УДК 004.925
DOI: 10.25686/978-5-8158-2474-4-2025-164-171

Гибридный алгоритм отображения облаков для систем с несколькими графическими адаптерами

М. К. Богданов, А. М. Суворов, А. П. Булаев
Университет ИТМО, г. Санкт-Петербург, Россия

Аннотация. Параллельный рендеринг представляет собой широко распространенное решение для повышения производительности в компьютерной графике, при котором задействуется несколько графических процессоров (GPU) для одного приложения. Такие методы позволяют эффективно балансировать нагрузку и уменьшать временные простои при рендеринге, которые могут ограничивать частоту и детализацию итогового изображения. Однако современные реализации в основном ориентированы на серверные решения с аппаратной поддержкой высокоскоростных шин передачи данных и не учитывают особенности игровых платформ, таких как ноутбуки и персональные компьютеры, оснащенные как интегрированной, так и дискретной видеокартами. В этой работе основной фокус ставится именно на пользовательские конфигурации [1], которые чаще всего оснащены двумя графическими адаптерами. Визуализация реалистичных облаков в компьютерной графике является сложной задачей, связанной с большим количеством вычислений. В работе представлен гибридный алгоритм отображения реалистичных облаков, распределяющий графические задачи между GPU на основе отдельных техник, которые выполняются на основном графическом процессоре. Такой подход позволяет использовать DirectX 12 в режиме EMA, не зависящем от аппаратных интерфейсов и специфики производителей видеоадаптеров. Разработанное решение использует дополнительный графический процессор для генерации текстур облаков, что обеспечивает рациональное использование всех доступных аппаратных ресурсов системы без потери качества итогового изображения. Предложенный метод легко интегрируется в существующие конвейеры рендеринга и позволяет достичь значительного прироста производительности в пользовательских гетерогенных мульти-GPU-системах при минимальных затратах на реализацию.

***Ключевые слова*:** компьютерная графика, адаптивные алгоритмы, облака, DirectX, GPU-симуляция, параллельный рендеринг.

Hybrid cloud simulation algorithm for multi-GPU systems

M. K. Bogdanov, A. M. Suvorov, A. P. Bulaev
ITMO University, Saint-Petersburg, Russia

Abstract. Parallel rendering represents a widely adopted solution for enhancing performance in computer graphics, utilizing multiple graphics processing units (GPUs) within a single application. Such methods enable efficient load balancing and reduce rendering latency, which may otherwise constrain frame rates and the level of detail in the final image. However, contemporary implementations primarily target server-based solutions equipped with hardware support for high-speed data buses and fail to address the specificities of gaming platforms, such as laptops and personal computers featuring both integrated and discrete graphics cards. The primary focus of this work is on user systems, which are most commonly equipped with two graphics adapters. Rendering realistic clouds in computer graphics poses a significant challenge due to its computationally intensive nature. This paper presents a hybrid algorithm for realistic cloud visualization, which distributes graphics tasks across GPUs by delegating specific rendering techniques to the primary graphics processor. This approach facilitates the use of DirectX 12 in Explicit Multi-Adapter (EMA) mode, eliminating dependencies on proprietary hardware interfaces and vendor-specific limitations. The proposed solution employs an additional GPU for cloud texture generation, ensuring optimal utilization of all available hardware resources without compromising output quality. The method is designed for seamless integration into existing rendering pipelines and demonstrates substantial performance improvements in heterogeneous multi-GPU systems with minimal implementation overhead.

***Keywords*:** computer graphics, adaptive algorithms, clouds, DirectX, GPU-simulation, parallel rendering.

Введение

Компьютерная графика является важной частью визуализации физических процессов, интерактивных развлечений и современных инженерии и дизайна. Степень реалистичности итогового рендера определяется корректностью используемых методов просчёта освещения и рендеринга. Эти факторы напрямую влияют на то, как пользователь воспринимает изображение. Особые трудности возникают при визуализации облаков, так как необходимо каждый кадр просчитывать многократное рассеивание света, динамически изменяемые вольюметрические (объёмные) структуры, а также поддерживать производительность на комфортном для пользователя уровне.

