DOI: 10.15514/ISPRAS-2026-38(3)-43

Функциональное распределение вычислительных задач компьютерной графики в гибридных системах

М.К. Богданов, ORCID: 0009-0002-3308-7643 <raeinforce@gmail.com>
А.М. Суворов, ORCID: 0009-0000-3921-4805 <andryusha.suvorov@gmail.com>
М.Е. Ивашечкина, ORCID: 0009-0004-4254-7722 <mari.avi00@mail.ru>
Р.Р. Султанов, ORCID: 0009-0009-9628-9737 <ravil.r.sultanov@gmail.com>
Г.С.Лях, ORCID: 0009-0009-0135-6723 <gleb_lyah@mail.ru>
А.П. Булаев, ORCID: 0009-0007-0148-9847 <bulaevap97@gmail.com>

Университет ИТМО,
Российская Федерация, 197101, Санкт-Петербург, Кронверкский пр., д. 49, лит. А.

Аннотация. В работе рассматривается проблема эффективного использования аппаратных ресурсов в компьютерных системах с несколькими графическими адаптерами в задачах визуализации компьютерной графики. Предлагается гибридная реализация к выполнению задач с использованием графического конвейера, которая основана на функциональном разделении вычислений между видеоадаптерами, в отличие от традиционных методов, где распределение нагрузки осуществляется по пространственным или временным критериям. Разработаны и протестированы алгоритмы для трех графических техник: моделирование облаков, расчет теней и окружающего затенения. Благодаря использованию API DirectX 12 в режиме Explicit Multi-Adapter было реализовано взаимодействие между GPU без необходимости использования специализированных аппаратных интерфейсов. Экспериментальные исследования показали прирост производительности до 28% для расчета теней, до 11% для окружающего затенения и до 59% для моделирования облаков в зависимости от конфигурации системы. Полученные результаты демонстрируют перспективность гибридного подхода для практического применения в потребительских устройствах, таких как игровые компьютеры и ноутбуки с комбинированной графикой.

Ключевые слова: гибридный рендеринг; многоадаптерные системы; семейство интерфейсов DirectX 12; компьютерная графика.

Для цитирования: Богданов М.К., Суворов А.М., Ивашечкина М.Е., Султанов Р.Р., Лях Г.С., Булаев А.П. Функциональное распределение вычислительных задач компьютерной графики в гибридных системах. Труды ИСП РАН, том 38, вып. 3, часть 3, 2026 г., стр. 161–176. DOI: 10.15514/ISPRAS–2026–38(3)–43.

Task Partitioning for Computer Graphics in Heterogeneous Computing Systems

M.K. Bogdanov, ORCID: 0009-0002-3308-7643 <raeinforce@gmail.com>
A.M. Suvorov, ORCID: 0009-0000-3921-4805 <andryusha.suvorov@gmail.com>
M.E. Ivashechkina, ORCID: 0009-0004-4254-7722 <mari.avi00@mail.ru>
R.R. Sultanov, ORCID: 0009-0009-9628-9737 <ravil.r.sultanov@gmail.com>
G.S. Lyakh, ORCID: 0009-0009-0135-6723 <gleb_lyah@mail.ru>
A.P. Bulaev, ORCID: 0009-0007-0148-9847 <bulaevap97@gmail.com>

ITMO University,
Kronverksky Ave., 49, bldg. A, St. Petersburg, 197101, Russia.

Abstract. This paper addresses the challenge of efficient hardware resource utilization in computer systems equipped with multiple graphics processing units for computer graphics rendering. We propose a hybrid pipeline execution approach based on functional partitioning of computational tasks across heterogeneous GPUs, diverging from conventional spatial or temporal load-balancing strategies. Algorithms were designed and evaluated for three rendering techniques: procedural cloud simulation, shadow mapping, and ambient occlusion. By leveraging the DirectX 12 API in Explicit Multi-Adapter mode, cross-GPU communication was achieved without reliance on specialized hardware interconnects (e.g., SLI/CrossFire bridges). Experimental results demonstrate performance gains of up to 28% for shadow mapping, 11% for ambient occlusion, and 59% for cloud rendering workloads, contingent upon system configuration. These findings validate the efficacy of functional partitioning in heterogeneous GPU environments and highlight its practical applicability for consumer-grade hardware, including gaming PCs and laptops boarded with discrete and integrated graphics subsystems.

Keywords: hybrid rendering; multi-GPU systems; DirectX 12; computer graphics.

For citation: Bogdanov M.K., Suvorov A.M., Ivashechkina M.E., Sultanov R.R., Lyakh G.S., Bulaev A.P. Task Partitioning for Computer Graphics in Heterogeneous Computing Systems. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 3, part 3, 2026, pp. 161-176 (in Russian). DOI: 10.15514/ISPRAS-2026-38(3)-43.

1. Введение

Все современные приложения компьютерной графики, такие как игры, системы виртуальной и дополненной реальности, а также инструменты для научной визуализации, предъявляют постоянно растущие требования к производительности графических систем. Для того, чтобы удовлетворить эти требования, производители компьютерных комплектующих предлагают все более мощные графические процессоры (GPU), но ресурсы даже самых современных видеокарт оказываются недостаточными для решения сложных вычислительных задач в режиме реального времени при высоком разрешении вывода изображений [1-2].

В большинстве современных персональных компьютеров и ноутбуков установлены как минимум два графических адаптера: дискретная видеокарта (dGPU) и интегрированное в процессор графическое ядро (iGPU). Согласно данным мониторинга Steam Hardware Survey, около 55% пользователей имеют системы с несколькими графическими адаптерами [3]. Однако существующие подходы к рендерингу с использованием нескольких видеокарт ориентированы на конфигурации с несколькими дискретными GPU, которые соединены специализированными высокоскоростными интерфейсами, такими как SLI, CrossFire, NVLink, XGMI, что делает их неприменимыми для большинства игровых потребительских устройств [4-5].

