Введение

В последнее время наблюдается бурное развитие искусственного интеллекта, которое находит свое применение во всех областях жизнедеятельности человека. Одним из перспективных направлений, где может быть применена эта инновация, являются не менее перспективные методы оптической диагностики в биомедицине. Эти методы основаны на использовании света в видимом и инфракрасном диапазонах для зондирования биологических объектов и анализа рассеянного света. По сравнению с другими методами диагностики, такими как рентгеновские лучи или магнитно-резонансная томография, оптические методы имеют несколько преимуществ, среди которых можно выделить безопасность, неинвазивность, высокое разрешение и низкую стоимость.

Одним из наиболее перспективных направлений в биомедицине являются методы оптической диагностики. Они основаны на использовании света в видимом и инфракрасном диапазонах для зондирования биологических объектов и анализа рассеянного света. По сравнению с другими методами диагностики, такими как рентгеновские лучи или магнитно-резонансная томография, оптические методы имеют несколько преимуществ, среди которых можно выделить безопасность, неинвазивность, высокое разрешение и низкую стоимость. В последние годы методы оптической диагностики продолжают активно развиваться и находят все большее применение.

Так же в связи с бурным развитием искусственного интеллекта возникает интерес ко внедрению интеллектуальных инноваций в сфере оптической диагностики. Однако эта задача осложняется потребность в большом количестве данных для обучения моделей искусственного интеллекта. Подготовка таких данных является достаточно долгим и трудоемким процессом, особенно с учетом требуемой квалификации от сотрудников, собирающих эти данные. В связи с этим возникает потребность в развитии соответствующего программного обеспечения для моделирования большого количества оптических измерений.

Метод Монте-Карло является мощным инструментом для решения различных задач, основанных на вероятностных математических моделях. Основная идея этого метода заключается в генерации большого количества случайных величин для моделирования сложных процессов, результат которых аналитически получить затруднительно или невозможно. В частности, в задаче физического моделирования распространения света внутри биологических тканей получить аналитическое описание в общем случае невозможно. Это объясняется тем, что интегро-дифференциальное уравнение переноса излучения в общем виде, тем более, в случае сложных граничных условий, не имеет аналитического решения. Поэтому для решения этой задачи используется метод Монте-Карло, который так же является эффективной заменой прямому облучению биологических образцов.

Метод Монте Карло выполняет многократное повторение случайных независимых испытаний, обсчитывая сложную математическую модель. Однако вычисление всех траекторий фотонов из пучка света может занять достаточно много времени, так как такая задача требует высокой точности и большого количества итераций. Наиболее принципиальным методом сокращения времени моделирования является параллелизация вычислений.

До недавнего времени наиболее доступными системами для параллельных расчетов были слабосвязные кластеры, в которых вычисления производились на центральных процессорах общего назначения (CPU, Central Processing Unit). Однако такие кластеры достаточно дороги и сложны в эксплуатации. В последние годы с развитием графических процессоров (GPU, Graphics Processing Unit) возможности вычислительных систем значительно расширились, так как такие процессоры специально разрабатывались как поточно-параллельные системы с большим количеством вычислительных блоков, конвейерной обработкой данных и памятью с максимальной пропускной способностью. В тоже время возросла сложность разработки и оптимизации ПО для эффективного применения GPU в вычислительных задачах.

В данной дипломной работе будет рассмотрено моделирование процессов методом Монте-Карло на GPU с применением технологии Intel OneAPI. Будет проведен анализ возможностей и ограничений вычислительных систем с использованием GPU для данного метода, а также исследованы способы оптимизации вычислений на GPU. Результаты исследования могут быть использованы для улучшения эффективности вычислительных систем при реализации метода Монте-Карло в различных приложениях.

------------------------

Одним из наиболее известных численных методов решения сложных математических систем является метод Монте Карло, который находит свое применение в разных задачах, начиная от простого расчёта интеграла до изображения сцен компьютерной графики. В том числе в моделировании распространения света внутри биологических тканей. Реализация этого метода заключается в процессе многократного расчета случайного транспорта одного фотона в среде. Такой программный подход может быть эффективной заменой действительному облучению и дальнейшему сбору информации о распространении фотонов в многослойных биологических структурах.

Метод Монте Карло выполняет многократное повторение случайных независимых испытаний, обсчитывая сложную математическую модель. Предсказание её характеристик и поведения требует большого количества вычислительных ресурсов. Наиболее принципиальным методом повышения производительности является распараллеливание вычислений.

