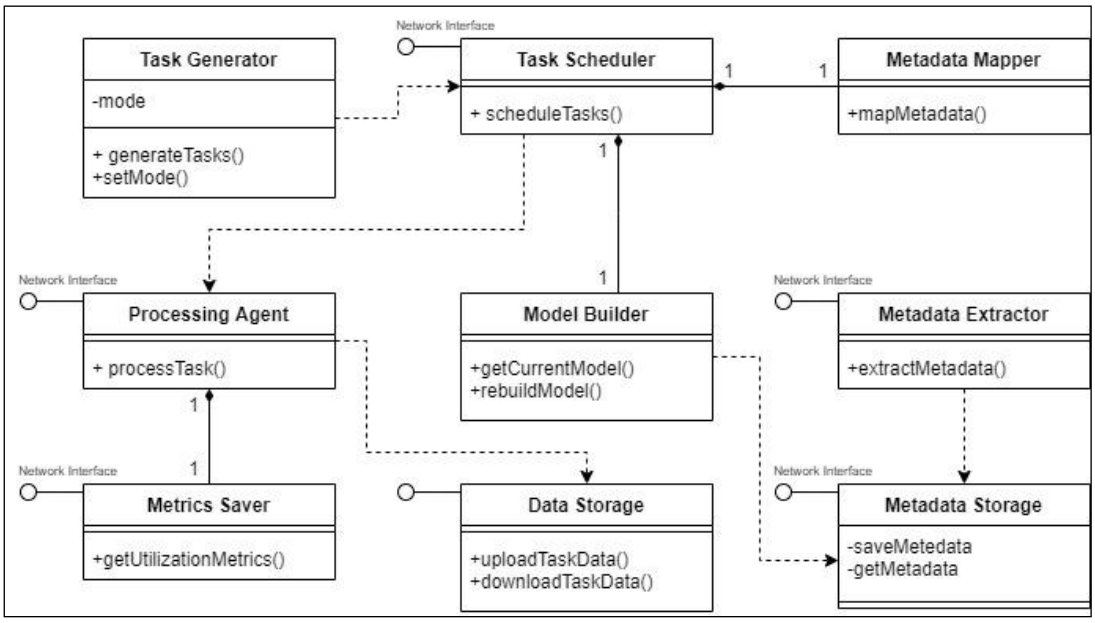

Рис. 3. Діаграма класів

У цьому контексті рівень використання процесорних ресурсів обчислювального вузла визначається з огляду на відсоток процесорного часу, який отримує паралельне завдання, а також зважаючи на кількість доступних процесорів.

За умови середнього навантаження метод *FIFO_LLF* незначно випереджає метод *Meta_Sched*. Це сталося тому, що *Meta_Sched* некоректно алокував деякі завдання, унаслідок чого ресурси стали перевантаженими. Крім того, в оптимальному діапазоні [0.8, 1] метод *Meta_Sched* незначно відстає від методу *FIFO_LLF*, якщо звернути увагу на кількість точок (70 проти 90).

Для адекватної інтерпретації отриманих показників важливо визначити ключові критерії, що оцінюють придатність розглянутого методу планування для реального впровадження. Основні оцінювальні параметри передбачають: аналіз часових показників оброблення завдань, зокрема середній час перебування заявки в системі, середня затримка й середній час оброблення. Другий критерій – це ефективність використання ресурсів. З погляду планування ідеальним вважається зайнятість ресурсу в діапазоні [0.8, 1]. Значення, нижчі від 0.8, свідчать про недостатнє використання ресурсів, тоді як значення понад 1 вказують на їх перевантаження.

За умови високого рівня завантаження метод планування, оснований на метаданих і ресурсних метриках, демонструє відчутну перевагу. Зокрема набір точок, який спочатку потрапив у діапазон [2, 6], був зміщений до діапазону [0, 2], що свідчить про зниження перевантаження системи.

Тимчасові показники, зокрема середній час перебування заявки в системі, середня затримка й середній час очікування на оброблення, є величинами, що мають схожий масштаб для обох розглянутих методів планування.

