ISSN 2707-1898 (print)

Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2023.01.001
Article received 14.04.2023 p.
Article accepted 02.05.2023 p.
UDK 004.925

Correspondence author
Y. V. Levus
Yevheniia.V. Levus@lpnu.ua

Є. В. Левус[1], П. Я. Пустельник[1], Р. О. Моравський[1], М. Ю. Морозов[2]
[1]Національний університет "Львівська політехніка", м. Львів, Україна
[2]Мюнхенський технічний університет, м. Мюнхен, Німеччина

АРХІТЕКТУРА РОЗПОДІЛЕНОГО ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ГЕНЕРУВАННЯ ЛАНДШАФТІВ ПЛАНЕТОЇДІВ

Зростання популярності генерування ландшафтів планетоїдів у відеоіграх, фільмах, інструментах симуляції посилює вимоги до якості, унікальності, масштабованості результатів візуалізації, які, водночас, зумовлюють підвищення вимог до обчислювальних ресурсів апаратних засобів. Запропоновано рішення щодо архітектури програмного застосунку для генерування ландшафтів планетоїдів, яке базується на поєднанні розподіленої обчислювальної системи і застосуванні паралелізму на підставі алгоритму Morsel-Driven Query Execution для подолання обмежень апаратного забезпечення. Розроблено обчислювальну модель генерування ландшафтів планетоїдів, яка містить компоненти: основний сервер, який підтримує gRPC з'єднання; сервери-працівники, які паралельно виконують завдання процедур генерування ландшафтів планетоїдів; геопросторова база даних, яка містить векторні дані згенерованих планетоїдів; бінарне сховище тривимірних моделей, які накладаються на згенерований ландшафт планетоїдів; tileset-сховище для зберігання растрових даних, необхідних для генерування; користувачі. Застосовано програмні агенти для уніфікації набору алгоритмів як єдиної сутності, що дає змогу вирішити проблему розширюваності програмного засобу. Для надсилання та опрацювання запитів у логічні канали, створені для послідовності агентів, використано розподілену систему обміну повідомленнями – брокер. Він базується на підході балансування навантаження для доставки запитів на генерування ландшафту до фонових процесів. Проведено обчислювальні експерименти для аналізу швидкодії затосунку при різних кількостях фонових процесів (1, 2; 4; 8; 16) та розмірах сегменту 512, 2048 пікселів. Отримано найменшу середню тривалість генерування одного сегменту при його розмірі 512 пікселів та кількості сегментів 64. Зменшено витрати процесорного часу від 2 до 5 разів, порівняно із застосунком, який використовує підхід збереження запитів у базі даних, за рахунок комплексного архітектурного рішення. Проведено порівняння роботи розробленого програмного засобу із застосунком MapGen. Ефективність рішення особливо помітна при великих обсягах даних, що визначається кількістю сегментів та їх розміром у пікселах.

Ключові слова: програмні агенти, обмін повідомленнями, технологія gRPC, брокер, швидкодія за стосунку, обчислювальна модель.

Вступ / Introduction

Індустрія створення цифрового контенту з кожним роком потребує все більш реалістичних результатів генерування великомасштабної віртуальної місцевості. Широке використання віртуальних світів можна спостерігати в кінематографії та відеоіграх, зокрема, це стосується планетоїдів для створення, зберігання та відображення великих ландшафтів. Планетоїди, згенеровані за допомогою програмних засобів, використовуються не тільки для розваг, а й для науки та освіти.

Образи цілих планет із реалістичними ландшафтами використовують у наукових дослідженнях, пов'язаних із моделюванням різних природних процесів у тривимірних віртуальних світах [1]. Окрім цього, планетоїди знаходять своє застосування у архітектурному моделюванні, зокрема для врахування кривизни планети. Якість та швидкість процедурного генерування ландшафтів планетоїдів залежать як від доступних обчислювальних ресурсів, так і від рівня оптимізації алгоритмів генерування.

Процес генерування планетоїдів є складним та оперує різними типами даних, які містять усю інформацію про ландшафт. Це, для прикладу, растрові дані, які використовуються для карти висот, векторні дані, які відображають структуру ландшафту, розміщення річок, озер, та, власне, тривимірні об'єкти, які наповнюють віртуальний світ.

Стан галузі процедурного генерування окремих ландшафтів чи цілих планетоїдів визначається обмеженою кількістю програмних засобів, які надають необхідні інструменти, зрозумілі для звичайних користувачів, та створюють достатньо реалістичні поверхні. Загалом, застосунки, які певним чином використовують генерування ландшафтів планетоїдів, можна поділити на дві кате-

горії: застосунки, основною метою яких є відображення планетоїдів для формування віртуальних світів у комп'ютерних іграх чи кіноіндустрії, та застосунки, які надають відповідні інструменти для параметризованого генерування ландшафтів планетоїдів для подальшого використання згенерованих моделей у сторонніх застосунках.

Незважаючи на переваги використання цих застосунків для процедурного генерування планетоїдів [1, 2], істотними недоліками, які звужують діапазон їх використання, є: складність у користуванні, обмеженість набору вхідних параметрів плагінів, неможливість інтегрування в інші системи, форма результату рендерингу тільки у вигляді фото або відео.