Применяемые сегодня алгоритмы, использующие растеризацию и трассировку лучей, демонстрируют компромисс между скоростью и точностью. Физически корректные методы обладают высокой вычислительной сложностью. В то же время упрощенные методы приводят к упрощению визуализации и худшему качеству изображения, все эти решения не подходят для гибридных систем. Растущая популярность игровых ноутбуков, оснащённых несколькими графическими адаптерами [1], подчеркивает важность разработки алгоритмов, оптимизированных для подобных конфигураций.

Цель работы – исследование возможности вынесения генерации текстур облаков на дополнительный графический процессор (включая интегрированные решения) и оценка эффективности такого распределения вычислений в условиях потребительских систем с ограниченной пропускной способностью шины PCI Express.

В данной работе:

1) предложен и разработан алгоритм гибридного рендеринга облаков, распределяющий нагрузку между графическими адаптерами и предназначенный для использования в приложениях, где требуется реалистичная визуализация неба без необходимости поддержки произвольных углов обзора;

2) показана применимость такого подхода для конфигураций с интегрированным и дискретным GPU без использования специализированных интерфейсов по типу NVLink и XGMI;

3) проведён анализ производительности и выявлены условия, в которых обмен данными между адаптерами не даёт прироста производительности системе.

Постановка задачи

Цель данного исследования – реализовать метод гибридного рендеринга облаков для multi-GPU-систем и провести сравнение с подобным методом, использующим один графический адаптер. Основное внимание уделяется разработке и тестированию алгоритма, в котором текстуры облаков вычисляются на вспомогательном GPU и передаются на основной GPU, на котором обрабатываются все остальные стадии графического конвейера.

Таким образом, задача состоит в том, чтобы:

1) написать алгоритмы рендеринга облаков, использующие как один графический адаптер, так и несколько;

2) измерить производительность и провести сравнение времени рендеринга кадра и FPS с этими алгоритмами на разных системах;

3) оценить преимущества алгоритма гибридного рендеринга облаков перед однопроцессорной конфигурацией;

4) провести анализ полученных данных, чтобы определить, в каких условиях использование второй видеокарты наиболее эффективно.

Теория

Рассмотрим основные подходы к созданию двумерных структур в виде облаков. Широкое распространение в компьютерной графике получила техника Billboard [2], где в сцену интегрируется плоское изображение, постоянно направленное к камере пользователя (рис. 1).

Другой популярный метод основан на наложении нескольких слоёв текстур с варьируемыми атрибутами прозрачности и позиции [3], (рис. 2). Оба подхода страдают от статичности результата, поскольку опираются на статичные изображения, что исключает динамическую трансформацию форм.

Алгоритм процедурной генерации с применением математических шумовых функций выступает альтернативой. Технология обеспечивает создание уникальных анимированных текстур, однако при повышении детализации или выборе сложных математических функций вычислительные затраты резко возрастают (рис. 3). Также недостатком является потеря объемности – критически важного аспекта в визуализации облаков.

Для преодоления указанных проблем разработан комбинированный алгоритм, объединяющий метод процедурной генерации шума с методом многослойного наложения текстур. Динамическое создание каждого слоя отдельно обеспечивает объемность, визуальную динамику и возможности имитации различных конфигураций облаков. Для симуляции неба используется техника Skybox, для верхней части которой каждый кадр процедурно вычисляется облачная текстура. Пример представлен на рисунке 4.

