2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Когда речь идет о проектировании архитектуры приложения, единственным и самим гибким решением остается разбиение проектируемого модуля на функциональные блоки. Такое деление уменьшает сложность системы, дает возможность масштабирования, разделенную на части программу легче воспринимать и сопровождать.

В разрабатываемом программном модуле можно выделить следующие функциональные блоки:

- блок декодирования изображений;
- блок кодирования изображений;
- блок управления графическими процессорами;
- блок инициализации;
- блок управления исполняемыми алгоритмами;
- блок конфигурирования;
- блок управления потоком данных;
- блок сетевого взаимодействия;
- блок управления исполняющими единицами;
- блок сохранения результатов.

Структурная схема, которая иллюстрирует связи взаимодействия между блоками, приведена на чертеже ГУИР.400201.040 С1. Каждый блок выполняет строго отведенную для него функциональность, что позволяет изменять каждый элемент независимо друг от друга. Такое «разделение ответственности» позволяет повысить надежность и адаптивность программного модуля. Более подробное описание блоков (общее назначение, основные взаимосвязи между элементами, использованные программные средства) представлено в нижеприведенных подразделах.

2.1 Блок декодирования изображений

Данный блок предназначен для преобразования изображений из формата, в котором оно было сохранено, в формат, пригодный для применения алгоритмов обработки.

Блок будет представлять собой набор классов для декодирования определённого формата изображений с максимальной скоростью.

2.2 Блок кодирования изображений

Блок кодирования изображений предназначен для преобразования данных изображения из формата, пригодного для обработки, в формат, пригодный для записи изображения в файл.

Данный блок будет представлять собой набор классов для максимально быстрого кодирования изображений.

2.3 Блок управления графическими процессорами

В этом блоке будут реализованы классы, которые будут определять, сколько в системе имеется CUDA-совместимых графических процессоров, параметры графических процессоров: название, версия Compute Capability, объём памяти, количество потоковых мультипроцессоров, пропускная способность памяти, и другие параметры.

Так же в данном блоке будет реализован функционал выбора графического процессора, на котором будут выполняться все этапы работы программы. Будут реализованы два метода выбора графического процессора: ручной и автоматический.

При ручном выборе пользователь сам сможет задать, какой графический процессор использовать.

При автоматическом выборе программа сама выберет графический процессор согласно заданной политике: наиболее новая (по версии Compute Capability), с наибольшим количеством памяти, с наибольшей пропускной способностью памяти, с наибольшим числом потоковых мультипроцессоров.

2.4 Блок инициализации

Данный блок будет ответственным за запуск приложения. Так как одной из задач проекта является достижения хорошей масштабируемости, довольно важную роль будет играть задание начального состояния каждого элемента системы.

Блок инициализации будет иметь связь с большинством управляющих блоков системы. Здесь же будет реализован функционал по получению текущего состояния любого компонента системы.

Основной принцип работы данного блока при запуске программы заключается в общении с блоком конфигурирования, получении от него конфигурационных параметров других компонентов системы, инициализации тех компонентов, для которых были получены конфигурационные параметры, проверке на ошибки при инициализации.

В случае, если все компоненты уже проинициализированы, данный блок будет иметь доступ к основным управляющим компонентам системы, благодаря чему будет реализован функционал запроса текущего состояния произвольного компонента системы в произвольный момент времени.

2.5 Блок управления исполняемыми алгоритмами

В классическом понимании алгоритм – последовательность конечных действий, которых нужно выполнить, чтобы получить решение поставленной задачи. В данном проекте это понятие несколько шире: алгоритм – последовательность конечных действий, а так же набор системных ресурсов, задействуемых при выполнении действий, которые нужно выполнить, чтобы

получить решение поставленной задачи. Здесь алгоритм – самостоятельная функциональная единица, которой для выполнения определённых действий необходимо предоставить ресурсы.

Данный блок представляет собой набор классов для создания алгоритмов для определённой исполняющей единицы, от которой данный блок будет получать набор конфигурационных параметров для каждого алгоритма, и который будет предоставлять ресурсы для выполнения.

2.6 Блок конфигурирования

Блок конфигурирования является одним из основных блоков системы. Благодаря блоку конфигурирования возможна работа всей системы в целом.

Данный блок отвечает за чтение конфигурационных файлов, их валидацию, предоставление конфигурационных параметров по запросу.

Этот блок имеет связь со всеми остальными блоками системы, так как все остальные блоки имеют внутренние параметры конфигурации, помимо тех, которые они получают от блока инициализации.

В данном блоке будут реализованы классы для работы с конфигурационными файлами в формате JSON.

2.7 Блок управления потоком данных

Блок управления потоком данных представляет собой набор классов для хранения данных, полученных на каждом этапе работы алгоритмов.

В данном блоке так же будут находиться все внутренние структуры данных и классы для работы с ними.

Данный блок назван таким образом, потому что в нём будут реализованы классы, которые отвечают за хранение данных, ограничение их количества, реструктуризация одного набора данных в другой.

Так же стоит отметить, что данный блок является связующим звеном между отдельно взятыми исполняющими единицами.

2.8 Блок сетевого взаимодействия

Данный блок представляет собой набор классов для сетевого взаимодействия с Kafka брокером, из которого на вход программы будут приходить данные.

Будут реализованы классы, отвечающие за получение сообщений и отправку сообщений, обработка результата взаимодействия с брокером.

Данный блок связан с блоком конфигурирования, так как есть необходимость получать внутренние конфигурационные параметры, определяющие такие параметры, как адрес брокера, Kafka topic, из которого необходимо считывать данные, и в который необходимо отсылать сообщения.

2.9 Блок управления исполняющими единицами

Управления исполняющими единицами является основным блоком системы. Данный блок тесно связан с блоком управления исполняемыми алгоритмами, так как исполняющие единицы будет предоставлять ресурсы, необходимые для выполнения алгоритма.

Исполняющая единица – сущность, которая хранит в себе набор алгоритмов, которые она должна исполнять, данные о месте, откуда необходимо брать данные на обработку, данные о месте, куда необходимо отдавать результаты работы алгоритмов, а так же системные ресурсы.

Существует два основных вида исполняющих единиц:

1. Задействующие только CPU.
2. Задействубщие как CPU, так и GPU.

Исходя из этого, определяются ресурсы, которыми владеют исполняющие единицы: поток процессора и CUDA stream графического процессора.

2.10 Блок сохранения результатов

Данный блок отвечает за сохранение результатов работы программы. Здесь будут реализованы классы для создания PLY моделей, а так же классы для взаимодействием с хранилищем результатов.

2.11 Описание архитектуры

Архитектура программы имеет некоторое сходство с архитектурой центрального процессора: обработка данных выполняется по принципу конвейера.

Работа программы будет выполняться в два этапа: инициализация всех объектов системы, взаимодействие компонентов системы.

На этапе взаимодействия компонентов системы будет выполняться следующая последовательность действий: считывание Kafka сообщения, извлечение данных из сообщения, сбор всех данных об одном объекте, определение начальных параметров изображений, извлечение ключевых точек изображений, поиск пар изображений с совпадающими ключевыми точками, построение рассеянного облака точек, построение плотного облака точек, построение трёхмерной сетки, минимизация плоскостей, текстурирование, сохранение результата. При этом каждый из перечисленных этапов выполняется параллельно, что будет обеспечивать хорошую пропускную способность конвейера, а использование GPU позволит достичь больших скоростей работы.