УДК 004.75:004.453.3 DOI https://doi.org/10.35546/kntu2078-4481.2024.1.29

М. О. ВОЛК
доктор технічних наук, професор,
професор кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0000-0003-4229-9904

В. Г. ЛАБАЗОВ
аспірант кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0004-2790-5130

Д. О. КИПАРЕНКО
магістрант кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0002-5172-5979

Б. В. ПРИВАЛОВ
магістрант кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0004-8261-3667

Д. О. ШУМОВ
магістрант кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0000-8922-2715

І. А. САМОЙЛОВ
магістрант кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0000-2829-2744

РОЗПОДІЛЕНЕ КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ В СИСТЕМАХ ХМАРНИХ ОБЧИСЛЕНЬ

У статті розглядаються питання підвищення ефективності систем розподіленого моделювання в хмарному віртуальному середовищі. Однією з основних задач, яка з'являється в процесі моделювання складних систем є масштабування обчислень – збільшення кількості ресурсів при зростанні розміру завдання. В роботі проводиться дослідження моделей, які використовуються для комп'ютерного моделювання, зокрема послідовна та паралельна модель з різними методами синхронізації. Розглядається хмарна модель імітаційного середовища з використанням тонких клієнтів, децентралізованим керуванням та асинхронною синхронізацією, що дозволяє масштабувати систему моделювання в горизонтальному напрямку. Дана модель дозволяє гнучке керування ресурсами залежно від складності об'єкта моделювання, динамічне призначення та звільнення обчислювальних віртуальних вузлів. В роботі описані експерименти з моделювання три вимірних об'єктів на різній кількості віртуальних машинах. Використання багатьох вузлів, з одного боку, збільшує швидкість обчислень, з іншого боку вимагає великої обчислювальної потужності для синхронізації, керування та обміну повідомленнями між вузлами, тому залежність зростання швидкості при використанні додаткових хмарних вузлів не є лінійною. Залучення розподіленого середовища моделювання дозволило подвоїти розмір симуляції у випадку використання 4 обчислювальних вузлів. Спостерігалось зростання продуктивності із збільшенням кількості обчислювальних вузлів. Результати дослідження можуть бути використані при розробленні нових методів розподілу ресурсів та технологій розподілених обчислень в хмарних системах, моделей ефективного управління обчислювальними вузлами, системах розподіленого імітаційного моделювання.

Ключові слова: *розподілені системи, комп'ютерні ресурси, імітаційне моделювання, хмарні обчислення, масштабування, віртуальні машини.*

M. O. VOLK
Doctor of Technical Sciences, Professor,
Professor at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0000-0003-4229-9904

V. H. LABAZOV
Postgraduate Student at Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0009-0004-2790-5130

D. O. KIPARENKO
Master's Student at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0009-0004-8261-3667

B. V. PARVALOV
Master's Student at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0009-0004-8261-3667

D. O. SHUMOV
Master's Student at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0009-0000-8922-2715

I. A. SAMOILOV
Master's Student at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0009-0000-2829-2744

DISTRIBUTED COMPUTER SIMULATION IN CLOUD COMPUTING SYSTEMS

The article examines issues of increasing the efficiency of distributed modeling systems in a cloud virtual environment. One of the main tasks that appears in the process of modeling complex systems is the scaling of calculations – increasing the number of resources when the size of the task increases. The paper examines the models used for computer simulation, in particular the serial and parallel model with different synchronization methods. A cloud-based model of the simulation environment with the use of thin clients, decentralized control and asynchronous synchronization is considered, which allows scaling the simulation system in the horizontal direction. This model allows flexible management of resources depending on the complexity of the modeling object, dynamic assignment and release of computing virtual nodes. Experiments on modeling 3D objects on different number of virtual machines are described in the paper. The use of many nodes, on the one hand, increases the speed of calculations, on the other hand, it requires a lot of computing power for synchronization, management and exchange of messages between nodes, so the dependence of the increase in speed when using additional cloud nodes is not linear. The involvement of a distributed simulation environment made it possible to double the size of the simulation in the case of using 4 computing nodes. An increase in productivity was observed with an increase in the number of computing nodes. The results of the research can be used in the development of new methods of resource allocation and technologies of distributed computing in cloud systems, models of efficient management of computing nodes, systems of distributed simulation modeling.