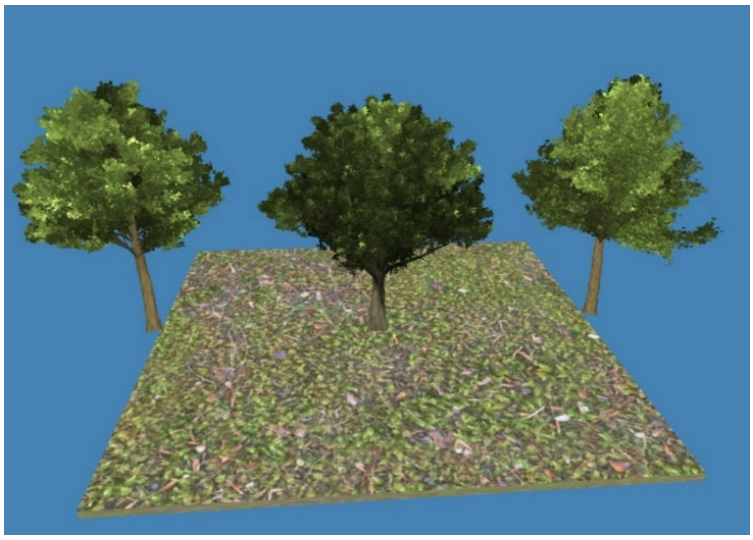

Рисунок 1. Демонстрация работы метода Billboard

Рисунок 2. Демонстрация работы метода Layered Texture

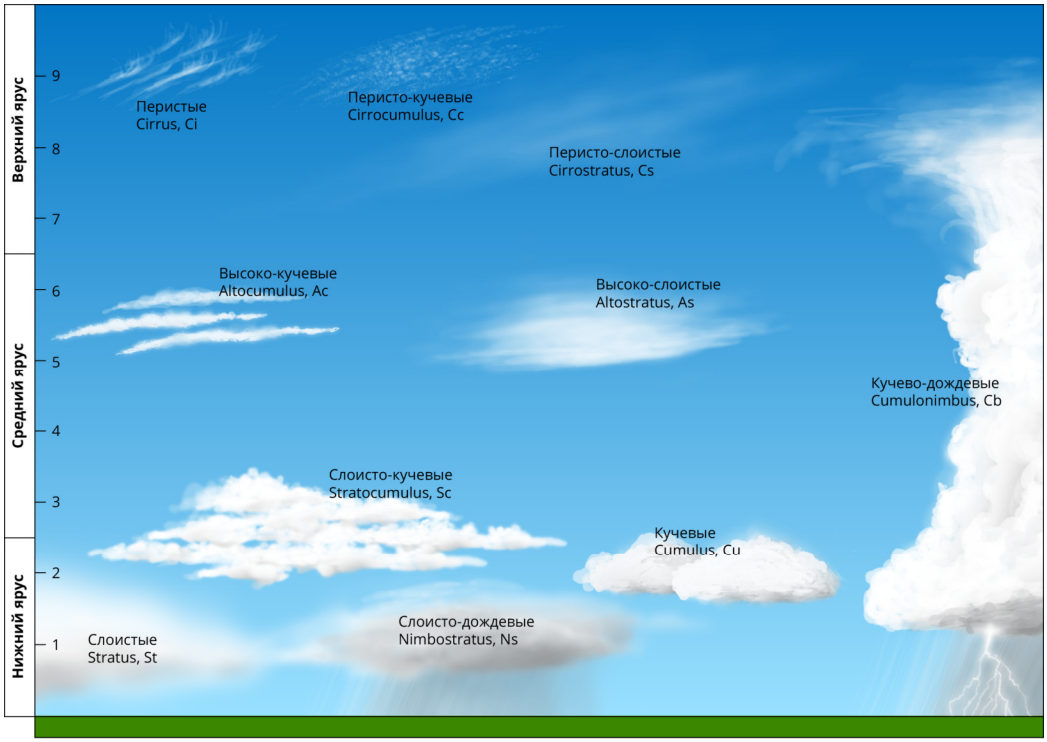

Рисунок 3. Типы облаков

Рисунок 4. Визуализация работы алгоритма

Рассмотрим способы работы с несколькими GPU [4,5]. В современных системах с несколькими графическими процессорами применяются три основных подхода к распределению вычислительной нагрузки.

