ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Санжаров В.В., Фролов В.А.

Исследование
масштабируемости
распределённых рендер-
систем на основе
алгоритмов адаптивной
трассировки путей и
Metropolis Light Transport в
гетерогенных сетях

Рекомендуемая форма библиографической ссылки: Санжаров В.В., Фролов В.А. Исследование масштабируемости распределённых рендер-систем на основе алгоритмов адаптивной трассировки путей и Metropolis Light Transport в гетерогенных сетях // Препринты ИПМ им. М.В.Келдыша. 2016. № 114. 22 с. doi:[10.20948/prepr-2016-114](https://doi.org/10.20948/prepr-2016-114)
URL: <http://library.keldysh.ru/preprint.asp?id=2016-114>

О р д е н а   Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Р о с с и й с к о й   а к а д е м и и   н а у к

В.В. Санжаров, В.А. Фролов

**Исследование масштабируемости
распределённых рендер-систем
на основе алгоритмов
адаптивной трассировки путей
и Metropolis Light Transport
в гетерогенных сетях**

Москва — 2016

Санжаров В.В., Фролов В.А.

Исследование масштабируемости распределённых рендер-систем на основе алгоритмов адаптивной трассировки путей и metropolis light transport в гетерогенных сетях

В данной работе предлагается новый подход к распределенному рендерингу как на GPU, так и на CPU, использующий идеи grid вычислений. Предложена новая модель поведения вычислительных узлов на основе акторного подхода. Проведено численное исследование свойств модели.

Ключевые слова: grid вычисления, акторы, распределенный рендеринг, GPU

Vadim Vladimirovich Sanzharov, Vladimir Aleksandrovich Frolov

Scaling properties of distributed rendering systems based on adaptive path tracing and metropolis light transport algorithms in heterogeneous networks

In this paper we present the new approach to distributed rendering on GPU and CPU based on grid computing paradigm. We propose the new actor-based model for peers' behavior and present the results of numerical study of its properties.

Key words: Grid computing, actors, distributed rendering, GPU

Работа выполнена при поддержке Российского фонда фундаментальных исследований (РФФИ) 16-31-60048.

Оглавление

1. Введение.....	3
2. Распределенный рендеринг.....	3
2.1 Рендеринг при помощи grid вычислений	3
3. Сетевые технологии	6
4. Предлагаемое решение	7
4.1 Распределённая адаптивная трассировка путей (ОМС)	8
4.2 Распределённый Metropolis Light Transport (MCMC).....	9
4.3 Выбор средства моделирования.....	9
4.4 Описание модели распределённых вычислений	10
4.5 Анализ результатов моделирования	12
5. Выводы.....	20
6. Благодарности	21
7. Литература	21

1. Введение

Задача синтеза фотореалистичных изображений в приложении к таким областям, как архитектурная, интерьерная визуализация и кинопроизводство, требует значительных вычислительных ресурсов. Зачастую| единственным способом получить результат за приемлемое время является распределение расчета освещения на несколько компьютеров. На сегодняшний день самый распространенный подход – использование так называемых рендер-ферм, высокопроизводительных компьютеров, объединенных в локальную сеть или вычислительный кластер. Однако рендер-фермы мало пригодны для небольших компаний и индивидуальных художников в силу высокой стоимости и определенных неудобств, связанных с использованием внешних сервисов.

2. Распределенный рендеринг

Одним из первых подходов к задаче распределенного рендеринга стали решения для суперкомпьютеров и кластеров [1, 2]. В работе [2] было предложено решение на основе интерактивной параллельной трассировки лучей, позволившее достичь 15-20 кадров в секунду в динамических сценах за счет их эффективного разбиения на независимые объекты. В работе [3] предлагается централизованная система рендеринга, в которой все расчеты выполняются на стороне сервера, который, однако, может использовать вычислительные ресурсы нескольких компьютеров. Клиенты же получают доступ к возможностям просмотра и редактирования сцены (в некоторых пределах) через веб-интерфейс.

2.1 Рендеринг при помощи grid вычислений

Современные исследования в области распределенного рендеринга в основном направлены на использование grid вычислений, основной характеристикой которых является гетерогенность сети: компьютеры, выполняющие вычисления, могут значительно отличаться по производительности и иметь различное географическое расположение. Среди основных проблем grid вычислений следует отметить (Рис. 1):

- Неопределённое количество вычислительных узлов. Количество машин, задействованных в вычислениях, может изменяться прямо во время расчёта. Одни узлы могут подключаться, другие – отключаться.
- Недетерминированное поведение вычислительных узлов и сети в целом. Вычислительный узел может получить свою порцию работы рано (в начале расчёта кадра) или поздно (ближе к концу расчёта). Он может выполнить её быстро или медленно. Его канал (сетевой) может быть широким либо узким.
- Асинхронный ответ. В момент ответа вычислительного узла его результаты могут быть ещё валидны или уже нет (т.е. уже не нужны

пользователю). Такая ситуация типична, поскольку пользователь достаточно часто останавливает расчёт и запускает его заново, например, с новой позиции камеры либо с изменёнными параметрами материалов.

