Санкт-Петербургский государственный университет
Кафедра моделирования электромеханических и компьютерных систем

Федоренко Максим Андреевич

Выпускная квалификационная работа

Применение высокопроизводительных вычислений в задачах моделирования течения крови в сосудистых системах

Уровень образования: магистратура
Направление 03.04.01 «Прикладные математика и физика»
Основная образовательная программа ВМ.5519.2019 «Прикладная информатика»

Научный руководитель:
кандидат физ.-мат. наук,
доцент
Кривовичев Г.В.

Рецензент:
специалист по системной биологии
ЗАО «БИОКАД»
Безрукова Е.С.

Санкт-Петербург
2021

Оглавление

Введение

По опубликованным Росстатом данным в 2019 доля смертей от заболеваний сердечно-сосудистой системы составляла около 57% [1]. В связи со столь высоким показателем изучение данных заболеваний и способов их предотвращения является крайне актуальной задачей. В связи с этим появляется необходимость изучения и математического моделирования процессов гемодинамики.

Математические модели процессов, происходящих в сердечно-сосудистой системе человека, представляют собой нелинейные системы уравнений в частных производных. Начально-краевые задачи для таких систем записываются на структурах, представляющих собой графы или деревья, состоящие из большого числа рёбер и узлов [2, 3]. Такие структуры моделируют реальные участки сосудистой системы. Из-за наличия нелинейностей такие задачи могут быть решены только с использованием численных методов. В связи с относительно большим числом ребер, такие задачи становятся весьма затратными для проведения расчетов даже на современных вычислительных устройствах. Другой же проблемой является необходимость использования сеток с достаточно большим числом узлов на каждом из участков такой структуры. В связи с этим возникает необходимость использования параллельных вычислений и высокопроизводительных систем с целью ускорения расчетов.

Целью данной работы является исследование возможностей применения высокопроизводительных систем (таких как многопроцессорные вычислительные системы и графические ускорители) к решению задач моделирования кровотока. Для выполнения данной цели поставлены следующие задачи:

- Исследование численных схем, применяемых для решения задач для уравнений гемодинамики.

- Разработка и анализ параллельных алгоритмов решения задач.
- Программная реализация для проведения расчетов с использованием высокопроизводительных систем и графических процессоров, и изучение сложностей, возникающих при реализации параллельных алгоритмов.

Данная работа имеет следующую структуру. В главе 1 рассмотрены математические модели, описывающие течение крови в системе сосудов (§1.1) и численные схемы, применяемые для решения систем уравнений, используемых в данных моделях (§1.2). Рассмотрены технологии применения параллельных вычислений, позволяющие добиться прироста производительности при решении задач для данных систем (§1.3), а также рассмотрены уже существующие программные пакеты, реализующие расчеты на основе различных моделей сосудистых систем (§1.4). В главе 2 предложены параллельные алгоритмы решения задач для уравнений, моделирующих течение крови. В §2.1 в демонстрационных целях предложены алгоритмы для решения уравнения переноса. В §2.2 предложены и реализованы алгоритмы, моделирующие поток крови в модельных системах. Также здесь обсуждаются особенности программной реализации предложенных алгоритмов, описываются возникающие трудности, причины их появления и возможные способы их решения. В данной главе рассмотрены и проанализированы результаты применения данных алгоритмов и проведено сравнение достоинств и недостатков различных методов задействования высокопроизводительных систем. В заключении подведены итоги по выполненной работе и представлены полученные в ней результаты.

Глава 1

Обзор литературы

Математическая модель

Кровь представляет из себя жидкую ткань организма, состоящую из жидкой части — плазмы, и форменных элементов — эритроцитов, лейкоцитов и тромбоцитов. В норме их соотношение составляет 55–60% плазмы и 40–45% форменных элементов [4]. Важно отметить, что считать кровь жидкостью имеет смысл лишь в крупных кровеносных сосудах, таких как вены и артерии. Диаметр же капиллярных сосудов сопоставим по размерам с размерами эритроцитов, что не позволяет рассматривать кровь в качестве жидкости. Плазма крови сама по себе является ньютоновской жидкостью, однако кровь в целом считается жидкостью неньютоновской, что обуславливается наличием в ней форменных элементов [5].

Для описания течения крови применяется трехмерные уравнения:

$$\mathbf{div}\ \boldsymbol{v} = 0, \tag{1.1}$$

$$\rho\frac{d\boldsymbol{v}}{dt} = \rho\mathbf{f} + \mathbf{div}\ \mathbf{T}, \tag{1.2}$$

где $\boldsymbol{v}$ — вектор скорости, ρ — плотность, $\mathbf{f}$ — вектор массовых сил, $\mathbf{T}$ — тензор напряжений [6]. Здесь уравнение (1.1) представляет из себя уравнение несжимаемости, а уравнение (1.2) — уравнение движения. Однако данная модель слабо годится для моделирования систем с большим количеством сосудов, так как проведение расчетов на основе такой модели требует использования большого числа вычислительных ресурсов. В связи с этим введем также ряд упрощений [7]:

- Кровеносный сосуд рассматривается в виде цилиндра с переменным

радиусом.

- Предполагается осевая симметрия, рассматриваемые величины не зависят от угла.
- Радиальная компонента вектора скорости полагается малой по сравнению с осевой.
- Давление считается постоянным вдоль поперечного сечения: $p = p(t, z)$
- Ось цилиндра считается несмещающейся по времени, происходят лишь радиальные смещения стенок сосуда.
- Внешние воздействия на сосуд не учитываются.

Данные упрощения позволяют получить одномерную модель течения крови в сосуде, которая несмотря на упрощения позволяет получить результаты, согласующиеся с трехмерной моделью [8]. Таким образом получаем модель, описываемую системой уравнений:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial z} = 0,$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial z}\left(\alpha \frac{Q^2}{A}\right) + \frac{A}{\rho}\frac{\partial P}{\partial z} = f(A, Q),$$

где P — давление $A(t, z)$ — площадь поперечного сечения сосуда, $Q(t, z)$ — объемный расход жидкости, а $f(A, Q)$ — функция учитывающая вязкие свойства крови.