1. Межкадровый параллелизм. Это технологии наподобие AFR (Alternate frame rendering) [6,7], при использовании которой каждый GPU рендерит отдельные кадры, чередующиеся с кадрами с другой видеокарты. Например, при двух видеокартах одна рендерит чётные кадры, а вторая – нечётные. Основные преимущества – простая реализация и минимальная связность между устройствами. Недостатки – повышенная задержка вывода, необходимость полного дублирования ресурсов на каждом адаптере, а также проблемы совместимости с темпоральными алгоритмами современного рендеринга (например TAA, TSR, SSR) [8]. Данный подход использовался в технологиях NVIDIA SLI и AMD CrossFire, но в настоящее время их поддержка сокращается [9,10].

2. Пространственное разделение. Технологии наподобие SFR (Split-frame rendering) [11,12] разделяют экран на отдельные области, которые обрабатываются разными GPU. У решений с таким подходом (например CHOPIN [13], MCM-GPU [14]) присутствуют сложность синхронизации при наложении регионов друг на друга, высокие требования к пропускной способности интерфейса и необходимость распределения геометрических данных. Современные реализации требуют специализированных высокоскоростных интерконнектов (NVLink, XGMI) [15,16] или высокопроизводительных сетевых технологий (Infiniband, RDMA) [12] и подходят в основном для серверных решений.

3. Функциональное разделение (гибридный подход). При таком подходе разные стадии графического конвейера закрепляются за разными GPU [17]. Например, один адаптер генерирует текстуры облаков, в то время как другой занимается основным рендерингом финального кадра со всеми стадиями. Подход эффективен, когда время выполнения конкретной стадии значительно превышает затраты на передачу данных. Это применимо ко многим генеративным алгоритмам и симуляциям в компьютерной графике. Ключевым ограничением является пропускная способность шины PCI Express, что особенно актуально для потребительских систем с гетерогенными конфигурациями GPU, например ноутбуками.

Современные графические API (DirectX 12, Vulkan) предоставляют необходимые механизмы для реализации гибридного подхода, включая кросс-адаптерные ресурсы и синхронизацию. Однако эффективность решения напрямую зависит от оптимизации межадаптерного обмена данными в условиях ограниченной пропускной способности PCIe.

В DirectX12, который мы используем в своей работе, за работу с несколькими GPU отвечает режим Explicit Multi-Adapter (EMA), у которого есть два варианта работы:

- Linked Mode. Несколько физических адаптеров работает как единое устройство с общей памятью, что требует однородности GPU и специализированных высокоскоростных интерконнектов (NVLink/XGMI);

- Unlinked Mode. Каждый GPU рассматривается как независимое устройство с отдельной памятью, позволяя работать с гетерогенными конфигурациями.

Для нашего гибридного алгоритма рендеринга облаков был выбран Unlinked Mode по следующим причинам:

1) подход к гетерогенными системам. Unlinked Mode позволяет задействовать разнородные GPU (дискретный + интегрированный) без требований к идентичности архитектуры;

2) отсутствие зависимости от специализированных интерконнектов. Unlinked Mode работает через стандартные PCIe-соединения, что соответствует стандартам потребительских систем;

3) гибкость распределения задач. Unlinked Mode предоставляет точный контроль над назначением стадий рендеринга конкретным адаптерам.

После создания алгоритма симуляции облаков была реализована его гибридная версия, алгоритм показан на рисунке 5. Данный подход, в отличие от последовательного расчета на одном GPU, организует параллельные вычисления на нескольких GPU, оптимизируя загрузку вычислительных ресурсов системы. На дополнительном GPU происходит генерация текстур облаков, после чего они копируются на другую видеокарту. Основной GPU использует уже созданную другим адаптером текстуру облаков при рендере кадра, а не вычисляет, в отличие от версии алгоритма для одного адаптера. Таким образом мы полностью снимаем достаточно затратную по вычислениям задачу генерации облаков с основного GPU.