Рис. 1. grid вычисления объединяют в сеть вычислительные узлы по всему миру с совершенно различными характеристиками

Указанные выше проблемы накладывают достаточно жёсткие требования на распределённую реализацию алгоритмов рендеринга:

1. Во-первых, работа должна быть делима на неравные части (или хотя бы на достаточно большое число мелких частей).
2. Во-вторых, разделение работы по частям должно уметь динамически переконфигурироваться, поскольку вычислительный узел, взявший на себя некоторую часть работы, ненадёжен – он может быть отключён, может выполнять работу слишком медленно, может получить все необходимые для выполнения работы данные сцены слишком поздно.
3. В-третьих, всё взаимодействие вычислительных узлов должно быть асинхронным.

Одно из решений на основе grid вычислений – BURP (Big Ugly Rendering Project) [4]. Этот проект представляет собой распределённую систему для рендеринга анимационных сцен. BURP основан на инфраструктуре BOINC (Berkeley Open Infrastructure for Network Computing) [5] и, тем самым, может быть отнесен к виду grid вычислений, называемому добровольными вычислениями. Желаящий рассчитать анимационную сцену пользователь загружает на сервер проекта свою задачу, которая распределяется между подключёнными к серверу пользователями, предоставляющими свои вычислительные ресурсы на добровольной основе. BURP поддерживает две рендер-системы – Cycles и встроенный рендер ПО Blender.

В работе [6] представлена рендер-система Yafriid, позволяющая распределить расчет изображения между гетерогенными компьютерами через Интернет. Одним из компонентов предлагаемой авторами системы является

модуль MAgArRO, использующий мультиагентный подход и нечеткую логику для анализа вычислительной сложности изображения и проведения оптимизации разбиения задачи расчета изображения на подзадачи.

В работах [7] и [8] представлена рендер-система, реализующая распределенный расчет изображений с помощью алгоритма кэша освещенности. Авторы предлагают подход к рендерингу с использованием алгоритма кэша освещенности на основе grid вычислений [7] и в Peer-to-peer (P2P) сетях [8]. Основу метода составляет использование вычислительными узлами результатов друг друга для ускорения процедур интерполяции и заполнения структур данных, специфичных именно для кэша освещенности.

Часть рассмотренных решений выполняют распределение задачи между вычислительными узлами только по отдельным кадрам и тем самым подходят только для анимационных сцен. Кроме того, при работе над одним кадром все рассмотренные выше решения используют особенности алгоритмов рендеринга на CPU и не могут быть напрямую применены к CPU и/или GPU рендер-системам, работающим в сети типа grid.

Например, наиболее распространённый способ – распределение работы по тайлам изображения (так называемым “бакетам”). При небольшом размере тайла стоимость его обратной пересылки (от машины, на которой он был рассчитан, к головной машине), как правило, пренебрежимо мала. Но этот способ для grid вычислений не подходит, поскольку при разной производительности машин в сети и различных скоростях соединения трудно спрогнозировать, какие машины и сколько тайлов должны получить, когда они начнут над ними работу и когда вернут результат. Пользователь при этом, скорее всего, увидит на экране хаотичную картинку из меняющихся блоков, что является практически неприемлемым. Современные требования к рендер-системам очень жёсткие: расчёт должен быть прогрессивным, и независимо от фактической скорости пользователь должен видеть промежуточный результат в адекватном виде, чтобы как можно раньше понять, хочет ли он что-либо переделать, и, соответственно, как можно раньше остановить расчёт. В этом случае единственный способ – пересылать и объединять Монте-Карло выборки для изображений целиком.

Именно по этой причине в распространенных GPU рендер-системах, например Vray RT [9], реализован такой способ, но архитектура сетевого взаимодействия проста и мало масштабируема, рассчитана только на локальную сеть. Одна из основных причин этого – возросший объем данных, которыми необходимо обмениваться вычислительным узлам, поскольку теперь они вынуждены пересылать всё изображение целиком.

Предлагаемая в данной работе модель распределенного рендеринга ориентирована на использование ресурсов как GPU, так и CPU, и рассчитана на распределение работы между вычислительными узлами в рамках расчета одного изображения в гетерогенной сети, состоящей из компьютеров с разной производительностью (grid – Рис. 2). Мы исследовали данную модель на

масштабируемость с учётом возрастающей нагрузки на пересылку всего изображения между вычислительными узлами (по сравнению с традиционными решениями на основе пересылки бакетов/тайлов).

Рис. 2. Фрагмент одной из модельных сетей в виде дерева.
В узлах – машины с различными GPU

3. Сетевые технологии

Одна из проблем, встающих в системах grid вычислений, связана с сетевым взаимодействием вычислительных узлов. Это обусловлено тем, что в соответствии с парадигмой grid вычислений компьютеры потенциально могут иметь любое географическое расположение, что приводит к необходимости передачи данных в условиях глобальных сетей, которые значительно отличаются от таковых для локальных сетей.

В задаче расчета освещения каждый вычислительный узел должен получить по сети всю информацию о 3d-сцене, включая геометрию и текстуры. В современных производственных задачах сцены могут содержать несколько миллионов треугольников в геометрии, а с распространением текстурирования в таком ПО, как The Foundry MARI с использованием технологии UDIM, – несколько сотен текстур. Таким образом, размер сцены, с учетом архивирования, может достигать нескольких гигабайт.

Кроме того, при распределенном рендеринге необходимо также передавать по сети результаты расчета освещения. А при необходимости получать на клиентском компьютере постоянные обновления это нужно делать довольно часто. Таким образом, проблема сетевого взаимодействия играет значительную роль при распределенном расчете освещения.