Key words: *distributed systems, computer resources, simulation modeling, cloud computing, scaling, virtual machines.*

Постановка проблеми

Наука і техніка розвиваються все швидше, зростають можливості усього людства та потреби повсякденного, соціального та економічного життя. Але технологічний розвиток сучасних обчислювальних ресурсів є специфічним. Тактова частота процесора на комерційному ринку вже давно не перевищує 4–5 ГГц. Це в першу чергу пов'язано з властивостями елементної бази та технологіями виробництва, які використовується для створення процесорів. Таким чином, прискорення в основному досягається шляхом використання множини ядер процесорів і та організацією системи (архітектурою) так, щоб забезпечити паралельність виконання обчислювальних завдань та розподіл програмного забезпечення за віддаленими ресурсами [1].

Можна виділити три типи моделювання. Фізичне моделювання полягає у використанні фізично існуючих систем у відповідності з підготовленим сценарієм, тобто проводячи натурні експерименти. Віртуальне моделювання вимагає

взаємодії об'єктів в комп'ютерно змодельованому середовищі. Це може вимагати від експериментатора прийняття рішень, моторики та комунікативних навичок. Конструктивне моделювання – це комп'ютерна програма, операції якої базуються на агрегованих об'єктах моделювання. Вони представляють реальні ситуації в контексті сценарію моделювання [2].

Віртуальне моделювання має більш високу роздільну здатність і відповідність реальному світу, ніж конструктивне моделювання. На жаль, у разі великих, детальних та/або динамічних систем, може виявитися, що обчислювальної потужності одного пристрою недостатньо. Однією з можливостей є перехід до конструктивного моделювання. На жаль, багато об'єктів, об'єднаних в один, можуть призвести до втрати точності і необхідності калібрування його характеристик. Віртуальну симуляцію можна поширити на багато комп'ютерних ресурсів. Тому обчислювальна потужність одного пристрою не є обмежуючим фактором.

Движок кожної комп'ютерної програми, що виконує віртуальну симуляції базується на певному шаблоні проєктування програмного забезпечення (англ. software design patterns) [3]. Більшість з них поділяє програмне забезпечення на окремі програмні модулі за функціональним призначенням. Ці модулі можуть розроблятися паралельно різними фахівцями або командами. Наприклад, одним з найбільш відомих шаблонів є MVC (модель–представлення–контролер, англ. Model-view-controller) [4; 5]. Згідно цього шаблону програмна система поділяється на три елемента: модель даних, інтерфейс та контролер. У загальному випадку кожен з цих елементів також може складатися з окремих модулів. Наприклад, інтерфейс може реалізовувати взаємодію з користувачем, , іншими програмами, комп'ютерною мережею або зовнішніми пристроями. Під час одного циклу виконання програмної системи обробляються пристрої введення, тоді як обчислена фізика, логіка, штучний інтелект і відтворена графіка відображаються на екрані. Вищезазначені кроки в класичному підході є послідовними. Лише після зупинки циклу може початися інший.

Розглянемо типовий цикл функціонування системи моделювання на прикладі шаблона MVC з урахуванням, що кожен компонент шаблону може складатися з декількох програмних модулів (рис. 1).