Найпоширеніші алгоритми в задачах процедурного генерування ландшафтів базуються на використанні шуму та машинного навчання. Більшість наукових праць, зокрема роботи К. Дж. Касла та А. Т. Крукса, зосереджені на впровадженні цих алгоритмів у програмних системах, проте цим роботам бракує розширюваності чи адаптації до сучасного апаратного забезпечення.

Зростаючі вимоги до унікальності, масштабованості результатів процедурного генерування ландшафтів планетоїдів в умовах ефективного використання обчислювальних ресурсів зумовлюють актуальність вирішення задачі розвитку методів процедурного генерування ландшафтів планетоїдів для різних предметних областей з врахуванням особливостей їх програмної імплементації.

Об'єкт дослідження – процес процедурного генерування ландшафтів планетоїдів.

Предмет дослідження – архітектура та алгоритми програмного забезпечення процедурного генерування ландшафтів планетоїдів.

Мета роботи – покращення продуктивності процесу процедурного генерування ландшафтів для планетоїдів на підставі використання програмних агентів та моделі архітектури, що базується на концепції розподіленої обчислювальної системи.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

- аналіз методів та програмних засобів у галузі процедурного генерування ландшафтів на предмет продуктивності отримання результатів візуалізації;
- створення масштабованої серверної архітектури із використанням двонаправленого потокового gRPC;
- реалізація механізму опрацювання запитів на генерування сегментів планетоїда;
- реалізація алгоритмічного забезпечення генерування природного ландшафту планетоїда;
- аналіз отриманих результатів швидкодії розробленого застосунку.

Аналіз останніх досліджень та публікацій. Незважаючи на те, що процес генерування віртуальних світів загалом, досяг хорошого рівня деталізації та реалістичності, постійно зростаючі вимоги до якості, продуктивності візуалізації диктують потреби розроблення як математичних моделей, алгоритмів, так і методів програмної імплементації в галузі процедурного генерування ландшафтів. Пошук оптимального рішення розподілений між багатьма спеціалізованими дослідницькими областями.

Автори робіт у цій галузі вирішують завдання процедур генерування ландшафтів на підставі використання різних методів. Серед таких методів алгоритми на підставі шуму виділяються як найбільш часто використовувані для процедурного генерування карт висот [2]. Тоді, як інші методи дають змогу більше контролювати характеристики рельєфу, такі як методи поділу [3] або методи синтезу Фур'є [4], алгоритми на підставі шуму мають очевидну перевагу у створенні майже нескінченних безшовних рельєфів. Існують також роботи, присвячені дослідженню використання машинного навчання в генеруванні рельєфу, спрямованого на вирішення проблеми повторюваності [5]. Хоча ці роботи загалом вирішують процедурну проблему генерування рельєфу, їм бракує масштабованості, розширюваності або більш точного контролю над етапами процесу генерування. Окрім цього, їх не можна легко модифікувати для введення в систему, призначену для безпосереднього генерування планетоїдів.

У роботі К. Дж. Касла та А. Т. Крукса [6], серед інших, пропонується використовувати процедурне генерування GIS (Geographic Information System) даних із фіксованим розташуванням об'єктів. Незважаючи на те, що цей підхід дає точні результати, інтегрувати таке рішення у великі системи без залучення програмних залежностей складно. У сфері машинного навчання ситуація розвивається інтенсивно. Тривають дослідження не тільки для створення, але й для імплементації цих методів у наявні системи [5], включаючи перетворення даних і візуалізацію. Щоб уніфікувати процес, програмні агенти були запропоновані як життєздатне рішення, що охоплює дослідницькі області з реальними застосунками, починаючи від оброблення цифрових зображень [7] до геодезичних досліджень [8]. Хоча їх принципи були викладені в попередніх дослідженнях [9], навіть включаючи вказування залежностей між агентами за допомогою послідовностей оброблення та орієнтованих графів, тільки нещодавно з'явилися дослідження, спрямовані на створення мультимедійного контенту, включаючи ігри та фільми [10].

До того ж, методи, описані вище, не враховують сучасне апаратне забезпечення та розмір даних, тільки використовують мережеві потокові дані для пом'якшення обмежень пам'яті [11]. З одного боку, у роботі [12] запропоновано використовувати розподілену систему хмарних обчислень, що підтримує програмні агенти, для боротьби з апаратними обмеженнями. З іншого боку, у [13] пропонується розділити запити на менші паралельні фрагменти обчислень для зменшення нерівномірного доступу до пам'яті та мережеві затримки.

Зважаючи на недоліки розглянутих методів, виникла потреба розроблення моделі обчислювальної системи для процедурного генерування ландшафту, яка дала б можливість задовільнити вимоги до масштабованості, однорідності та узгодженості, необхідні для ефективного контролю процесу генерування ландшафту в масштабі планети.

Результати дослідження та їх обговорення / Research results and their discussion

У запропонованому рішенні архітектура програмного застосунку базується на концепції, яка поєднує розподілену обчислювальну систему [14] і підхід паралелізму на підставі алгоритму Morsel-Driven Query Execution для подолання обмежень апаратного забезпечення.

Серверна архітектура для генерування планетоїдів є комплексним поняттям для системи, яка містить усі не-

обхідні компоненти та надає інтерфейс доступу для застосунків-клієнтів (рис. 1). Обчислювальна модель містить компоненти:

1) основний сервер, який підтримує gRPC з'єднання;

2) обчислювальні сервери, які містять фонові процеси для виконання завдань генерування ландшафтів планетоїдів;

3) геопросторова база даних, яка містить векторні дані згенерованих планетоїдів (річки та водойми; географічні регіони – біоми, лісові масиви та інші сегменти; дороги);

4) бінарне сховище тривимірних моделей, які накладаються на згенерований ландшафт планетоїдів;

5) tileset-сховище для зберігання растрових даних, необхідних для генерування;

6) користувачі, наприклад дизайнери, які використовують програмний засіб генерування ландшафтів планетоїдів для реалізації власних цілей.