Одним из эффективных подходов к организации сетевого взаимодействия являются P2P технологии [10]. В [11] приводится классификация P2P технологий в контексте их применения к задачам grid вычислений. Одна из классификаций делит P2P по структуре на структурированные, неструктурированные и гибридные.

В неструктурированных P2P сетях поиск вычислительных ресурсов под задачу выполняется в основном путем так называемого «затопления» (flooding) – отправки запросов всем узлам, подключенным к сети. Такой подход быстро становится неэффективным с ростом числа узлов и тем самым препятствует

масштабируемости сети. В структурированных сетях узлы логически организуются, например, на основе распределенных таблиц хэшей (distributed hash tables, DHT). В работах [12-14] было показано, что структурирование позволяет снизить трафик и, соответственно, нагрузку на канал в глобальных P2P сетях, а также решает проблему несбалансированного распределения ресурсов между узлами.

На сегодняшний день ни одно из существующих решений в области распределенного рендеринга не использует возможностей P2P технологий, несмотря на их перспективность в grid вычислениях. К преимуществам использования P2P технологий в grid вычислениях, имеющим значение в задаче распределенного рендеринга, можно отнести возможность использования более сложной модели параллельных вычислений (для традиционных grid сетей она обычно довольно проста): единственный главный компьютер напрямую раздает задачи подчиненным и принимает от них результаты расчетов [11]. Технология P2P вносит большую гетерогенность в структуру сети, что позволяет добиться большей гибкости в процессе распределения задач между вычислительными узлами и получения результатов. Кроме того, все существующие системы обладают весьма ограниченной масштабируемостью при распределённом рендеринге одного кадра (а grid вычисления используются только для рендеринга анимации), не позволяя пользователю получать быстрый отклик от системы.

Мы предлагаем исследование проблемы неограниченной масштабируемости производительности для несмещённого рендеринга одного кадра и её решение при помощи алгоритмов адаптивной трассировки путей и переноса света Метрополиса (Metropolis Light Transport) на основе P2P технологий.

4. Предлагаемое решение

Для того чтобы эффективно распараллелить работу по расчёту освещения и синтезу изображения, нами были выбраны несмещённые методы – трассировка путей, являющаяся применением обычного Монте-Карло (Ordinary Monte Carlo, OMC) к расчёту освещённости, и Metropolis Light Transport (MLT) – применение Монте-Карло по схеме Марковских цепей (Markov Chain Monte Carlo - MCMC).

Ключевая особенность этих алгоритмов – возможность распараллеливать вычисления по отдельным Монте-Карло выборкам. Изображения, рассчитанные на разных машинах, можно усреднять с весами для отдельных пикселей, пропорциональными числу Монте-Карло выборок, фактически сделанных на данном вычислительном узле. Таким образом, использование несмещённых методов позволяет автоматически решить проблему перераспределения работы – каждый вычислительный узел внесёт в изображение вклад пропорционально фактически сделанному (а не запланированному) объёму работы.

4.1 Распределённая адаптивная трассировка путей (ОМС)

Одна из существенных проблем Монте-Карло трассировки на графических процессорах – сильно неравномерное распределение работы по изображению. В то время как для одних пикселей изображения достаточно всего нескольких сотен Монте-Карло выборок на пиксел, другие могут потребовать десятков тысяч для достижения необходимой точности. Один из возможных вариантов решения этой проблемы – адаптивная трассировка путей на основе распределения работы GPU по тайлам изображения [15]. Данный алгоритм со временем исключает из расчёта тайлы изображения, для которых достигнута заданная точность. Однако этот алгоритм изначально не рассчитан на распределённый рендеринг, поскольку при объединении изображений с нескольких вычислительных узлов необходимая точность для некоторых тайлов уже может быть достигнута, но знает об этом только тот вычислительный узел, на котором произошло объединение. Остальные вычислительные узлы будут продолжать считать тайлы, которые были исключены из расчёта на “главном вычислительном узле” (используем пока этот термин, не вдаваясь в детали того, что он означает) – Рис. 3.

Рис. 3. После объединения промежуточных изображений от 2 подчинённых машин на главной машине остаются активными только области, отмеченные оранжевыми прямоугольниками. Подчинённые машины, однако, об этом не знают и продолжают обсчитывать всё изображение целиком

Для того чтобы исправить эту проблему, нами был введён механизм пересылки сообщений об исключённых тайлах на дочерние машины и исследована его эффективность.

4.2 Распределённый Metropolis Light Transport (MCMC)

В отличие от обычного Монте-Карло, проблема неравномерного распределения работы в Монте-Карло по схеме Марковских цепей решается автоматически самим алгоритмом. Metropolis Light Transport (MLT) сам помещает больше выборок в наиболее сложные области изображения [16]. Однако плата за это – увеличенное время старта, необходимость выполнять “прожиг” для того, чтобы правильно разместить начальные точки для цепей Маркова в необходимые области многомерного пространства (включающего в себя плоскость изображения), пропорционального сложности этих областей, и таким образом устранить “начальное смещение” (startup bias) при расчёте на GPU [15]. В рамках данной работы мы исследовали масштабируемость рендера при использовании Metropolis Light Transport на нескольких узлах, содержащих GPU, и сравнили её с масштабируемостью адаптивной трассировки путей.

