УДК 004.92

О. Н. Романюк, д-р техн. наук, проф.; В. П. Майданюк, канд. техн. наук, доц.; О. В. Романюк, канд. техн. наук, доц.; О. Л. Бобко; О. О. Новосельцев

РОЗПОДІЛЕННЯ ОБЧИСЛЕНЬ І БАЛАНСУВАННЯ НАВАНТАЖЕННЯ У ВИСОКОПРОДУКТИВНИХ СИСТЕМАХ ПАРАЛЕЛЬНОГО РЕНДЕРИНГУ

У статті досліджено питання організації розподіленого рендерингу в умовах динамічних графічних сцен, де навантаження змінюється в реальному часі залежно від руху об'єктів, анімацій, змін освітлення або взаємодії користувача. У таких випадках традиційні статичні підходи до розподілу задач не забезпечують бажаної ефективності та стабільності, що актуалізує потребу в адаптивних методах балансування навантаження. Розглянуто сучасні стратегії управління ресурсами, зокрема методи динамічного перерозподілу підзадач між обчислювальними вузлами, прогнозування складності сцени, а також засоби гібридного керування на основі даних попередніх кадрів.

Особливу увагу приділено інтелектуальним підходам, які передбачають використання методів машинного навчання для виявлення зон підвищеної складності та динамічного корегування топології обчислень. Аналізуються технології попереднього кешування, делегування даних, реплікації об'єктів і побудови підсцен, що дозволяє значно зменшити обсяг переданих даних у кластерних середовищах. Описано застосування методів прогресивного завантаження спрощених моделей сцени та передбачення майбутніх запитів на графічні ресурси.

У частині комбінування результатів рендерингу розглянуто техніки об'єднання часткових зображень із дотриманням глибини, прозорості та правильного порядку відображення об'єктів. Визначено переваги багатоступеневого композитингу та алгоритмів обробки прозорих елементів. Пропонується систематизований підхід до створення адаптивних систем розподіленого рендерингу, які ефективно функціонують в умовах високої графічної складності, що властива сучасним інтерактивним і віртуальним середовищам, цифровим двійникам, науковим візуалізаціям та ігровим рушіям. Результати можуть бути використані для проєктування гнучких архітектур, здатних до масштабування, адаптації та мінімізації затримок у візуалізації.

Ключові слова: *паралельний рендеринг, розподілені обчислення, балансування навантаження, гетерогенні архітектури, графічні процесори, кластерна візуалізація.*

Вступ

Комп'ютерна графіка стала невід'ємною складовою багатьох галузей – від кіноіндустрії та комп'ютерних ігор до наукових візуалізацій і проєктування. З ускладненням тривимірних сцен і збільшенням обсягів графічної інформації зростають і вимоги до продуктивності та якості рендерингу. Використання лише одного центрального процесора вже не задовольняє потреби сучасних систем, що стимулює розвиток паралельних і розподілених підходів. Особливо актуальним стає питання ефективного розподілу навантаження між процесорами, графічними прискорювачами та іншими обчислювальними вузлами. Окрім самого поділу обчислень, важливо забезпечити синхронну взаємодію між компонентами системи та оптимізувати обмін даними. У цьому контексті важливими є як архітектурні рішення, так і алгоритмічні методи управління ресурсами та об'єднання результатів обробки.

Аналіз літератури

Упродовж останніх років у сфері комп'ютерної графіки значно зросла зацікавленість до питань розподіленого рендерингу, що є важливим інструментом для досягнення

масштабованості, високої ефективності та інтерактивності при візуалізації складних графічних сцен. Ідеї, започатковані ще в 2001 році в системі WireGL [1], яка вперше реалізувала концепцію перехоплення команд OpenGL для віддаленого рендерингу, отримали новий імпульс у світлі сучасних кластерних та хмарних обчислювальних платформ [2], [3].

Сучасна наукова література активно досліджує класифікацію стратегій рендерингу – таких як сортування за зображенням, сортування після рендерингу та комбіновані методи, які забезпечують гнучке масштабування у багатовузлових системах [4]. Останні дослідження демонструють, що вибір оптимального методу залежить від особливостей сцени, ступеня динаміки та рівня інтерактивності [5]. Зокрема, гібридні рішення з адаптивною зміною обчислюваних ділянок показали високу ефективність у нерівномірно навантажених середовищах [6].

У літературі детально вивчається розподілення обчислень між центральними та графічними процесорами з урахуванням механізмів синхронізації, буферизації й асинхронної передачі даних [7], [8]. Зростає інтерес до гетерогенних обчислювальних платформ, у яких частина графічних обчислень реалізується на спеціалізованих прискорювачах [9]. Це відкриває нові можливості для зменшення енергоспоживання та підвищення продуктивності, особливо у портативних пристроях.