Для клієнт-серверного зв'язку застосовується gRPC [15]. Фреймворк gRPC базується на моделі RPC (Remote Procedure Call), яка надає механізми для віддаленого виклику методів сервісів, розміщених фізично на різних апаратних системах. Дана технологія має переваги – використання буферів протоколу (Protobuf), потокова передача даних, використання HTTP/2.

gRPC підтримує двонаправлений зв'язок між клієнтом та сервером (рис. 2), що дає змогу передавати дані у режимі реального часу, як тільки вони стають готовими.

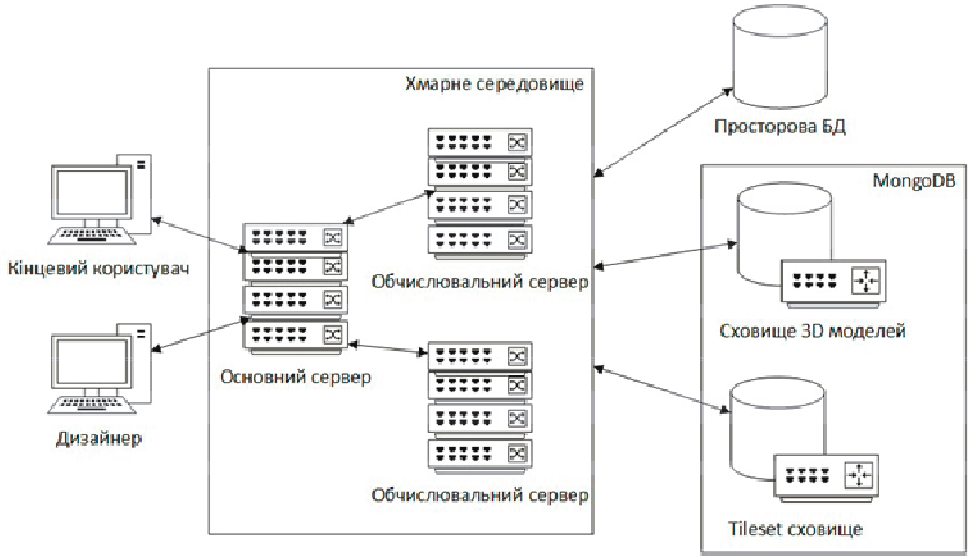

Рис. 1. Архітектура розподіленого застосунку / Architecture of a distributed application

Рис. 2. Приклад двонаправленої потокової передачі / An example of bidirectional streaming

У термінах генерування планетоїдів це є істотною перевагою, адже надає можливість надсилати клієнту згенерований ландшафт частинами, не чекаючи на його повне генерування. При цьому застосунок-клієнт отримує більшу гнучкість у відображенні отриманих даних.

Серверна архітектура складається з кількох фізичних слабко пов'язаних серверів, які взаємодіють із базою даних, включаючи центральний сервер, який прослуховує запити клієнта та готує відповіді [16]. Інші сервери використовують фонові процеси для отримання завдань генерування з бази даних і їх виконання. Фоновий процес – це процес, який постійно виконується на сервері, очікує на появу нових завдань та, коли одне з таких з'являється, приступає до його виконання. Це дає змогу розподілити ресурсоємне виконання генерування планетоїдів на кілька серверів у хмарному середовищі.

Фоновий процес реалізований на підставі алгоритму Morsel-Driven Query Execution [13]. Виконання запитів на підставі Morsel-Driven Query Execution – це концепція, яка використовує всі переваги багатоядерних систем. Morsel-Driven Query Execution вводить такі поняття, як диспетчер (фоновий процес) і конвеєри виконання (завдання процедур генерування планетоїдів). Диспетчер контролює та призначає завдання робочим потокам. Потоки можуть бути переміщені в інші конвеєри після завершення поточного завдання, тобто жоден робочий елемент не буде простоювати, поки є завдання для виконання.

Систему зберігання даних розділено за типом даних, якими оперує сервер при генеруванні планетоїдів. Окрім необхідних реляційних і векторних даних, процес генерування планетоїдів також залучає використання тривимірних моделей, а також растрових об'єктів, зокрема карти висот. Оскільки зберігання таких типів даних через їх розмір та структуру є неефективним у реляційній базі даних, обрано розподілити процес збе-

реження таких типів даних і винести їх в окрему нереляційну документно-орієнтовану базу даних MongoDB.

Процес генерування природних ландшафтів – це зазвичай не один, а декілька алгоритмів. Саме тому для програмних засобів генерування ланшафтів потрібно використовувати комплексне рішення, яке об'єднує конкретні алгоритми та підходи до реалізації різних етапів генерування. Важливим є не тільки реалізація цих алгоритмів, а й спосіб їх поєднання. Адже що більше задач та підходів до їх вирішення, то складнішим та менш розширюваним може стати остаточне рішення. Використання програмних агентів на відміну від інших альтернатив – плагіни, монолітний підхід – дає змогу уніфікувати певний алгоритм чи набір алгоритмів як єдину сутність, яка може бути використана для певного етапу генерування ландшафту та вирішити проблему розширюваності програмного застосунку.

