УДК 004.925
DOI: 10.25686/978-5-8158-2474-4-2025-172-182

# Гибридный метод расчета окружающего затенения для систем с несколькими видеоадаптерами

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

***Аннотация.*** Параллельный рендеринг – это популярный подход, используемый в компьютерной графике для повышения производительности системы за счет использования связки видеокарт. Однако большинство современных реализаций полагается на серверные версии видеокарт, у которых есть аппаратная поддержка связи по скоростной шине передачи данных, а также полностью игнорирует рынок игровых устройств, таких как ноутбуки и десктопные компьютеры, в которых есть дискретная видеокарта и интегрированный в процессор видеочип. Мы решили разработать механизм разделения графических задач не на основе данных, а на основе техник, выполняемых на основной видеокарте; это позволит нам использовать DirectX 12 в режиме EMA и не полагаться на аппаратные интерфейсы, а также избавиться от привязки к производителю видеоадаптеров. Как идеальный кандидат для оценки эффективности такого разделения нами был выбран алгоритм расчета окружающего затенения. Наш алгоритм использует дополнительную видеокарту для вычисления карты затенения, тем самым эффективно применяя все доступные в системе аппаратные ресурсы, сохраняя при этом качество и точность рассчитываемых данных. Такой подход легко интегрируется в существующие конвейеры рендеринга и позволяет без особых затрат увеличить производительность рендеринга на системах с несколькими GPU.
***Ключевые слова*:** Ambient Occlusion, SSAO, DirectX 12, Multi-GPU, компьютерная графика, HBAO

# Hybrid ambient occlusion method for systems with multiple video adapters

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

***Abstract.*** Parallel rendering is a popular approach used in computer graphics to enhance system performance by utilizing multiple graphics cards. However, most of all these approaches rely on server-grade GPUs, which feature hardware support for high-speed data bus communication, and completely ignore the consumer device market, such as laptops and desktop systems that include both a discrete GPU and an integrated graphics chip within the processor. We have developed a mechanism for distributing graphics tasks not based on data, but rather on techniques executed on the primary GPU. This allows us to use DirectX 12 in Explicit Multiadapter (EMA) mode, avoiding dependence on specific hardware interfaces and eliminating vendor lock-in for graphics adapters. As an ideal candidate for evaluating the efficiency of such distribution, we selected the ambient occlusion calculation algorithm, which takes into account multiple GPUs in the system. Our algorithm utilizes an additional GPU to compute the shading map, thereby efficiently leveraging all available hardware resources in the system while maintaining the quality and accuracy of the computed data. This approach can be easily integrated into existing rendering pipelines and enables a cost-effective performance boost for rendering on systems with multiple GPU.
***Keywords:*** Ambient Occlusion, SSAO, DirectX 12, Multi-GPU, computer graphics, HBAO

## Введение

Графические процессоры (GPU) разрабатывались для ускорения обработки графики – процесса генерации 2D-изображений из 3D-моделей [1]. Несмотря на развитие видеокарт и исследование методов по их использованию для универсальных вычислений, высокопроизводительная обработка графики все так же составляет одну из основных долей спроса на GPU.

Более того, графика остается доминирующим источником дохода для поставщиков GPU: например, доходы от рынка графики для NVIDIA за 2024 год принесли примерно 44 % от суммарной годовой прибыли и исчисляются миллиардами долларов [2]. Это обусловлено тем, что многим приложениям, включая игры, научную визуализацию данных, компьютерное проектирование, виртуальную реальность (VR), дополненную реальность (AR) и так далее, необходимы все более производительные графические адаптеры. Сама игровая индустрия продолжает развиваться: современные игры в 4K и VR требуют производительности в 4x и 7x больше, чем игры в формате 1080p HD соответственно, в то время как сами игры содержат миллионы или миллиарды треугольников, часто меньших, чем пиксель [3].

Постоянные нововведения увеличивают потребность в повышении графической производительности, однако ее становится все труднее удовлетворить с помощью традиционных одиночных графических

адаптеров. Чтобы продолжить масштабирование производительности GPU, поставщики GPU недавно создали продвинутые аппаратные механизмы передачи данных и синхронизации видеокарт [4, 5], на использование которых полагаются различные распределенные архитектуры, такие как Multi-Chip-Module GPU (MCM-GPU) MCM [6].

В теории эти платформы могут предложить значительные возможности для повышения производительности; на практике же их решения касательно производительности для обработки графики отличаются от однокристальных GPU, и полная реализация преимуществ требует использования распределенных алгоритмов рендеринга.

Параллельный рендеринг не является чем-то новым; разработчики видеокарт уже давно объединяют от двух до четырех видеоадаптеров с помощью аппаратных интерфейсов, таких как SLI и Crossfire [5, 7, 8]. Благодаря этим интерфейсам можно распределять работу, необходимую для отображения кадра либо с помощью альтернативного рендеринга (AFR), где разные связанные видеокарты последовательно обрабатывают кадры, либо с помощью разделенного рендеринга (SFR), который делит отображаемый кадр на части, а обработка каждой непересекающейся области осуществляется отдельной видеокартой. В AFR каждая видеокарта в системе обрабатывает кадры независимо, что увеличивает частоту отображения кадров в системе, но это справедливо только для двух идентичных видеокарт: при подключении двух и более видеокарт с разной производительностью возникает проблема бутылочного горлышка и суммарная производительность системы ограничивается самой «слабой» видеокартой [9]. В свою очередь, SFR может улучшить как частоту кадров, так и уменьшить рендер одиночного кадра [10–13]. Из-за этого SFR чаще используется на практике, и, как следствие, существует огромное количество различных модификаций данного подхода к рендерингу с несколькими видеокартами.