У напрямі динамічного балансування навантаження інтенсивно досліджуються алгоритми гнучкого перегрупування об'єктів сцени та пріоритетне керування потоком даних через буфери [10 – 12]. Це важливо під час обробки рухомих і змінних сцен, де статичне розподілення ресурсів є неефективним.

Задача обмеження швидкості мережевих обмінів залишається є надзвичайно важливою, оскільки передача геометричних даних, текстур і часткових результатів рендерингу часто виступає стримуючим фактором масштабування [13]. Як свідчать роботи [14], [15], ефективними засобами зменшення мережевих навантажень є попередня обробка команд, стиснення даних, багатопотокове передавання, а також використання сучасних протоколів.

Перспективним напрямом [16] вважається використання алгоритмів штучного інтелекту для прогнозування навантаження, інтелектуального розподілення задач між вузлами та автоматичного стиснення текстур без втрати якості Наприклад, система Neural-Sort Rendering [17] поєднує класичні стратегії сортування з AI-аналізом для зменшення часу обробки складних сцен.

Аналіз літератури свідчить про стрімкий розвиток теорії та практики розподіленого рендерингу в останні роки. Найбільш актуальними залишаються питання обробки прозорих об’єктів, інтерактивної графіки в доповненій та віртуальній реальності, а також масштабування рендеринг-систем.

Метою статті є дослідження та обґрунтування адаптивних методів балансування навантаження **у** динамічних графічних сцена**х**, де традиційні статичні підходи до розподілу задач виявляються неефективними.

Методи розподілення обчислень і балансування навантаження

У комп’ютерній графіці можна виділяють різні рівні паралелізму. Поширеним є паралелізм по кадрах. У цьому випадку кілька кадрів сцени можуть бути оброблені одночасно різними обчислювальними вузлами. Це особливо корисно у сценаріях, де кадри не залежать один від одного, наприклад, при попередньому рендерингу анімації або відео. У деяких випадках, частини одного кадру можуть бути розподілені по вузлах, де кожен вузол відповідає за певний часовий інтервал або шар анімації. Головна перевага полягає в простоті реалізації, але обмеженням є відсутність взаємодії між кадрами, що обмежує гнучкість підходу при динамічних сценах у реальному часі [2], [5], [13].

При паралелізмі по об’єктах [3], [6], [11] сцена розбивається на підсцени або множину об’єктів, які можуть оброблятися незалежно. Кожен вузол формує свою підмножину

об’єктів, що дозволяє ефективно використовувати ресурси за наявності складних ієрархічних структур. Наприклад, віртуальне місто може бути поділено по кварталах, і кожен вузол рендерить окрему ділянку. Важливо враховувати перекриття об’єктів та необхідність синхронізації за остаточного формування зображення. Такий підхід добре масштабується для великомасштабних сцен і дозволяє реалізувати спеціалізацію обчислень по типу об’єктів.

При паралелізмі по пікселях [7 – 9] зображення ділиться на піксельні блоки, і кожен вузол обчислює значення кольору для відповідної частини зображення. Цей рівень паралелізму широко використовується в графічних процесорах (GPU), де обчислення можуть виконуватися незалежно. Такий тип паралелізму особливо ефективний для рендерингу у реальному часі, трасування променів і постобробки кадрів. Основна складність – забезпечення балансу навантаження між обчислювальними потоками, оскільки складність піксельної обробки може змінюватись у різних зонах зображення.

В комбінованих методах [1], [10], [12], [14], [15]. поєднуються кілька рівнів паралелізму для досягнення максимальної продуктивності та ефективного використання обчислювальних ресурсів. Наприклад, можна об’єднати паралелізм по об’єктах зі зображенням у sort-first режимі: спочатку розподіляється сцена, а потім кожен вузол ділить свій регіон зображення по пікселях. Такі гібридні стратегії дозволяють адаптувати систему до конкретних характеристик сцени, апаратного забезпечення та вимог до затримки. Основна перевага – гнучкість і масштабованість, проте реалізація потребує складного механізму синхронізації, моніторингу продуктивності та адаптивного балансування.

У комп’ютерній графіці виділяють рівні паралелізму, залежно від типу сцени, вимог до продуктивності й обчислювальної архітектури [4], [6], [10].

