

## Введение

Развитие современного общества зачастую ставит перед наукой цели, решение которых требует решения самых разнообразных систем дифференциальных уравнений. В том числе и систем диффузионных уравнений. К таким задачам можно отнести уравнение теплопроводности !!! (примеры других задач). Каждую из них можно решать с помощью численных методов, например, методом Эйлера, многошаговыми методами Рунге-Кутты или методами Дормана-Принца. Подобные вычисления удобно автоматизировать, чтобы в дальнейшем иметь возможность быстро производить расчеты.

## 1 Постановка задачи

Требовалось создать программный комплекс, с помощью которого можно было бы автоматизировать моделирования конечно-разностных диффузионных задач на гибридных вычислительных кластерах.

Задача реакции диффузии имеет следующий вид.

$$\dot{u} = D \Delta u + F(u);$$

$$\frac{\partial u}{\partial \nu} = 0, F(0) = 0.$$

Точечная задача  $\dot{u} = f(u)$ , «определяющая реакцию», имеет устойчивый период  $\dot{u}_0(f)$ , который также !!!(слово) однородное решение задачи.

Для получения результата может быть применен один из следующих методов численного решения дифференциальных уравнений, а именно: метод Эйлера, четырехстадийный метод Рунге-Кутты, семистадийный метод Дормана-Принца.

Разрабатываемый программный комплекс должен поддерживать работу с уравнениями и системами уравнений, которые были бы распределены в одномерных, двумерных и трехмерных областях. Каждая из таких областей может быть представлена неким набором блоков. Для одномерного случая этими блоками являются отрезки, в случае плоскости - прямоугольники, параллелепипеды - если область трехмерна !!!примеры. Каждый из блоков характеризуется координатами в пространстве и размерами, а также информацией о своих границах. Границы блока могут состоять из нескольких частей, каждая из которых представлена !!! Нейман Дирихле, либо является местом соединения с другим блоком.

Кроме того, необходима поддержка современного оборудования с неоднородной архитектурой и иерархической организацией памяти. К системам с неоднородной архитектурой относятся, например, вычислительные кластеры, которые используют для расчетов мощности центрального процессора и видеокарт. Иерархическая организация памяти предполагает малые объемы высокоскоростной памяти и большие медленной. Подобный подход к реализации памяти подразумевает экономное расходование ресурсов в процессе выполнения с целью повешения общей производительности за счет ускоренного обращения к данным.

Программный комплекс должен иметь возможность по окончании вычислений сохранять полученный результат, а также выполнять сохранение текущего состояния в процессе выполнения для последующего возобновления вычислений с любого из сохраненных состояний. Кроме того, необходимо обладать инструментами для запуска расчетов до определенного момента времени с данным шагом или выполнять заданное количество итераций.

В качестве языка программирования используемого для реализации поставленных задач был выбран C++ как наиболее подходящий для

данных целей. Этот язык обладает необходимыми инструментами для удобной разработки подобного приложения, примером того может послужить механизм наследования. Кроме того, язык C++ дает возможность низкоуровневой работы с памятью, что позволяет производить оптимизацию программного кода на высоком уровне. Также, важным достоинством языка является библиотека MPI, которая позволяет осуществлять передачу данных между вычислительными устройствами, который расположены на разных машинах, используя протокол Ethenet !!! проверить.

## 2 Архитектура приложения

### 2.1 Класс Solver и его потомки

Класс Solver и его потомки обеспечивают работу различных численных методов, которые могут применяться для решения поставленной задачи. Сам класс Solver является абстрактным классом, который определяет интерфейс, необходимый для дальнейшей работы, содержит две переменных: матрица текущего состояния (mState) и количество элементов в этой матрице (mCount).

EulerSolver, RK4Solver, DP45Solver являются потомками класса Solver и описывают основную информацию о реализуемых методах. Например, является ли данный численный метод методом с изменяемым шагом или требуется ли ему предварительная подготовка перед первой итерацией, сообщает количество итераций которые необходимы для данного метода (для метода Эйлера это значение равно единице). Кроме того, они хранят набор констант необходимых конкретно для данного метода, например для метода Рунге-Кутты это числа, используемые при вычислении нового состояния. !!! информация о методе Все эти классы в полной мере реализуют все необходимые функции, описанные в классе предке, поэтому объект каждого из них можно создать. Именно они используются в качестве хранителей информации о выбранном методе решения задачи.