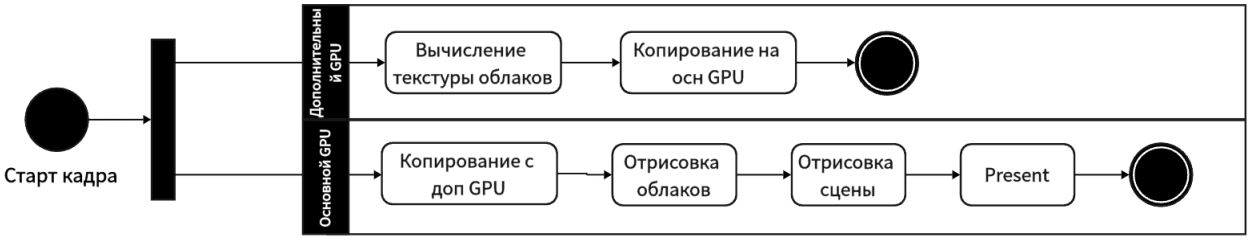

Рисунок 5. Гибридный алгоритм

Результаты экспериментов

Для обеспечения значимости результатов выполнено по 10 прогонов на различных тестовых стендах и замерены данные. Для примера на представленных графиках (рис. 6 и 7) показаны усредненные значения времени рендера кадра на двух стендах. Каждый запуск включал 30-секундную фазу инициализации и прогрева кэшей, после чего в течение 120 секунд осуществлялись основные расчеты и замеры производительности. Для каждого кадра регистрировалось время, затраченное на его рендеринг.

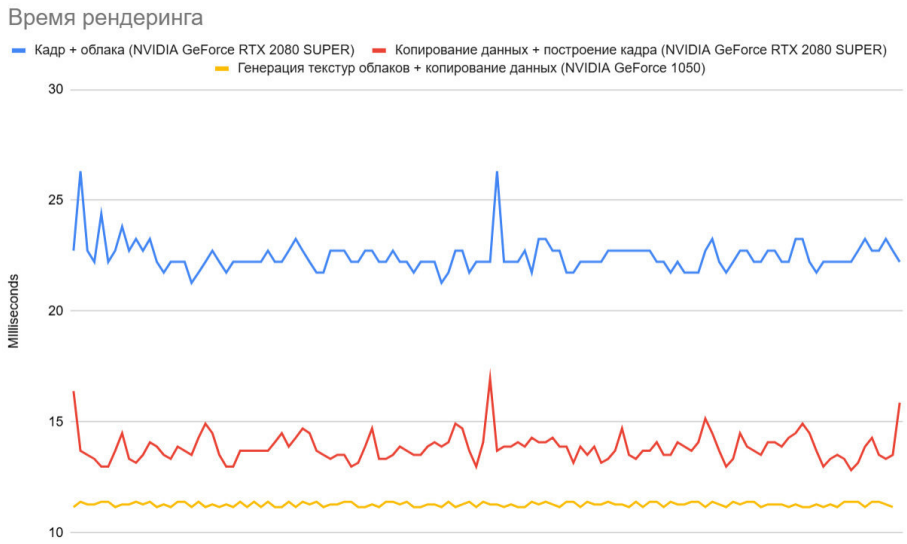

Рисунок 6. Результаты тестирования RTX 2080 SUPER + NVIDIA GeForce 1050 (Процессор AMD Ryzen 2800X)