Паралелізм на рівні кадрів [5], [14] передбачає одночасну обробку кількох кадрів різними вузлами. Такий метод є ефективним для задач попереднього рендерингу відео чи анімації, де відсутня залежність між послідовними кадрами. У деяких реалізаціях також можливий розподіл одного кадру по часових шарах або ефектах. Основна перевага цього методу – простота впровадження, проте він не підходить для реалістичних інтерактивних сцен, де кадри тісно пов’язані між собою

Паралелізм на рівні об’єктів [6], [10], [11] або сцен передбачає поділ візуального простору на логічні підсцени або набори об’єктів, кожен з яких обробляється окремим вузлом. Наприклад, за візуалізації великої урбаністичної сцени різні вузли можуть відповідати за окремі квартали міста. Це забезпечує масштабування та дає можливість використання вузлів із різною спеціалізацією. Однак виникає потреба в синхронізації під час формування фінального зображення, особливо в разі перекриття геометричних об’єктів або тіней.

Паралелізм на рівні пікселів [4], [7], [12] реалізується шляхом розбиття кадру на піксельні блоки, які паралельно рендеряться різними потоками або обчислювальними модулями. Це підхід, властивий сучасним графічним процесорам, які дозволяють ефективно обробляти зображення в реальному часі, включаючи трасування променів і ефекти постобробки. Його головна складність полягає у варіативності навантаження: одні зони кадру можуть бути простими, інші – насиченими складною геометрією чи освітленням, що вимагає динамічного балансування

Комбіновані або гібридні методи [6], [8], [16].інтегрують різні рівні паралелізму з метою досягнення вищої продуктивності. Наприклад, попередньо розподілена сцена може бути далі оброблена у форматі зображення шляхом поділу на піксельні сектори, які формуються паралельно в межах вузла. Це дозволяє адаптувати систему до конкретних апаратних умов і сцени, проте вимагає ретельно організованої синхронізації, оцінювання навантаження й адаптивної корекції під час виконання

Можна запропонувати використання нейромереж для прогнозування навантаження в динамічних сценах, що дозволяє завчасно розподіляти ресурси обчислення між вузлами. Актуальним є впровадження динамічних багаторівневих сіток, де межі між блоками

зображення або підсценами можуть змінюватись залежно від складності вмісту. Можливо реалізувати розподілений рендеринг безпосередньо між клієнтами, що створює новий клас легких графічних обчислень. Пропонується використати контекстно-залежну точність візуалізації, де рівень деталізації змінюється відповідно до важливості об'єкта для користувача. Це особливо ефективно у віртуальній реальності та ігрових застосунках.

У паралельній побудові зображення традиційно використовуються стратегії розподілу навантаження [3], [4], [5]: «сортування на початку», «сортування після» та комбінований підхід**.** Кожна з них характеризується логікою обробки сцени й передавання даних, що істотно впливає на загальну ефективність в умовах масштабованої обчислювальної архітектури.

У варіанті «сортування на початку» [1], [2], [12]. екран ділиться на ділянки, які розподіляються між вузлами, і кожен обробляє лише ті об’єкти, що потрапляють у його зону відповідальності. Хоча це зменшує обсяг результатів, які потрібно передавати після побудови кадру, але потреба надсилати повну геометричну інформацію всім вузлам і складнощі з об’єктами, що перетинають межі регіонів, знижують загальну ефективність.

Метод «сортування після» [6], [7], [9], [13] передбачає розподіл об'єктів між вузлами незалежно від їхнього положення на екрані, а результати поєднуються у фінальне зображення за допомогою процедури злиття. Цей метод краще масштабується, однак потребує потужного мережевого каналу та додаткових обчислювальних ресурсів для коректного об’єднання часткових зображень.

Комбінована стратегія 8], [15] поєднує переваги обох попередніх підходів: наприклад, об’єкти спочатку розподіляються між вузлами, а далі виконується локальне сортування їхнього відображення в межах кожного вузла. Така адаптивність дозволяє системі гнучко реагувати на зміну навантаження, динамічність сцени.

Ці методи спрямовані на оптимальне використання обчислювальних ресурсів**,** зменшення затримок і підвищення продуктивності. У сучасних реалізаціях дедалі частіше застосовуються автоматизовані системи вибору або перемикання стратегії побудови зображення**,** що забезпечує максимальну ефективність у різних умовах візуалізації.