В квазилинейной форме данная система будет иметь следующий вид:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{H}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial z} = \mathbf{f}(\mathbf{U}),$$

где

$$\mathbf{U} = \begin{pmatrix} A \\ Q \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 0 & 1 \\ \frac{A}{\rho}\frac{dP}{dA} - \alpha(\frac{Q}{A})^2 & 2\alpha\frac{Q}{A} \end{pmatrix}.$$

Запишем данную систему в следующем виде:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial z} = \mathbf{f}(\mathbf{U}), \tag{1.3}$$

где

$$\mathbf{F}(\mathbf{U}) = \begin{pmatrix} Q \\ \alpha \frac{Q^2}{A} + \int\limits_{A_{min}}^{A_{max}} \frac{\tau}{\rho} \frac{dP}{dA}(\tau) d\tau \end{pmatrix},$$

$$\mathbf{f}(\mathbf{U}) = \begin{pmatrix} 0 \\ f(A, Q) \end{pmatrix}.$$

Разностные схемы

Для численного решения задач для системы уравнений (1.3) в работе используются разностные схемы первого и второго порядков, часто применяемые в литературе. Более подробную информацию о различных методах, применяемых при решении подобных задач можно найти в работе [9]. В качестве схем первого порядка были рассмотрены:

- Схема Лакса–Фридрихса [9]:

$$\mathbf{U}_i^{s+1} = \frac{1}{2}\left(\mathbf{U}_{i+1}^s + \mathbf{U}_{i-1}^s\right) - \frac{\Delta t}{2h}\left(\mathbf{F}\left(\mathbf{U}_{i+1}^s\right) - \mathbf{F}\left(\mathbf{U}_{i-1}^s\right)\right) + \\ + \frac{\Delta t}{2}\mathbf{f}\left(\frac{1}{2}\left(\mathbf{U}_{i+1}^s + \mathbf{U}_{i-1}^s\right)\right). \tag{1.4}$$

  Здесь и далее $\mathbf{U}_i^s \approx \mathbf{U}(s\Delta t, ih)$; Δt и h — шаги разбиения по времени и пространственной координате соответственно.

- Схема с применением потока Русанова [9]:

$$\mathbf{U}_i^{s+1} = \mathbf{U}_i^s - \frac{\Delta t}{h}\left(\frac{1}{2}\left(\mathbf{F}\left(\mathbf{U}_i^s\right) + \mathbf{F}\left(\mathbf{U}_{i+1}^s\right)\right) + \frac{1}{2}|\lambda|\left(\mathbf{U}_i^s - \mathbf{U}_{i+1}^s\right)\right) - \\ - \frac{\Delta t}{h}\left(\frac{1}{2}\left(\mathbf{F}\left(\mathbf{U}_{i-1}^s\right) + \mathbf{F}\left(\mathbf{U}_i^s\right)\right) + \frac{1}{2}|\lambda|\left(\mathbf{U}_{i-1}^s - \mathbf{U}_i^s\right)\right) + \Delta t \mathbf{f}(\mathbf{U}_i^s), \tag{1.5}$$

где $\lambda = \max\limits_k (|\lambda_k(\mathbf{U}_i^s)|)$, $\lambda_k(\mathbf{U}_i^s)$ — собственные значения матрицы $\mathbf{H}$.

В качестве схем второго порядка были рассмотрены двухшаговые схемы:

- Схема Лакса – Вендроффа [10]. На первом шаге производится расчет значений $\mathbf{U}$ в узлах с полуцелыми индексами:

$$\mathbf{U}_{i+\frac{1}{2}}^{s+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{U}_i^s + \mathbf{U}_{i+1}^s\right) - \frac{\Delta t}{2h}\left(\mathbf{F}\left(\mathbf{U}_{i+1}^s\right) - \mathbf{F}\left(\mathbf{U}_{i+1}^s\right)\right) + \\ + \frac{\Delta t}{2}\mathbf{f}\left(\frac{1}{2}\left(\mathbf{U}_i^s + \mathbf{U}_{i+1}^s\right)\right), \tag{1.6}$$

  а затем в узлах с целыми индексами:

$$\mathbf{U}_i^{s+1} = \mathbf{U}_i^s - \frac{\Delta t}{h}\left(\mathbf{F}\left(\mathbf{U}_{i+\frac{1}{2}}^{s+\frac{1}{2}}\right) - \mathbf{F}\left(\mathbf{U}_{i-\frac{1}{2}}^{s+\frac{1}{2}}\right)\right) + \\ + \Delta t\mathbf{f}\left(\frac{1}{2}\left(\mathbf{U}_i^s + \mathbf{U}_{i-1}^s\right)\right). \tag{1.7}$$

- Схема Мак-Кормака [10]. Данная схема состоит из двух этапов: предиктора, имеющего вид

$$\overline{\mathbf{U}}_i^{s+1} = \mathbf{U}_i^s - \frac{\Delta t}{h}\left(\mathbf{F}\left(\mathbf{U}_{i+1}^s\right) - \mathbf{F}\left(\mathbf{U}_i^s\right)\right) + \Delta t\mathbf{f}(\mathbf{U}_i^s) \tag{1.8}$$

  и корректора, имеющего вид

$$\mathbf{U}_i^{s+1} = \frac{1}{2}\left(\mathbf{U}_i^s + \overline{\mathbf{U}}_i^{s+1}\right) - \frac{\Delta t}{2h}\left(\mathbf{F}\left(\mathbf{U}_i^s\right) - \mathbf{F}\left(\mathbf{U}_{i-1}^s\right)\right) + \frac{\Delta t}{2}\mathbf{f}(\mathbf{U}_i^s). \tag{1.9}$$

Технологии параллельных вычислений

В качестве технологий, позволяющих использовать высокопроизводительные вычисления, в данной работе рассматриваются интерфейсы прикладного программирования OpenMP (предназначенный для создания многопоточных приложений для систем с общей памятью) и CUDA (предназначенная для создания приложений с использованием графических ускорителей).