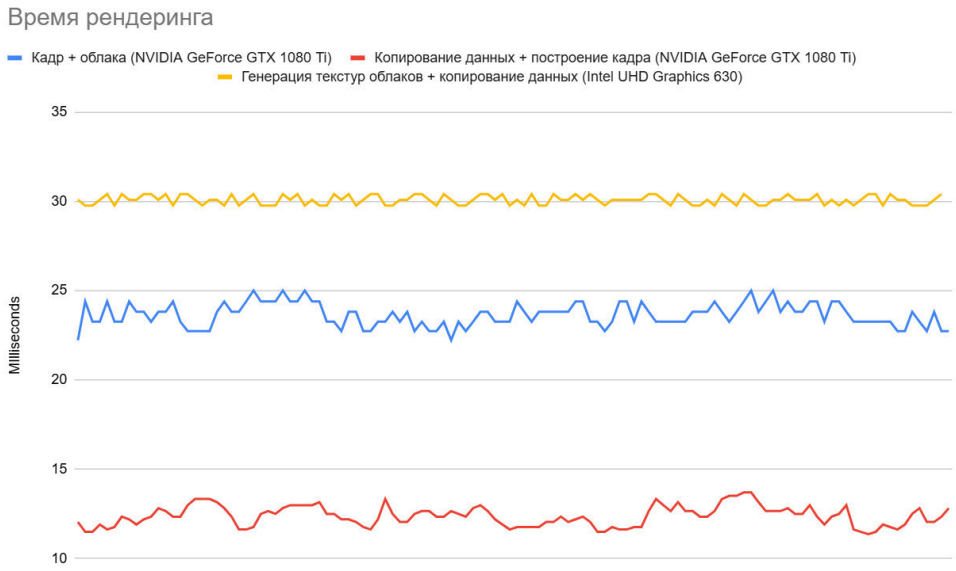

Рисунок 7. Результаты тестирования GTX 1080 Ti + Intel UHD Graphics 630 (Процессор Intel Core i7 9700)

Сцена для теста состояла из вращающейся вокруг центра мира камеры, статических объектов, системы частиц с симуляцией 20000 полупрозрачных частиц и пяти источников освещения, один из которых представлял собой Directional light, эмулирующий солнечный свет. Количество отображаемых полигонов составляло примерно два миллиона. Динамические тени от всех источников освещения рассчитывались для каждого кадра. Для симуляции облачного неба использовалась техника Skybox, для верхней части которой в каждом кадре процедурно вычислялась текстура облаков. Пример можно наблюдать на рисунке 8. Все эксперименты проводились в разрешении Full HD (1920×1080 пикселей) с использованием последних стабильных версий драйверов с официальных сайтов производителей, доступных на момент тестирования.

Анализ данных показал прирост суммарной производительности рендеринга в реальном времени на 40–60 % в зависимости от конфигурации системы.

Однако тесты показали возникновение визуальных артефактов при значительном отрыве в производительности GPU. Это обусловлено существенным отставанием частоты обновления данных на дополнительном адаптере от частоты рендера кадра на основном. Для устранения данного недостатка в алгоритм внедрён коэффициент масштабирования, изменяющий размер текстуры облаков.

При запуске приложения замеряются время генерации текстуры (на дополнительном адаптере, iGPU) и время построения кадра (на основном адаптере, dGPU) для одного кадра.

Если дополнительный адаптер завершил кадр быстрее основного или за то же время, то разрешение текстуры не меняется (в нашем примере изначально 4096x4096 пикселей). Если же основной адаптер рендерит один кадр быстрее дополнительного, то разрешение уменьшается в 2 раза по каждой стороне и начинается цикл изменения разрешения для поиска подходящего; если при замере времени видеокарт в следующем кадре время обеих видеокарт примерно совпадает, происходит выход из цикла и текущее разрешение сохраняется на всё время работы приложения. Если время iGPU меньше времени dGPU, разрешение увеличивается на 50 % от прошлого диапазона, если больше – уменьшается на 50 % от прошлого диапазона. После этого цикл начинается заново с измерения времени рендера на видеокартах.

После добавления алгоритма изменения размера текстуры скорости рендеринга на обоих GPU стали примерно одинаковыми и артефакты, вызванные сильным отставанием дополнительного адаптера от основного, пропали.