Следует отметить, правда, что MLT не всегда можно использовать как замену для обычной Монте-Карло трассировки путей на практике, поскольку он плохо справляется с расчётом первичного освещения (в силу высокой яркости источников освещения и отдельных сильно освещённых областей). По этой причине мы также исследовали масштабируемость совместного решения, когда первичное освещение считается Монте-Карло трассировкой путей, а вторичное рассчитывается при помощи Metropolis Light Transport (Рис. 4). Стоит отметить, что в этом случае нагрузка на сеть при передаче изображений возрастает в 2 раза (т.к. нужно передавать 2 отдельных изображения – для первичного и для вторичного освещения).

Рис. 4. Иллюстрация применения MLT на практике. Прямой свет был посчитан при помощи обычной Монте-Карло трассировки путей (Path Tracing, PT, слева). MLT был использован только для расчёта вторичного освещения (посередине).

Итоговая картинка – результат сложения изображений (справа)

4.3 Выбор средства моделирования

Исследованная в данной работе модель построена с использованием акторного подхода, в основе которого лежит асинхронный обмен сообщениями

[17], и модели начальник-подчинённый (или заказчик-исполнитель). Акторный подход позволяет моделировать и анализировать поведение широкого круга систем распределённых параллельных вычислений и находит широкое применение в построении в силу масштабируемости и устойчивости к отказам [19]. Мы использовали реализацию акторной модели в популярной для построения высоконагруженных параллельных систем платформе Akka [19] для создания программной реализации предлагаемого решения.

Основная цель использования акторов – организация асинхронной модели вычислений и решение проблем (описанных ранее), связанных с низкой надёжностью и недетерминированностью вычислительных узлов.

4.4 Описание модели распределённых вычислений

В предлагаемой модели вычислительные узлы объединяются в древовидную структуру (рис. 1). Корнем дерева является компьютер, инициировавший задачу рендеринга, другими словами «заказчик». В узлах располагаются компьютеры, предоставляющие свои вычислительные ресурсы для решения задачи – «исполнители». В качестве допущения принимается, что каждый из компьютеров знает своих ближайших (с точки зрения скорости передачи данных) соседей-исполнителей. Таким образом, компьютер-заказчик отправляет вычислительную задачу своим соседям-исполнителям и те, в свою очередь, получив запрос, начинают выполнять расчеты. Исполнители далее также отправляют вычислительную задачу уже своим соседям-исполнителям, выступая для них в роли заказчика. Результатом вычислений в случае расчета освещения является изображение с некоторым числом сэмплов (Монте-Карло выборки) на пиксель.

После выполнения некоторого минимального объема вычислений, регламентированного компьютером-заказчиком, и по запросу вышестоящего в иерархии компьютера вычислительные узлы передают текущие результаты вычислений вверх по иерархии. Изображения, полученные от дочерних узлов и в результате локальных расчетов, объединяются в одно (при помощи усреднения сумм Монте-Карло выборки в акторе `renderNodeA`) в каждом узле графа и передаются на уровень выше. Таким образом, вышестоящая в иерархии машина (заказчик) “видит” каждого исполнителя как одну, но очень мощную машину, не имея информации о том, что на самом деле происходит ниже в иерархии. В реализованной модели учитывается скорость передачи данных между узлами сети, которая влияет на время, через которое каждый из компьютеров получит сцену и сможет начать расчет, а также на время передачи результатов. В соответствии с акторным подходом в модели выделены три типа акторов, условно обозначенных как `renderNodeA`, `renderNodeB`, `localRender`. Эти три актора присутствуют на каждом вычислительном узле сети. Схема взаимодействия этих акторов представлена на Рис. 5.

Рис. 5. Схема взаимодействия акторов в рамках одного вычислительного узла. Всё, что внутри серого прямоугольника, расположено на 1 машине

Актор `renderNodeA` является родительским для двух остальных типов акторов в системе. Этот актор получает через дочерний актор `renderNodeB` результаты вычислений с более низких уровней дерева, а также от локальной рендер-системы и объединяет их. Результат объединения `renderNodeA` отправляет на уровень вверх актору `renderNodeB` другого вычислительного узла. Актор `renderNodeB`, помимо передачи результатов расчетов, занимается установлением соединения с другими вычислительными узлами.

Актор `localRender` занимается взаимодействием с расчетным ядром рендера на данном узле – запускает и останавливает расчет, получает текущие результаты и передает их родительскому актору `renderNodeA`.

Таким образом, модель работает в соответствии со следующей логикой:

- `renderNodeA` на корневом компьютере инициирует задачу рендеринга, создает два дочерних актора – `renderNodeB` и `localRender`;
- `renderNodeA` регламентирует время получения результатов расчетов от дочерних узлов, отсылая через равные промежутки времени сообщения с запросом результатов;
- `renderNodeB` создает акторов типа `renderNodeA` на соседних узлах и передает им сообщение с информацией о задаче;
- `localRender` запускает расчет освещения локально;
- акторы `renderNodeA` соседних узлов действуют аналогично корневому актору того же типа.