Основные методы, которые используются в современных подходах к мульти-GPU рендерингу, можно разделить на отдельные категории: альтернативный рендеринг кадров (Alternate Frame Rendering, AFR) и разделенный пространственный рендеринг (Split Frame

Rendering, SFR) [6-8]. При использовании AFR каждый адаптер в системе последовательно обрабатывает отдельные кадры. Такой способ рендеринга обеспечивает простоту реализации, но не позволяет эффективно задействовать все ресурсы системы при использовании существенно различающихся по производительности GPU, а также плохо взаимодействует с алгоритмами, в которых есть зависимости от ресурсов, используемых между кадрами [9]. При использовании SFR, в свою очередь, кадр разделяется на отдельные части, каждая из которых обрабатывается отдельным GPU. Этот подход обеспечивает лучшую балансировку нагрузки, но требует сложной синхронизации и передачи данных между адаптерами, особенно при работе с прозрачными объектами и эффектами, затрагивающими всю сцену [8,10]. SFR также плохо масштабируется в рамках конфигураций, в которых используются видеокарты с разным уровнем производительности.

Более современным решением является подход MDScale, который объединяет полноценные системы (CPU+GPU) в единую сеть. Он позволяет обеспечить близкое к линейному масштабирование производительности на большее количество GPU, но требует специализированных интерфейсов связи, что исключает его применение для большинства потребительских конфигураций [11].

Для того чтобы преодолеть ограничения как современных, так и традиционных подходов мы предлагаем гибридный метод распределения вычислительных задач между несколькими адаптерами, который основан на функциональном разделении вычислений. В отличие от существующих методов, наш подход:

1. Позволяет эффективно использовать любые GPU, включая комбинации дискретных и интегрированных видеокарт от разных производителей;
2. Не требует специализированных аппаратных интерфейсов связи между адаптерами, благодаря использованию шины PCIe для обмена и синхронизации данных;
3. Распределяет нагрузку на основе функциональных особенностей задач, отбирая для переноса на дополнительный GPU такие алгоритмы, в которых имеются компактные выходные данные и минимальные зависимости от ресурсов, используемых между кадрами;
4. Может быть легко интегрирован в существующие графические приложения без значительной модификации исходного кода;

В работе представлены результаты разработки и тестирования гибридных алгоритмов для трех графических техник: симуляция и визуализация облаков, расчет теней и расчет окружающего затенения. Эксперименты были проведены на различных конфигурациях оборудования, включая как пары дискретных GPU, так и гибридные системы с дискретными и интегрированными графическими адаптерами.

2. Предпосылки и мотивации

2.1 Ограничения существующих подходов к параллельному рендерингу

Современные подходы к реализации параллельного рендеринга сталкиваются с рядом фундаментальных проблем и ограничений при работе с гибридными конфигурациями. Например, в исследованиях Eilemann и др. [12-13] подчеркивается, что традиционные подходы, такие как AFR и SFR, предполагают, что в системе будут использоваться примерно одинаковые по производительности GPU. Данные условия совершенно не соответствуют реальным конфигурациям распространённых гибридных систем с комбинацией дискретного и интегрированного GPU. Результат тестирования алгоритма SFR в гибридной системе представлен на рис. 1.

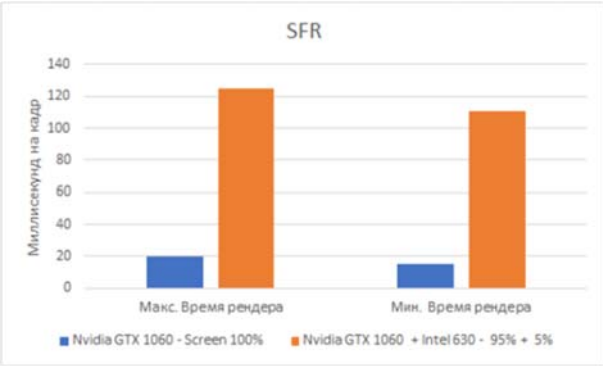

Рис. 1. Результат работы подхода SFR на гибридной системе с iGPU и dGPU.
Fig. 1. Results of the SFR approach on a hybrid system with iGPU and dGPU.

В работах Ren и Lis [14], а также Kim и др. [15], отмечено, что SFR требует значительных затрат на обмен данными между видеопроцессорами в системе, особенно в ситуациях обработки прозрачных объектов и эффектов. Также стоит отметить, что в системах, где нет возможности использования высокоскоростных интерфейсов связи, таких как NVLink и XGMI, пропускная способность шины PCIe существенно ограничивает эффективность передачи данных между большим количеством видеокарт [4,16].

Кроме того, исследования Arunkumar и др. [5] и Miic [17] показывают, что современные алгоритмы глобального освещения, постобработки и временных эффектов имеют сильные зависимости от результатов вычислений между различными кадрами и другими алгоритмами, что делает применение AFR проблематичным. Темпоральные алгоритмы, такие как TAA (Temporal Anti-Aliasing) [18], требуют сохранения состояния между кадрами и не могут быть эффективно распределены по разным GPU без дополнительных затрат на синхронизацию.

Еще одним важным ограничением является зависимость от производителя видеокарты и доступности оборудования, приобретаемого потребителем. Решения, предлагаемые NVIDIA SLI [19] и AMD CrossFire [20], работают только с конфигурациями GPU одного производителя и постепенно теряют как программную поддержку в драйверах, так и аппаратную, поскольку все меньше выпущенных видеокарт имеют механизмы для такого взаимодействия [4].