У таблиці 1 наведено порівняльний аналіз стратегій розподілу навантаження**.**

Таблиця 1

Порівняльний аналіз стратегій розподілу

Критерій	**Сортування спочатку (sort-first)**	**Сортування після (sort-last)**	**Комбінована стратегія**
Принцип роботи	Розподіл простору екрана між вузлами до побудови зображення	Розподіл геометрії сцени між вузлами; об’єднання результатів після побудови	Поєднання розподілу сцени та екранного простору
Передача геометрії	Вся геометрія надсилається на всі вузли	Кожен вузол отримує лише частину геометрії	Частковий розподіл геометрії з подальшим уточненням
Передача результатів побудови	Мінімальна (кожен вузол формує частину зображення)	Висока (повне зображення надходить з кожного вузла)	Середній рівень, залежно від налаштування
Об’єднання зображення	Мінімальне, потрібне лише при накладанні об’єктів	Необхідний складний механізм об’єднання	Гнучкий механізм залежно від конфігурації
Складність реалізації	Відносно невелика	Висока через потребу у злитті зображень	Дуже висока через адаптивність і багаторівневу обробку
Балансування навантаження	Посереднє (залежить від змісту регіонів екрана)	Добре масштабується при рівномірному розподілі об’єктів	Адаптивне, з можливістю переналаштування у реальному часі

Критерій	Сортування спочатку (sort-first)	Сортування після (sort-last)	Комбінована стратегія
Придатність до динамічних сцен	Обмежена (потребує частих переналаштувань)	Вища (менш залежить від положення об’єктів)	Висока, за рахунок можливості адаптації
Переваги	Менше трафіку після побудови, проста обробка частин зображення	Менше дублювання геометрії, кращий розподіл навантаження	Гнучкість, адаптивність до зміни сцени й обчислювальних ресурсів
Недоліки	Високе дублювання геометрії, складність обробки при перекриттях	Значне навантаження на мережу при об’єднанні зображень	Висока складність впровадження, потреба в розширеному моніторингу

У системах комп’ютерної графіки дедалі ширше застосовуються гетерогенні архітектури, що поєднують центральні процесори (ЦП), графічні процесори (ГП) та спеціалізовані прискорювачі. Такий підхід дозволяє ефективніше розподіляти обчислювальні задачі залежно від їх характеру: ЦП обробляє логіку, ГП – масово-паралельні обчислення, а додаткові модулі, як от FPGA чи тензорні блоки, – специфічні алгоритми [7], [9], [11].

Особливу роль відіграє правильна організація передачі даних між компонентами [2], [14], де використовуються сучасні високошвидкісні шини та технології спільної пам’яті. Це мінімізує затримки синхронізації та забезпечує узгодженість даних між різними типами пристроїв

Гібридне використання ресурсів дозволяє значно прискорити рендеринг складних сцен, зменшити затримки й оптимізувати енергоспоживання. У динамічних візуалізаційних задачах все частіше застосовуються алгоритми розумної диспетчеризації завдань, які враховують поточний стан кожного обчислювального блоку та обирають оптимальний виконавець для кожного типу задачі [6], [10], [15].

Гетерогенна архітектура (рис. 1) [1], [3], [4]. також забезпечує масштабованість – нові модулі можуть підключатися без повної перебудови системи, що спрощує розширення візуалізаційних кластерів [8], [12]. Це відкриває перспективи для створення високопродуктивних систем з балансованим навантаженням і підтримкою візуалізації в реальному часі.

Рис. 1. Структура гетерогенної архітектури рендерингу

Рис. 1 ілюструє структуру гетерогенної архітектури рендерингу, яка включає кілька спеціалізованих обчислювальних блоків для ефективного виконання графічних і нейромережевих завдань. Керуючий блок (CPU), який відповідає за планування, синхронізацію та передачу даних між компонентами. Графічний процесор (GPU) виконує рендеринг, обчислення шейдерів та постобробку, працюючи тісно з AI-прискорювачами, які спеціалізуються на обробці нейронних мереж і стисненні текстур. FPGA-модулі реалізують гнучке фільтрування та обробку геометрії, з можливістю адаптації до специфічних задач. Всі

ці блоки взаємодіють з пам’яттю – як кешованою, так і спільною, забезпечуючи швидкий доступ до даних.

У рамках розвитку гетерогенних архітектур, що поєднують центральні процесори, графічні прискорювачі та інші обчислювальні модулі (FPGA, AI-чипи), можна запропонувати впровадження інтелектуальної системи управління обчислювальними ресурсами на основі динамічного аналізу сцени. Така система може здійснювати адаптивне профілювання вхідних даних і автоматично визначати оптимальну конфігурацію для обробки: наприклад, за умов складної сцени з багатьма прозорими об'єктами – задіювати комбінацію GPU + FPGA, а для простих геометричних структур – обмежитися CPU.

Доцільним є впровадження механізмів спільного використання оперативної пам’яті між вузлами через прямий віддалений доступ, що дозволить вилучити дублювання великих обсягів геометричних і текстурних даних. Іншим кроком удосконалення може стати створення апаратних мікроядер, спеціалізованих для виконання повторюваних операцій, таких як фільтрація, Z-порівняння або обробка глибини. Це дає змогу зменшити навантаження на основні графічні процесори й підвищити загальну швидкодію системи.