4.5 Анализ результатов моделирования

В качестве исходных данных мы взяли две архитектурные сцены (Рис. и рис. 7) вместе с текстурами. Первая сцена занимает на диске порядка 500 мегабайт, вторая – порядка 350 мегабайт. Данные сцены были рассчитаны с использованием алгоритма трассировки пути [20] в рендер-системе Hydra Renderer [21] на GPU от Nvidia модели GTX 670. В результате было получено, что на данном GPU для первой сцены (рис. 6) скорость расчета составляет приблизительно 0.5 сэмпла на пиксел в секунду, а для второй сцены (рис. 7) – около 0.4 сэмплов на пиксел в секунду.

Рис. 6. Сцена, характеристики которой были использованы для исследования свойств распределённой адаптивной трассировки путей

Рис. 7. Сцена, характеристики которой были использованы для исследования свойств распределённого Metropolis Light Transport. Весь оранжевый свет – отражённый, каустический

В первом численном эксперименте был зафиксирован GPU для каждого компьютера сети (GTX 670) и канал связи между соседними компьютерами (download 10 мегабайт/сек. и upload 5 мегабайт/сек.). Рассмотрены три топологии – бинарное и тернарные деревья (с глубиной 3) и плоское дерево (все узлы подключены к корню). Один узел – один компьютер с одним GPU, всего доступных компьютеров 15. Моделирование проводилось до достижения числа сэмплов, равного 1024 для трех сетей. Сначала рассмотрим случай, когда архив со сценой необходимо передавать по сети между узлами (Рис. , обозначены пунктирными линиями). Видно, что меньшее число подключений на один узел приводит к лучшему результату. И в случае подключения всех вычислительных узлов напрямую к корневому компьютеру (обозначен синим на графике) к моменту завершения передачи сцены бинарная конфигурация (красный пунктирный график) практически успевает посчитать такую сцену дважды.

Далее рассмотрим случай, когда вычислительные узлы скачивают сцену с некоторого сервера или из облачного хранилища, т.е. нагрузка на каналы между узлами связана исключительно с передачей результатов (показаны на Рис. 8 точечными графиками).

Рис. 8. Рост числа сэмплов для трёх топологий с одинаковыми GPU и сетевыми каналами. По оси X – время. По оси Y – число сэмплов на пиксел. Binary tree – бинарное дерево из одинаковых машин с GTX670. Ternary tree – тернарное дерево из одинаковых машин с GTX670. Flat tree – дерево глубиной 1, когда все дочерние машины напрямую подключены к основной машине-заказчику.

Пометка (cloud) указывает на использование облачного хранилища для доставки сцены на расчётные узлы. Ideal – идеальное время рендера, полученное как время рендера на гипотетической машине, обладающей суммой производительностей всех машин

В этом случае время, требуемое на расчет, падает более чем в два раза. Меньшее количество подключений на узел все так же дает наилучший результат.

Если увеличить временной интервал, через который компьютер-заказчик запрашивает из сети промежуточные результаты, то расчет также ускоряется (Рис. 8, розовый график). Таким образом, в случае, когда важен только финальный результат, интервал получения обновлений может быть увеличен для ускорения расчетов.

Теперь рассмотрим результаты моделирования сети с топологией бинарного дерева, с фиксированными каналами связи с высокой пропускной способностью, одинаковой вычислительной мощностью узлов и довольно большим интервалом передачи обновлений. Моделирование проводилось до достижения 32768 сэмплов для разного числа компьютеров – 31, 63 и 127 (глубина дерева 4, 5 и 6 соответственно). В этом случае видно (Рис. 9), что для большого числа сэмплов, типичного для финального рендера, и с достаточно большим интервалом отправления результатов скорость расчета освещения сходится к идеальной (как если бы вся производительность сети была бы доступна на одном компьютере). Есть лишь отставание, связанное с необходимостью загрузить сцену. Однако такая конфигурация сети сама по себе близка к идеальной из-за одинаково высоких скоростей каналов связи у всех узлов (12 мегабайт/сек.), которые без труда обеспечивают доставку результатов раз в 20 секунд (Рис. 9).

Рис. 9. Рост числа сэмплов для топологии бинарного дерева разной глубины.

По оси X – время. По оси Y – число сэмплов на пиксел. Графики предоставлены для сетей из 31 (красный), 63 (зелёный) и 127 (синий) машин. Сплошные линии соответствуют производительности идеального случая, когда производительность всех машин просто суммируется. Точечные линии соответствуют моделированию производительности предлагаемого решения

Наконец, рассмотрим сценарий моделирования со случайными конфигурациями сети. Каждый компьютер сети оборудован GPU, выбранным случайным образом исходя из частоты встречаемости данного GPU в популярном сервисе цифрового распространения компьютерных игр и программ Steam (были взяты 19 самых популярных GPU, а также GTX 670) [22].

Производительность каждого GPU была оценена исходя из известной производительности GTX 670 и баллов, набранных каждым из GPU в системе тестирования 3d Mark [23]. Каналы связи заданы случайным образом (с равномерным распределением) в диапазоне от 0.5 до 12.5 мегабайт/сек. Моделирование проводилось до достижения 16384 сэмплов, максимальная глубина дерева – 3.