2.2 Возможности DirectX 12 Explicit Multi-Adapter

Появление DirectX 12 расширило возможности разработчиков и предоставило возможность прямого управления несколькими GPU в системе без зависимости от драйверов производителя. В режиме Explicit Multi-Adapter (EMA) приложение само управляет распределением задач и синхронизацией между адаптерами [21]. В EMA доступны два режима работы: Linked Mode и Unlinked Mode. Linked Mode предполагает использование одинаковых GPU с общей виртуальной адресацией памяти, что похоже на традиционные решения SLI/CrossFire. Unlinked Mode позволяет работать с любыми GPU, используя отдельные девайсы, очереди и механизмы синхронизации [21-22]. На рис. 2 представлен пример работы двух GPU в режиме EMA с Unlinked Mode.

Использование Unlinked Mode позволяет работать с любыми конфигурациями графического оборудования, поскольку синхронизация и обновление данных на адаптерах находится в зоне ответственности разработчика и работает через PCIe шину, что убирает требование использования специализированных аппаратных интерфейсов.

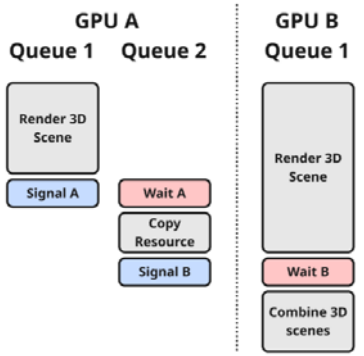

Рис. 2. Механизм работы разделенных между адаптерами вычислений.
Fig. 2. Working principle of computations distributed between adapters.

2.3 Анализ рендер-графов и выбор алгоритмов для разделения

Для того чтобы определить наиболее подходящие задачи для разделения вычислений между GPU, мы провели анализ рендер-графов современных игровых приложений и движков с использованием инструментов профилирования [23-24]. Исходя из результатов анализа, мы выработали список критериев оценки пригодности задач для разделения вычислений между адаптерами:

- Относительная независимость от других этапов конвейера.
- Компактные выходные данные, минимизирующие объем передаваемых между GPU данных.
- Высокая вычислительная сложность, позволяющая компенсировать накладные расходы на обмен данными.
- Минимальные зависимости между входными и выходными данными

3. Гибридные алгоритмы

Предлагаемая нами архитектура гибридного рендеринга основана на функциональном разделении задач между основным и дополнительным GPU. Основной принцип заключается в выносе на вторичный адаптер вычислительно затратных задач, которые подходят под описанные ранее критерии. Общий механизм работы можно описать следующим образом:

1) Основной GPU выполняет предварительные этапы рендеринга, такие как G-buffer, Prepass глубины и нормалей;
2) Необходимые данные передаются на дополнительный GPU через кросс-адаптерные ресурсы;
3) Дополнительный GPU рассчитывает технику на основе полученных данных;
4) Результаты вычислений передаются обратно на основной GPU;
5) Основной GPU завершает рендеринг сцены с использованием полученных данных

Такая архитектура позволяет полностью перекрывать время вычислений на вторичном GPU с работой основного адаптера, минимизируя общее время построения кадра. На рис. 3 можно наблюдать рендер-граф, который используется в разработанном нами бенчмарке. В нем красным цветом отмечены все исследуемые нами техники. Для каждой из исследуемых графических техник были разработаны специализированные алгоритмы разделения вычислений.

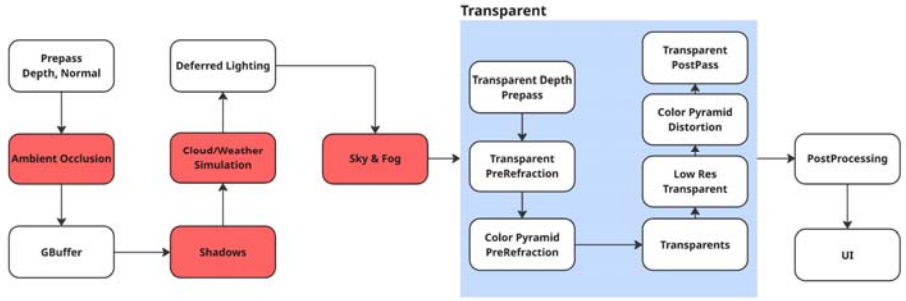

Рис. 3. Пример рендер-графа приложения.
Fig. 3. Example of an application render graph.

3.1 Гибридный алгоритм отображения облаков

Для оценки эффективности разделения вычислительных техник, которые считаются параллельно основному рендер графу, нами был выбран алгоритм отображения облаков посредством создания слоёной кубической текстуры неба с использованием различных шумов [25-26]. Алгоритм выносит на дополнительный GPU вычисление текстуры облаков, в то время как основной GPU обрабатывает остальные этапы рендеринга. Схематичное описание алгоритма представлено на рис. 4. Процесс включает следующие шаги:

- Основной GPU занимается отображением сцены;
- Дополнительный GPU выполняет симуляцию и рендеринг облаков в текстуру, и по завершении расчетов текстура передается на основной GPU как shared-ресурс;
- Основной GPU копирует текстуру из shared-ресурса и использует ее для финальной отрисовки

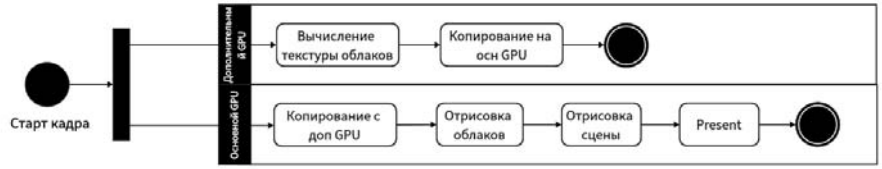

Рис. 4. Гибридный алгоритм отображения облаков.
Fig. 4. Hybrid cloud rendering algorithm.