Компромисс состоит в том, что SFR требует, чтобы GPU обменивались как данными, необходимыми для создания кадра, так и результатами расчетов, что создает значительные проблемы пропускной способности и задержки из-за синхронизации [5]. Хотя недавнее появление высокопроизводительных соединений, таких как NVLink и XGMI, направлено на уменьшение ограничений пропускной способности между различными GPU, определенные аппаратные ограничения все так же остаются.

К сожалению, все популярные и актуальные алгоритмы рендеринга на нескольких видеокартах акцентируются только на системы, в которых есть аппаратная возможность соединить между собой графические адаптеры. Такие подходы не допускают возможность использования видеокарт разных производителей и к тому же полностью игнорируют состав отображаемого кадра с точки зрения используемых графических техник.

Эти ограничения полностью исключают возможность использования существующих подходов к параллельному рендерингу на самых распространённых игровых платформах, таких как ноутбуки и персональные компьютеры, в которых есть одна дискретная видеокарта и интегрированное в процессор видео ядро [14].

В данной статье мы предлагаем новый гибридный метод расчета окружающего затенения, который использует в своей работе несколько видеокарт, вне зависимости от их производителя. Для этого мы используем разделение отрисовки не по данным, а по задачам, которые выполняются на каждой видеокарте. Такой подход позволил снизить время построения кадра, а также уменьшить требования к видеопамяти, необходимой для построения кадра по сравнению с актуальными алгоритмами.

В рамках работы были выполнены следующие задачи:

- рассмотрены основные популярные алгоритмы рендеринга с использованием нескольких видеокарт и выявлены их эксплуатационные затраты и ограничения, которые мешают использовать данные подходы на распространённых гибридных игровых конфигурациях;
- на основе анализа графа построения кадра в современных игровых движках и играх предложена гибридная реализация техники окружающего затенения, которую потенциально можно эффективно перенести для параллельного вычисления;
- разработан графический бенчмарк и протестированы разные реализации алгоритмов окружающего затенения на разных аппаратных платформах с учетом разделения вычислений на разные графические адаптеры.

Исходя из анализа полученных тестовых данных, разработанный нами гибридный подход к расчету локального затенения, позволяет уменьшить время построения кадра до 20 % в зависимости от конфигурации системы. Он избавлен от необходимости дублирования всех данных между устройствами и не привязан к одному производителю аппаратного обеспечения, а также его можно интегрировать в существующие игровые и графические решения реального времени.

**Предпосылки и мотивации**

*A. Ограничения существующих подходов к параллельному рендерингу*

Для удовлетворения растущих требований к качеству графики и суммарной производительности системы разрабатываются различные техники параллельного рендеринга. Большинство из них [6, 10–12, 15–33] разработаны для серверного применения с использованием нескольких однородных GPU, соединенных между собой через высокоскоростные аппаратные интерфейсы [4, 5, 7, 8, 34–36], что позволяет эффективно распределять нагрузку и синхронизировать данные. Однако эти подходы не применимы к популярным пользовательским конфигурациям, таким как ноутбуки или настольные ПК, в которых комбинируется дискретная (dGPU) и интегрированная (iGPU) видеокарта.

Между серверными и пользовательскими системами существуют принципиальные различия:

1. Неоднородность производительности, выражаемая в существенной разнице между более мощной дискретной видеокартой и значительно менее мощной интегрированной, что делает все традиционные схемы и методы балансировки нагрузки неэффективными;

2. Ограниченная пропускная способность, связанная с тем, что вместо высокоскоростных шин по типу NVLink и XGMI, соединение между dGPU и iGPU осуществляется через PCIe. Из-за этого мы имеем значительно меньшую пропускную способность и большую задержку;

3. Отсутствие различных специализированных аппаратных средств синхронизации и общения между устройствами из-за ограничений на уровне драйверов и операционной системы.

Все эти ограничения заставляют взглянуть на гибридные пользовательские системы под другим углом с целью разработки алгоритмов, которые бы нивелировали все описанные ограничения. Для этого мы решили рассмотреть возможность реализации параллельного алгоритма, абстрагируясь от разделения расчетов по данным, а сконцентрировавшись на разделении вычислений между адаптерами на основе выполняемых ими задач.

*B. Анализ рендер-графов и выбор стратегии*

Для того чтобы определить наиболее подходящую задачу для распределения между графическими адаптерами, мы провели анализ рендер графов современных игр и игровых движков [37–41] с использованием инструментов профилирования, таких как NVIDIA Nsight [42, 43].

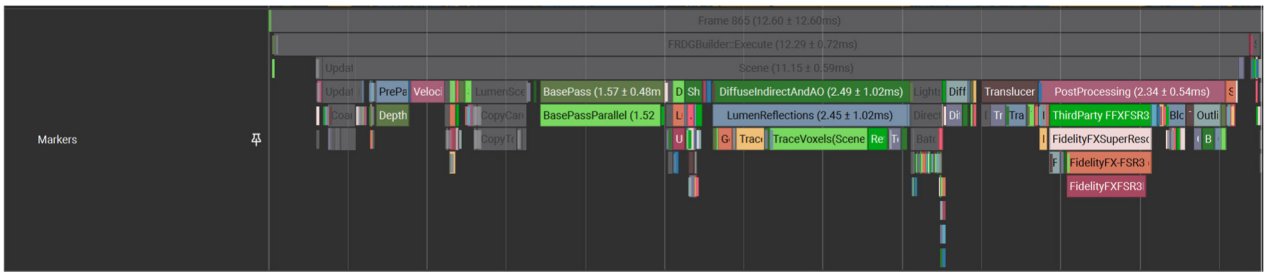


Рисунок 1. Nvidia Nsight профайлер

Исходя из проведенного анализа приложений и игр, мы выявили, что этап локального затенения (Ambient Occlusion, AO) – идеальный кандидат для переноса расчетов на дополнительную видеокарту.