Завдання окремого агента – виконати певний етап генерування чи оброблення сегменту ландшафту планетоїда (рис. 3). Прикладами можуть бути агенти, що виконують такі дії: генерування карти висот, формування клімату та біомів, генерування каньйонів, річок, озер тощо.

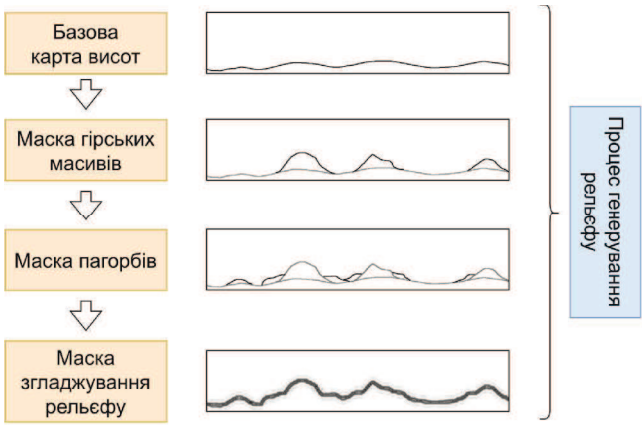

Рис. 3. Схема роботи масок в агенті генерування рельєфу / Scheme of processing of masks in the relief generation agent

Процес виконання запиту на певну ділянку планетоїда можна подати у вигляді залежності:

$$result = \sum_{j=1}^{n} mask(agent_j) \cdot t_i \ ,$$

де *result* – вихідні результати роботи всіх програмних агентів; t_i – *i*-й сегмент ландшафту; *mask()* – функція накладання маски, вхідним параметром якої є конкретний програмний агент; $agent_j$ – *j*-й програмний агент; *n* – загальна кількість програмних агентів.

Для надсилання та опрацювання запитів запропоновано використання розподіленої системи обміну повідомленнями – брокер повідомлень [17]. Розподілені системи обміну повідомленнями дають змогу відправникам асинхронно надсилати повідомлення, незалежно від того, скільки отримувачів існує та де вони знаходяться. У розробленому програмному застосунку повідомлення – це запити на генерування чи опрацювання заданого сегменту ландшафту планетоїда окремим агентом, а сервіси, які працюють із цими повідомленнями, поділяються на дві категорії:

1. Виробник (producer) – основний мікросервіс, з яким взаємодіють застосунки-клієнти (gRPC чи REST запити на генерування сегментів). Публікує повідомлення в "теми" (іменовані логічні канали).
2. Споживач (consumer) – мікросервіс, який використовує фонові процеси, основним завданням яких є опрацювання запитів на генерування чи опрацювання сегментів певними агентами. Вичитує повідомлення із іменованих логічних каналів.

Задля забезпечення правильного порядку опрацювання сегменту ландшафту планетоїда програмними агентами розроблено алгоритм зчитування повідомлень на підставі пріоритетної черги (рис. 4).

Описане вище архітектурне рішення реалізовано в застосунку PlanetoidGen2, який дає змогу генерувати ландшафти планетоїдів (рис. 5).

Для верифікації отриманих результатів проведено низку обчислювальних експериментів із різними вхідними параметрами: кількість фонових процесів (workers), растровий розмір сегменту в пікселях та кількість сегментів для опрацювання (табл. 1). Використано сервер: CPU AMD Ryzen 7 5800H (1 CPU, 16 логічних і 8 фізичних ядер), GPU Nvidia GeForce RTX 3060, RAM 16GB DDR4, Drive SSD NVME2 2TB, ОС Windows 11. Середовище проведення експериментів – Docker Desktop 4.17.0 із вбудованим кластером Kubernetes 1.25.4 з використанням Linux контейнерів.

Для наведених експериментів середня тривалість генерування сегменту становила від 0,33 до 9,32 с. Найменша середня тривалість генерування індивідуального сегменту є для розміру сегменту 512 пікселів та кількості сегментів 64 (експерименти 5, 13, 9, 1) – від 0.33 до 0.51 с. Для такого ж самого розміру генерованого сегменту, але меншої загальної кількості сегментів (16), середня тривалість генерування збільшилася приблизно в два рази. Це пояснюється тим, що при більшій кількості одночасних запитів на генерування мінімізується часові витрати очікування при зчитуванні повідомлень із брокера. Проте у експерименті 17 можна спостерігати протилежну ситуацію – середня тривалість генерування більше ніж в 2 рази від описаних раніше експериментів. Зумовлено це обмеженою кількістю сервісів-споживачів, які одночасно вичитують повідомлення (запити), при використанні брокера. Одночасно до каналу можуть звертатися максимум 16 споживачів без потреби перебудови споживацької групи [18]. Найбільшу середню тривалість для розміру сегменту 512 пікселів зафіксовано для експериментів 14 (1,18 с) та 18 (1,34 с).

Для розміру сегменту 2048 пікселів найменша середня тривалість генерування індивідуального сегменту складає 2,78 с та спостерігається у експерименті 19 із кількістю сегментів – 64, та фонових процесів – 16. Такий результат досягається через те, що більша кількість фонових процесів допомагає краще розподілити навантаження на генерування великого об'єму даних при мінімальних затримках брокера, враховуючи кількість запитів на генерування. Найбільшу середню тривалість для розміру сегменту 2048 пікселів зафіксовано для експериментів 3 (7,42 с) та 4 (9,32 с), де використано по одному фоновому процесу.