До недавнего времени наиболее доступными системами для параллельных расчетов были слабосвязные кластеры, в которых вычисления производились на центральных процессорах общего назначения (CPU, Central Processing Unit). Однако такие кластеры достаточно дороги и сложны в эксплуатации, поэтому проблема эффективных параллельных вычислений остается актуальной. С развитием индустрии трехмерной компьютерной графики и массивно-параллельных ускорителей, появились более конкурентные решения в виде графических процессоров (GPU, Graphics Processing Unit), которые специально разрабатывались как поточно-параллельные системы с большим количеством вычислительных блоков, конвейерной обработкой данных и памятью с максимальной пропускной способностью.

Безусловным лидером среди индустрии массивно-параллельных ускорителей является компания Nvidia. Эта компания предоставляет графическое процессоры на базе архитектуры CUDA. Разработчикам ПО для GPU Nvidia [1] предоставляется набор инструментов CUDA Toolkit[2]. Важным фактом для программ, разработанных с применением CUDA Toolkit [3], является поддержка всей линейки видеокарт с той же архитектурой на протяжении уже нескольких лет.

Архитектура построения программ с массивно-параллельными вычислениями идейно очень похожа между разными вендорами графических ускорителей. Но, к сожалению, из-за отсутствия единой стандартизации программно-аппаратного взаимодействия, программные реализации сильно зависимы от своего вендора и не могут быть взаимозаменяемы.

Ошибки, совершенные на этапе планирования, зачастую обходятся намного дороже в сравнение с ошибками последующих этапов. Архитекторы ПО и проектные менеджеры зачастую стоят перед сложным выбором, изучая все ограничения и сопутствующие затраты для каждой возможной архитектуры массивно параллельного ускорителя.

В попытке решить возникающие проблемы было создано несколько проектов, среди которых наиболее новым и современным является решение от компании Intel – открытый стандарт OneAPI [4], язык программирования DPC++ и сопутствующий набор библиотек для кросс платформенного программирования на вычислительных устройствах.

В данной работе проведен обзор технологий Nvidia CUDA и Intel OneAPI для программирования массивно-параллельных вычислений и описан опыт разработки решений с их применением. В частности, реализован параллельный алгоритм метода Монте Карло в задаче распространения света.

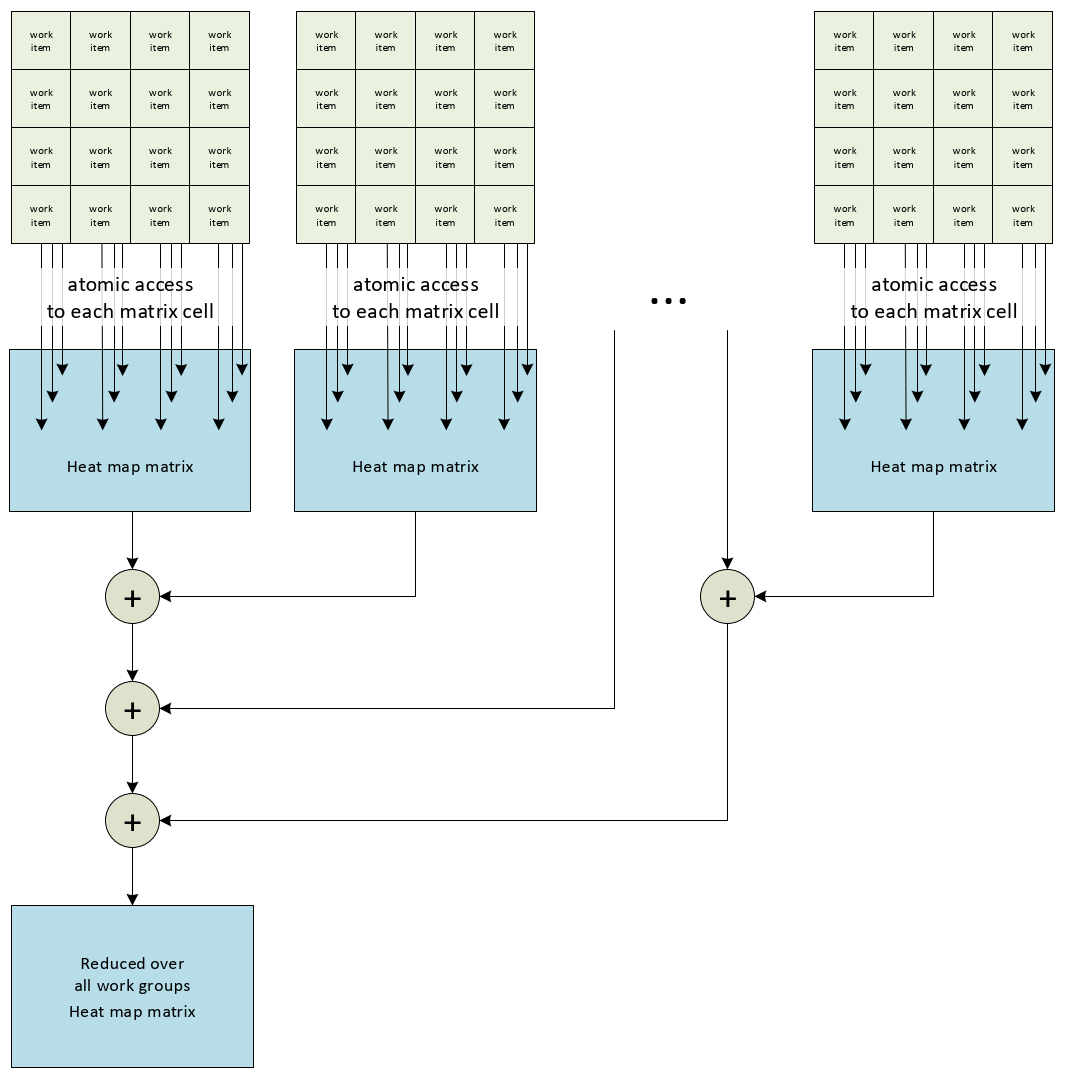
# Подсистема памяти многопоточного приложения