Классы EulerSolverCpu, EulerSolverGpu и другие уже являются потомками второго уровня, они наследуются от EulerSolver, RK4Solver, DP45Solver соответственно и предназначены для работы на центральном процессоре и видеокарте соответственно. Именно эти классы в полной мере реализуют выбранный численный метод. Разберем работу данных классов на примере RK4SolverCpu. Фактически данный класс является большим хранилищем данных. Реализованный метод Рунге-Кутты предполагает четыре стадии, поэтому предок этого класса имеет четыре переменных для временного хранения данных (например mTempStore1, mTempStore2). Именно в них записывается информация о выделенных под хранилища ресурсах. В процессе работы от объекта данного класса можно получить источник и приемник информации.

## 2.2 Класс Block и его потомки

Класс Block и его потомки обеспечивают непосредственно расчеты, производимые с данными. Сам класс Block является абстрактным и, как следствие, его создание невозможно. Он описывает интерфейс для блоков, который позволит полноценно работать с реальными блоками - потомками данного класса. Кроме того, он имеет переменные для хранения всей необходимой информации о блоке, например, его положение в пространстве, размеры, тип и номер устройства, на котором блок будет выполняться.

Класс BlockCpu является наследником класса Block, предназначенным для работы на центральном процессоре. Он является реальным блоком, который хранит информацию и осуществляет ее обработку. Для осуществления распараллеливания используется библиотека OpenMP.

Класс BlockGpu является наследником класса Block, предназначенным для работы на видеокарте. Он также является реальным блоком, но в отличие от класса BlockCpu всю информацию он хранит в памяти видеокарты, расчеты выполняются также на ней, для этого используется специальный язык CUDA.

Класс BlockNull единственный класс-наследник Block, который не является реальным блоком. Он нужен для того чтобы обозначить, что часть области, относящаяся к данному блоку на самом деле обрабатывается другим потоком и на другой машине. Но он позволяет выполнить все функции присущие блоку, но в данном случае ничего не будет выполнено. Кроме того, объект данного класса содержит основную информацию о блоке: его размеры и местоположение в пространстве. корректной работы.

### 2.3 Класс Domain

Главным управляющим классом приложения является класс Domain. Объект этого класса создается в единственном экземпляре каждым из потоков выполнения. Данный класс обладает широким набором возможностей. Чтение загрузочной информации из файла, во время которой фактически происходит инициализация всех переменных приложения. Вывод результатов вычисления как по завершению работы, так и во время выполнения. Также этот класс осуществляет сохранение текущего состояния вычислений и загрузку любого ранее сохраненного состояния, относящегося к той же задаче.

## 2.4 Класс Interconnect

Для пересылки данных между блоками, который расположены на разных узлах вычислительного кластера используется класс Interconnect. Каждый его экземпляр хранит номер потока, являющегося источником передаваемой информации, номер потока, который данную информацию должен принять, а также длину массива, который необходимо переслать. Экземпляры этого класса создаются на каждую связь между блоками каждым потоком. При вызове функции передачи/приема объект выполнит либо отправку сообщения, либо прием в зависимости от номера потока, который вызвал данную функцию у себя, либо не будет выполненно ничего, если пересылка осуществляться не должна, например поток не имеет к ней вообще никакого отношения, не является ни источником, ни приемником, либо пересылка для данной связи не требуется - блоки принадлежат одному потоку и, как следствие, расположены на одной машине и могут получать данные напрямую.

### 3 Компоненты программного комплекса

Приложение состоит из нескольких частей. Каждая из них отвечает за определенную часть работы, либо подготовительной, либо относящейся непосредственно к самим вычислениям, либо реализующей взаимодействие с пользователем.

Пользовательский интерфейс позволяет создавать и модифицировать проекты. Кроме того, он дает возможность управлять текущими вычислениями, а именно запускать, прекращать, приостанавливать, а также выполнять сохранение текущего состояния и его загрузку.