Однако в результате тестирования выяснилось, что данная реализация гибридного алгоритма может вызывать визуальные артефакты. Причина их появления связана с разной частотой отображения сцены и шага симуляции облаков, что особенно распространено в гибридных системах, где есть значительный разброс в уровне производительности видеокарт. Для решения этой проблемы, была внедрен коэффициент масштабирования, расчет которого запускался на старте приложения, а размер результирующей текстуры облаков масштабировался согласно данному коэффициенту. Схематичное описание алгоритма поиска коэффициента масштабирования представлен на рис. 5.

Этот коэффициент показывает во сколько раз необходимо уменьшить результирующую текстуру облаков, чтобы дополнительная видеокарта успевала заполнить ее данными до того, как она понадобится основной видеокарте. Применение коэффициента позволило сбалансировать вычислительную нагрузку на дополнительном адаптере и избежать визуальных артефактов в гибридных системах со значительно отличающимися по

производительности видеокартами, а также оставить возможность для увеличения детализации текстуры в системах, где видеокарты сопоставимы по производительности.

Пример сгенерированных алгоритмом облаков показан на рис. 6.

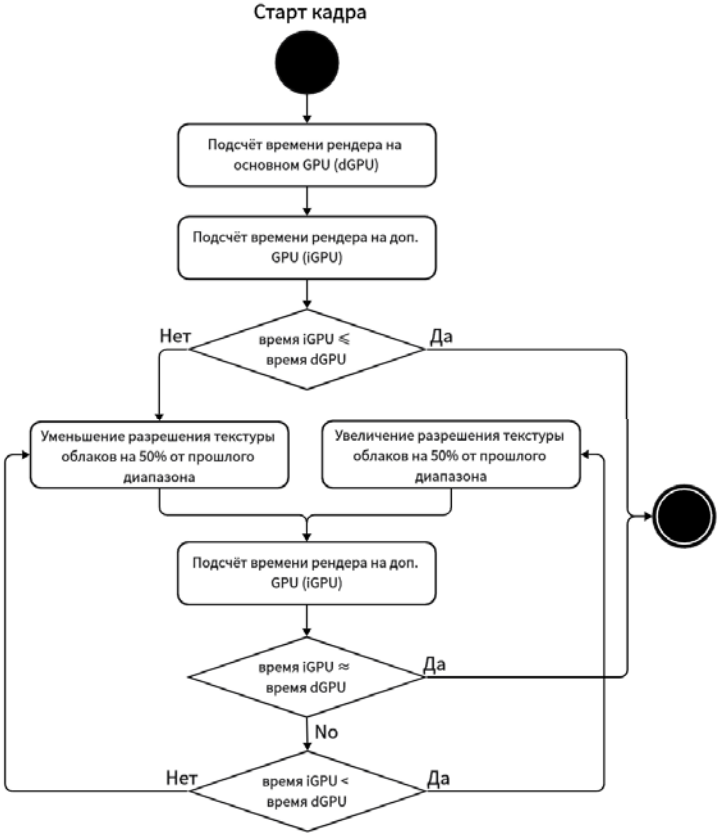

Рис. 5. Расчет коэффициента масштабирования.
Fig. 5. Scaling factor calculation.

Рис. 6. Результат работы гибридного алгоритма отображения облаков.
Fig. 6. result of the hybrid cloud rendering algorithm.

3.2 Гибридный алгоритм расчета теней

Для оценки эффективности раздельного вычисления графических техник, которые не имеют входных данных, но чьи результаты используются для финального построения сцены, были выбраны техники отображения теней. На основе анализа работы техник Shadow Maps [27-28], Cascaded Shadow Maps (CSM) [29] и были разработаны следующие методы разделения задач:

- Shadow Map рассчитывается полностью на дополнительном адаптере;
- Cascade Shadows полностью рассчитываются на дополнительном адаптере;
- Hybrid Cascaded Shadows (HCSM). Добавлен механизм дифференциации каскадов: ближние к камере каскады с высоким разрешением вычисляются на основном GPU, в то время как дальние каскады с меньшими требованиями к качеству выносятся на дополнительный адаптер.

Пример рассчитанных каскадов алгоритма HCSM показан на рис. 7.

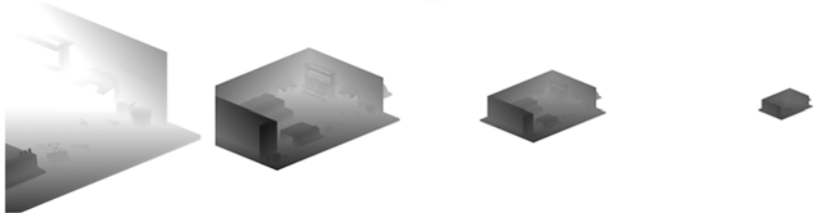

Рис. 7. Результат работы гибридного алгоритма Cascaded Shadow Maps.
Fig. 7. Result of the Cascaded Shadow Maps hybrid algorithm.