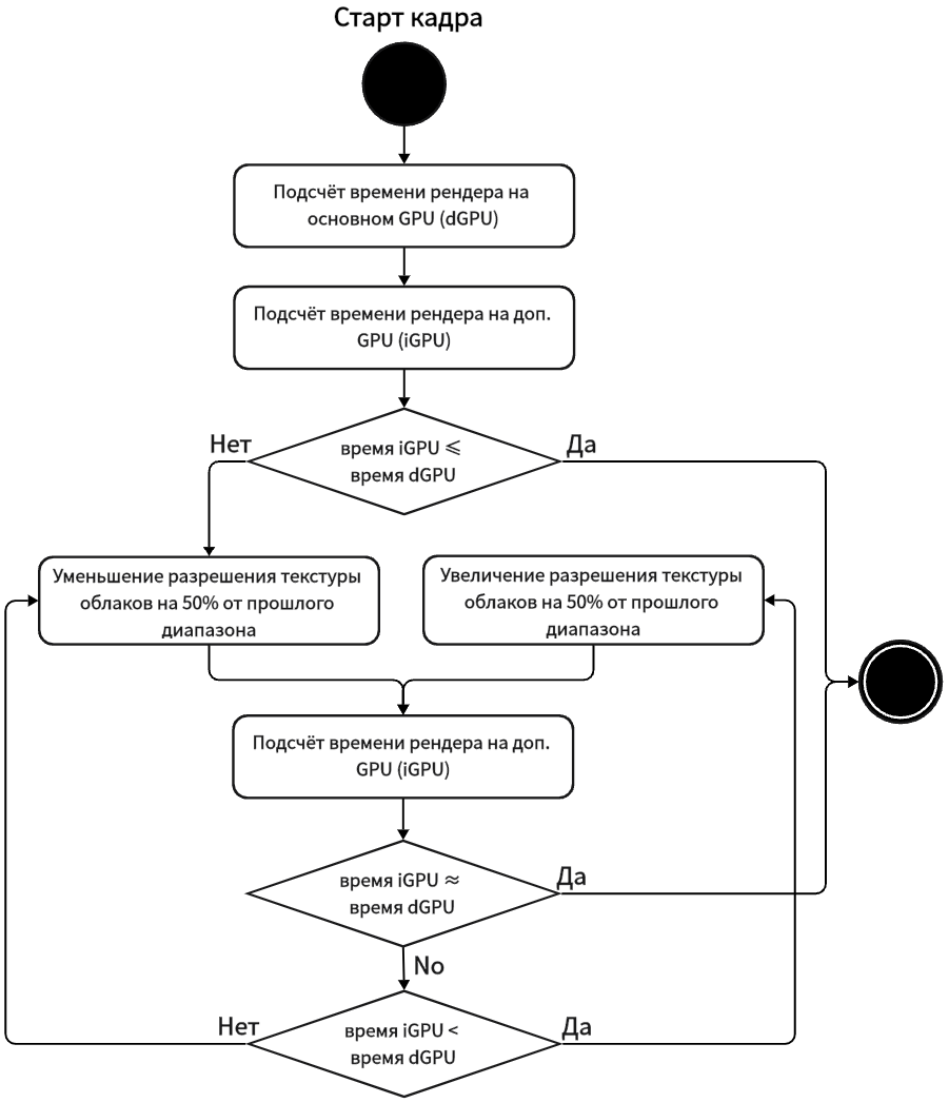

Рисунок 8. Алгоритм расчета коэффициента масштабирования текстуры облаков

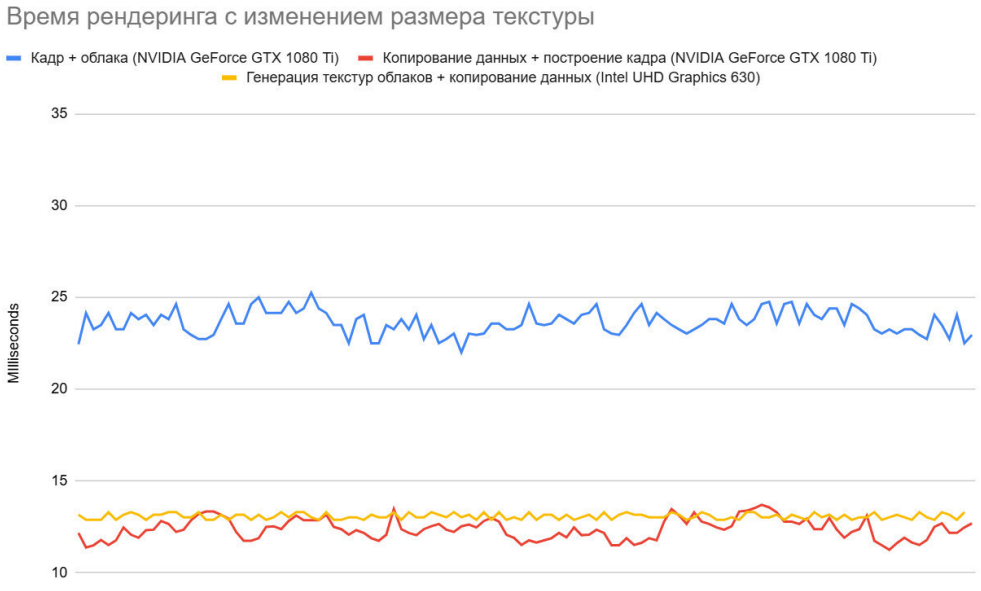

Рисунок 9. Результаты тестирования GTX 1080 Ti + Intel UHD Graphics 630 после добавления коэффициента масштабирования текстуры (процессор Intel Core i7 9700)

Выводы

Результаты данного исследования демонстрируют обеспечение прироста производительности графической части системы с несколькими GPU благодаря разработанному методу отображения облаков, а также создают перспективы для дальнейших исследований в данной области. Было выявлено, что гибридные вычисления сильно зависят от конфигурации системы. В ходе работы были созданы подходы для решения проблем в системах с кардинально различающимися по мощности видеокартами.

В перспективе планируется решить проблему с уменьшением детализации неба и исследовать эффективность распределения вычислительных алгоритмов на несколько видеоадаптеров для объёмных 3D-облаков, а также разработать методы компенсации качества генерируемой текстуры при её использовании в системах с GPU разной производительности.

Список литературы

1. Steam Данные об оборудовании пользователей за апрель. URL: https://store.steampowered.com/hwsurvey/videocard (дата обращения: 19.05.2025).

2. Tomas Akenine-Miller, Eric Haines Billboarding. URL: https://www.flipcode.com/archives/Billboarding-Excerpt_From_iReal-Time_Renderingi_2E.shtml (дата обращения: 14.05.2025).

3. Roden T., Parberry I. Clouds and stars. New York, NY, USA: ACM, 2005.C. 434–437.

4. A Sorting Classification of Parallel Rendering / Molnar S., et al. 1994.

5. Humphreys G., Hanrahan P. Sort-first, sort-middle, sort-last parallel rendering // ACM SIGGRAPH Course Notes. 2002.