Часть предварительной обработки выполняет подготовку исходных данных. В ней происходит обработка геометрии и свойств области, производится определение размерности, расположения и размера блоков, граничных условий. Здесь же выполняется распределение блоков по вычислительным устройствам, формируется библиотека пользовательских функций.

Параллельный фреймворк и алгоритмы обработки данных составляют основу приложения и занимаются непосредственно самими вычислениями. К алгоритмам обработки данных относятся метод Эйлера, явные схемы Рунге-Кутты, методы Дормана-принца. Стоит отметить, что присутствует возможность расширения списка доступных методов решения.



## 4 Параллельный фреймворк

Параллельный фреймворк занимается непосредственной реализацией распределенных вычислений. С его помощью выполняется взаимодействие всех вычислительных устройств во время работы приложения. Он же реализует пересылку данных между блоками. Подобные пересылки необходимы в случае, когда одному блоку нужна информация от другого для продолжения собственных вычислений. Кроме того, фреймворк занимается обработкой событий, которые генерирует пользовательский интерфейс, а именно начало вычислений с заданными параметрами, приостановка вычислений, их прекращении, сохранение и загрузка состояний.

Мелкозернистый параллелизм на различных устройствах осуществляется по-разному в зависимости от типа устройства. Если работа ведется с центральным процессором, то для использования всех доступных ресурсов используется библиотека OpenMP.

OpenMP реализует параллельные вычисления с помощью многопоточности, в которой «главный» (master) поток создает набор подчиненных (slave) потоков и задача распределяется между ними. Предполагается, что потоки выполняются параллельно на машине с несколькими процессорами (количество процессоров не обязательно должно быть больше или равно количеству потоков). Количество создаваемых потоков может регулироваться как самой программой при помощи вызова библиотечных процедур, так и извне, при помощи переменных окружения. Данная библиотека работает в системах с общей памятью.

Конкретно в данном случае библиотека осуществляет распараллеливание расчетов нового состояния блока. Каждой нити исполнения дается на обработку часть матрицы. Нить выполняет расчет новых значений основываясь на текущих данных своей части, кроме того она имеет возможность получить данные, которые обрабатывают другие нити, если таковые потребуются для расчета своих значений. Так как работа ведется с общей памятью никакой подготовки для такого «чтения» не требуется.

Пересылка данных между разными машинами (вычислительными узлами) осуществляется с помощью библиотеки MPI. Message Passing Interface (MPI, интерфейс передачи сообщений) - программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу. В стандарте MPI описан интерфейс передачи сообщений, который должен поддерживаться как на платформе, так и в приложениях пользователя. В первую очередь MPI ориентирован на системы с рас-

пределенной памятью, то есть когда затраты на передачу данных велики, в то время как OpenMP ориентирован на системы с общей памятью (многоядерные с общим кешем). Обе технологии могут использоваться совместно, чтобы оптимально использовать в кластере многоядерные системы.

Крупнозернистый параллелизм осуществляется путем разбиения области на блоки и распределения блоков по вычислительным устройствам еще на этапе подготовки, о чем было изложено ранее. Данный подход позволяет использовать все вычислительные мощности имеющиеся в наличии и равномерно распределять нагрузку на устройства. На одном устройстве может располагаться несколько блоков одновременно, в таком случае их вычисление будет осуществляться в порядке очереди. Немаловажным фактом является то, что в случае если количество блоков значительно (в несколько раз) превышает количество вычислительных устройств, то блоки, которые имеют общие границы разумно располагать на одном устройстве, либо в пределах одной машины, а блоки не имеющих таких границ - на разных, сохраняя равномерность распределения блоков по устройствам в целом. Такой подход позволяет увеличить скорость расчетов для небольших, по количеству используемых узлов, областей. Достигается это уменьшение времени, которое необходимо на пересылку данных между блоками между итерациями вычислений, так как если блоки расположены на одном вычислительном устройстве или одной машине, то пересылка не будет выполняться в принципе, потому что данные уже доступны блоку. Очевидно, что в общем случае невозможно распределить все блоки таким образом чтобы пересылка не было вообще, но такой подход позволяет минимизировать количество пересылаемой информации, что положительно сказывается на производительности.