OpenMP представляет из себя открытый стандарт программирования, включающий в себя спецификации набора директив компилятора, вспомогательных функций и переменных среды. Данный стандарт применим к языкам Фортран и C/C++. В данном стандарте параллельные вычисления осуществляются при помощи многопоточности, позволяя при помощи директив распределить выполнение задачи на некоторое число потоков, выполняющихся параллельно на машине с несколькими процессорами [11]. Подробное описание принципов работы с OpenMP, а также объяснение различных сложностей, возникающих при применении данного стандарта могут быть найдены в работе [12]. В работе же [11] представлено краткое описание доступных для применения директив, а также приведены примеры их использования.

CUDA представляет из себя интерфейс, позволяющий реализовывать (на упрощенных диалектах языков C/C++ и Фортран) алгоритмы, с целью их последующего выполнения на графических процессорах Nvidia. В отличие от центрального процессора, в котором число ядер, позволяющих вести одновременное вычисление различных данных, исчисляется десятками, в современных графических процессорах используется от нескольких сотен до тысяч ядер. Несмотря на то, что ядра графических процессоров обычно менее производительны, чем ядра центрального процессора, большее их количество компенсирует этот недостаток. В работе [13] подробно рассмотрено как устройство графических процессоров NVIDIA, так и практическое применение технологии CUDA для разработки програм. В этой работе приведены примеры решения некоторых задач и рассмотрены средства отладки и диагностики программ.

В работе [14] больший аспект уделяется процессу написания программ с применением CUDA. Однако большое количество примеров позволяет рассмотреть и объяснить некоторые особенности использования CUDA и помочь выяснить, какие методы наиболее подходят для решения определенных задач. В процессе разбора некоторых примеров также объясняется и архитектура работы графических процессоров.

Модельные задачи

В работе были рассмотрены две задачи моделирования кровотока. Первая задача была предложена в данной работе, а вторая была взята из известной статьи. В качестве тестового примера была рассмотрена задача для системы, состоящей из 12 сосудов, представленной на рисунке 1.1. Данная система состоит из сосудов, имеющих одинаковые параметры длины ($L = 20$ см) и начальной площади ($A = \frac{\pi}{24}$ см^2). Также в данной системе рассматривается невязкая модель течения крови, в связи с чем $f(A, Q) = 0$. В качестве начального значения поток задается равным нулю. Моделированное данной системы проводилось для времени $t = 1$ с.

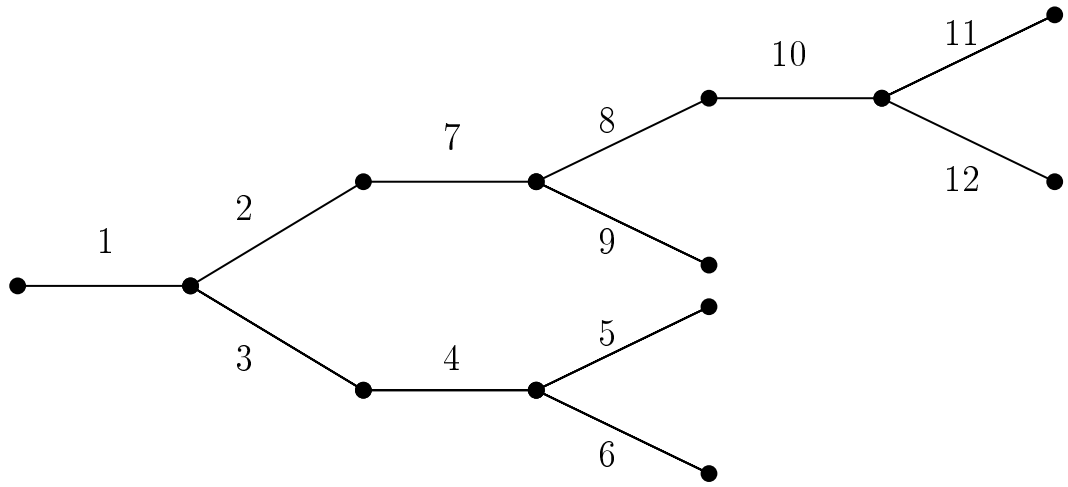

Рис. 1.1: Модель системы из 12 сосудов

В качестве входного значения потока была задана функция

$$Q(t, 0) = \frac{\pi}{8} \cdot \exp\left(-5000(t - 0.05)^2\right),$$

а также для нахождения значения $A(t, 0)$ поставлено условие совместности, описывающее поведение системы вдоль характеристики, покидающей область [7]:

$$\mathbf{l}_1 \left(\frac{\partial \mathbf{U}}{\partial t} + \mathbf{H} \frac{\partial \mathbf{U}}{\partial z} \right) = 0 \iff \mathbf{l}_1 \frac{\partial \mathbf{U}}{\partial t} + \lambda_1 \mathbf{l}_1 \frac{\partial \mathbf{U}}{\partial z} = 0.$$

В узлах, отвечающих выходам из системы было поставлено неотражающее условие, позволяющее без отражения выйти решению в виде волны за границу области [7]:

$$\mathbf{l}_1 \frac{\partial \mathbf{U}}{\partial t} = 0, \tag{1.10}$$

и условие совместности:

$$\mathbf{l}_2 \left(\frac{\partial \mathbf{U}}{\partial t} + \mathbf{H} \frac{\partial \mathbf{U}}{\partial z} \right) = 0 \iff \mathbf{l}_2 \frac{\partial \mathbf{U}}{\partial t} + \lambda_2 \mathbf{l}_2 \frac{\partial \mathbf{U}}{\partial z} = 0.$$

Здесь и далее $\mathbf{l}_1$ и $\mathbf{l}_2$ собственные вектора матрицы $\mathbf{H}$ отвечающие собственным значениям $\lambda_1(\mathbf{U})$ и $\lambda_2(\mathbf{U})$.