Щодо ефективності використання обчислювальних можливостей, то це питання буде досліджено для кожного тестового сценарію. В умовах низької навантаженості метод *FIFO_LLF* демонструє вищу кількість точок у діапазоні [0, 0.8), що свідчить про недостатнє використання ресурсів. Також цей метод має завдання, для яких виділено недостатньо обчислювальної потужності (з використанням процесора понад 2). У діапазоні [0.8, 1] обидва методи планування показують подібні результати. Запропонована архітектура програмної системи для планування завдань у розподіленій обчислювальній системі втілює метод, оснований на метаданих і ресурсних метриках.

Аналітичний огляд результатів свідчить про таке: застосування методу планування, основаного на метаданих, призводить до зменшення загального часу перебування заявок у системі на 20 % за умови високої інтенсивності вступу завдань, однак цей показник збільшується на 4,5 % у разі середньої інтенсивності та на 10,7 %, якщо інтенсивність низька. Також середній час очікування на оброблення заявок скорочується на 7,58 %, коли висока

Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 3 (25)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

інтенсивність вступу завдань, але зростає на 10,55 % за умови середньої інтенсивності та на дуже значні 70,6 %, якщо інтенсивність низька.

Висновки

Розглянуто системи управління ресурсами та сучасні методи планування завдань у розподілених системах оброблення даних, що вказало на необхідність розроблення нових програмних рішень, які ґрунтуються на плануванні в умовах недостатньої інформації про ресурсні потреби.

Розглянуто недоліки методів розподілення запланованих завдань без попереднього знання про ресурсні потреби.

Створено метод планування завдань на основі метаданих, оснований на модифікованому алгоритмі пошуку найближчих сусідів через локалізоване хешування з використанням алгебри скінченних предикатів, що бере до уваги типи атрибутів та їх важливість для ресурсопотреби, щоб спростити процес пошуку найближчих завдань.

З метою тестування зазначеного методу створено програмну архітектуру для системи планування завдань у РСОД. Визначено критерії для характеристик даних (метаданих) та метрик ресурсопотреби, взаємозв'язок між якими дає змогу створювати прогнозні моделі для оцінювання потреб нових завдань, зважаючи на історію виконання попередніх завдань.

Реалізовано програмне рішення з використаного методу на прикладі практичного сценарію – у завданні планування декодування відеоінформації. Проведено серію експериментів, результати яких показали, що середній час виконання завдань зменшився приблизно на 20 % за умови високого рівня надходження запитів.

Список літератури

1. Endo, P., de Almeida Palhares, A., Pereira, N., Goncalves, G., Sadok, D., Kelner, J., Melander, B. & Mangs, J.E. Resource allocation for distributed cloud: concepts and research challenges. *IEEE Network*. 2011. Vol. 25, No. 4. P. 42–46. DOI: 10.1109/mnet.2011.5958007

2. Sunyaev A. Principles of Distributed Systems and Emerging Internet-Based Technologies. *Internet Computing. Cham: Springer International Publishing,* 2020. Vol. XVIII. 413 p. DOI: 10.1007/978-3-030-34957-8

3. Siva Prasad B. V., Sucharitha G., Venkatesan K. G., Patnala T. R., Murari T., Karanam S. R., Optimisation of the Execution Time Using Hadoop-Based Parallel Machine Learning on Computing Clusters. *Computer Networks, Big Data and IoT*. Singapore, 2022. P. 233–244. DOI: 10.1007/978-981-19-0898-9_18

4. Kamran M. Fundamentals of Smart Grid Systems. Elsevier Science & Technology Books, 2023. 500 p. URL: https://shop.elsevier.com/books/fundamentals-of-smart-grid-systems/kamran/978-0-323-99560-3

5. Hasimi L., Penzel D. A Case Study on Cloud Computing: Challenges, Opportunities, and Potentials. *Studies in Systems, Decision and Control*. Cham, 2023. P. 1–25. DOI: 10.1007/978-3-031-27506-7_1

6. Zolotariov D. Microservice Architecture for Building High-Availability Distributed Automated Computing System in A Cloud Infrastructure. *Сучасний стан наукових досліджень і технологій в промисловості*. 2021. No. 3 (17). P. 13–22. DOI: 10.30837/itssi.2021.17.013