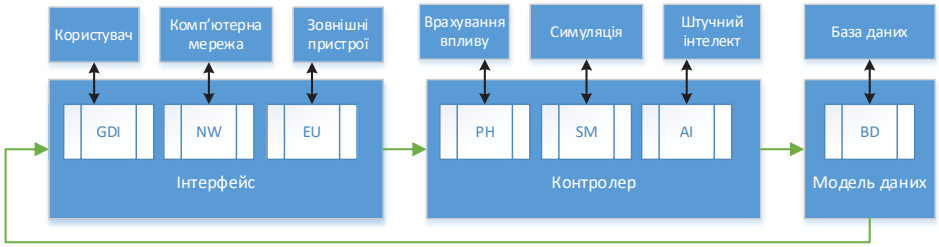

Рис. 1. Послідовне виконання процесу симуляції

На цьому рисунку можна продемонструвати, що кожен з модулів шаблону може мати свої окремі модулі: інтерфейс складається з модулів GDI (Graphic Design Interface), що забезпечує візуалізацію на екрані, модуля мережевої взаємодії NM (Network) та іншими зовнішніми пристроями EU (External Units), до яких мужуть відноситись клавіатура, датчики та інше. Так же саме контролер може включати модулі моделювання SM (simulation), штучного інтелекту AI та інших модулів, наприклад, бібліотек моделей фізичних явищ PH (physic).

Щоб підтримувати процес моделювання, а зображення у користувача, що відображає процес моделювання було реалістичним, кадри екрана повинні формуватися з певною частотою. Щоб отримати 60 кадрів на секунду, усі обчислення, пов'язані з одним циклом, мають тривати приблизно 17 мілісекунд. Допустима нижня межа для нормального сприйняття – 16 кадрів [ ]. В іншому випадку динамічне зображення не буде стабільним. У табл. 1 показано це явище шляхом порівняння кількості об'єктів із кадрами, що відтворюються за секунду. Дані були отримані в проекті моделювання системи візуалізації 3D об'єктів, виконаному на віртуальних машинах, з такими параметрами: 4 ядра процесора Intel Core i7-9700K/3,6GHz/ з 12MB оперативної пам'яті.

Таблиця 1

Порівняння кількісних параметрів для однієї віртуальної машини

Кількість циклів	Кількість об'єктів	Час формування кадру, сек.	Кількість кадрів в секунду
10	100	16	60
10	500	18	55
10	1000	22	45
15	100	24	42
15	500	28	35
15	1000	32	31
20	100	33	30
20	500	45	22
20	1000	57	17

Як можна побачити з табл. 1, зі зростанням кількості ітерацій моделювання та кількості об'єктів, що моделюються в системі, зростає час формування одного кадру зображення, тож зменьшується кількість кадрів, що система може відобразити на екрані. Те ж саме стосується і інших модулів системи. Зрозтання кількості зовнішнішніх джерел впливу, розмірності, наприклад, нейромереш в системах штучного інтелекту, призводить до аналогічного результату.

При впровадженні архітектури багатоядерних процесорів виявилося, що послідовні системи моделювання виявились неефективними. Тому ідеальним рішенням є асинхронний цикл, де завданням не потрібно чекати результатів інших завдань. Замість цього враховується останній обчислений результат, взятий зі спільного ресурсу (рис. 2). На жаль, така модель може бути надто складною для реалізації на практиці. Деякі завдання необхідно виконувати послідовно.

У випадку з комп'ютерним моделюванням однорангова мережа та клієнт-сервер є найбільш часто використовуваними типами архітектури багатокористувацької програмної системи. Головною перевагою однорангового зв'язку є відсутність центрального сервера. Вартість створення та підтримки такої мережі невисока. На жаль, рішення не масштабується. Швидкість з'єднань обмежує пропускну здатність, а впровадження є складним. Повільне підключення одного з клієнтів може спричинити затримки в процесі моделювання.

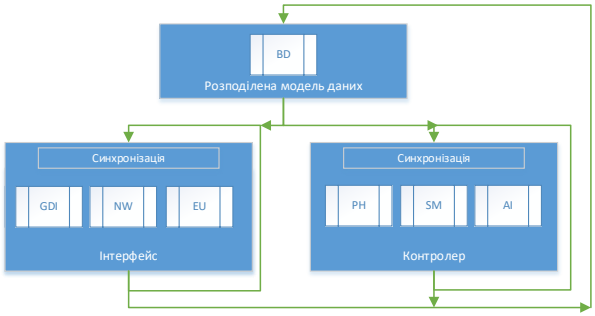

Рис. 2. Паралельне виконання процесу симуляції

Виникають не тільки деякі проблеми з синхронізацією, але і необхідно підтримувати фіксовану кількість навантаження для всіх клієнтів, що призводить до необхідності вирішувать завдання балансування навантаження [6]. Архітектура клієнт-сервер вимагає одного комп'ютера, який використовується як сервер і робить послугу доступною для клієнтів. Користувачі використовують клієнтські програми, які взаємодіють із сервером. Програми обмінюються повідомленнями та отримують список змін, які необхідно зробити в локальному віртуальному пространстві [7].