В точке бифуркации сосудов были поставлены следующие условия:

$$\begin{aligned}
&Q_1 = Q_2 + Q_3, \\
&P_2 = P_1, \\
&P_3 = P_1, \\
&\mathbf{l}_2(\mathbf{U}_1) \left(\frac{\partial \mathbf{U}_1}{\partial t} + \lambda_2(\mathbf{U}_1) \mathbf{l}_2(\mathbf{U}_1) \frac{\partial \mathbf{U}_1}{\partial z} \right) = 0, \\
&\mathbf{l}_1(\mathbf{U}_2) \left(\frac{\partial \mathbf{U}_2}{\partial t} + \lambda_1(\mathbf{U}_2) \mathbf{l}_1(\mathbf{U}_2) \frac{\partial \mathbf{U}_1}{\partial z} \right) = 0, \\
&\mathbf{l}_1(\mathbf{U}_3) \left(\frac{\partial \mathbf{U}_3}{\partial t} + \lambda_1(\mathbf{U}_3) \mathbf{l}_1(\mathbf{U}_3) \frac{\partial \mathbf{U}_1}{\partial z} \right) = 0.
\end{aligned} \tag{1.11}$$

Первое из этих условий указывает на необходимость равенства потока до прохождения точки бифуркации сумме потоков после её прохождения. Второе и третье условие — это условия непрерывности давления. Оставшиеся три — условия совместности.

В точке соединения сосудов:

$$\begin{aligned}
&P_1 = P_2, \\
&Q_1 = Q_2, \\
&\mathbf{l}_2(\mathbf{U}_1) \left(\frac{\partial \mathbf{U}_1}{\partial t} + \lambda_2(\mathbf{U}_1) \mathbf{l}_2 \frac{\partial \mathbf{U}_1}{\partial z} \right) = 0, \\
&\mathbf{l}_1(\mathbf{U}_2) \left(\frac{\partial \mathbf{U}_2}{\partial t} + \lambda_1(\mathbf{U}_2) \mathbf{l}_1 \frac{\partial \mathbf{U}_2}{\partial z} \right) = 0.
\end{aligned} \tag{1.12}$$

Здесь первое условие является условием непрерывности давления. Второе

— условие сохранения расхода, и оставшиеся два — условия совместности.

В качестве системы, моделирующей часть сердечно-сосудистой системы рассматривается совокупность из 37 сосудов различных длин и начальных площадей (рисунок 1.2) [2]. Для данной модели все числовые характеристики (такие как например длины сосудов L и их радиусы R_{i0} и R_{i1}) взяты из [2]. В качестве начального значения поток задается равным нулю, а начальные значения площади сосудов задаются как $A_i(0,z) = \pi\left(\frac{R_{i1}-R_{i0}}{L}z + R_{i0}\right)^2$. Для данной задачи моделирование проводилось для времени $t = 15$ с.

Значение потока на входе в систему (на верхнем конце сосуда 1 (см. рис. 1.2)) задано модельной функцией из [2]:

$$Q(t,0) = f(t),$$

описывающей поток использовавшийся при проведении эксперимента, а также на этом конце задается условие совместности для нахождения значения $A_1(t,0)$:

$$\mathbf{l}_1\left(\frac{\partial \mathbf{U}}{\partial t} + \mathbf{H}\frac{\partial \mathbf{U}}{\partial z} - \mathbf{f}(\mathbf{U})\right) = 0 \iff \mathbf{l}_1\frac{\partial \mathbf{U}}{\partial t} + \lambda_1\mathbf{l}_1\frac{\partial \mathbf{U}}{\partial z} - \mathbf{l}_1\mathbf{f}(\mathbf{U}) = 0.$$

Для узлов, отвечающих выходам из системы сосудов в рассматриваемой модели сети, было поставлено условие, отвечающее наличию давления в периферийной части сердечно-сосудистой системы [2]:

$$Q = \frac{P - P_{out}}{R_p},$$

где P_{out} — постоянное давление равное 426 Па (это значение отвечает давлению на периферии системы сосудов), R_p — периферическое сопротивление потоку, взятое из [2], а также задано условие совместности.

В точке бифуркации сосудов были поставлены практически идентичные (1.11) условия, за исключением наличия ненулевого значения $\mathbf{f}(\mathbf{U})$ в условиях совместности, а в точке же соединения сосудов были поставлены условия, практически идентичные (1.12) условия, с учетом ненулевого значения $\mathbf{f}(\mathbf{U})$.