У сценах з змінами [4], [7], [10], такими як рух об’єктів, анімація, динамічне освітлення чи взаємодія користувача – навантаження на систему змінюється в реальному часі, і статичні алгоритми розподілу ресурсів виявляються неефективними. Для адаптації до змін середовища застосовуються динамічні підходи до балансування, що дозволяють гнучко перерозподіляти обчислення.

Одним із таких методів є механізм «перехоплення завдань» [11], [13], коли менш завантажені вузли отримують завдання від перевантажених. Альтернативою є адаптивне балансування навантаження, яке базується на безперервному відстеженні стану ресурсів і прогнозуванні навантаження на основі аналізу попередніх кадрів.

Також застосовують гібридні підходи[1], [9], [12]., що оцінюють складність кадру (враховуючи кількість об’єктів, глибину сцени, освітлення) для динамічного перерозподілу підзадач. Машинне навчання використовується для розпізнавання зон з високою обчислювальною складністю і адаптивного перепланування архітектури обчислень.

У складних додатках, таких як віртуальна реальність чи цифрові двійники, можуть комбінуватись об'єктно-орієнтовані і зображувальні методи розподілу [2], [8], [15]. що підвищує ефективність за змінах навантаження.

У системах розподіленого рендерингу необхідно забезпечити ефективну організацію обміну геометрією, текстурами та супутньою інформацією (нормалі, матеріали, індекси). Щоб усунути надлишкове дублювання, застосовуються стратегії кешування, реплікації лише потрібних частин сцени та механізм делегування володіння даними конкретним вузлам. Один із методів – поділ сцени на підсцени, кожна з яких охоплює окрему ділянку простору і передається вузлам, що обробляють лише релевантні об’єкти. У випадках обмеженої пропускної здатності мережі застосовують стиснення або агрегацію ресурсів у батчі. Прогресивне завантаження дає змогу почати рендеринг з використанням спрощених рівнів деталізації (LOD), а повноцінні ресурси – дозавантажувати асинхронно. Прогнозування майбутніх запитів до текстур і геометрії (на основі попередніх кадрів) дозволяє більш раціонально використовувати кеш на вузлах.

У методах розподілу типу гібридних схем, коли кадр ділиться на регіони екранного простору, часто виникають ситуації перекриття об’єктів у кількох регіонах. Це створює потребу у композиціюванні часткових результатів у фінальне зображення. Ключовими викликами тут є коректне об’єднання буферів глибини (z-buffer), прозорості (alpha blending) та порядку відображення об'єктів. Для цього використовуються алгоритми сортування за глибиною, alpha compositing з попередньо перемноженою альфою, а також багаторівневий композитинг через ієрархію вузлів, що дозволяє знизити трафік і покращити масштабованість.

У методах розподілу типу гібридних схем**,** коли кадр ділиться на регіони екранного простору, часто виникають ситуації перекриття об’єктів у кількох регіонах. Це створює потребу у композиціюванні [3], [5], [10] часткових результатів у фінальне зображення

Важливим є коректне об’єднання буферів глибини**,** прозорості та порядку відображення об'єктів. Для цього використовуються алгоритми сортування за глибиною**,** а також багаторівневий композитинг через ієрархію вузлів, що дозволяє знизити мережевий трафік і підвищити масштабованість у кластерних системах рендерингу [1], [2], [6], [9].

У візуалізації прозорих об'єктів застосовують методи пошарового відслідковування або порядконезалежну прозорість, що зменшує артефакти при накладенні складних прозорих елементів у тривимірному просторі [4], [7], [8], [11]. Такі методи активно використовують у сучасних фреймворках розподіленого рендерингу, де важливо забезпечити узгодженість зображення за глибиною та прозорістю між різними вузлами [10], [12], [13].

У сучасних високопродуктивних системах рендерингу, зокрема за візуалізації динамічних сцен у середовищах з високим ступенем зміни, постає важлива задача забезпечення ефективного балансування навантаження між обчислювальними вузлами**.** Традиційні статичні методи розподілу задач виявляються недостатньо гнучкими в умовах, коли зміст сцени постійно змінюється: переміщення об'єктів, зміни освітлення, взаємодія користувача чи запуск анімаційних подій призводять до неоднорідного навантаження [1], [2].

Для адаптації до таких умов застосовуються динамічні методи балансування навантаження, серед яких важливими є перехоплення завдань та адаптивне балансування (рис. 1). У першому випадку менш завантажені вузли можуть автономно перехоплювати частини обчислювальних задач у перевантажених вузлів, що дозволяє уникнути простоїв та підвищити загальну продуктивність [3], [4].