Сервер може виступати в якості посередника в процесі обміну повідомленнями між клієнтами або брати активну участь в процесі моделювання. У такому випадку саме сервер приймає рішення щодо подій, що відбуваються. Деякі затримки, спричинені мережевою інфраструктурою, можуть ускладнити процес моделювання та забезпечення його реалістичності. В результаті можуть виникати відмінності в статусі об'єкта на сервері і клієнтських пристроях.

Можливим вирішенням цієї проблеми є використання механізму прогнозування, який оцінює майбутні властивості об'єкта на основі поточних даних (наприклад, положення, швидкість, напрямок). Архітектура традиційних мережевих симуляторів передбачала використання одного сервера в архітектурі клієнт-сервер. У більш складному випадку одного сервера недостатньо для обробки такої кількості з'єднань і даних. Тому потрібно багато серверів. Такі сервери можуть бути незалежними (статуси про стан моделювання зберігаються на кожному сервері окремо, без синхронізації), або мають загальну базу даних. У першому випадку користувач може вибрати сервер для входу. У другому випадку є машини, які протидіють переміщенню на серверах і призначають користувачів до найменш завантаженої машини [7]. У контексті симуляторів найчастіше використовуються наступні два типи архітектури: розподілене інтерактивне моделювання (DIS) і архітектура високого рівня (HLA). DIS є стандартом обміну інформацією між симуляторами. Він не має центрального сервера керування. Симулятори можуть приєднуватися до симуляції та залишати її в будь-який момент. Статус симуляції зберігається в повідомленні під назвою Protocol Data Unit (PDU) і обмінюється між симуляторами через доступний протокол транспортного рівня, використовуючи багатоадресний або широкомовний метод. Поточна версія DIS 7 визначає 72 різних типи PDU [8].

Архітектура високого рівня (HLA) – це архітектура, розроблена для комп'ютерних систем (зокрема, систем моделювання). Ідея цього рішення полягає в тому, щоб зробити зв'язок незалежним від платформ, на яких встановлені системи. Тренажер відповідно до HLA називається федеративним. Системи пов'язані з інфраструктурою

часу виконання (RTI) і разом створюють так звані федерації. Елементи системи спільно використовують дані про об’єкти, які знаходяться в об’єктній моделі об’єднання (FOM). Додатково деякі події (взаємодії) з параметрами пересилаються між симуляторами. На жаль, застосування вищезазначених типів архітектури для багатьох симуляторів має один головний недолік. З логічної точки зору, віртуальний світ моделюється на кожному федераті. Основним призначенням архітектури DIS і HLA є синхронізація різних типів симуляторів. Це не дозволяє розподілити віртуальний світ між різними симуляторами.

Формулювання мети дослідження

Мета даної роботи полягає у підвищенні ефективності розподіленого комп'ютерного моделювання шляхом створення моделі використання хмарного середовища для виконання окремих програмних модулів.

Виклад основного матеріалу дослідження

Використання обчислювальних хмарних рішень у віртуальному моделюванні (рис. 3) передбачає використання тонких клієнтів. Зараз використовуються товсті клієнти з швидкими процесорами та відеокартами. Використовувати тонкі клієнти можна лише за умови обробки завдань кожного клієнта сервером. У середовищі моделювання це називається хмарним моделюванням. Наприклад щодо комп'ютерної графіки, фірма NVIDIA під комерційною назвою NVIDIA GRID пропонує таку послугу. Рішення дозволяє відправляти вхідні дані від клієнта до сервера. Клієнт отримує відеопотік, який декодується і виводиться на екран. Тому симуляція не залежать від платформи, яку повинен мати клієнт.