7. Tom L., Bindu V. R. Task Scheduling Algorithms in Cloud Computing: A Survey. *Inventive Computation Technologies*. Cham, 2019. P. 342–350. DOI: 10.1007/978-3-030-33846-6_39

8. Chen C., Shi H., Wang Z., Yu Z. A Task Scheduling Algorithm Based on Big.LITTLE Architecture in Cloud Computing". *In: 2020 6th International Conference on Big Data and Information Analytics (BigDIA),* 4–6 December 2020, Shenzhen, China IEEE. 2020. P. 94–99. DOI: 10.1109/bigdia51454.2020.00023

9. Chan C., Cooper B. Debugging incidents in Google's distributed systems. *Communications of the ACM*. 2020. Vol. 63, No. 10. P. 40–46. DOI: 10.1145/3397880

10. Langhnoja H. K., Hetal A Joshiyara P. Multi-Objective Based Integrated Task Scheduling In Cloud Computing. *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 12–14 June 2019. 2019. DOI: 10.1109/iceca.2019.8821912

11. Zolotariov D. Automated deployment of a software environment for microservices in a rapidly changing technology stack. *Сучасний стан наукових досліджень і технологій в промисловості*. 2021. No. 4 (18). P. 23–30. DOI: 10.30837/itssi.2021.18.023.

12. Danielsson J., Seceleanu T., Jagemar M., Behnam M., Sjodin M., Resource Depedency Analysis in Multi-Core Systems. *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain, 13–17 July 2020. 2020. DOI: 10.1109/compsac48688.2020.00021

13. Liu F., Guo W. Optimized Min-Min Dynamic Task Scheduling Algorithm in Grid Computing. *Advances in Intelligent Systems and Computing*. Cham, 2019. P. 745–752. DOI: 10.1007/978-3-030-25128-4_92

14. Stavrinides G. L., Karatza H. D. Scheduling Single-Task Jobs along with Bag-of-Task-Chains in Distributed Systems. *ICFNDS '19: 3rd International Conference on Future Networks and Distributed Systems*, Paris France. New York, NY, USA, 2019. P. 1–6. DOI: 10.1145/3341325.3342023

15. Yadav S., Mohan R., Yadav P. K. Fuzzy based task allocation technique in distributed computing system. *International Journal of Information Technology*. 2018. Vol. 11, No. 1. P. 13–20. DOI: 10.1007/s41870-018-0172-6

16. Li C., Liu F., Wang B., Philip Chen C. L., Tang X., Jiang J., Liu J. Dependency-Aware Vehicular Task Scheduling Policy for Tracking Service VEC Networks. *IEEE Transactions on Intelligent Vehicles*. 2022. P. 1–15. DOI: 10.1109/tiv.2022.3224057

17. Kozyriev A., Litvin S. Methods of Creating Service-oriented Software Systems. CEUR Workshop Proceedings. Vol. 3171, 2022. P. 763–774. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85134768124&partnerID=40&md5=6ed7a3d1bcc4e527548c2ade3019562d

18. Shubin I., Karataiev O. Reuse of Information Based on The Interpretation of Knowledge. *Сучасний стан наукових досліджень і технологій в промисловості*. 2023. No. 2 (24). P. 62–71. URL: https://doi.org/10.30837/itssi.2023.24.062 (дата звернення: 01.10.2023).

References

1. Endo, P., de Almeida Palhares, A., Pereira, N., Goncalves, G., Sadok, D., Kelner, J., Melander, B. & Mangs, J.E. (2011), "Resource allocation for distributed cloud: concepts and research challenges". *IEEE Network*. Vol. 25(4), P. 42–46. DOI: 10.1109/mnet.2011.5958007

2. Sunyaev A. (2020), "Principles of Distributed Systems and Emerging Internet-Based Technologies". *Internet Computing. Cham: Springer International Publishing,* 2020. Vol. XVIII. 413 p. DOI: 10.1007/978-3-030-34957-8