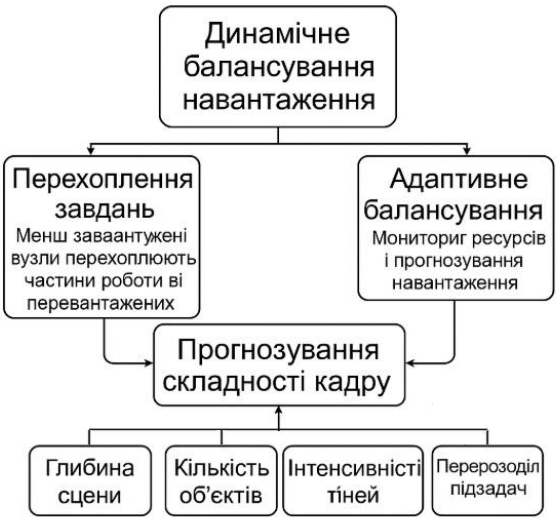

Рис. 2. Принцип динамічного балансування навантаження

Схема демонструє принципи динамічного балансування навантаження у системах рендерингу. Основу становлять два підходи: перехоплення завдань менш завантаженими вузлами та адаптивне балансування через моніторинг ресурсів. Обидва методи спираються на прогнозування складності кадру за параметрами глибини сцени, кількості об’єктів, тіней тощо. Це дозволяє ефективно перерозподіляти підзадачі у змінних умовах

У другому випадку, адаптивне балансування передбачає безперервний моніторинг стану обчислювальних ресурсів із залученням механізмів прогнозування - наприклад, на основі статистики попередніх кадрів – для випереджувального розподілу задач [5]. Центральним компонентом обох підходів є прогнозування складності кадру, яке враховує кілька параметрів: глибину сцени, кількість активних об'єктів, інтенсивність тіней, ступінь прозорості тощо. На основі цих показників система оцінює обчислювальну складність

майбутніх кадрів і адаптує розподіл підзадач відповідно [6], [7].

У складніших реалізаціях, наприклад у хмарних рендерингових середовищах або в багатокластерних інфраструктурах, застосовуються гібридні підходи**,** що поєднують об'єктно-орієнтований розподіл навантаження **з** фрагментною (екранно-просторовою) сегментацією кадру**,** що підвищує масштабованість і стійкість до нерівномірних обчислювальних потоків [8], [9].

Крім того, останні дослідження показують ефективність використання методів машинного навчання для аналізу та прогнозування складності сцен. ML-моделі можуть у режимі реального часу розпізнавати "гарячі зони" з підвищеним навантаженням та адаптивно змінювати топологію обчислень, що особливо актуально для мобільних або енергообмежених пристроїв [10 –

12].

У системах з розподіленим рендерингом, навіть за наявності потужного апаратного забезпечення, важливим чинником залишається ефективність мережевої взаємодії між обчислювальними вузлами. Основними проблемами є затримки під час передавання інформації та обмежена пропускна здатність каналів зв'язку, особливо коли йдеться про передавання великих обсягів геометричних моделей, текстурних даних, проміжних зображень або буферів глибини. Навіть незначні затримки можуть спричиняти збої у синхронізації, зменшення швидкості оновлення кадрів або порушення цілісності візуалізації.

Щоб зменшити перевантаження мережі, застосовуються різноманітні засоби оптимізації. Зокрема, широко використовується метод буферизації команд, коли завдання попередньо накопичуються у спеціальних чергах, а потім передаються блоками. Це дозволяє скоротити кількість окремих запитів до мережі. Також ефективним є попереднє сортування і об'єднання команд в логічні групи, що підвищує узгодженість обчислень і знижує накладні витрати на керування передаванням. Не менш важливим є впровадження алгоритмів стиснення даних, зокрема геометрії та текстур, що дає змогу зменшити обсяг інформації без відчутного впливу на якість графіки.

Крім того, усе частіше реалізуються підходи до асинхронної передачі, які дозволяють виконувати обчислення одночасно з відправленням або прийманням даних. Це так зване перекриття обчислень і комунікації забезпечує більш повне використання ресурсів і зменшує час простою. При цьому зменшується кількість примусових синхронізацій між вузлами, що позитивно впливає на плавність рендерингу.

У якості прикладу доцільно згадати систему WireGL, створену у Стенфордському університеті. Вона дозволяє зменшити трафік шляхом попереднього впорядкування викликів графічних функцій і відкидання надлишкових або дубльованих команд. Цей підхід забезпечив зниження затримок під час дистанційного рендерингу на багатьох вузлах і став основою для подальших розробок у сфері інтерактивної візуалізації.