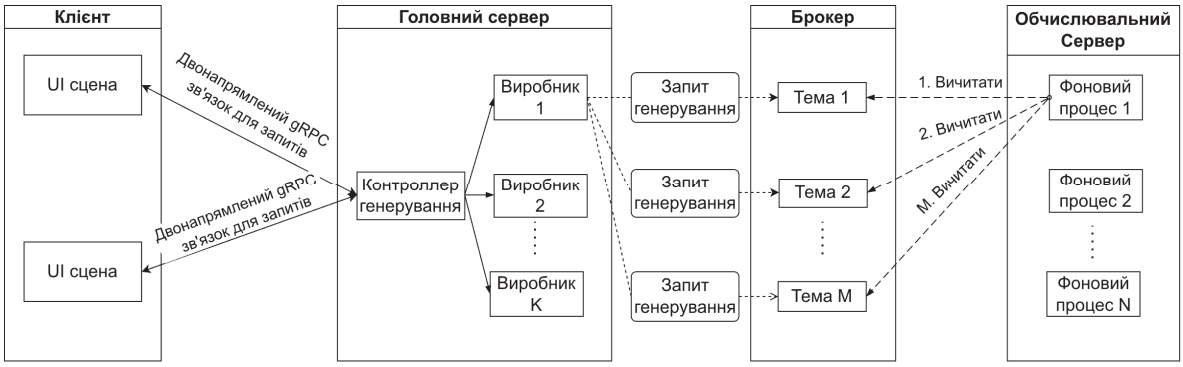

Рис. 4. Схема зчитування повідомлень на підставі пріоритетної черги / Diagram of the message reading based on the priority queue

Рис. 5. Приклад візуалізації сегменту згенерованого ландшафту у PlanetoidGen2 / Example of a segment visualization of the generated landscape in PlanetoidGen2

Табл. 1. Швидкодія генерування сегментів у PlanetoidGen2 / Segment generation performance inPlanetoidGen2

№ обчислювального експерименту	Кількість процесів	Растровий розмір сегменту, пікселі	Кількість сегментів	Загальна тривалість генерування, с	Середня тривалість генерування індивідуального сегменту, с
1	1	512	64	32,34	0,51
2	1	512	16	10,20	0,64
3	1	2048	64	474,81	7,42
4	1	2048	16	149,06	9,32
5	2	512	64	21,42	0,33
6	2	512	16	10,20	0,64
7	2	2048	16	104,52	6,53
8	2	2048	64	460,78	7,20
9	4	512	64	27,96	0,44
10	4	512	16	12,86	0,80
11	4	2048	64	269,74	4,21
12	4	2048	16	77,52	4,84
13	8	512	64	26,57	0,42
14	8	512	16	18,89	1,18
15	8	2048	16	78,51	4,91
16	8	2048	64	466,75	7,29
17	16	512	64	72,26	1,13
18	16	512	16	21,44	1,34
19	16	2048	64	178,23	2,78
20	16	2048	16	76,20	4,76

При використанні брокера повідомлень залежність середньої тривалості генерування сегментів ландшафту від кількості фонових процесів не є детермінованою, на відміну від застосунку PlanetoidGen1 [19], у якій використано інший підхід для опрацювання запитів на генерування – збереження запитів у базу даних. Недетермінованість з'являється, якщо при зчитуванні повідомлень із логічних каналів кількість одночасних сервісів-споживачів на певний момент часу наближається до кількості партицій [1] і, як наслідок, виникають затримки, пов'язані із перебудовою споживацької групи всередині брокера повідомлень.

При значній кількості обчислень є тенденція зменшення тривалості генерування сегменту зі збільшенням кількості фонових процесів (рис. 6). Часові затримки на зчитування із логічних каналів нівелюються.

Виконано порівняння швидкодії розробленого застосунку із схожим застосунком MapGen [20, 19], який також використовує програмні агенти, та попередньою версією застосунку PlanetoidGen1, яка не використовує брокер (табл. 2).

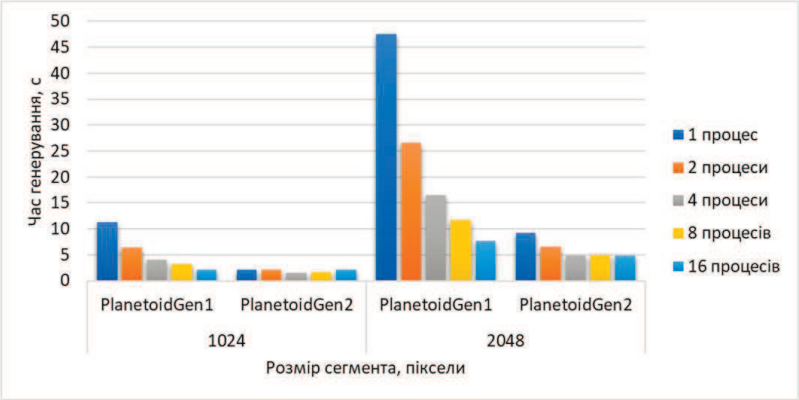

Рис. 6. Залежність тривалості генерування від кількості фонових процесів для систем PlanetoidGen1 та PlanetoidGen2 / Time dependence on the number of background workers for PlanetoidGen1 and PlanetoidGen2 systems