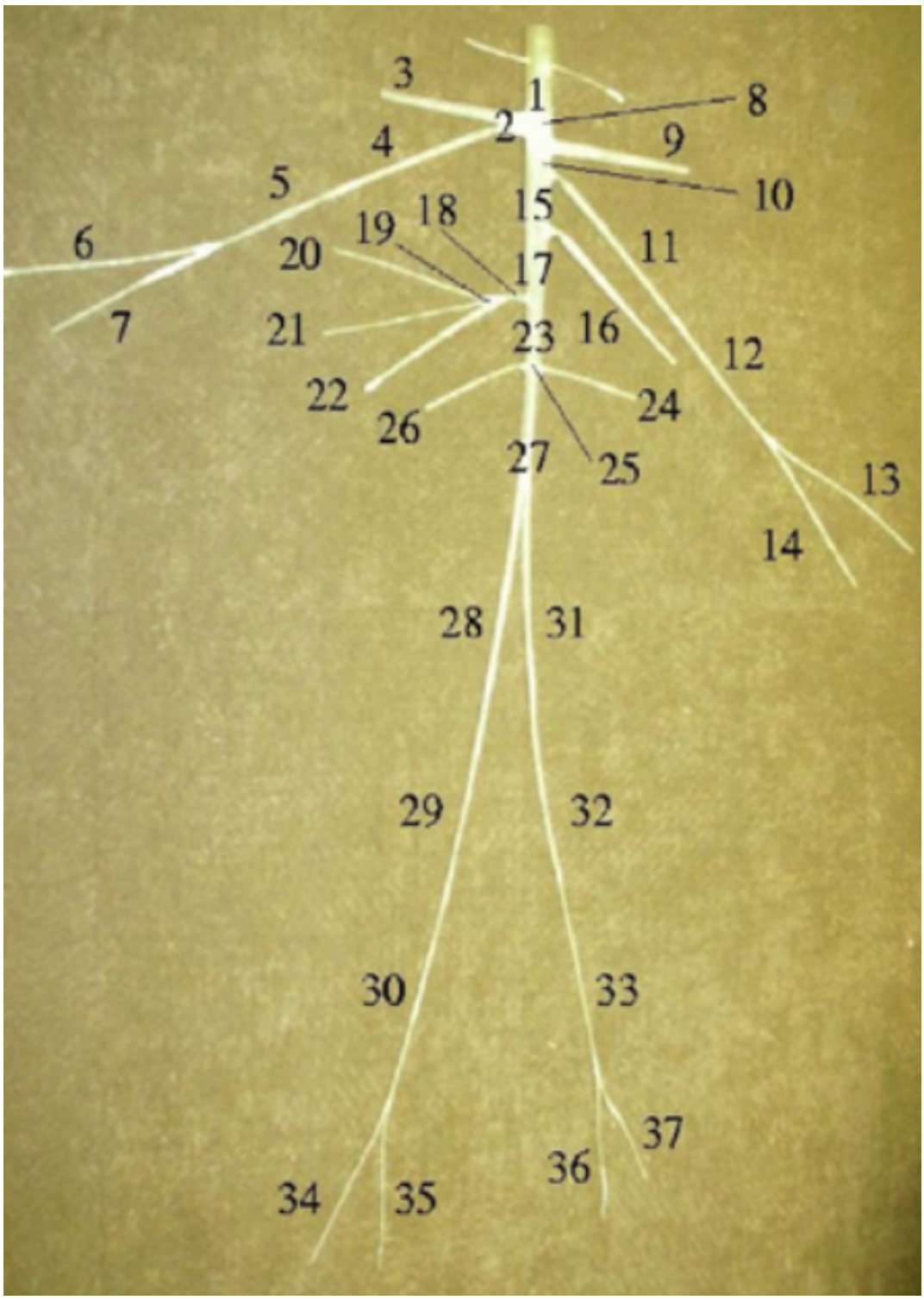

Рис. 1.2: Модель системы сосудов [2]

Существующие программные пакеты

На данный момент уже существуют программные пакеты, позволяющие проводить расчеты с использованием одномерной модели крови. Так, например, пакет на языке Python Artery.FE позволяет проводить подобные вычисления с применением фреймворка FEniCS. Данный пакет позво-

ляет описать геометрию сосудистой системы и, используя схему Кранка–Николсона произвести моделирование данной системы [15].

Помимо этого также существует пакет на языке Python VaMpy [16]. Данный пакет позволяет проводить моделирование системы с применением метода Лакса–Вендроффа.

Однако минусом данных пакетов является то, что несмотря на высокую ресурсоемкость решаемой ими задачи, они не применяют методы высокопроизводительных вычислений, позволяющие значительно увеличить быстродействие.

Вывод

Рассмотренная в данной главе литература позволяет сделать следующий вывод: моделирование кровотока является актуальной и весьма ресурсоемкой задачей, однако использование одномерной модели моделирования кровотока позволяет добиться достаточно точных результатов. При этом существующие программные пакеты не используют методы высокопроизводительных вычислений, что с учетом вычислительной сложности данной задачи служит обоснованием высокой актуальности данной работы.

Глава 2

Параллельные алгоритмы и их реализация

Уравнение переноса

В целях демонстрации основных принципов применения высокопроизводительных систем для ускорения расчетов в первую очередь рассмотрим их применение к решению уравнения переноса:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \tag{2.1}$$

со следующим начальным условием:

$$u(x, 0) = f(x). \tag{2.2}$$

Для решения данной задачи используется явная разностная схема:

$$\frac{u_n^{k+1} - u_n^k}{\Delta t} + c\frac{u_n^k - u_{n-1}^k}{h} = 0. \tag{2.3}$$

Можно предложить два варианта алгоритмов, использующих высокопроизводительные системы для численного решения такого уравнения. Суть первого алгоритма состоит в том, что значения в узлах сетки на одном шаге по времени вычисляются независимо друг от друга на различных процессорах вычислительной системы. Суть второго алгоритма же состоит в том, что сетка разбивается на некоторое число участков, и каждый из процессоров системы выполняет вычисления для своего участка. Таким образом данные алгоритмы имеют следующий вид:

Алгоритм 1:

1. Задаются шаги сетки h, τ так, что $\tau = \frac{h}{c}$ и конечный момент времени

T.

2. В параллельном по i режиме вычисляются значения по формуле

$$u_i^{n+1} = u_i^n - \tau c \frac{u_i^n - u_{i-1}^n}{h}.$$

3. Если $t_{\text{тек}} >= T$, то вычисления завершаются. Иначе $n = n+1$, $t_{\text{тек}} = t_{\text{тек}} + \tau$ и возвращаемся к п. 2.

Алгоритм 2:

1. Задаются шаги сетки h, τ так, что $\tau = \frac{h}{c}$ и и конечный момент времени T.

2. Сетка разбивается на $N_{\text{уч}}$ участков $(a_j; b_j]$ содержащих $N_{\text{уз}}$ узлов каждый, значения решения в узла каждого участка вычисляются независимо от других участков.

3. Для $x_i \neq a_j$ вычисляются значения по формуле

$$u_{ij}^{n+1} = u_{ij}^n - \tau c \frac{u_{ij}^n - u_{i-1}^n}{h},$$

где $i = 1, \ldots, N_{\text{уз}}$, $j = 1, \ldots, N_{\text{уч}}$.

4. Для $x_i = a_j$ считаем $u_{1j} = u_{N_{\text{уз}} j-1}$.

5. Если $t_{\text{тек}} >= T$, то вычисления завершаются. Иначе $n = n+1$, $t_{\text{тек}} = t_{\text{тек}} + \tau$ и возвращаемся к п. 2.