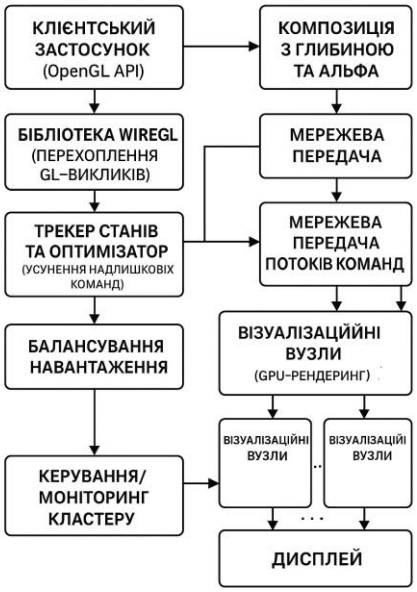

Рис. 3. Принцип роботи системи WireGL

На наведеній структурній схемі (рис. 3) показано принцип роботи системи WireGL. Клієнтський застосунок, який використовує OpenGL API, надсилає графічні виклики, що перехоплюються бібліотекою WireGL. Далі трекер станів та оптимізатор аналізує і видаляє надлишкові або дубльовані команди, зменшуючи обсяг переданої інформації. Після цього відбувається балансування навантаження між обчислювальними вузлами кластера, що забезпечує ефективне використання ресурсів. Команди передаються по мережі до візуалізаційних вузлів, де кожен вузол виконує GPU-рендеринг окремої частини зображення. Отримані результати проходять стадію композиції з урахуванням глибини та альфа-каналу, після чого формуються у фінальне зображення, яке відображається на дисплеї. Роботою всієї системи керує модуль моніторингу і керування кластером, що відповідає за синхронізацію процесів і стабільність обчислень. Такий підхід дозволяє значно зменшити мережевий трафік і затримки, підвищуючи швидкість і якість віддаленої візуалізації.

Додатково слід зазначити значущість сучасних високошвидкісних інтерфейсів, які забезпечують мінімальну затримку за великої пропускній здатності. До них належать такі рішення, як InfiniBand – спеціалізована мережа для надшвидкого передавання даних у суперкомп’ютерах, NVLink – для зв’язку між графічними процесорами та центральним процесором, а також RDMA-технологія, що дозволяє здійснювати прямий доступ до пам’яті одного вузла з іншого, оминаючи центральний процесор. Такі інструменти суттєво знижують навантаження на систему та прискорюють обробку сцен із великою кількістю елементів.

У контексті динамічних сцен та розподіленого рендерингу можна запропонувати нові методи, які поєднують переваги наявних стратегій із можливостями адаптивного машинного навчання та гетерогенного обчислення. Однією з інновацій може стати інтелектуальна система прогнозування складності кадру на основі попередніх сцен за допомогою рекурентних нейронних мереж , що дозволить передбачити не лише кількість об'єктів, а й зміну тіней, прозорості та анімаційних ефектів у часі.

Другою пропозицією є розробка гібридного розподільчого модуля, який може адаптивно перемикатись між стратегіями sort-first, sort-last та image-based compositing на основі реального навантаження та доступності ресурсів у мережі, наприклад, через метрики затримки, пропускної здатності та локальної зайнятості GPU/CPU.

Можливо запропонувати глибоку кешовану структуру сцен, яка зберігатиме багаторівневу інформацію про попередні конфігурації об'єктів, текстур і освітлення в умовах взаємодії користувача, що дозволить зменшити трафік у розподілених середовищах.

Висновок

У статті проаналізовано сучасні методи балансування навантаження при візуалізації динамічних сцен у розподілених середовищах. Розглянуто особливості динамічних сцен, які супроводжуються змінами положення об’єктів, освітлення, прозорості та взаємодією з користувачем. Показано, що традиційні статичні стратегії розподілу обчислень виявляються неефективними в умовах високої динаміки сцен. Обґрунтовано доцільність використання адаптивного балансування з моніторингом ресурсів і прогнозуванням складності кадру. Окрема увага приділена ефективним стратегіям розподілу даних у мережі, зокрема кешуванню, делегуванню прав власності на об’єкти сцени та використанню технологій потокового завантаження. Аналізовано особливості композитингу за перекривання об’єктів на межах регіонів при сортуванні зображення. Показано важливість алгоритмів сортування за глибиною, глибинних буферів і прозорості з незалежним порядком.