Табл. 2. Порівняння швидкодії застосунків MapGen та PlanetoidGen / Performance comparison with MapGen

Застосунок	Растровий розмір сегменту, пікселі			
	256	512	1024	2048
	Тривалість, с	Тривалість, с	Тривалість, с	Тривалість, с
MapGen	2,08	8,42	49,21	323,01
PlanetoidGen1	1,16	2,98	11,26	47,32
PlanetoidGen2	0,50	0,64	2,14	9,32

Загалом, реалізована модель обчислювальної системи дає можливість зменшити витрати процесорного часу для процедурного генерування ландшафтів планетоїдів при забезпеченні необхідної якості отриманих результатів. Для наведених експериментальних даних при різних значеннях кількості фонових працівників (1, 2, 4, 8, 16), кількості сегментів (16, 64) та розміру сегментів (512, 2048) відбулося пришвидшення генерування одного сегменту ландшафту від 2 до 5 разів, порівняно з попередньою версією системи PlanetoidGen1, яка використовувала збереження запитів у базу даних, а не розподілену систему обміну повідомленнями. Зменшення тривалості генерування сегменту зі збільшенням кількості фонових процесів найбільш відчутно при значній кількості обчислень, що визначаються, зокрема, розміром сегмента у 2048 пікселів.

Використання розподіленої системи обміну повідомленнями (брокера) дало можливість мінімізувати часові витрати на запити до бази та операції із обчисленням пріоритету запиту. Запропоноване рішення дало можливість посилити переваги над застосунком MapGen.

Обговорення результатів дослідження. У сфері процедурного генерування ландшафтів представлено роботи, які мають на меті покращити кількісні та якісні показники моделювання задля підвищення реалістичності результату відображення та продуктивності процесу відображення.

У роботі [21] авторами описано новий підхід ерозії для створення різноманітного та реалістичного рельєфу. Цей метод використовує визначення двох типів обмежень, які забезпечують виразнішу специфікацію бажаних особливостей рельєфу. Встановлено, що запропонований метод дає змогу генерувати більш різноманітний та реалістичний рельєф завдяки інтеграції додаткових параметрів та процесів моделювання, який при цьому працює швидше, пропонує більшу гнучкість та простоту використання для дизайнерів ландшафту, завдяки природі та інтуїтивності нових параметрів.

У роботі [22] автори пропонують метод генерування на підставі шуму із введенням додаткового контролю над процесом, який відсутній у базових алгоритмах. Описано спосіб процедурного генерування із використанням висотних обмежень. Для цього у запропонованому методі встановлюються фіксовані значення у функції карти висот та створюється система рівнянь для отримання всіх точок, які залежать від заданих обмежень. Це дало можливість авторам отримати все ще випадковий рельєф, але з врахуванням бажаних обмежень та без втрати роздільної здатності.

У роботі [23] автори розглянули використання хмарної платформи для географічної інформаційної системи замість локальних рішень. Для зберігання, аналізу та обміну геопросторовими даними використано хмарну базу даних. Векторні дані про сегменти ландшафту, які містять інформацію про право власності та атрибути

вартості землі, надаються безпосередньо з бази даних у вигляді векторних даних. Результати досліджень показали, що використана база даних швидше реагує на запити і масштабується під час високих навантажень для зменшення часових затримок обчислень.

У роботі [24] автори презентують розроблену систему для візуалізації тривимірних ландшафтів на підставі розподіленої клієнт-серверної архітектури. За допомогою хмарного підходу виявлено та вирішено поширені обмеження обробки великої кількості даних. Також автори дослідили вимоги візуалізації потенційно великих ландшафтів на декількох пристроях та зібрали відгуки для покращення функцій архітектури.

Аналіз практичних результатів наведеного дослідження виконано на підставі порівняння із застосунком MapGen, який також базується на генеруванні рельєфу ландшафту з використанням підходу програмних агентів. Отримані результати дають змогу сформулювати наукову новизну та практичну значущість проведеного дослідження.

Отже, за результатами роботи можна сформулювати таку наукову новизну і практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – удосконалено модель обчислювальної системи для процедурного генерування ландшафтів планетоїдів на підставі архітектурного рішення розподіленого генерування із використанням програмних агентів, вивантажених до хмарних обчислень.

Практична значущість результатів дослідження – створено програмний застосунок на підставі розробленої моделі обчислювальної системи для процедурного генерування ландшафтів планетоїдів. Застосунок може використовуватися в індустрії розваг, науці, освіті і має здатність стати еволюційною розробкою.

Висновки / Conclusions

Вирішена задача підвищення продуктивності процесу процедурного генерування ландшафтів для планетоїдів на підставі побудованої обчислювальної моделі, яка передбачає:

- архітектуру розподіленого генерування із використанням програмних агентів, вивантажених до хмарних обчислень;
- асинхронну обробку запитів генерування на підставі Morsel-Driven Query Execution і потокової передачі даних;
- розширюваний уніфікований інтерфейс програмного агента з підтримкою завантаження та збереження даних до реляційних таблиць довільної структури.

Рішення програмно імплементовано у вигляді застосунку PlanetoidGen2. Обчислювальні експерименти при різних значеннях кількості фонових процесів, кількості та розміру сегментів засвідчили пришвидшення генерування одного сегменту ландшафту від 2 до 5 разів, порівняно з попередньою версією застосунку PlanetoidGen1, яка використовувала збереження запитів у базу даних, а не розподілену систему обміну повідомленнями.