6. Krawczyk G. Microstuttering in AFR-based multi-GPU rendering // Computer Graphics Forum. 2009. Vol. 28, no. 2. P. 367–374.

7. Haines E. Temporal reprojection and frame dependencies in rendering // ACM Queue. 2018. Vol. 16, no. 6. P. 40-61.

8. Karis B. High-quality temporal supersampling // ACM SIGGRAPH Courses. 2014.

9. VR SLI – Scalable Per-Eye Rendering for VR // NVIDIA Developer. URL: https://developer.nvidia.com/vrworks/graphics/vrsli (дата обращения: 01.09.2025).

10. NVIDIA. SLI Best Practices Guide. 2015.

11. Moreland K., Angel E. The Distributed Framebuffer // Proceedings of the IEEE Symposium on Parallel Visualization and Graphics. 2001. P. 61–68.

12. Parallel volume rendering using binary-swap compositing / Ma K.-L., et al. // IEEE Comput Graph Appl. 1994. Vol. 14, no. 4. P. 59–68.

13. Ren X., Lis M. CHOPIN: Scalable Graphics Rendering in Multi-GPU Systems via Parallel Image Composition // Proceedings – International Symposium on High-Performance Computer Architecture. IEEE Computer Society, 2021. Vol. 2021-February. P. 709–722.

14. MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scaling / Jain A., et al. // Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA). 2017. P. 320–332.

15. NVLink High-Speed GPU Interconnect // NVIDIA. URL: www.nvidia.com/en us/data-center/nvlink (дата обращения: 01.09.2025).

16. Infinity Fabric Link // AMD. URL: www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/white-papers/overview-amd-epyc7003-series processors-microarchitecture.pdf (дата обращения: 01.09.2025).

17. Computer Graphics: Principles and Practice / Foley J., et al. 3rd изд. Addison-Wesley, 2014.