3.3 Гибридный алгоритм расчета окружающего затенения

Чтобы оценить эффективность раздельного вычисления графических техник, которым необходимы результаты других техник и чьи данные используются для финального построения кадра, были выбраны техники локального затенения (Ambient Occlusion). На основе анализа техник Screen Space Ambient Occlusion (SSAO) [30-31] и Horizon-Based Ambient Occlusion (HBAO) [32-33] был разработан гибридный метод:

- Основной GPU рассчитывает буфер глубины и буфер нормалей сцены;
- Эти данные передаются на дополнительный GPU;
- Вторичный GPU вычисляет карту окружающего затенения;
- Результат передается обратно на основной GPU для использования в проходе освещения.

Выбор SSAO и HBAO в качестве основы для реализации обусловлен их относительной автономностью, предсказуемым объемом данных и совместимостью со всеми тестируемыми комбинациями видеокарт.

Пример рассчитанного окружающего затенения показан на рис. 8.

3.4 Механизмы синхронизации и обмена данными

В ходе разработки предложенных ранее гибридных техник было выявлено, что ключевой проблемой в гибридных системах является обеспечение эффективной синхронизации данных между GPU, а также минимизация накладных расходов на обмен этими данными.

В рамках исследования были использованы различные механизмы взаимодействия:

1) Кросс-адаптерные ресурсы: Использование shared-ресурсов DirectX 12 для прямого доступа к данным на другом GPU без копирования через системную память в тех

гибридных системах, где аппаратное обеспечение предоставляет такую возможность [34];

2) Неблокирующая синхронизация: использование механизма аппаратной синхронизации между адаптерами, которая позволяет основному GPU продолжать работу до момента, когда ему понадобятся результаты вычислений с дополнительного адаптера;
3) Адаптивная частота обновления: для задач с переменной вычислительной сложностью (например, симуляция облаков и расчета теней) был реализован механизм динамического регулирования частоты обновления на основе производительности каждого GPU;
4) Кэширование результатов: для статичных или медленно изменяющихся элементов сцены реализовано кэширование результатов вычислений между кадрами для сокращения объема передаваемых данных.

Рис. 8. Результат работы гибридного алгоритма Ambient Occlusion.
Fig. 8. Result of the Ambient Occlusion hybrid algorithm.

4. Реализация и экспериментальные исследования

Для тестирования гибридных алгоритмов были использованы конфигурации оборудования, описанные в табл. 1.

Табл. 1. Конфигурации тестовых систем.
Table 1. Test system configurations.

	Компьютер 1	Компьютер 2	Компьютер 3	Компьютер 4	Ноутбук
CPU	Intel Core i9-10900K	Intel Core i9-9900K	Intel Core i5-8400	Intel Core i5-3470	Intel Core i7-8750H
1 GPU	NVIDIA GeForce RTX 2080 SUPER	NVIDIA GeForce GTX 1080	NVIDIA GeForce GTX 1060	NVIDIA GeForce GTX 660	NVIDIA GeForce GTX 1650 (Mobile)
2 GPU	Intel UHD Graphics 630	Intel UHD Graphics 630	Intel UHD Graphics 630	AMD Radeon RX 470	AMD Radeon Vega 8
RAM	32 ГБ DDR4	16 ГБ DDR4	16 ГБ DDR4	8 ГБ DDR3	8 ГБ DDR4

Тестовая сцена включала камеру, совершавшую вращение вокруг центра мира, набор статических объектов, систему частиц с симуляцией 20 000 полупрозрачных анимированных частиц, а также 15 источников света. Один из источников представлял собой направленный свет, имитирующий солнечное освещение. Общее количество отображаемых полигонов составляло порядка двух миллионов. Для всех источников света в каждом кадре вычислялись динамические тени. Для отображения окружающего пространства сцены иисполь зовался алгоритм skybox для которого в рамках гибридных реализаций алгоритма генерации облаков каждый кадр процедурно создавалась трехмерная текстура облаков, во всех других случаях использовалась статическая HDRI текстура окружения. Все эксперименты выполнялись в разрешении Full HD (1920×1080 пикселей) с применением последних стабильных версий драйверов, загруженных с официальных сайтов производителей и актуальных на момент проведения тестирования. В рамках одного запуска бенчмарка осуществлялся сбор статистики о производительности системы каждый кадр в течении 5 минут. Проводилось минимум по 5 запусков для прогревания кэшей драйверов и операционной системы, после чего еще 10 опорных запусков, по которым происходил анализ результатов.

На рис. 9 представлены столбчатые диаграммы, показывающие сравнение среднего времени кадра (Average Frame Time) при использовании одной (Single GPU) и двух (Multi-GPU) видеокарт для семи рассматриваемых техник рендеринга. CloudsV1 и CloudsV2 на диаграммах это алгоритмы отображения облаков до и после введения коэффициента масштабирования соответственно. Каждый график соответствует определенной системной конфигурации. Оси абсцисс соответствуют техники рендеринга. Оси ординат соответствует среднее время кадра, выраженное в относительных процентах, где значение Single GPU (синий столбец) принято за 100%. Внутри столбцов указаны абсолютные значения времени кадра; над красными столбцами приведен процентный прирост производительности, рассчитанный как снижение времени кадра при переходе с Single GPU на Multi-GPU.

Анализ полученных результатов позволяет сделать несколько важных выводов:

1) Эффективность гибридного подхода зависит от типа задачи. Наибольший прирост производительности был получен для моделирования облаков (до 59%), за которым следует расчет теней с использованием CSM (до 28%) и расчет окружающего затенения (до 11%). Такое распределение связано с различной вычислительной сложностью и соотношением объема входных/выходных данных.
2) Ключевым фактором является баланс производительности GPU. В системах с сильно различающейся производительностью адаптеров (например, высокопроизводительная dGPU и слабая iGPU) эффект от гибридного подхода снижается, поскольку дополнительный GPU становится бутылочным горлышком для выделенных ему задач.
3) Отношение вычислительной сложности к объему передаваемых данных является критическим для эффективности гибридного рендеринга. Задачи с высоким соотношением (как в случае с CSM на сложных сценах) демонстрируют лучшие результаты.
4) Гибридный подход также эффективно работает в системах, которые включают GPU от разных производителей. Это подтверждается результатами для пары NVIDIA/AMD (GTX 660 + RX 470).
5) Влияние разрешения рендеринга. При увеличении разрешения эффективность гибридного подхода снижается для всех исследованных алгоритмов, что связано с возрастающей нагрузкой на основной GPU для обработки остальных этапов конвейера.