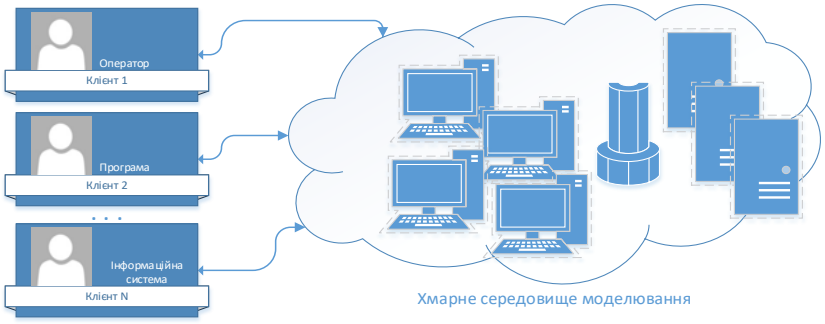

Рис. 3. Імітаційна хмарна модель

Однією з особливостей хмари моделювання є асинхронний механізм, який дозволяє масштабувати в горизонтальному напрямку. Таким чином можна ініціалізувати обчислювальні вузли, якщо виникне потреба. Важливо, щоб кожен вузол відповідав за різні об’єкти моделювання та дозволяв взаємодію між об’єктами, які керуються іншими вузлами. Об'єкти можуть бути закріплені за обчислювальними вузлами за територіальною ознакою. Коли об’єкт знаходиться в зоні, керованою даним вузлом, то ним керує такий вузол. Розміри зони можуть відрізнятися. Наприклад, міста щільніше заповнені симуляційними об’єктами, ніж ліси. Вузли ініціалізуються та призначаються як статично, так і динамічно. Під час моделювання та аналізу можна підготувати задовільний розподіл призначених обчислювальних вузлів. Операцію слід повторювати для кожного сценарію. Динамічне призначення означає постійний аналіз завантаження обчислювальних вузлів, ініціювання нових і звільнення таких вузлів.

Такий механізм дозволяє масштабувати ресурси в залежності від складності моделювання [7]. Перевагою динамічного розподілу є гнучкість і економія хмарних ресурсів, якщо їх використання є обов'язковим. З іншого боку, може виникнути певне навантаження через тестування та аналіз поточного та прогнозованого навантаження вузлів. Ще однією перевагою динамічного розподілу обчислювальних вузлів є більша стійкість до збоїв. У разі несподіваної втрати функціональності вузла, ініціюється новий вузол і моделювання продовжується. Крім стандартних обчислювальних блоків, можна використовувати спеціальні та спеціалізовані блоки. Проста модель симулятора підтримує фізику, логіку та штучний інтелект. Виділені вузли можуть використовувати спеціальні графічні карти, процесори або компоненти, що підтримують мобільні пристрої для організації хмари [9].

Платформа SpatialOS від Improbable є одним із комерційних продуктів, які використовують розподілену обробку в обчислювальній хмарі для віртуального моделювання. Рішення базується на платформі Google Cloud. Віртуальна симуляція в SpatialOS управляється обчислювальними вузлами, відомими як Workers. Якщо зростає потреба в обчислювальній потужності моделювання, ініціюються нові екземпляри обчислювальних вузлів. Розрізняють два види працівників: керовані та зовнішні. Керовані працівники – це ті, життєвим циклом яких керує SpatialOS. Зовнішні працівники зазвичай є клієнтами. SpatialOS не визначає, коли зовнішній робочий пристрій підключається або відключається. До платформи SpatialOS підключені всі типи працівників. Якщо подія відбувається в одному з вузлів (клієнт або робочий), вона надсилається в SpatialOS, яка інформує про це інші вузли. У випадку з клієнтами події обробляються лише для відображення 3D-моделей, текстур і анімації. Керовані

працівники займаються обчисленнями частини світу, призначеної їм платформою SpatialOS. Зовнішні працівники знають лише ту частину світу, яка оточує об'єкт симуляції, з якою вони пов'язані. В рамках проекту можливе налаштування способу організації обчислювальних вузлів. У разі статичного розподілу встановлюються координати та номер вузла. Динамічний розподіл вимагає визначення максимальної кількості вузлів, а також автомасштабування на основі максимальної кількості керованих об’єктів [15].