Отримані результати засвідчують перспективність досліджень та вказують на потребу подальшого опрацювання. Майбутні дослідження спрямовані на покращення якості результатів генерування, що містить розробку нових програмних агентів, які урізноманітнять згенерований ландшафт (озера, річки, ерозія, скелі, і т. д.), та вдосконалення процесів генерування ландшафту на базі розподіленої системи обчислень для зменшення часових затримок генерування. Зокрема, варто розглянути питання масштабованості застосунку при збільшенні обсягу генерованих даних, можливим рішенням якого є використання розподіленого хмарного сховища даних.

References

[1] Cassol, V. J., Marson, F. P., Vendramini, M., Paravisi, M., Bicho, A. L., Jung, C. R., & Musse, S. R. (2011). Simulation of autonomous agents using terrain reasoning. In Proc. the Twelfth IASTED International Conference on Computer Graphics and Imaging (CGIM 2011), Innsbruck, Austria. IASTED/ACTA Press. http://dx.doi.org/10.2316/P.2011.722-017

[2] Hyttinen, T., Mäkinen, E., & Poranen, T. (2017). Terrain synthesis using noise by examples. In Proceedings of the 21st International Academic Mindtrek Conference. (pp. 17–25). https://doi.org/10.1145/3131085.3131099

[3] Kamal, K. R., & Uddin, Y. S. (2007). Parametrically controlled terrain generation. In Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia. (pp. 17–23). https://doi.org/10.1145/1321261.1321264

[4] Henham, W., Holloway, D., & Panton, L. (2016). Broadband acoustic scattering with modern aesthetics from random 3D terrain surfaces generated using the Fourier Synthesis algorithm. Acoustics 2016: The Second Australasian Acoustical Societies Conference. (pp. 203–212).

[5] Spick, R. J., Cowling, P., & Walker, J. A. (2019). Procedural generation using spatial GANs for region-specific learning of elevation data. 2019 IEEE Conference on Games (CoG), (pp. 1–8). IEEE. https://doi.org/10.1109/CIG.2019.8848120

[6] Castle, C. J., & Crooks, A. T. (2006). Principles and concepts of agent-based modelling for developing geospatial simulations.

[7] Hofmann, P., Andrejchenko, V., Lettmayer, P., Schmitzberger, M., Gruber, M., Ozan, I., & Blaschke, T. (2016). Agent based image analysis (ABIA)-preliminary research results from an implemented framework. https://doi.org/10.3990/2.455

[8] Jahanbani, M., Vahidnia, M. H., & Aspanani, M. (2022). Geographical agent-based modeling and satellite image processing with application to facilitate the exploration of minerals in Behshahr. Iran. Arabian Journal of Geosciences, 15(9), 901. https://doi.org/10.1007/s12517-022-10165-8

[9] Ali, S. M., Doolan, M., Wernick, P., & Wakelam, E. (2018). Developing an agent-based simulation model of software evolution. Information and Software Technology, 96, 126–140. https://doi.org/10.1016/j.infsof.2017.11.013

[10] Aderum, O., & Åkerlund, J. (2016). Controllable Procedural Game Map Generation using Software Agents and Mixed Initiative.

[11] Roohitavaf, M., Ren, K., Zhang, G., & Ben-Romdhane, S. (2019). LogPlayer: Fault-tolerant Exactly-once Delivery using gRPC Asynchronous Streaming. https://doi.org/10.48550/arXiv.1911.11286

[12] Krämer, M., & Senner, I. (2015). A modular software architecture for processing of big geospatial data in the cloud. Computers & Graphics, 49, 69–81. https://doi.org/10.1016/j.cag.2015.02.005

[13] Leis, V., Boncz, P., Kemper, A., & Neumann, T. (2014). Morsel-driven parallelism: a NUMA-aware query evaluation framework for the many-core age. Proceedings of the 2014 ACM SIGMOD international conference on Management of data, (pp. 743–754). https://doi.org/10.1145/2588555.2610507

[14] Jacobsen, R. H., Jeppesen, J. H., Laursen, K. F., Skovsgaard, J., Jensen, H. N., & Toftegaard, T. S. (2017). A scalable cloud computing infrastructure for geospatial data analytics for change