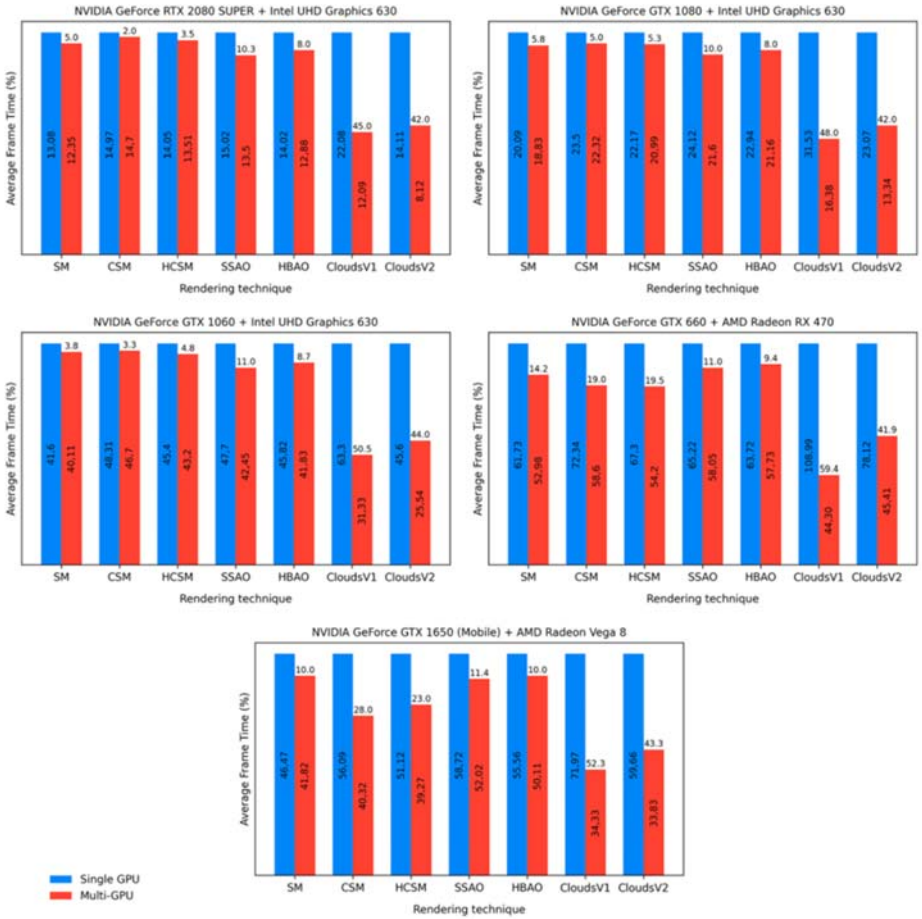

Рис. 9. Результаты тестов.
Fig. 9. Test results.

Сравнение с традиционными методами мульти-GPU рендеринга показывает преимущества нашего гибридного подхода в системах с несколькими видеокартами. В отличие от AFR, гибридный подход работает эффективнее с алгоритмами, в которых используются данные нескольких кадров. По сравнению с алгоритмами, основанными на SFR [13–15], гибридный подход требует меньше данных, как для построения кадра, так и для передачи между GPU, а также проще в реализации и не требует специализированных аппаратных интерфейсов, что позволяет им работать с любой конфигурацией GPU, которые поддерживают DirectX 12.

Среди ограничений подхода можно выделить зависимость от графического API DirectX 12, что снижает совместимость с некоторыми старыми аппаратными конфигурациями систем. Также трудность может представлять необходимость ручной модернизации и модификации графического конвейера, используемого в приложения, для создания, прототипирования и интеграции гибридных алгоритмов, а также сложность балансировки нагрузки в системах с существенно различающейся производительностью GPU.

5. Заключение и перспективы

В ходе исследования была разработана и протестирована архитектура гибридного рендеринга для систем с несколькими графическими процессорами. Предложенный нами подход основан на функциональном распределении задач между GPU с использованием современных возможностей DirectX.

Основные результаты работы:

1) Разработаны и реализованы гибридные алгоритмы для трех графических техник: моделирование облаков, расчет теней (Shadow Maps и CSM) и окружающее затенение (SSAO и HBAO).
2) Проведены экспериментальные исследования на различных аппаратных конфигурациях, включая как персональные компьютеры с парой GPU, так и переносные гетерогенные системы.
3) Показана и доказана эффективность гибридного подхода в системах с несколькими графическими адаптерами без необходимости использовать специализированных аппаратных интерфейсов связи между GPU.
4) Получен существенный прирост производительности для рассматриваемых техник, до 49% для расчета теней (CSM), до 60% для моделирования облаков и до 20% для окружающего затенения в зависимости от конфигурации системы.