Расчет реалистичного глобального освещения является сложной задачей в области компьютерной графики. Эта техника моделирует количество фонового освещения, достигшего точки на поверхности видимого объекта в зависимости от ее доступности [44], т.е. доли направлений, с которых свет может достичь этой точки, помогает пользователю лучше воспринять геометрию сцены. Такая аппроксимация не является точной с точки зрения физики, однако вычисления проводятся сравнительно быстро и дают убедительный результат на практике, что и привело к широкому распространению этого подхода [45].

В большинстве реализаций для реального времени применяют локальное затенение в экранном пространстве [46], которое использует буфер глубины и дополнительные данные в качестве исходных. Подобное решение снижает вычислительную сложность и увеличивает производительность, однако вычисления осуществляются на одном графическом адаптере, что накладывает ограничение на другие этапы в конвейере отрисовки, поскольку существует максимальный предел во времени построения кадра для систем реального времени.

Рисунок 2. Пример влияния окружающего затенения на поверхность

Результаты расчета AO представляют собой текстуру, размер которой заранее известен и пропорционален размеру экрана, что значительно меньше объёмов данных, необходимых для отрисовки сцены. AO может быть рассчитано с небольшой задержкой относительно основного конвейера отрисовки, выполняемого на основной видеокарте, что позволяет смягчить проблемы, связанные с задержкой коммуникаций между видеоадаптерами.

Популярные подходы к параллельному рендерингу, такие как SFR, распределяют пространственные области кадра между GPU, что требует сложной синхронизации и приводит к избыточным вычислениям [15, 16]. На основе проведенного анализа мы разработали алгоритм, который эффективно использует комбинацию dGPU и iGPU в популярных пользовательских системах. В отличие от существующих решений, которые стремятся распределить всю нагрузку рендеринга между адаптерами, наш подход фокусируется на выносе конкретной, вычислительно емкой задачи – расчета Ambient Occlusion – на дополнительную видеокарту.

**Выбор техники AO**

Алгоритм построения локального затенения довольно прост: для каждого пикселя, отображаемого пользователю, рассчитывается коэффициент затенения (occlusion factor) на основе значений из буфера глубины.

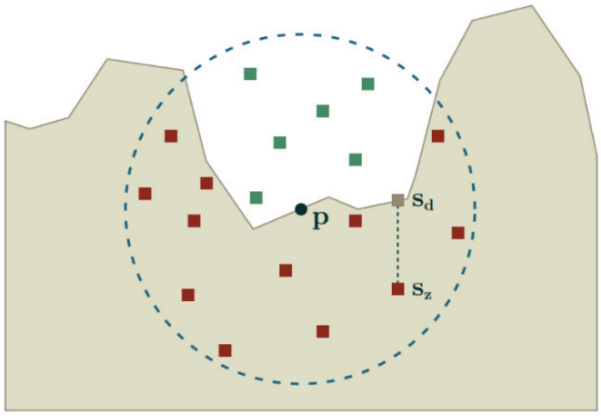


Рисунок 3. AO на основе буфера глубины

Вычисленный коэффициент затенения используется для уменьшения интенсивности фонового освещения, вплоть до полного его исключения. Для расчета этого коэффициента требуется сбор данных о глубине от множества выборок из сферической области, окружающей рассматриваемый фрагмент, и сравнения полученных значений глубины с глубиной рассматриваемого фрагмента [46, 47]. Число выборок, у которых значение глубины больше, нежели глубина непосредственно текущего фрагмента, определяют коэффициент затенения. Пример можно наблюдать на рисунке 3.

На нем изображено следующее: $p$ – затеняемая точка, $s_z$ – координата сэмпла, а $s_d = depth(s_x, s_y)$ – значение буфера глубины в соответствующем пикселе, где $s_x, s_y$ координаты пикселя. Здесь также легко заметить, что каждая красная точка лежит внутри некоторого геометрического объекта, а потому осуществляет вклад в значение коэффициента затенения. Чем больше выборок окажется внутри геометрии окружающих объектов, тем меньше будет остаточная интенсивность фонового затенения в этой области.

При разработке нашего подхода к распределённому рендерингу между двумя видеоадаптерами мы тщательно оценили различные алгоритмы Ambient Occlusion (AO) и выбрали именно Screen Space Ambient Occlusion (SSAO) [46] и Horizon-Based Ambient Occlusion (HBAO) [48–51] для реализации, несмотря на существование более современных вариантов, таких как Ground-Truth AO (GTAO) [52–54] и Ray-Traced AO (RTAO) [55, 56].

Наш выбор обусловлен несколькими причинами. Во-первых, эффективное распределение вычислений между GPU критически зависит от баланса нагрузки и соответствия вычислительных требований возможностям каждого устройства: интегрированные видеокарты, особенно в ноутбуках, имеют значительно меньшую вычислительную мощность по сравнению с дискретными. Во-вторых, для нашего подхода критически важен объём передаваемых и обновляемых каждый кадр данных на каждом устройстве, что накладывает ограничения на количество пересылаемых данных, а HBAO+ и GTAO используют дополнительные структуры данных, которые требуют расчетов и пересылки на дополнительную видеокарту [48, 49]. RTAO также исключается, поскольку для этой техники необходимо обновлять геометрию сцены в случае использования динамических объектов, которые распространены в современных компьютерных играх. В-третьих, учет аппаратной совместимости графических техник. В-четвертых, наш фокус направлен на принципиальную возможность осуществления и эффективности предложенного решения, поэтому выбор более простых и самодостаточных алгоритмов позволит оценить именно накладные расходы предложенного нами способа разделения, а не оценивать сложность самого алгоритма.