Данные алгоритмы были реализованы на языке C++ с применением OpenMP и CUDA для их использования на высокопроизводительных системах. Ниже представлен реализующий основные моменты данных алгоритмов код:

Листинг 2.1: Реализация алгоритма 1 с применением OpenMP

```
1 double currentTime = 0;
2 while (currentTime < t)
3 {
4   int i;
```

```
5   #pragma omp parallel for shared(Unew,Uold) private(i)
6   for (i = 1; i<N; i++){
7    Unew[i] = Uold[i] - tau*c*(Uold[i]-Uold[i-1])/h;
8   }
9   currentTime += tau;
10 }
```

Листинг 2.2: Реализация алгоритма 2 с применением CUDA

```
1  __global__ void CalculateU(double *Unew, double *Uold, double tau, double
       c,double h)
2  {
3   int i = blockIdx.x * BSIZE + threadIdx.x;
4   for (int j = 0; j<MULT; j++){
5    int index = i+1024*j;
6    Unew[index] = Uold[index] - tau*c*(Uold[index]-Uold[index-1])/h;
7   }
8  }
9  double currentTime = 0;
10 while (currentTime < t)
11 {
12  CalculateU<<<grid,threads>>>(d_Unew, d_Uold,tau,c,h);
13  currentTime += tau;
14 }
```

При реализации алгоритма с применением CUDA возникла сложность, связанная с тем, что для расчета следующего шага по времени требуется использовать данные, полученные на предыдущем шаге, однако копирование данных в переменную Uold значительно замедляет работу программы. В данном случае необходимости копирования данных можно легко избежать, чередуя какая из переменных Uold и Unew используется для хранения значений предыдущего шага, а какая используется для записи вычисленных данных.

Расчеты для OpenMP производились на процессоре Intel Xeon X5365, для CUDA расчеты производились на системе с процессором Intel Pentium 3805U и графическим процессором Nvidia GeForce 920M. Ниже приведено время работы (в секундах) программы T_{OMP_k}, реализующей алгоритм 1 при $N = 51200$ и k потоков:

$$T_{OMP_1} = 12.5759, \quad T_{OMP_2} = 13.0835, \quad T_{OMP_3} = 9.83985, \quad T_{OMP_4} = 7.82068,$$
$$T_{OMP_5} = 6.61141, \quad T_{OMP_6} = 6.21981, \quad T_{OMP_7} = 6.04904, \quad T_{OMP_8} = 5.27881.$$

Как можно видеть, использование 8 ядер OpenMP позволило добиться прироста производительности в 2,5 раза.

Для алгоритма 2 при $N = 51200$ представлено время расчетов на CPU, а также с применением одного или двух мультипроцессоров (так как методика запуска вычислений с применением CUDA не позволяет контролировать запуск с точностью до 1 ядра (на системе, на которой производились вычисления 1 мультипроцессор содержит 192 ядра)) :

$$T_{CPU} = 8.30366, \quad T_{CUDA_1} = 1.6657, \quad T_{CUDA_2} = 1.04007.$$

Для алгоритма 2 при $N = 1024$:

$$T_{CPU} = 0.1146, \quad T_{CUDA_1} = 0.2582, \quad T_{CUDA_2} = 0.2509.$$

Для CUDA также были рассмотрены методы, которыми можно увеличить эффективность её применения. Так как запуск вычислений сопровождается накладными расходами, следует минимизировать число запусков по сравнению с числом операций, что может быть достигнуто, например, увеличением числа узлов сетки. В более сложных задачах вычислений соответственно будет больше в связи с чем отпадет необходимость использовать большое количество узлов сетки. В итоге использование CUDA позволило добиться максимального ускорения вычислений в 8 раз.

Модель системы сосудов

Используя схожий алгоритм, возможно произвести и разбиение более сложной задачи моделирования кровотока. Как и в случае с уравнением переноса, возможно осуществлять разбиение вычислений двумя способами: по узлам сетки и по сосудам системы.

Для получения значений A и Q в точках бифуркации система (1.11) дискретизовалась. В результате дискретизации получается система нели-

нейных алгебраических уравнений следующего вида:

$$F(\overline{\mathbf{U}}) = \begin{cases} f_1(A_1, A_2, A_3, Q_1, Q_2, Q_3), \\ f_2(A_1, A_2, A_3, Q_1, Q_2, Q_3), \\ \quad \vdots \\ f_6(A_1, A_2, A_3, Q_1, Q_2, Q_3); \end{cases}$$

$$F(\overline{\mathbf{U}}) = 0.$$

Которая решалась методом Ньютона:

$$\overline{\mathbf{U}}^{(k+1)} = \overline{\mathbf{U}}^{(k)} - W^{-1}(\overline{\mathbf{U}}^{(k)}) \cdot F(\overline{\mathbf{U}}^{(k)}),$$

здесь

$$W(\overline{\mathbf{U}}) = \begin{pmatrix} \dfrac{\partial f_1(\overline{\mathbf{U}})}{\partial A_1} & \cdots & \dfrac{\partial f_1(\overline{\mathbf{U}})}{\partial Q_3} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n(\overline{\mathbf{U}})}{\partial A_1} & \cdots & \dfrac{\partial f_n(\overline{\mathbf{U}})}{\partial Q_3} \end{pmatrix};$$

а значения $W^{-1}(\overline{\mathbf{U}}^{(k)}) \cdot F(\overline{\mathbf{U}}^{(k)})$ вычислялись с применением LU разложения матрицы $W(\overline{\mathbf{U}})$. Так как при вычислении LU разложения осуществляется деление на элемент $W_{1,1}$ матрицы $W(\overline{\mathbf{U}})$ крайне важно записывать условия в таком порядке, чтобы данный элемент не оказался равным нулю. Аналогичным образом дискретизуя систему (1.12) получаем систему нелинейных алгебраических уравнений, решение которой методом Ньютона позволяет найти значения A и Q в точках соединения сосудов.

Таким образом, алгоритм, осуществляющий решение задачи для системы сосудов, будет иметь следующий вид:

1. Выбрать число шагов по времени m, $\tau = \frac{T}{m-1}$ и число узлов n в которых будут вестись расчеты значения $\mathbf{U}$ для каждого из сосудов, $h_k = \frac{L_k}{n-1}$, $k = 1, \ldots, N$.

2. Положить счетчик шагов по времени s равным нулю.