Рис. 10. Сеть со случайными характеристиками. По оси X – время. По оси Y – число сэмплов на пиксел. Красные линии для сети с суммарной производительностью 9.25 сэмплов на пиксел. Зелёная линия для сети с суммарной производительностью 11.08 сэмплов в секунду (но с худшими каналами связи). Сплошные линии – для идеального случая. Пунктирные – моделирование предлагаемого решения

В итоге, значительную роль играет скорость каналов связи. Сеть с большей суммарной производительностью – 11.08 сэмплов/сек. (Рис.10, зеленый пунктирный график) справляется с задачей медленнее на 40 минут, чем сеть с меньшей производительностью (9.25 сэмплов/сек., красный пунктирный график) именно за счет худших каналов в сети. Для первой сети результат хуже идеального в 5.5 раз, а для второй – в 3.3 раза. Суммарная производительность второй сети превосходит производительность корневого

компьютера в 18.5 раз, а скорость расчета выше в 5.6 раз. Для первой сети суммарная производительность больше в 22 раза, скорость – в 4 раза.

В следующей группе экспериментов было проведено сравнение адаптивного и обычного алгоритмов трассировки. Моделирование производилось до достижения 16384 сэмплов на пиксел на первой тестовой сцене (рис. 11). Были рассмотрены следующие сценарии:

1. Гомогенная сеть из 15 компьютеров с топологией бинарного дерева глубины 3, зафиксированным GPU для каждого компьютера сети (GTX 670) и одинаковыми каналами связи между соседними компьютерами (download 12 мегабайт/сек. и upload 6 мегабайт/сек.);
2. Один очень мощный компьютер, обладающий производительностью, эквивалентной производительности сети сценария №1;
3. Гетерогенная сеть с компьютерами и каналами связи, заданными случайным образом аналогично предыдущей серии экспериментов, максимальная глубина дерева – 4.

Результаты моделирования приведены на рис. 11:

Рис. 11. Сравнение адаптивного и обычного алгоритмов трассировки пути. По оси X – время. По оси Y – число сэмплов на пиксел. Пунктирные линии соответствуют обычной трассировке путей. Сплошные – адаптивной. Чёрные линии – одна гипотетическая машина, обладающая суммарной производительностью сети (идеальный случай). Красные линии – бинарная сеть с одинаковыми машинами. Синие линии – сеть со случайными характеристиками

Будем считать, что для сцены на рис. 11 для одного мощного компьютера (на графике показан черным цветом) расчет алгоритмом адаптивной трассировки пути (сплошная линия) примерно в 2 раза быстрее обычного алгоритма трассировки пути (пунктирная линия) (*конкретное число определяется характеристикой самой сцены и настройками точности; чем точнее нужен расчёт, тем больше выигрыш от адаптивной трассировки путей, но обычно он не превышает 2-3 раз [15]*). Чтобы промоделировать поведение адаптивного алгоритма в сети, будем считать, что при рендеринге на каждой машине в среднем каждые 1024 сэмпла из расчёта исключаются 25% тайлов после достижения некоторого минимального числа сэмплов на пиксел (в эксперименте это 4096 сэмплов). Эти числа соответствуют экспериментально наблюдаемому поведению алгоритма из работы [15] в рассмотренной нами рендер-системе при рендеринге сцены, изображённой на рис.11 с высокими настройками качества на одной машине.

Результаты моделирования показали (рис.11), что как для гомогенной бинарной сети (графики красного цвета на рис.11), так и для гетерогенной сети со случайно заданными характеристиками (графики синего цвета на рис.11) адаптивный подход дал меньший прирост скорости – в ~1.3-1.4 раз против 2 раз на одной машине. Причина замедления заключается в том, что распространение сообщения об остановке расчёта некоторых тайлов изображения происходит в сети с определённой задержкой (вызванной не скоростью распространения самого сообщения, а тем фактом, что для отправки такого сообщения машина-заказчик сначала должна получить изображения от дочерних машин) и ситуация, когда машина исполнитель продолжает обсчитывать тайлы, уже не нужные машине-заказчику, по-прежнему встречается довольно часто. Тем не менее алгоритм адаптивной трассировки пути в сетевых сценариях всё ещё даёт ощутимый прирост в производительности в – 30-40% от максимально-возможного ускорения (полученного на одной машине – в 2 раза).

Заключительная группа экспериментов исследует поведение распределенных алгоритмов – MLT и комбинированного подхода MLT + RT, описанного ранее. Моделирование производилось также до достижения 16384 сэмплов на пиксел на второй тестовой сцене (рис. 12). Основная исследованная конфигурация – гомогенная сеть из 15 компьютеров с топологией бинарного дерева глубины 3, зафиксированным GPU для каждого компьютера сети (GTX 670) и одинаковыми каналами связи между соседними компьютерами (download 12 мегабайт/сек. и upload 6 мегабайт/сек.). Также была рассмотрена гетерогенная сеть с компьютерами и каналами связи, заданными случайным образом аналогично предыдущей серии экспериментов, максимальная глубина дерева – 4. Проведенные расчеты показали, что необходимость «прожига» в алгоритме MLT не оказывает влияния на масштабируемость сети – точка начала расчета просто отодвигается на время, необходимое для «прожига» (рис.12).

Разница между чистым MLT (синие графики на рис. 12) и комбинированным подходом (фиолетовые графики на рис. 12), с точки зрения поведения сети, наблюдается только в случае необходимости часто передавать результаты расчета по сети, т.е. когда требуется отслеживание результатов рендеринга в реальном времени. При минимальном числе сэмплов на пиксель в передаваемом изображении, равном 8 комбинированному подходу (синяя сплошная линия на рис. 12), требуется в два раза больше времени для расчета изображения по сравнению с чистым MLT (пурпурная сплошная линия на рис. 12), что обусловлено необходимостью передавать в два раза больше информации по сети.