Проблема многих многопоточных реализаций заключается в синхронизированном доступе к общим ресурсам, например, таким как память. В частности, одной из таких проблем является одновременный доступ, data race. Data race т.н. состояние гонки – это ошибка проектирования многопоточной системы, при которой нарушен синхронизированный доступ к общим ресурсам, от которых зависят результаты вычислений. Проблема устраняется введением блокировок при доступе, что, несомненно, сказывается на общей скорости вычислений. Существуют способы уменьшения количества совместных блокировок, некоторые из которых решаются на уровне архитектуры ПО.

Параллельная реализация метода Монте-Карло предполагает одновременный доступ к глобальной памяти, в которой хранится матрица потерянной энергии фотонов в каждой точке пространства. Учитывая особенность реализации, невозможно предсказать какой следующий адрес будет использован для сохранения результатов, поэтому применить какой-то потокобезопасный и, одновременно, эффективный алгоритм записи данных не представляется возможным. При обновлении значений предполагается эксклюзивный доступ для чтения и записи значений с помощью применения атомарных операций над областью памяти. Доступ к одним и тем же ячейкам памяти для разных ядер осуществляется поочередно, что приводит к простаиванию ядер, ожидающих запись. С увеличением количества вычислителей неизбежно растет время ожидания такого доступа к общему ресурсу, поэтому общий прирост производительности снижается. И наоборот, снижение количества вычислителей приведет к снижению времени ожидания на запись.

Несмотря на это, рост времени ожидания доступа может быть снижен путем разделения областей памяти и балансировки вычислений на них.

Наиболее приятной особенностью метода Монте-Карло является независимость работы вычислителей от результатов работы других вычислителей. С учетом этого, может быть предложена следующая архитектура параллельной реализации метода Монте-Карло с разделением памяти между группами вычислителей:



Основные положения такой реализации:

1. Вычислители (work-items) объединяются в несколько групп (work-group) фиксированного размера.
2. Каждая группа имеет собственную область памяти, к которой не имеет доступ другие группы.
3. Внутри группы каждый вычислитель имеет атомарный доступ к каждой ячейке матрицы весов.
4. Работа вычислителей происходит независимо от результатов других вычислителей.
5. После завершения работы всех вычислителей выполняется объединение все матриц в итоговую результирующую матрицу.

Так же стоит отметить возможность использования локальной памяти внутри группы вычислителей для малых размеров памяти, что должно положительно сказываться на скорости доступа. К сожалению, матрицы большого размера не получится оптимизировать таким образом, т.к. объем локальной памяти сильно ограничен, и как правило составляет десятки килобайт.

Выделим преимущества и недостатки такой реализации. К плюсам можно отнести:

1. Среднее время ожидания доступа для записи/чтения определяется размером группы.
2. Количество групп вычислителей не ограничено, хорошая масштабируемость.

К минусам:

1. Большое потребление памяти.

Таким образом получается увеличить скорость вычислений распространения фотонов в плоскопараллельной среде.

**Локальная память**

Локальная память – это тип памяти, доступный для использования в графических ускорителях, который расположен непосредственно рядом с ядрами исполнения. С одной стороны, это даёт ей более высокую пропускную способность. С другой стороны, её объем, как правило, существенно меньше. Так же, она не имеет прямой адресации с глобальной областью памяти, поэтому при необходимости обмена данных приходится выполнять копирование. Локальная память обычно используется для хранения данных, которые не могут быть эффективно переданы между глобальной памятью и регистрами ядер исполнения. Поэтому локальная память обычно используется для хранения промежуточных результатов вычислений или для кэширования данных, которые часто используются внутри ядер исполнения.

В рамках данной задачи применение локальной памяти в общем случае становится невозможным ввиду большого размера матриц, которые используются для хранения результатов моделирования.