3. Используя формулы (1.4) – (1.9) вычислить значения $\mathbf{U}$ в узлах сосудов на шаге времени s.

4. Используя метод Ньютона для нахождения решения системы (1.11) вычислить значения $\mathbf{U}$ в узлах бифуркации.

5. Расчитать значения $\mathbf{U}$ в точках соединения сосудов.

6. Если $(s+1)\tau > T$, то вычисления завершить, иначе положить $s = s+1$ и повторить шаги 3-5.

В связи с тем, что для тестовой задачи из 12 сосудов разница в работе программ, реализующих данный алгоритм при разбиении вычислений между процессорами по узлам сосудов и по сосудам целиком составила порядка 10^{-3} времени работы программы, в дальнейшем вычисления проводились лишь при разбиении по сосудам.

Данный алгоритм численного решения представленной задачи для модели из 12 сосудов был реализован в виде программы на языке C++. В связи с тем, что для расчетов значений в узлах сосудов на шаге 3 не требуется знание значений в узлах сосуда, то вычисления можно производить параллельно. Также в связи с тем, что вычисления проводятся для каждого из n узлов, этот шаг в программе использует большую часть вычислительных ресурсов, что подкрепляет необходимость производить вычисления на этом этапе в параллельном режиме. В связи с этим был реализован параллельный алгоритм, идентичный приведенному выше, с применением OpenMP.

С использованием представленных в §1.2 схем были проведены расчеты для различного числа узлов сетки n и получены результаты, представленные в виде графиков на рисунках 2.1–2.3. Для каждой из схем построены два графика, первый из них показывает зависимость ускорения от числа узлов сетки для различного числа потоков, второй же показывает эффективность распараллеливания. Как можно видеть из представленных графиков, для всех рассмотренных схем эффективность увеличивается с увеличением числа узлов сетки. Так, например, для сетки $n = 200$ узлов наблюдается наименьшая эффективность для всех рассмотренных схем. Это объясняется тем, что при увеличении числа узлов сетки также изменяется отношение выполняемых параллельно расчетов к последовательным. Наибольшая эффективность достигается при использовании схемы Лакса–Вендроффа, что объясняется тем, что данная схема, в отличие от других,

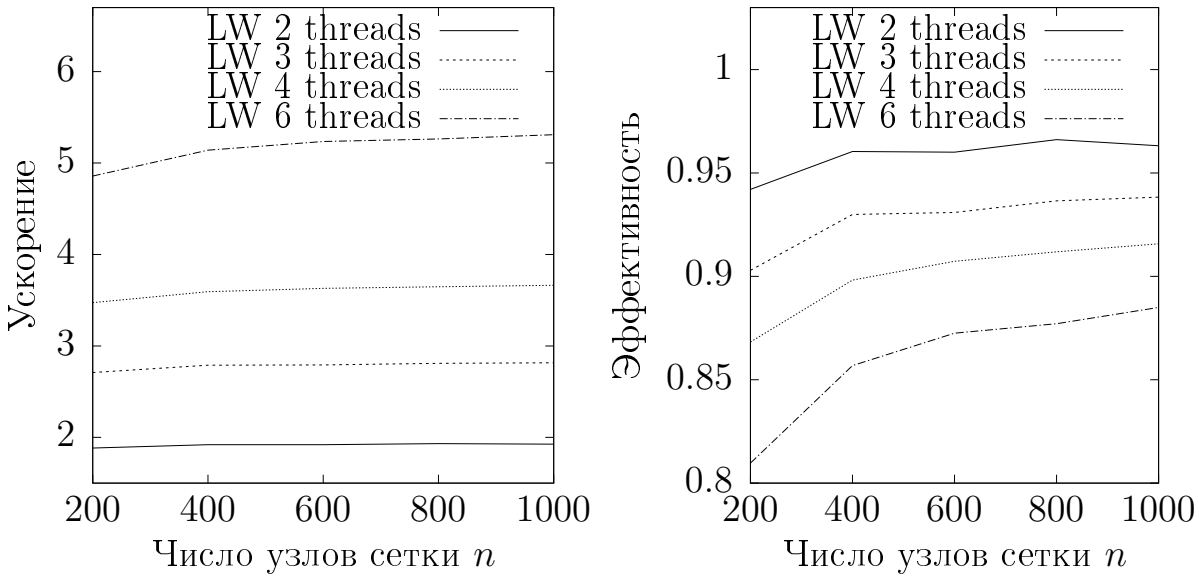

Рис. 2.1: Ускорение и эффективность для схемы Лакса–Вендроффа в зависимости от числа узлов сетки и потоков OpenMP