У роботі наведено приклади інноваційних рішень, що базуються на прогнозуванні обчислювальної складності сцени із застосуванням машинного навчання. Також запропоновано адаптивну зміну стратегій рендерингу в режимі реального часу залежно від навантаження. Розроблено концепцію глибокого кешу сцен для зменшення затримок у мережі. Запропоновано створення модуля балансування з урахуванням споживання ресурсів. Окреслено перспективи інтеграції запропонованих підходів у сучасні ігрові рушії, VR/AR-системи та цифрові двійники.

Потреба в реальному часі обумовлює подальші дослідження в напрямку гібридних стратегій, машинного навчання та оптимізації обміну даними. Отримані результати можуть бути використані як основа для створення нових алгоритмів балансування в умовах гетерогенних обчислювальних архітектур.

СПИСОК ЛІТЕРАТУРИ

1. Morikawa T., Sasaki S. Dynamic Task Redistribution in Interactive Rendering. *ACM Transactions on Graphics*. 2023. Vol. 42, № 4. P. 1–15.

2. Survey on Distributed Rendering Systems. *Computers & Graphics* / P. Sander et al. 2022. Vol. 106. P. 1–24. DOI: 10.1016/j.cag.2022.03.010.

3. Visual Cluster Rendering with Chromium / M. Riede et al. *Journal of Parallel and Distributed Computing*. 2022. Vol. 160. P. 12–27. DOI: 10.1016/j.jpdc.2022.01.004.

4. DHR+S: Distributed Hybrid Rendering with Realistic Real-time Shadows for Interactive Thin Client Metaverse and Game Applications / Y. W. Tan et al. *Proceedings of IEEE Conference on Computer Graphics and Applications*. 2024. P. 233–241.

5. Schroeder W., Martin K. Visualization Performance on High-latency Networks. *Kitware Reports*. Clifton Park, NY : Kitware Inc., 2021. 34 p.

6. Mendez G., Klein J., Zimmermann K. Communication-aware Scheduling for Distributed Image Synthesis. *Eurographics Conference Proceedings*. 2021. P. 145–158. DOI: 10.2312/egp.20211045.

7. Efficient Command Stream Compression in Remote Rendering / X. Zhang et al. *IEEE Access*. 2022. Vol. 10. P. 11834–11845. DOI: 10.1109/ACCESS.2022.3156724.

8. Li Q., Guo Z. CPU–GPU Collaborative Rendering: A Review. *Visual Informatics*. 2020. Vol. 4, № 3. P. 109–122. DOI: 10.1016/j.visinf.2020.08.001.

9. Kawashima H., Fukuda K. Hybrid Rendering Strategies in Parallel Graphics Pipelines. *IEEE Transactions on Visualization and Computer Graphics*. 2023. Vol. 29, № 7. P. 2921–2936. DOI: 10.1109/TVCG.2023.3234567.

10. Zhang H., Huang Y., Xu T. Real-time Scheduling of Rendering Tasks in Hybrid Architectures. *Concurrency and Computation: Practice and Experience*. 2021. Vol. 33, № 24. DOI: 10.1002/cpe.6201.

11. Xu L., Fan Z. FPGA-assisted Path Tracing System. *Sensors*. 2022. Vol. 22, №18. P. 6733. DOI: 10.3390/s22186733.

12. Yu M., Liu Q. Network-aware Rendering Pipelines for VR Applications. *Multimedia Tools and Applications*. 2023. Vol. 82. P. 19123–19141. DOI: 10.1007/s11042-023-16180-y.

13. RadSplat: Radiance Field-Informed Gaussian Splatting for Robust Real-Time Rendering with 900+ FPS / M. Niemeyer et al. *arXiv preprint arXiv*: 2403. 2024. P. 13806.

14. Sort-Last Parallel Rendering for View-Dependent Visualization / G. Humphreys et al. *IEEE Visualization Conference Proceedings*. Boston : IEEE Computer Society, 2002. P. 59–68. DOI: 10.1109/VISUAL.2002.1183775.

15. Stoll G. WireGL and Beyond: Scalable Interactive Rendering on Clusters. Stanford Graphics Group Technical Report TR-2003-05. Stanford : Stanford University, 2003. 18 p.

Стаття надійшла до редакції 18.09.2025.
Стаття пройшла рецензування 27.09.2025.

Романюк Олександр Никифорович – д-р техн. наук, професор, завідувач кафедри програмного забезпечення, e-mail: rom8591@gmail.com.

Майданюк Володимир Павлович – канд. техн. наук, доцент кафедри програмного забезпечення.

Романюк Оксана Володимирівна – канд. техн. наук, доцент кафедри програмного забезпечення.

Бобко Олексій Леонідович – аспірант кафедри програмного забезпечення.

Новосельцев Олександр Олександрович – аспірант кафедри програмного забезпечення.
Вінницький національний технічний університет.