Именно поэтому SSAO и HBAO, с их понятной и легко реализуемой математикой, являются идеальными кандидатами для демонстрации разработанного нами механизма разделения вычислений в системах с несколькими видеокартами. Оптимизация и внедрение более сложных и продвинутых алгоритмов является следующим шагом по развитию предложенного нами способа взаимодействия видеокарт.

**Разделение вычислений AO**

С появлением DirectX 12 Microsoft представила три режима для взаимодействия с видеокартами в системе. В самом простом случае используется упомянутая выше технология AFR с видеокартами одного производителя – AMD или NVIDIA [5, 7, 8, 57].

Но, как мы выяснили из обзора существующих подходов, данный режим ограничивает варианты использования видеокарт в системе, хоть и уменьшает этим вероятность потенциального возникновения ошибок. В этом режиме большая часть работы для поддержки выполняется на уровне драйвера, а не на уровне DirectX. Однако DirectX 12 обеспечивает более расширенный доступ к аппаратной части системы. Для этого существует режим Explicit Multi-Adapter (EMA) [57]. Для каждого одиночного GPU определяется доступ к памяти, описывается взаимодействие GPU между собой – вся эта поддержка должна быть запрограммирована заранее. В EMA доступны два разных варианта использования: Linked Mode и Unlinked Mode.

Режим Linked Mode можно рассматривать как форму SLI или Crossfire под DirectX 12: аппаратные ресурсы в Linked Mode комбинируются в один большой виртуальный адаптер с общей памятью и

пользователь работает с ним как с единой видеокартой в системе. Однако комбинация оборудования в данном режиме ограничена из-за аппаратных особенностей, например, оба физических видеоадаптера должны быть от производителя NVIDIA.

В режиме Unlinked Mode становится возможно комбинировать видеокарты в том числе и от разных производителей. Также появляется возможность комбинирования дискретного и интегрированного GPU. В этом режиме каждый графический процессор считается самостоятельной аппаратной единицей со своими аппаратными ресурсами. Демонстрацию того, как работает данный режим можно наблюдать на рисунке 4.

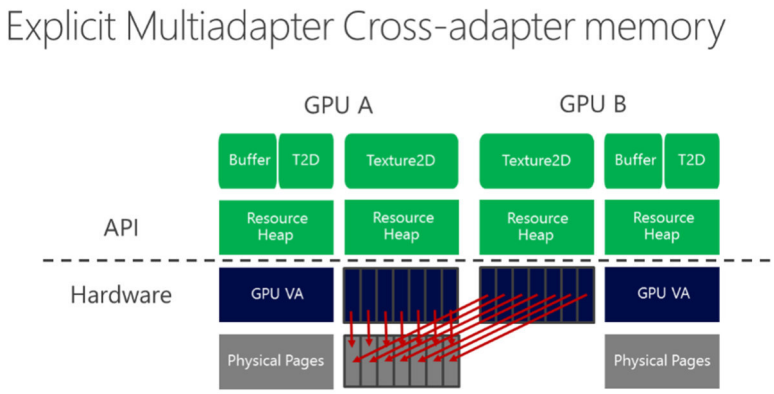


Рисунок 4. Механизм работы разделенных между адаптерами ресурсов [57]

В этом режиме EMA позволяет обмениваться данными между видеокартами, причем на глубоком уровне, а не просто готовыми кадрами, что является ключевой особенностью, которую не используют современные AFR алгоритмы [6, 15, 16, 28, 34, 58]. Обмен возможен как частично просчитанными кадрами, так и данными в буферах, что позволяет разрабатывать новые механизмы взаимодействия на нескольких GPU. Но присутствует также дополнительное ограничение в виде использования PCI Express шины, обмен данными через которую намного медленнее, нежели с применением аппаратных интерфейсов [5]. Пример того, как осуществляется взаимодействие между видеокартами, представлен на рисунке 5.

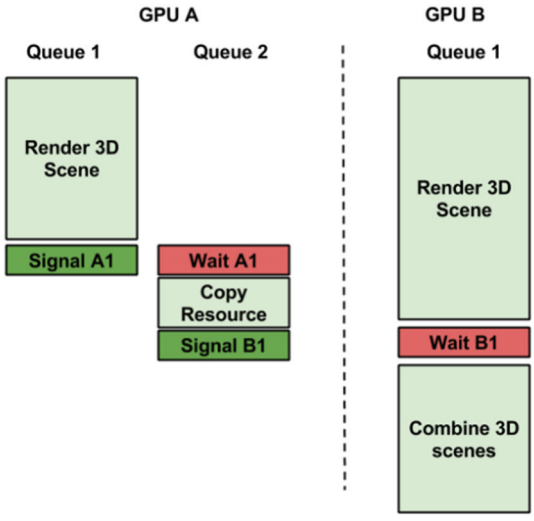


Рисунок 5. Механизм взаимодействия видеокарт в режиме EMA

Для реализации гибридного расчета AO мы разработали графический бенчмарк и изменили конвейер рендеринга, а именно разделили расчет локального затенения на две составляющие: подготовка данных для алгоритма и непосредственно сам алгоритм расчета затенения. Подготовкой

данных будет заниматься основная видеокарта в системе, так как эти данные необходимы будут не только AO, но и другим этапам в конвейере. Общую схему иллюстрирует рисунок 6.