В качестве направлений дальнейших наших исследований можно выделить следующие:

1) Исследование темпоральных методов для решения проблемы различающейся частоты симуляции между GPU.
2) Адаптация метода для волюметрических облаков и вокселей с высокой детализацией.
3) Дифференциация каскадов карт теней между GPU с учетом их производительности.
4) Интеграция гибридного подхода с другими алгоритмами, такими как процедурная симуляция растительности и расчеты водных поверхностей.
5) Масштабирование подхода на системы с тремя и более GPU.
6) Оптимизация для мобильных платформ и использование энергосберегающих алгоритмов.

Полученные результаты открывают новые возможности для разработки эффективных графических приложений, способных адаптироваться к наличию аппаратных ресурсов в системе и обеспечивать оптимальное качество визуализации при заданных ограничениях производительности.

Список литературы / References

[1]. Luebke D., Humphreys G. How GPUs work. Computer, 2007, vol. 40, no. 2, pp. 96-100. DOI: 10.1109/MC.2007.59.
[2]. NVIDIA. NVIDIA 2023 Annual Review. Available at: https://s201.q4cdn.com/141608511/files/doc_financials/2023/ar/2023-Annual-Report-1.pdf, accessed 21.08.2024.
[3]. Steam. Steam Hardware Stats. Available at: https://store.steampowered.com/hwsurvey, accessed 14.12.2024.
[4]. Li A., Song S. L., Chen J., Li J., Liu X., Tallent N., Barker K. Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect. IEEE Transactions on Parallel and Distributed Systems, 2020, vol. 31, no. 1, pp. 94-110. DOI: 10.1109/TPDS.2019.2928289.
[5]. Arunkumar A., Bolotin E., Cho C., Milic U., Villella M., Katibah G.M., Olukotun K., Nellans D. MCM-GPU: Multi-chip-module GPUs for continued performance scalability. Proceedings of the International Symposium on Computer Architecture, 2017, pp. 320-332. DOI: 10.1145/3079856.3080231.
[6]. Molnar S., Cox M., Ellsworth D., Fuchs H. A Sorting Classification of Parallel Rendering. 1994, vol. 14, no. 4, pp. 23-32. DOI: 10.1109/38.291528.

[7]. Mueller C. Sort-first rendering architecture for high-performance graphics. Proceedings of the Symposium on Interactive 3D Graphics, 1995, pp. 75-84. DOI: 10.1145/199404.199417.
[8]. Gao T., Luo X., Guo N. Multi-frame prediction load balancing algorithm for sortfirst parallel rendering. ACM International Conference Proceeding Series, 2017. DOI: 10.1145/3110224.3110240.
[9]. Ryder I.W. Microstuttering in AFR-based multi-GPU rendering. Available at: https://www.tomshardware.com/reviews/radeon-geforce-stutter-crossfire,2995-2.html, accessed 16.11.2024.
[10]. Yang C., Chen C., Hu X., Yang H. Dynamic Load Balancing Algorithm Based on Per-pixel Rendering Cost Estimation for Parallel Ray Tracing on PC Clusters. Communications in Computer and Information Science, 2019, vol. 1043, pp. 591-601. DOI: 10.1007/978-981-13-9917-6_56.
[11]. Barreales G. N., Novalbos M., Otaduy M.A., Sanchez A. MDScale: Scalable multi-GPU bonded and short-range molecular dynamics. Journal of Parallel and Distributed Computing, 2021, vol. 157, pp. 243-255. DOI: 10.1016/j.jpdc.2021.07.006.
[12]. Eilemann S., Makhinya M., Pajarola R. Equalizer: A scalable parallel rendering framework. IEEE Transactions on Visualization and Computer Graphics, 2009, vol. 15, no. 3, pp. 436-452. DOI: 10.1109/TVCG.2008.104.
[13]. Eilemann S., Steiner D., Pajarola R. Equalizer 2.0: convergence of a parallel rendering framework. IEEE Transactions on Visualization and Computer Graphics, 2020, vol. 26, no. 2, pp. 1292-1307. DOI: 10.1109/TVCG.2018.2870822.
[14]. Ren X., Lis M. CHOPIN: Scalable Graphics Rendering in Multi-GPU Systems via Parallel Image Composition. Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA), IEEE, 2021, pp. 709-722. DOI: 10.1109/HPCA51647.2021.00065.
[15]. Kim Y., Jo J.E., Jang H., Rhu M., Kim H., Kim J. GPUpd: A fast and scalable multi-GPU architecture using cooperative projection and distribution. Proceedings of the Annual International Symposium on Microarchitecture (MICRO), IEEE Computer Society, 2017, pp. 574-586. DOI: 10.1145/3123939.3123968.
[16]. PCI-SIG. PCI Express Base Specification Revision 5.0 Version 1.0. 2019, Beaverton, USA.
[17]. Milic U., Villa O., Bolotin E., Arunkumar A., Ebrahimi E., Jaleel A., Ramirez A., Nellans D. Beyond the socket: NUMA-aware GPUs. Proceedings of the Annual International Symposium on Microarchitecture (MICRO), 2017, pp. 123-135. DOI: 10.1145/3123939.3124534.
[18]. Wronski B. Temporal supersampling and antialiasing. Available at: https://bartwronski.com/2014/03/15/temporal-supersampling-and-antialiasing/, accessed 07.01.2025.
[19]. NVIDIA Corporation. Introduction to SLI Technology. Available at: https://www.geforce.com/whats-new/guides/introduction-to-sli-technology-guide#2, accessed 07.01.2025.
[20]. Advanced Micro Devices. AMD Crossfire Technology. Available at: https://www.amd.com/ru/technologies/crossfire, accessed 07.01.2025.
[21]. Microsoft Corporation. Multi-Adapter. Available at: https://devblogs.microsoft.com/directx/directx-12-multiadapter-lighting-up-dormant-silicon-and-making-it-work-for-you/, accessed 07.01.2025.
[22]. Derivative.ca. Using Multiple Graphic Cards. Available at: https://docs.derivative.ca/Using_Multiple_Graphic_Cards, accessed 07.01.2025.
[23]. Iyer K., Kiel J. GPU debugging and profiling with NVIDIA Parallel Nsight. Game Development Tools, 2016, pp. 303-324.
[24]. Suh J. W., Kim Y. Installing NVIDIA Nsight into Visual Studio. Accelerating Matlab with GPUs, 2014.
[25]. Chan A. B., Vasconcelos N. Layered dynamic textures. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009, vol. 31, no. 10, pp. 1862-1879. DOI: 10.1109/TPAMI.2009.110.
[26]. Wang W.C., Chen S.H., Xiang G. Layered textures for image-based rendering. Journal of Computer Science and Technology, 2004, vol. 19, no. 5, pp. 633-642. DOI: 10.1007/BF02945589.
[27]. Williams L. Casting curved shadows on curved surfaces. Computer Graphics (ACM), 1978, vol. 12, no. 3, pp. 270-274. DOI: 10.1145/965139.807402.
[28]. Valient M. Taking Killzone Shadow Fall Image Quality into the Next Generation. 2015, Available at: https://www.guerrilla-games.com/read/taking-killzone-shadow-fall-image-quality-into-the-next-generation-1, accessed 04.09.2025.
[29]. Liang X.H., Ma S., Cen L.-X., Yu Z. Light space cascaded shadow maps algorithm for real time rendering. Journal of Computer Science and Technology, 2011, vol. 26, no. 1, pp. 176-186. DOI: 10.1007/s11390-011-9424-7.
[30]. Bavoil L., Sainz M. Screen space ambient occlusion. NVIDIA Developer Information, 2008, September.