detection. 2017 Euromicro Conference on Digital System Design (DSD), (pp. 403–410). IEEE. https://doi.org/10.1109/DSD.2017.49
[15] Jamzuri, E. R., Mandala, H., Analia, R., & Susanto, S. (2022). Cloud-Based Architecture for YOLOv3 Object Detector using gRPC and Protobuf. Jurnal Teknik Elektro, 14(1), 18–23. https://doi.org/10.15294/jte.v14i1.36537
[16] Minailenko R., Sobinov O., Konoplitska-Slobodenyuk K., Buravchenko K. (2021). Architectural Features of Distributed Computing Systems. Central Ukrainian Scientific Bulletin. Technical Sciences, 35(4), 16–23. https://doi.org/10.32515/2664-262X.2021
[17] Sharvari, T., & Sowmya Nag, K. (2019). A study on Modern Messaging Systems – Kafka. RabbitMQ and NATS Streaming. https://doi.org/10.48550/arXiv.1912.03715
[18] Wu, H., Shang, Z., & Wolter, K. (2019, August). Performance prediction for the Apache Kafka messaging system. 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Confeence on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), (pp. 154–161). IEEE. https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00036
[19] Levus, Y., Westermann, R., Morozov, M., Moravskyi, R., & Pustelnyk, P. (2022). Using Software Agents in a Distributed Computing System for Procedural Planetoid Terrain Generation. 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT). (pp. 446–449). IEEE. https://doi.org/10.1109/CSIT56902.2022.10000868
[20] Doran, J., & Parberry, I. (2010). Controlled procedural terrain generation using software agents. IEEE Transactions on Computational Intelligence and AI in Games, 2(2), 111–119. https://doi.org/10.1109/TCIAIG.2010.2049020
[21] Lim, F. Y., Tan, Y. W., & Bhojan, A. (2022). Visually improved erosion algorithm for the procedural generation of tile-based terrain. arXiv preprint arXiv:2210.14496. https://doi.org/10.5220/0010799700003124
[22] Gasch, C., Chover, M., Remolar, I., & Rebollo, C. (2020). Procedural modelling of terrains with constraints. Multimedia Tools and Applications, 79, 31125–31146. https://doi.org/10.1007/s11042-020-09476-3
[23] Mete, M. O., & Yomralioglu, T. (2021). Implementation of serverless cloud GIS platform for land valuation. International Journal of Digital Earth, 14(7), 836–850. https://doi.org/10.1080/17538947.2021.1889056
[24] Fanini, B., Pescarin, S., & Palombini, A. (2019). A cloud-based architecture for processing and dissemination of 3D landscapes online. Digital Applications in Archaeology and Cultural Heritage, 14, e00100. https://doi.org/10.1016/j.daach.2019.e00100

Y. V. Levus[1], P. Ya Pustelnyk[1], R. O. Moravskyi[1], M. Yu. Morozov[2]

[1] Lviv Polytechnic National University, Lviv, Ukraine
[2] Technical University of Munich, Munich, Germany

ARCHITECTURE OF A DISTRIBUTED SOFTWARE SYSTEM FOR PROCEDURAL PLANETOID TERRAIN GENERATION

The procedural generation of planetoids finds its place in the field of visualization of virtual worlds in video games, movies, and simulation tools. Due to the growing popularity of the application, the requirements for the quality, uniqueness, and scalability of visualization results are increasing, which, in turn, leads to higher requirements for hardware computing resources. This paper proposes a solution for the architecture of a software system for generating planetoid landscapes, based on a combination of a distributed computing system and the use of parallelism based on the Morsel-Driven Query Execution algorithm to overcome hardware limitations. The computing model includes the following components: a main server that supports gRPC connections; worker servers that perform the task of generating planetoid landscapes in parallel; a geospatial database containing vector data of generated planetoids (rivers and reservoirs; geographic regions – biomes, forests, and other road segments) binary storage of three-dimensional models that are superimposed on the generated planetoid landscape; tileset-storage for storing raster data required for a generation; users who use the software system for generating planetoid landscapes to realize their own goals. The use of software agents in the built system allows unifying a set of algorithms as a single entity used for a particular stage of landscape generation and solving the problem of software system extensibility. A distributed messaging system, a broker, is used to send and process requests using a topic per software agent sequence position. The broker utilizes load balancing to deliver landscape generation requests to background workers. To analyze the system's performance, experiments were conducted with different numbers of background workers (1, 2: 4, 8; 16) and segment sizes of 512, and 2048 pixels. The lowest average time for generating one segment was obtained when the segment size was 512 pixels and the number of segments was 64. The average segment generation time for the above experiments ranged from 0.33 to 9.32 seconds. The integrated architectural solution allowed to reduce the CPU time by 2 to 5 times compared to a system that uses the approach of storing queries in the database. The solution's efficiency is especially noticeable with large amounts of data, which is determined by the number of segments and their size in pixels.

Keywords: software agents, messaging, gRPC technology, broker, system performance, computational model.

Інформація про авторів:
Левус Євгенія Василівна, канд. техн. наук, доцент, кафедра програмного забезпечення.
Email: Yevheniia.V. Levus@lpnu.ua; https://orcid.org/0000-0001-5109-7533
Пустельник Павло Ярославович, магістр, кафедра програмного забезпечення.
Email: pavlo.pustelnyk.mnpzm.2021@lpnu.ua; https://orcid.org/0000-0003-3273-0211
Моравський Роман Орестович, магістр, кафедра програмного забезпечення.
Email: roman.moravskyi.mnpzm.2021@lpnu.ua; https://orcid.org/0000-0002-6695-1920
Морозов Микола Юрійович, магістр, школа обчислювальної техніки, інформації та технологій.
Email: mykola.morozov@tum.de; https://orcid.org/0000-0002-5892-6936

Цитування за ДСТУ: Левус Є. В., Пустельник П. Я., Моравський Р. О., Морозов М. Ю. Архітектура розподіленого програмного застосунку для генерування ландшафтів планетоїдів. *Український журнал інформаційних технологій.* 2023. Т. 5, № 1. С. 01–08.
Citation APA: Levus, Y. V., Pustelnyk, P. Ya., Moravskyi, R. O., Morozov M. Yu. (2023). Architecture of a distributed software system for procedural planetoid terrain generation. *Ukrainian Journal of Information Technology*, 5(1), 01–08. https://doi.org/10.23939/ujit2023.01.001