Рисунок 6. Диаграмма выполнения вычислений между видеоадаптерами в рамках одного кадра

После предварительного расчета глубины сцены и нормалей осуществляется копирование полученных данных в общую для устройств память. Перед моментом расчета освещения добавляется копирование рассчитанной текстуры с данными о локальном затенении из общей памяти видеокарт, а затем продолжается вычисление кадра. Параллельно этому дополнительная видеокарта копирует в свою память данные для расчета локального затенения, считает окклюзию и копирует результат обратно в общую память.

**Сравнительный анализ**

Нагрузочная сцена состоит из вращающейся камеры, направленной в центр сцены, на которой происходит симуляция 200000 частиц и расположены статические объекты – их суммарное количество вертексов составляет примерно 2 миллиона. На сцене присутствует пять источников освещения, включая направленный источник освещения, эмулирующий солнце. Также каждый кадр пересчитывается карта теней.

Для оценки эффективности разработанного алгоритма мы производили тестирование на следующих системах: Intel Core I7 3770 со связкой Nvidia GTX 660 и AMD RX 470, а также Intel Core i7 9900K с интегрированным видеоядром Intel UHD Graphics 630 в связке с Nvidia GTX 1050Ti. Все тесты производились на системе Windows 10 в разрешении Full HD (1920x1080). Каждый запуск первые 30 секунд происходило прогревание кэшей, после чего в течение 120 секунд каждый кадр собиралась

статистика по времени построения кадра на каждом из адаптеров. На изображениях с 7 по 10 отображено время построения кадра, где по вертикальной оси указываются миллисекунды, рассчитанные как средние для 10 запусков бенчмарка, а по горизонтальной – время.

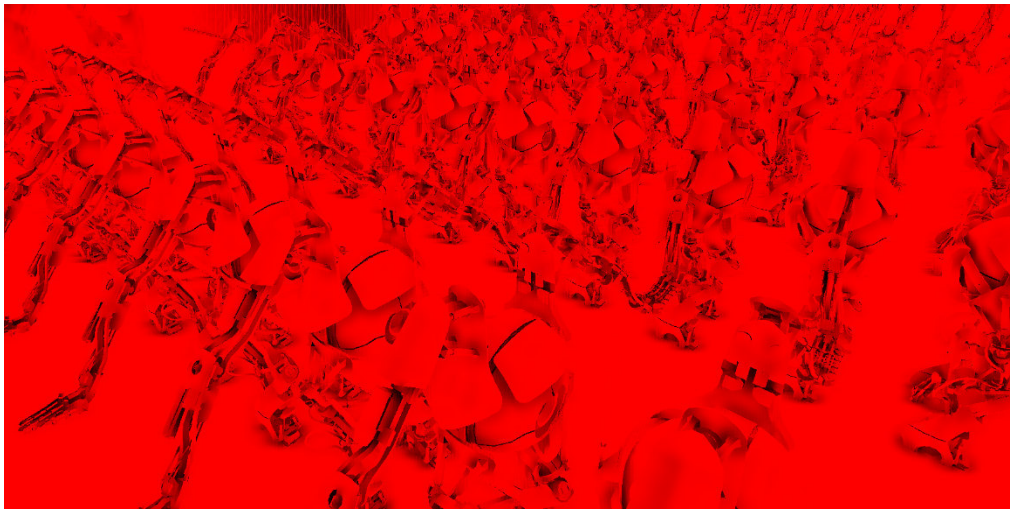

Рисунок 7. Демонстрация рассчитанного AO

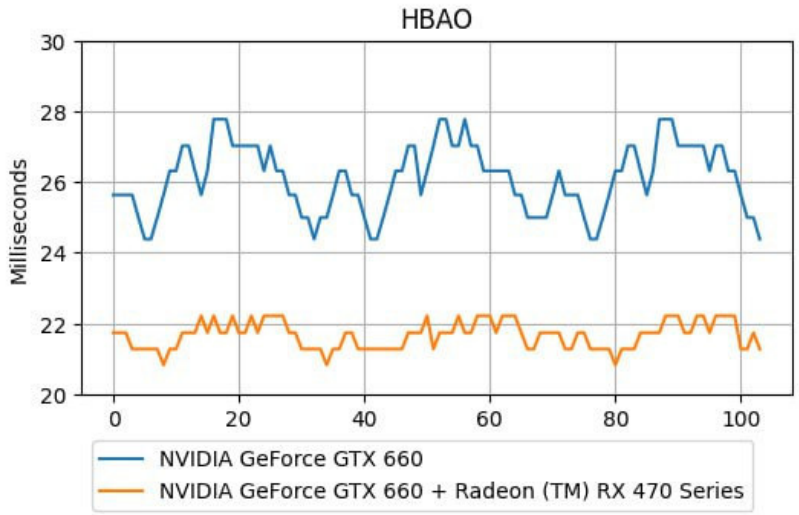


Рисунок 8. Рендер кадра при Native и Shared реализации HBAO на NVIDIA GTX 660 + AMD RX 470