[31]. McGuire M., Osman B., Bukowski M., Hennessy P. The Alchemy screen-space ambient obscurance algorithm. Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, 2011, pp. 25-32. DOI: 10.1145/2018323.2018327.
[32]. Bavoil L., Sainz M., Dimitrov R. Image-space horizon-based ambient occlusion. Proceedings of the SIGGRAPH'08: ACM SIGGRAPH Talks, 2008, p. 1. DOI: 10.1145/1401032.1401061.
[33]. Song Y., Liu F., Xu J. Horizon-based screen-space ambient occlusion using mixture sampling. Proceedings of the ACM SIGGRAPH Asia Posters, 2010, p. 1. DOI: 10.1145/1900354.1900410.
[34]. DirectX 12 Cross Adapter Resources. Available at: https://learn.microsoft.com/en-us/windows-hardware/drivers/display/supporting-caso, accessed 04.09.2025.

Информация об авторах / Information about authors

Михаил Константинович БОГДАНОВ – аспирант факультета прикладной информатики в Университете ИТМО. Научные интересы включают методы визуализации в реальном времени, разработку и анализ графических алгоритмов, параллельные вычисления на графических процессорах, а также масштабирование рендеринга в многопроцессорных графических системах.

Mikhail Konstantinovich BOGDANOV – postgraduate student at the Faculty of Applied Informatics, ITMO University. Research interests include real-time rendering, development and analysis of graphics algorithms, parallel computing on GPUs, and scaling rendering techniques in multi-GPU systems.

Андрей Максимович СУВОРОВ – студент бакалавриата факультета "Школа разработки видеоигр" в Университете ИТМО. Научные интересы включают физически корректный рендеринг, процедурную генерацию графического контента, а также исследование и реализацию графических систем на базе низкоуровневых API.

Andrey Maximovich SUVOROV – bachelor's student of the School of Game Development, ITMO University. Research interests include physically based rendering, procedural graphics content generation, and research and implementation of graphics systems based on low-level APIs.

Мария Евгеньевна ИВАШЕЧКИНА – студент магистратуры факультета "Школа разработки видеоигр" в Университете ИТМО. Научные интересы связаны с компьютерной графикой, включая разработку и оптимизацию алгоритмов визуализации, процедурную генерацию контента, а также методы повышения производительности интерактивных систем.

Maria Evgenievna IVASHECHKINA – master's student of the School of Game Development, ITMO University. Research interests include computer graphics, development and optimization of rendering algorithms, procedural content generation, and performance improvement methods for interactive systems.

Равиль Радикович СУЛТАНОВ – студент магистратуры факультета "Школа разработки видеоигр" в Университете ИТМО. Научные интересы включают высокопроизводительные вычисления, параллельные алгоритмы обработки данных, а также оптимизацию вычислительных процессов в системах реального времени и графических приложениях.

Ravil Radikovich SULTANOV – master's student of the School of Game Development, ITMO University. Research interests include high-performance computing, parallel data processing algorithms, and optimization of computational processes in real-time systems and graphics applications.

Глеб Сергеевич ЛЯХ – студент магистратуры факультета "Школа разработки видеоигр" в Университете ИТМО. Научные интересы включают разработку игровых движков, проектирование архитектуры программных систем, а также методы оптимизации и эффективного управления вычислительными ресурсами.

Gleb Sergeevich LYAKH – master's student of the School of Game Development, ITMO University. Research interests include development of game engines, software architecture design, and optimization methods and efficient resource management in computational systems.

Андрей Петрович БУЛАЕВ – разработчик, графический инженер. Научные интересы включают разработку графических движков, физически корректный рендеринг, оптимизацию алгоритмов на графических процессорах, а также построение высокопроизводительных систем визуализации.

Andrey Petrovich BULAEV – software developer, graphics engineer. Research interests include graphics engine development, physically based rendering, GPU algorithm optimization, and development of high-performance visualization systems.