3. Siva Prasad, B. V., Sucharitha, G., Venkatesan, K. G., Patnala, T. R., Murari, T., Karanam, S. R. (2022), "Optimisation of the Execution Time Using Hadoop-Based Parallel Machine Learning on Computing Clusters". *Computer Networks, Big Data and IoT*. Singapore, 2022. P. 233–244. DOI: 10.1007/978-981-19-0898-9_18

4. Kamran, M. "Fundamentals of Smart Grid Systems". Elsevier Science & Technology Books, 2023. 500 p. available at: https://shop.elsevier.com/books/fundamentals-of-smart-grid-systems/kamran/978-0-323-99560-3

5. Hasimi L., Penzel D. (2023), "A Case Study on Cloud Computing: Challenges, Opportunities, and Potentials". *Studies in Systems, Decision and Control*. Cham. P. 1–25. DOI 10.1007/978-3-031-27506-7_1

6. Zolotariov, D. (2021), "Microservice Architecture for Building High-Availability Distributed Automated Computing System in A Cloud Infrastructure". *Сучасний стан наукових досліджень і технологій в промисловості.* No. 3 (17). P. 13–22. DOI: 10.30837/itssi.2021.17.013

7. Tom, L., Bindu, V. R. (2019), "Task Scheduling Algorithms in Cloud Computing: A Survey". *Inventive Computation Technologies*. Cham. P. 342–350. DOI: 10.1007/978-3-030-33846-6_39

8. Chen, C., Shi, H., Wang, Z., Yu, Z. (2020), "A Task Scheduling Algorithm Based on Big.LITTLE Architecture in Cloud Computing". *In: 2020 6th International Conference on Big Data and Information Analytics (BigDIA),* 4–6 December 2020, Shenzhen, China IEEE. P. 94–99. DOI: 10.1109/bigdia51454.2020.00023

9. Chan, C., Cooper, B. (2020), "Debugging incidents in Google's distributed systems". *Communications of the ACM*. 2020. Vol. 63, No. 10. P. 40–46. DOI: 10.1145/3397880

10. Langhnoja, H.K., Hetal, A., Joshiyara, P. (2019), "Multi-Objective Based Integrated Task Scheduling In Cloud Computing". *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 12–14 June 2019. DOI: 10.1109/iceca.2019.8821912

11. Zolotariov, D., (2021). "Automated Deployment of a Software Environment for Microservices in a Rapidly Changing Technology Stack". *Innovative Technologies and Scientific Solutions for Industries*. Vol. 4 (18), P. 23–30. DOI: 10.30837/itssi.2021.18.023

12. Danielsson, J., Seceleanu, T., Jagemar, M., Behnam, M., Sjodin, M. (2020), "Resource Depedency Analysis in Multi-Core Systems". *In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC),* 13–17 July 2020, Madrid, Spain. DOI: 10.1109/compsac48688.2020.00021

13. Liu, F., Guo, W. (2019), "Optimized Min-Min Dynamic Task Scheduling Algorithm in Grid Computing". *Advances in Intelligent Systems and Computing.* Cham: Springer International Publishing. P. 745–752. DOI: 10.1007/978-3-030-25128-4_92

14. Stavrinides, G. L., Karatza, H. D. (2019), "Scheduling Single-Task Jobs along with Bag-of-Task-Chains in Distributed Systems". *ICFNDS '19: 3rd International Conference on Future Networks and Distributed Systems*, Paris France. New York, NY, USA. P. 1–6. DOI: 10.1145/3341325.3342023

15. Yadav, S., Mohan, R., Yadav, P. K. (2018), "Fuzzy based task allocation technique in distributed computing system". *International Journal of Information Technology*. Vol. 11, No. 1. P. 13–20. DOI: 10.1007/s41870-018-0172-6

16. Li, C., Liu, F., Wang, B., Philip Chen, C. L., Tang, X., Jiang, J. & Liu, J. (2022), "Dependency-Aware Vehicular Task Scheduling Policy for Tracking Service VEC Networks". *IEEE Transactions on Intelligent Vehicles*. P. 1–15. DOI: 10.1109/tiv.2022.3224057

Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 3 (25)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

17. Kozyriev, A., Litvin, S. "Methods of Creating Service-oriented Software Systems". CEUR Workshop Proceedings. Vol. 3171, 2022. P. 763–774. available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85134768124&partnerID=40&md5=6ed7a3d1bcc4e527548c2ade3019562d