Мета наших тестів полягала в тому, щоб виявити різницю в потужності між симуляціями на основі 1, 2 або 4 обчислювальних вузлів. Моделювання проводилось на платформі SpatialOS. У сценарії використовувався статичний метод розподілу вузлів. Визначено 4 положення вузлів за координатами: <-250,-250>, <-250,250>, <250,-250> та <250,250>. Об'єкти були розміщені випадковим чином, на а відстань 500 одиниць від позиції <0,0>.

Тест проводився в конфігурації 1, 2, 4 вузлів і 300, 600, 900, 1200, 1500 об'єктів, з яких 30 були статичними елементами оточення. Решта були динамічними об'єктами, що рухалися у випадковому напрямку. Результати тестування наведені в табл. 2.

Таблиця 2

Порівняння кількості циклів моделювання, об'єктів і кадрів для різної кількості віртуальних машин

Кількість об'єктів	Кількість робочих вузлів					
	1		2		4	
	Кількість кадрів в секунду	Кількість об'єктів на вузел	Кількість кадрів в секунду	Кількість об'єктів на вузел	Кількість кадрів в секунду	Кількість об'єктів на вузел
300	45	300	55	150	60	75
600	31	600	52	300	58	150
900	15	900	34	450	49	225
1200	8	1200	17	600	25	300
1500	3	1500	9	750	13	375

Результати тестування показують, що збільшення кількості кадрів в секунду не є лінійним у порівнянні зі збільшенням кількості обчислювальних вузлів. Одного вузла не вистачає на 780 об'єктів. На рисунку 4 показана панель діаграми, що представляє дані з табл. 2. Застосування 4 вузлів дозволило плавне моделювання на задовільному рівні з 1200 об’єктами.

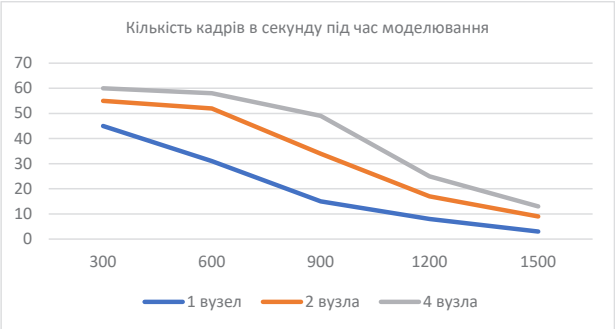

Рис. 4. Діаграма залежності кількості кадрів від кількості об'єктів для різної кількості обчислювальних вузлів

Найбільший приріст FPS помітний при переході з 1 вузла на 2 вузла. Найменший приріст FPS помітний при переході з 2 вузлів на 4 вузли. Отже, очевидно, що використання більшої кількості вузлів вимагає використання додаткових ресурсів для роботи та синхронізації таких вузлів.

Низьке збільшення потужності, ймовірно, є результатом високої надлишковості об’єктів, призначених вузлам. У випадку 4 обчислювальних вузлів загальна сума всіх об’єктів у вузлах вдвічі перевищує кількість усіх об’єктів у моделюванні. Це викликано перекриттям зон відповідальності вузлів і високою щільністю розподілу об'єктів між усіма вузлами.

Висновки

У статті розглянуто концепцію використання розподіленого віртуального моделювання. Показані типові проблеми, з якими зараз стикається моделювання складних систем. Представлено рішення з використанням обчислювальної хмари для розподіленого моделювання. Застосування багатьох вузлів вимагає великої обчислювальної потужності для керування та обміну повідомленнями між ними, тому залежність з залученням додаткових хмарних вузлів не є лінійною. Використання розподіленого середовища дозволило подвоїти розмір симуляції з 4 обчислювальними вузлами.

Чим більше вузлів, тим менше збільшення продуктивності. Для тестування використовувався сценарій, за якого об'єкти не змінювали свого розташування протягом короткого часу. На практиці мінливе віртуальне середовище може викликати різницю в навантаженні обчислювальних вузлів для різних сценаріїв. Подальші дослідження будуть присвячені методам і прийомам призначення завдань (об'єктів моделювання) обчислювальним вузлам, розробці ефективної моделі управління та зв'язку обчислювальних вузлів як одного з хмарних модулів моделювання.