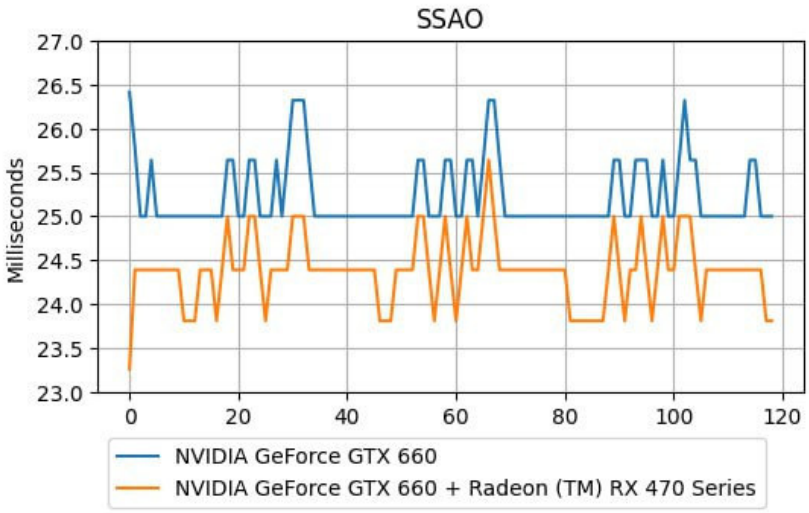


Рисунок 9. Рендер кадра при Native и Shared реализации SSAO на NVIDIA GTX 660 + AMD RX 470

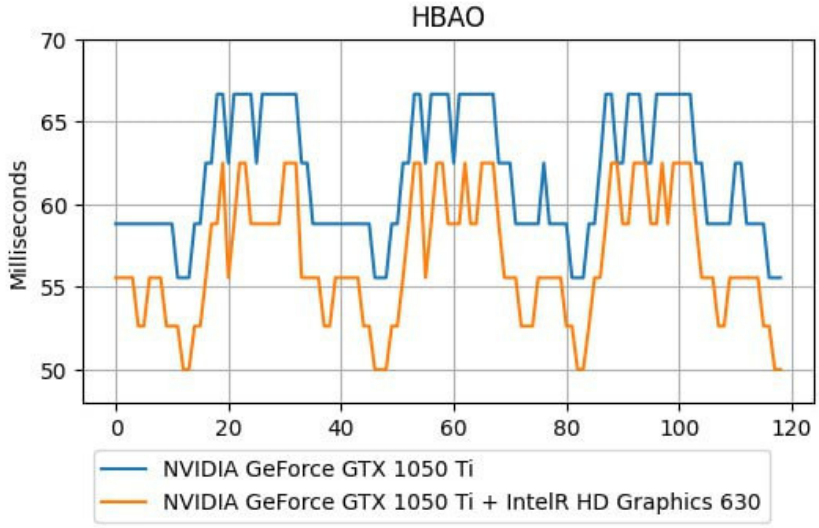


Рисунок 10. Рендер кадра при Native и Shared реализации HBAO на NVIDIA GTX 1050 Ti + Intel HD 630

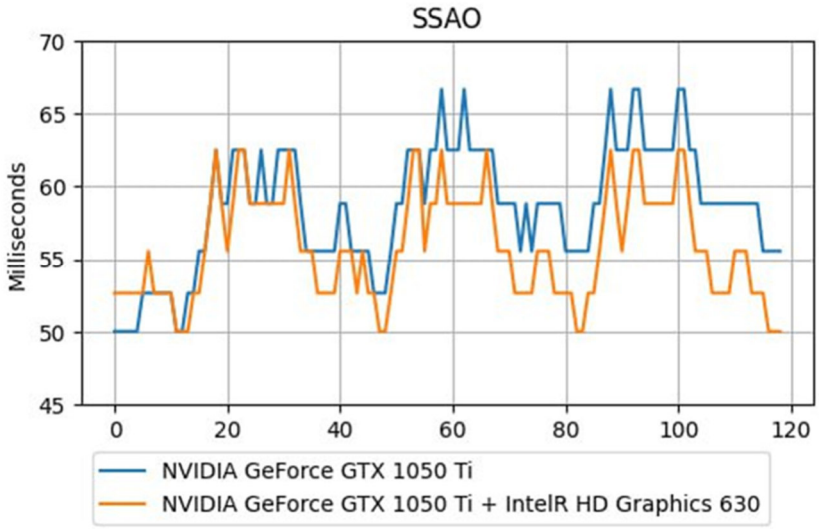


Рисунок 11. Рендер кадра при Native и Shared реализации SSAO на NVIDIA GTX 1050 Ti + Intel HD 630

Из анализа полученных данных видно, что гибридные реализации, вне зависимости от используемых устройств, показывают прирост производительности в разделенном режиме. В среднем копирование между адаптерами в зависимости от конфигурации занимало от 0.3 до 1 миллисекунды, что существенно быстрее, чем расчет локального затенения на основной видеокарте. Для сравнения: на конфигурации Nvidia GTX 1050 Ti и Intel HD Graphics 630 основная видеокарта 1050 Ti рассчитывала карту локального затенения в среднем 3 мс, что в 3 раза медленнее копирования данных между адаптерами. Для оценки загруженности нами были предприняты попытки использования сторонних профилировщиков, таких как Nvidia Nsight и AMD GPU Profiler, но указанные профилировщики не поддерживают работу с приложениями DirectX 12 в режиме EMA с включенным Unlinked Mode. Загруженность определялась по встроенному в Windows профилировщику видеокарт.