18. Shubin, I., Karataiev, O. (2023), "Reuse of Information Based on The Interpretation of Knowledge". *Сучасний стан наукових досліджень і технологій в промисловості*. No. 2 (24). P. 62–71. available at: https://doi.org/10.30837/itssi.2023.24.062 (last accessed: 01.10.2023)

Надійшла 28.08.2023

Відомості про авторів / About the Authors

Козирєв Андрій Дмитрович – Харківський національний університет радіоелектроніки, аспірант кафедри програмної інженерії, Харків, Україна; e-mail: andrii.kozyriev@nure.ua; ORCID ID: https://orcid.org/0000-0001-6383-5222

Шубін Ігор Юрійович – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, професор кафедри програмної інженерії, Харків, Україна; e-mail: igor.shubin@nure.ua; ORCID ID: https://orcid.org/ 0000-0002-1073-023X

Kozyriev Andrii – Kharkiv National University of Radio Electronics, Postgraduate, Kharkiv, Ukraine.

Shubin Ihor – PhD (Engineering Sciences), Associate Professor, Kharkiv National University of Radio Electronics, Professor at the Software Department, Kharkiv, Ukraine.

METHOD OF PLANNING DATA PROCESSING TASKS IN DISTRIBUTED SYSTEMS WITH LIMITED INFORMATION ABOUT AVAILABLE RESOURCES

In today's digital landscape, distributed data processing systems (DDPs) are becoming increasingly critical to efficiently process, analyze, and manage large volumes of data. These systems are often used in commercial, scientific and social domains to process complex data in real-time or batch mode. One of the key components of such systems is task scheduling, which is an extremely complex process, particularly when information about resource requirements is not complete or accurate. The **subject** of research are algorithms, methods and approaches used for scheduling tasks between nodes in distributed systems. The **purpose** of the study is to create an optimized method of task planning in the RSOD with limited availability of information about available resources. The **task of the research**: to analyze the limitations of modern methods for scheduling tasks in distributed data processing systems (DDS); optimize the method of scheduling tasks based on metadata between RSOD nodes, based on the methodology of searching for nearest neighbors using the method of localized hashing and the algebra of finite predicates; develop the architecture of the software solution and its implementation based on the optimized method; test the algorithm on the example of a video decoding task. The following **methods** were used: statistical algorithms and techniques such as classification and cluster analysis were used to predict resource requirements, and visualization techniques assisted in the analysis and interpretation of results. As a **result** of the work: the limitations of modern methods for the distribution of tasks in distributed data processing systems (DDPs) were analyzed; an optimized method of task planning based on metadata in RSOD was created, based on the methodology of searching for nearest neighbors using the method of localized hashing and the algebra of finite predicates; the processes in the modified nearest neighbor search algorithm are detailed; the architecture of the software solution was developed, which integrates an optimized method of task planning based on metadata and resource allocation; validation of the software solution was carried out with the help of a practical scenario – the use of the created algorithm in the planning task for decoding video information. The **conclusions** of this study confirmed that the proposed method, based on the methodology of localized hashing and the use of finite predicate algebra, is effective even with insufficient or limited information about resource needs. This highlights the possibility of using dynamic scheduling strategies to adapt to changing load conditions and resource availability.

Keywords: software engineering; distributed systems; data processing; database; task scheduling; algebra of finite predicates.

Бібліографічні описи / Bibliographic descriptions

Козирєв А. Д., Шубін І. Ю. Метод планування завдань оброблення даних у розподілених системах з обмеженою інформацією про доступні ресурси. *Сучасний стан наукових досліджень та технологій в промисловості*. 2023. № 3 (25). С. 27–39. DOI: https://doi.org/10.30837/ITSSI.2023.25.027

Kozyriev, A., Shubin, I. (2023), "Method of planning data processing tasks in distributed systems with limited information about available resources", *Innovative Technologies and Scientific Solutions for Industries*, No. 3 (25), P. 27–39. DOI: https://doi.org/10.30837/ITSSI.2023.25.027