Рис. 12. Сравнение алгоритмов MLT, MLT + PT и адаптивного PT. По оси X – время. По оси Y – число сэмплов на пиксел. Пунктирным графикам соответствует минимальное число сэмплов на пиксел для передачи по сети, равное 256, а сплошным графикам – равное 8. Красные линии – бинарная сеть из одинаковых машин и адаптивная трассировка путей. Пурпурные линии – бинарная сеть из одинаковых машин и комбинированный подход (PT + MLT). Синие линии – бинарная сеть из одинаковых машин и только MLT. Зелёная линия – сеть со случайными характеристиками и только MLT

Как было показано в предыдущих экспериментах, влияние сети на время расчета можно уменьшить, увеличив временной интервал между передачей промежуточных результатов. Поэтому если увеличить минимальное число сэмплов на пиксель в передаваемом изображении с 8 до 256, то чистый MLT и комбинированный подход практически совпадают по динамике расчета (синий и пурпурный пунктирные графики на рис. 12). Кроме того, оба этих подхода при условии минимального передаваемого числа сэмплов, равного 256, в значительной мере повторяют динамику адаптивной трассировки пути (красный пунктирный график на рис. 12), присутствует только сдвиг графиков на время, необходимое для «прожига». Структура сети в данном случае влияния не оказала – характер поведения для гетерогенной сети (зеленый график на рис. 12) и гомогенной бинарной сети (синий и пурпурный графики) практически одинаковы: графики сдвигаются вправо на небольшой интервал.

В заключение приведем сравнение рассмотренных в работе алгоритмов для двух топологий сети (бинарной и плоской) из 63 узлов с суммарной производительностью 31.5 сэмплов на пиксел в секунду на первой тестовой сцене (рис. 6) по времени, затраченному вычислительной сетью для достижения 16384 сэмплов на пиксел (рис. 13).

Для достаточно небольшого минимального числа сэмплов в изображении, передаваемом по сети, равного 32, разница между бинарной топологией (синие столбцы на рис. 13) и плоской (оранжевые столбцы на рис. 13) составляет порядка двух-трех раз. Только для комбинированного подхода она становится несколько меньше. При увеличении минимального числа сэмплов в изображении, передаваемом по сети, до 128, разница между бинарной (серые столбцы на рис. 13) и плоской (желтые столбцы на рис. 13) топологиями значительно уменьшается, что связано с уменьшением общей нагрузки на сеть. При этом наибольший выигрыш во времени за счет более редкой передачи промежуточных результатов наблюдается для комбинированного подхода, что логично, т.к. он требует сетевой передачи в два раза большего объема информации.

Таким образом, предложенный метод распределённого рендеринга на основе организации вычислительных узлов в дерево позволяет в 2-3 раза повысить эффективность расчёта (по сравнению с отсутствием иерархии, рис.13) при необходимости интерактивного отслеживания расчёта пользователем (когда пользователь должен часто получать обновления изображения и видеть результаты промежуточного расчёта). При расчете же финального изображения, когда можно снизить частоту обновления итогового изображения, влияние топологии сети снижается, хотя и не пропадает полностью (рис. 13) – предложенная организация вычислительных узлов в дерево продолжает показывать лучшие результаты (до +30% скорости).

Рис.13. Сравнение алгоритмов обычного РТ, адаптивного РТ, MLT, MLT + РТ для разных топологий и разного минимального количества сэмплов в передаваемом по сети изображении (промежуточном результате). По оси Y – время в секундах. Синие столбцы – бинарная сеть из одинаковых машин, минимальное количество сэмплов при передаче равно 32. Оранжевые столбцы – плоская сеть (все узлы подключены напрямую к машине-заказчику), минимальное количество сэмплов при передаче равно 32. Серые столбцы – бинарная сеть из одинаковых машин, минимальное количество сэмплов при передаче равно 128. Желтые столбцы – плоская сеть (все узлы подключены напрямую к машине-заказчику), минимальное количество сэмплов при передаче равно 128

5. Выводы

Разработанная модель позволила оценить перспективность распределенных grid технологий для задачи расчета освещения с учетом особенностей решения этой задачи на GPU. Предложенная схема организации распределённого рендеринга (на основе организации вычислительных узлов в дерево) позволяет повысить производительность расчёта в 2-3 раза для интерактивного рендеринга (когда пользователь должен видеть промежуточные результаты расчета с относительно частыми обновлениями).

Скорости каналов связи 12.5 мегабайт/сек. (т.е. 100Mb/s) хватает для того, чтобы интервал обновления изображения (10-20 сек, рис. 3-5) был достаточен для интерактивного отслеживания рендеринга. Результаты исследования модели не показали наличия предела масштабируемости для такой организации сетевого взаимодействия вычислительных узлов по сети как для распределенной адаптивной трассировки пути (РТ), так и для распределенного алгоритма MLT и для комбинированного подхода (MLT+РТ).

При обеспечении хорошей связи между вычислительными узлами возможно добиться практически идеальной скорости расчета по сравнению с несетевым решением. Если же связь относительно медленная, скорость расчета может быть увеличена за счет большего временного интервала между получением результатов расчетов из сети. Но при этом предложенный распределённый алгоритм адаптивной трассировки путей даёт меньшее ускорение – порядка 30-40% от максимально возможного ускорения в работе [15] (когда расчёты производятся на одной машине).