Список використаної літератури

1. Farhanaaz, Sanju V. Compiling with MultiCores. 2nd International Conference for Innovation in Technology, INOCON 2023. P. 1–8 DOI: 10.1109/INOCON57975.2023.10101054

2. Weatherly R. M., Wilson A. L., Canova B. S., Page E. H., Zabek A. A., Fisher M. C. Advanced Distributed Simulation though the Aggregate Level Simulation Protocol, *Proceedings of the 29thHawaii International Conference on Systems Sciences.*1996. P. 407–414. DOI: 10.1109/HICSS.1996.495488

3. Aratchige R., Manujaya K., Weerasinghe P. An Overview of Structural Design Patterns in Object-Oriented Software Engineering. Sofware Modeling. 2024. P.1-3. DOI: 10.13140/RG.2.2.16089.90724.

4. García R. MVC: Model–View–Controller. iOS Architecture Patterns. 2023. P.45-106. DOI: 10.1007/978-1-4842-9069-9_2.

5. Thakur R.N., Pandey U.S. The Role of Model-View Controller in Object Oriented Software Development. Nepal Journal of Multidisciplinary Research. No 2. 2019. P. 1–6. DOI: 10.3126/njmr.v2i2.26279.

6. Ivanisenko I.M., Volk M.O. Simulation methods for load balancing in distributed computing. Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2017), Novi Sad, Serbia, September 27 – October 2, 2017. P. 690–695. DOI: 10.1109/EWDTS.2017.8110078

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

8. Peng Y., Dang W., Yin Q. Distributed simulation of MAS-based interactive applications with HLA. WIT Transactions on Information and Communication Technologies. No60. 2014. P. 229–238. DOI: 10.2495/CTA140281.

9. Mamchych O., Volk M. Smartphone Based Computing Cloud and Energy Efficiency.12th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 2022, pp. 1–5, DOI: 10.1109/DESSERT58054.2022.10018740

References

1. Farhanaaz, Sanju, V. (2023) Compiling with MultiCores. 2nd International Conference for Innovation in Technology, INOCON 2023. P. 1–8 DOI: 10.1109/INOCON57975.2023.10101054

2. Weatherly, R., M., Wilson, A.,L., Canova, B., S., Page, E., H., Zabek, A., A., Fisher, M., C. (1996) Advanced Distributed Simulation though the Aggregate Level Simulation Protocol, *Proceedings of the 29thHawaii International Conference on Systems Sciences.* P. 407–414. DOI: 10.1109/HICSS.1996.495488

3. Aratchige, R., Manujaya, K., Weerasinghe, P. (2024) An Overview of Structural Design Patterns in Object-Oriented Software Engineering. Sofware Modeling. P. 1–3. DOI: 10.13140/RG.2.2.16089.90724.

4. García, R.(2023) MVC: Model–View–Controller. iOS Architecture Patterns. P.45-106. DOI: 10.1007/978-1-4842-9069-9_2.

5. Thakur, R.,N., Pandey, U.,S. (2019) The Role of Model-View Controller in Object Oriented Software Development. Nepal Journal of Multidisciplinary Research. No 2. P. 1–6. DOI: 10.3126/njmr.v2i2.26279.

6. Ivanisenko, I.,M., Volk, M.,O. (2017) Simulation methods for load balancing in distributed computing. Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2017), Novi Sad, Serbia, September 27 – October 2. P. 690–695. DOI: 10.1109/EWDTS.2017.8110078

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

8. Peng, Y., Dang, W., Yin, Q. (2014) Distributed simulation of MAS-based interactive applications with HLA. WIT Transactions on Information and Communication Technologies. Vol. 60. P.229-238. DOI: 10.2495/CTA140281

9. Mamchych, O., Volk, M. (2022) Smartphone Based Computing Cloud and Energy Efficiency.12th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, pp. 1–5, DOI: 10.1109/DESSERT58054.2022.10018740