Согласно нашим наблюдениям, производительность разработанного нами подхода к разделению зависит от используемой пары видеоадаптеров. Так, при включении разделённого режима загруженность видеокарт в связке Nvidia GTX 1050 Ti и Intel HD Graphics 630 была 83 и 95 % соответственно, согласно встроенному в Windows профилировщику. Однако при использовании связки Nvidia GTX 660 и AMD RX 470 загруженность видеокарт в разделенном режиме была 80 и 60 % соответственно. Разница в загруженности показывает, что эффективность разделения техники зависит не только от выбранного алгоритма, но и от разницы в производительности связки видеокарт, что необходимо учитывать в дальнейших исследованиях.

В ходе дальнейших исследований мы планируем оценить эффективность переноса других популярных графических техник и возможность комбинированного запуска выявленных техник на дополнительном видеоадаптере в гибридных системах.

**Выводы**

Наша гибридная реализация окружающего затенения с копированием данных между устройствами позволяет сократить время рендеринга приблизительно на 20 % по сравнению с использованием одной GPU. Гибридный подход может улучшить пользовательский опыт из-за большей частоты кадров, а также позволить использовать усложненные вычислительные техники из-за сэкономленного времени при построении кадра на основной видеокарте.

**Список литературы**

1. Luebke D., Humphreys G. How GPUs work // Computer (Long Beach Calif). 2007. Vol. 40, no. 2.

2. NVIDIA 2023 Annual Review. URL: https://s201.q4cdn.com/141608511/files/doc_financials/2023/ar/2023-Annual-Report-1.pdf (accessed: 21.08.2024).

3. Nanite as a Disruptive Technology for the Interactive Visualisation of Cultural Heritage 3D Models: A Case Study / Díaz-Alemán M.D., et al. // Heritage. 2023. Vol. 6, no. 8.

4. Scaling Deep Learning workloads: NVIDIA DGX-1/Pascal and Intel Knights Landing / Gawande N.A., et al. // Future Generation Computer Systems. 2020. Vol. 108.

5. Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect / Li A., et al. // IEEE Transactions on Parallel and Distributed Systems. 2020. Vol. 31, no. 1.

6. MCM-GPU: Multi-chip-module GPUs for continued performance scalability / Arunkumar A., et al. // Proceedings - International Symposium on Computer Architecture. 2017. Vol. Part F128643.

7. NVIDIA Corporation. Introduction to SLI Technology. 2017. URL: https://www.geforce.com/whats-new/guides/introduction-to-sli-technology-guide#2 (accessed: 07.01.2025).

8. Advanced Micro Devices. AMD Crossfire Technology. 2017. URL: https://www.amd.com/ru/technologies/crossfire (accessed: 07.01.2025).

9. How to Fix Stuttering of Crossfire? URL: https://www.reddit.com/r/crossfire/comments/ddlcl9/how_to_fix_stuttering_of_crossfire/ (accessed: 23.03.2024).

10. Eilemann S., Makhinya M., Pajarola R. Equalizer: A scalable parallel rendering framework // IEEE Trans Vis Comput Graph. 2009. Vol. 15, no. 3.

11. Eilemann S., Steiner D., Pajarola R. Equalizer 2.0-convergence of a parallel rendering framework // IEEE Trans Vis Comput Graph. 2020. Vol. 26, no. 2.

12. Chromium: A stream-processing framework for interactive rendering on clusters / Humphreys G., et al. // Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02. 2002.

13. Mueller C. Sort-first rendering architecture for high-performance graphics // Proceedings of the Symposium on Interactive 3D Graphics. 1995.

14. Steam Hardware Stats. URL: https://store.steampowered.com/hwsurvey (accessed: 14.12.2024).

15. Ren X., Lis M. CHOPIN: Scalable Graphics Rendering in Multi-GPU Systems via Parallel Image Composition // 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2021. P. 709–722.

16. GPUpd: A fast and scalable multi-GPU architecture using cooperative projection and distribution / Kim Y., et al. // Proceedings of the Annual International Symposium on Microarchitecture, MICRO. IEEE Computer Society, 2017. Part F131207. P. 574–586.

17. Eilemann S., Steiner D., Pajarola R. Equalizer 2.0-convergence of a parallel rendering framework // IEEE Trans Vis Comput Graph. IEEE Computer Society, 2020. Vol. 26, no. 2. P. 1292–1307.

18. Molnar S., Eyles J., Poulton J. PixelFlow: high-speed rendering using image composition // Computer Graphics (ACM). 1992. Vol. 26, no. 2.

19. Moll L., Heirich A., Shand M. Sepia: Scalable 3D compositing using PCI Pamette // IEEE Symposium on FPGAs for Custom Computing Machines, Proceedings. 1999.

20. Laine S., Karras T. High-performance software rasterization on GPUs // Proceedings - HPG 2011: ACM SIGGRAPH Symposium on High Performance Graphics. 2011.

21. WireGL: A scalable graphics system for clusters / Humphreys G., et al. // Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001. 2001.

22. Hsu W.M. Segmented ray casting for data parallel volume rendering // Proceedings of the 1993 Parallel Rendering Symposium. 1993.

23. Eldridge M., Igehy H., Hanrahan P. Pomegranate: A fully scalable graphics architecture // Proceedings of the ACM SIGGRAPH Conference on Computer Graphics. 2000.

24. ATTILA: A cycle-level execution-driven simulator for modern GPU architectures / Del Barrio V.M., et al. // ISPASS 2006: IEEE International Symposium on Performance Analysis of Systems and Software, 2006. Vol. 2006.

25. Griffin: Hardware-software support for efficient page migration in multi-GPU systems / Baruah T., et al. // Proceedings - 2020 IEEE International Symposium on High Performance Computer Architecture, HPCA 2020. 2020.