Использование алгоритма Metropolis Light Transport, с другой стороны, дало лучшие результаты – задержка, вызванная необходимостью “прожига”, пренебрежимо мала (рис.12 – небольшой сдвиг графика вначале). Однако его использование оправдано только на сценах с доминирующим вторичным освещением (поскольку напрямую для расчёта первичного освещения MLT использовать нельзя). В связи с этим мы считаем, что идеальным кандидатом на параллельный расчёт в grid сети мог бы стать алгоритм Gradient Domain Metropolis Light Transport [24], который позволяет полностью исключить из расчёта обычный Монте-Карло и считать при помощи МСМС освещение целиком, однако это тема для будущего исследования.

6. Благодарности

Работа выполнена при поддержке РФФИ, грант 16-31-60048 мол_а_дк.

7. Литература

- [1] DeMarle D.E., Parker S., Hartner M., Gribble C., Hansen C.: Distributed interactive ray tracing for large volume visualization. IEEE Symposium on Parallel and Large-Data Visualization and Graphics (2003), IEEE Computer Society, p. 12.
- [2] Parker S., Martin W., Sloan P.-P. J., Shirley P., Smits B., Hansen C.: Interactive ray tracing. I3D '99 Proceedings of the 1999 symposium on Interactive 3D graphics (1999), pp. 119-126.
- [3] Барладян Б.Х., Волобой А.Г., Вьюкова Н.И., Галактионов В.А., Дерябин Н.Б. Моделирование освещенности и синтез фотореалистичных изображений с использованием Интернет технологий // Программирование, №5, 2005, с.66-80.
- [4] Открытая распределенная система рендеринга BURP на основе платформы BOINC. 2016 <https://burp.renderfarming.net/>
- [5] Платформа grid вычислений BOINC. 2016 <https://boinc.berkeley.edu/>
- [6] Gonzales-Morcillo C., Weiß G., Vallejo-Fernandez D., Liminez-Linares L., Albusac-Jimenez J. 3D distributed rendering and optimization using free software, European Journal of the Informatics Professional, 6, 8(2008), pp. 45-53.
- [7] Aggarwal V., Debattista K., Bashford-Rogers T., Dubla P., Chalmers A.: High-fidelity interactive rendering on desktop grids. IEEE Computer Graphics and Applications 32, 3 (2012), pp. 24–36.

- [8]Bugeja K., Debattista K., Spina S., Chalmers A. Collaborative high-fidelity rendering over peer-to-peer networks, PGV '14 Proceedings of the 14th Eurographics Symposium on Parallel Graphics and Visualization, pp. 9-16.
- [9]Chaos group, производитель рендер-систем Vray и Vray RT. 2016 <http://chaosgroup.com>
- [10]Foster, I., Iamnitchi, A.: On death, taxes, and the convergence of peer-to-peer and grid computing. In: IPTPS '03: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (2003)
- [11]Zhao H., Liu X., Li X. A taxonomy of peer-to-peer desktop grid paradigms, Cluster Computing, 14, 2(2011), pp.129-144.
- [12]Cheema, A.S., Muhammad, M., Gupta, I.: Peer-to-peer discovery of computational resources for grid applications. In: GRID'05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, pp. 179–185 (2005).
- [13]Darlagiannis, V., Liebau, N., Heckmann, O., Mauthe, A., Steinmetz, R.: Caching indices for efficient lookup in structured overlay networks. Agents and P2P Computing, pp. 81–93 (2006).
- [14]Gupta, R., Sekhri, V., Somani, A.K.: CompuP2P: An architecture for internet computing using peer-to-peer networks. IEEE Trans. Parallel Distrib. Syst. 17(11), 1306–1320 (2006).
- [15]Frolov V.A., Kharlamov A.A., Ignatenko A. Biased solution of integral illumination equation via irradiance caching and path tracing on GPUs. Programming and Computer Software, 2011, Vol 37, № 5, p. 252-259.
- [16]Фролов В., Галактионов В. Исследование особенностей Metropolis Light Transport на GPU. Труды Международной научной конференции GraphiCon2016, г.Нижний Новгород, ННГАСУ, 2016, с.227-231.
- [17]Hewitt C., Bishop P., Steiger R. A Universal Modular Actor Formalism for Artificial Intelligence. IJCAI'73 Proceedings of the 3rd international conference on Artificial intelligence, pp.235-245.
- [18]Hewitt C. ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing, IEEE Internet Computing, 12, 5(2008), pp. 96-99.
- [19]Платформа распр. вычислений Akka. 2016
- [20]Kajiya, J. T.: The rendering equation. Computer Graphics 20(4): 143–150. Proceedings of SIGGRAPH '86
- [21]Рендер-система Hydra Renderer. 2016 <http://raytracing.ru/>
- [22]Статистика по комплектующим компьютеров пользователей платформы Steam. 2016 <http://store.steampowered.com/hwsurvey>
- [23]Система тестирования 3dMark. 2016 <http://3dmark.com/>
- [24]Jaakko Lehtinen and Tero Karras and Samuli Laine and Miika Aittala and Fr'edo Durand and Timo Aila. Gradient Domain metropolis Light Transport. ACM Trans. Graph.2013. Vol 32. №4.