26. Arnau J.M., Parcerisa J.M., Xekalakis P. Eliminating redundant fragment shader executions on a mobile GPU via hardware memoization // Proceedings - International Symposium on Computer Architecture. 2014.

27. Tauray: A Scalable Real-Time Open-Source Path Tracer for Stereo and Light Field Displays / Ikkala J., et al. // Proceedings - SIGGRAPH Asia 2022: Technical Communications. 2022.

28. Granja P., Pereira J. Hybrid-Rendering Techniques in GPU. 2023.

29. GPU Accelerated Path Tracing of Massive Scenes / Jaros M., et al. // ACM Trans Graph. 2021. Vol. 40, no. 2.

30. MDScale: Scalable multi-GPU bonded and short-range molecular dynamics / Barreales G.N., et al. // J Parallel Distrib Comput. 2021. Vol. 157. P. 243–255.

31. MIGPerf: A Comprehensive Benchmark for Deep Learning Training and Inference Workloads on Multi-Instance GPUs / Zhang H., et al. 2023.

32. Balın M.F., Sancak K., Çatalyürek Ü. V. MG-GCN: Scalable Multi-GPU GCN Training Framework. 2021.

33. Dong Y., Peng C. Multi-GPU multi-display rendering of extremely large 3D environments // Vis Comput. 2023. Vol. 39, no. 12. P. 6473–6489.

34. rCUDA: Reducing the number of GPU-based accelerators in high performance clusters / Duato J., et al. // Proceedings of the 2010 International Conference on High Performance Computing and Simulation, HPCS 2010. 2010. P. 224–231.

35. Beyond the socket: NUMA-aware GPUs / Milic U., et al. // Proceedings of the Annual International Symposium on Microarchitecture, MICRO. 2017. Part F131207.

36. Kirk D. NVIDIA CUDA software and GPU parallel computing architecture // International Symposium on Memory Management, ISMM. 2007.

37. O'Donnell Y. FrameGraph: Extensible Rendering Architecture in Frostbite // Gdc 2017. 2017. Vol. 1999, no. December.

38. Electronic Arts. Halcyon: Rapid Innovation using Modern Graphics. 2019. URL: https://media.contentapi.ea.com/content/dam/ea/seed/presentations/wihlidal2019-rebootdevelopblue-halcyon-rapid-innovation.pdf (accessed: 23.08.2025).

39. Unity. HDRP Render Graph. URL: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@12.0/manual/Custom-Post-Process.html (accessed: 14.03.2023).

40. Epic Games. Unreal Rednering Pipeline. URL: https://dev.epicgames.com/community/learning/tutorials/lyJK/unreal-engine-rendering-pipeline-ue5 (accessed: 23.07.2025).

41. Epic Games. A first look at Unreal Engine 5. 2020. URL: https://www.unrealengine.com/en- US/blog/a-first-look-at-unreal-engine-5 (accessed: 23.08.2024).

42. Installing NVIDIA Nsight into Visual Studio // Accelerating Matlab with GPUs. 2014.

43. Iyer K., Kiel J. GPU debugging and profiling with NVIDIA parallel nsight // Game Development Tools. 2016.

44. Zhukov S., Iones A., Kronin G. An ambient light illumination model. 1998.

45. Landis H. Production-Ready Global Illumination. 2004.

46. The Alchemy screen-space ambient obscurance algorithm / McGuire M., et al. // Proceedings - HPG 2011: ACM SIGGRAPH Symposium on High Performance Graphics. 2011.

47. Bavoil L., Sainz M. Screen space ambient occlusion // NVIDIA developer information: …. 2008. no. September.

48. Bavoil L., Sainz M., Dimitrov R. Image-space horizon-based ambient occlusion // SIGGRAPH'08: ACM SIGGRAPH Talks 2008. 2008.

49. Song Y., Liu F., Xu J. Horizon-based screen-space ambient occlusion using mixture sampling // ACM SIGGRAPH ASIA 2010 Posters, SA'10. 2010.

50. Mcguire M., Mara M., Luebke D. Scalable ambient obscurance // High-Performance Graphics 2012, HPG 2012 - ACM SIGGRAPH: Eurographics Symposium Proceedings. 2012.

51. Mara M., McGuire M., Luebke D. Lighting Deep G-Buffers: Single-Pass, Layered Depth Images with Minimum Separation Applied to Indirect Illumination // Graphics.Cs.Williams.Edu. 2013.

52. Jimenez J. Practical Realtime Strategies for Accurate Indirect Occlusion // Siggraph 2016. 2016.

53. Vardis K., Papaioannou G., Gaitatzes A. Multi-view ambient occlusion with importance sampling // Proceedings of the Symposium on Interactive 3D Graphics. 2013.

54. Vermeer J., Scandolo L., Eisemann E. Stochastic-Depth Ambient Occlusion // Proceedings of the ACM on Computer Graphics and Interactive Techniques. 2021. Vol. 4, no. 1.

55. Gautron P. Real-time ray-traced ambient occlusion of complex scenes using spatial hashing // Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks, SIGGRAPH 2020. 2020.

56. Multi-GPU room response simulation with hardware raytracing / Thoman P., et al. // Concurr Comput. 2022. Vol. 34, no. 4.

57. Microsoft Corporation. Multi-Adapter. 2017. URL: https://devblogs.microsoft.com/directx/directx-12-multiadapter-lighting-up-dormant-silicon-and-making-it-work-for-you/. (accessed: 07.01.2025).

58. Marchesin S., Mongenet C., Dischler J.-M. Multi-GPU Sort-Last Volume Visualization. 2008. 1–8 p.