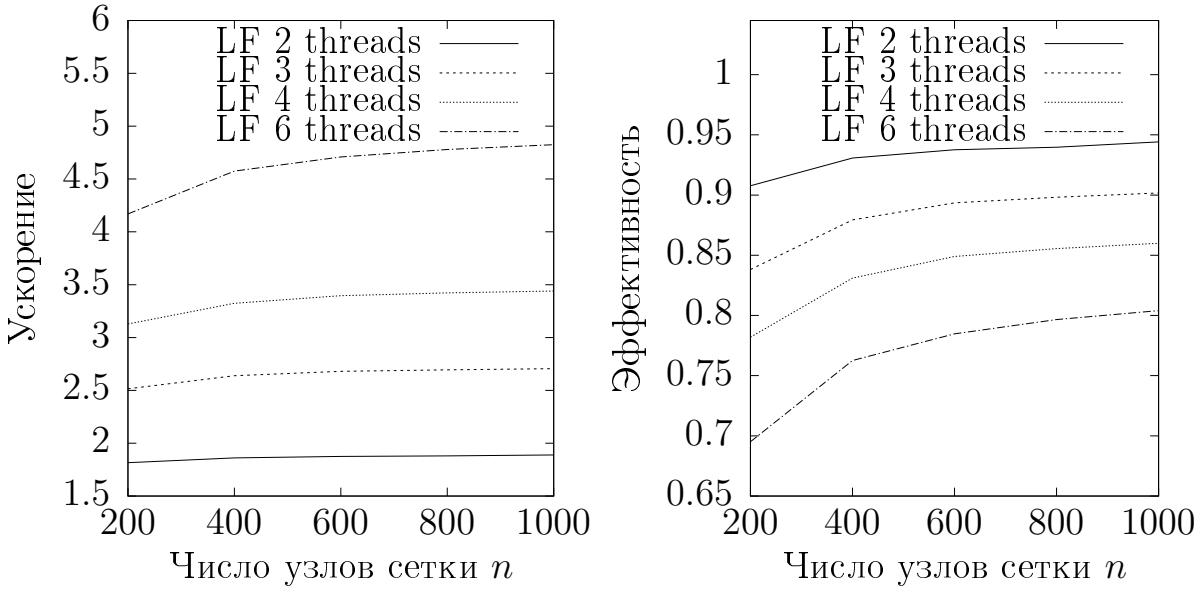

Рис. 2.2: Ускорение и эффективность для схемы Лакса–Фридрихса в зависимости от числа узлов сетки и потоков OpenMP

является двухшаговой, следовательно в ней наблюдается наиболее высокое отношение части алгоритма, выполняемого параллельно, к части, выполняемой последовательно. Снижение же эффективности при увеличении числа ядер обусловленно тем, что некоторая доля вычислений в данном алгоритме может быть произведена лишь последовательными расчетами (а именно вычисление значений для шагов по времени), что иллюстрирует закон Амдала.

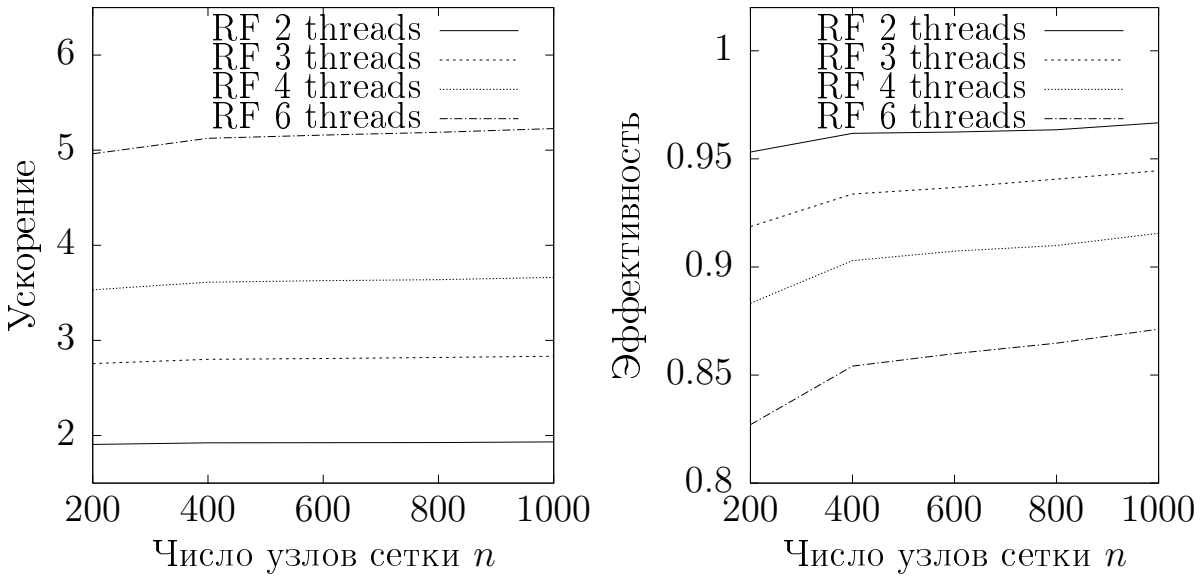

Рис. 2.3: Ускорение и эффективность для потока Русанова в зависимости от числа узлов сетки и потоков OpenMP

Для модели из 37 сосудов данный алгоритм был реализован в виде программы на языке C++, а также были написаны его реализации с применением OpenMP и CUDA. В связи с тем, что на модели из 12 сосудов наибольшая эффективность была получена при помощи двухшаговой схемы Лакса–Вендроффа, на данном этапе также было решено применять двухшаговую схему. Однако в связи с необходимостью вычисления в схеме Лакса–Вендроффа значений в полуцелых узлах была выбрана двухшаговая схема Мак-Кормака, являющаяся вариацией схемы Лакса–Вендроффа.

Для запуска вычислений с применением CUDA необходимо перед выполнением шага 3 осуществить копирование данных, полученных для предыдущего шага по времени в память графического адаптера, а также после этого шага осуществить копирование результатов вычислений в оперативную память. Так как графический процессор поддерживает одновременное выполнение копирования с проведением расчётов (так называемые асинхронные функции копирования) был проведен запуск копирования и вычислений различными методами. Первый рассмотренный метод (метод 1) подразумевал схему `COPYIN(1) CALCULATE(1) COPYOUT(1) COPYIN(2) ... COPYOUT(12)`, где `COPYIN(1)` — функция асинхронного копирования данных для первого сосуда, `CALCULATE(1)` - функция вычислений и `COPYOUT(1)` — функция асинхронного копирования результатов. Второй же рассмотренный метод (метод 2) подразумевал схему `COPYIN(1)`

`... COPYIN(12) CALCULATE(1) ... CALCULATE(12) COPYOUT(1) ... COPYOUT(12)`. Помимо этого был рассмотрен запуск программы без применения асинхронных функций копирования.

При запуске вычислений методом 1 вычисления заняли $T_{CUDA1} = 19355$ мс. Это связано с тем, что запуск вычислений для следующих сосудов осуществлялся лишь после того, как было завершено копирование результатов вычислений предыдущего сосуда. Аналогичное время выполнения было получено и без применения асинхронных функций копирования памяти. Время выполнения программы методом 2 составило $T_{CUDA2} = 4492$ мс. На рисунках 2.4,2.5 показана временная шкала выполнения функций на одном шаге итерации (серо-желтым указаны функции копирования, а синим функции вычисления). Таким образом данное незначительное изменение программы может позволить добиться значительного прироста производительности.

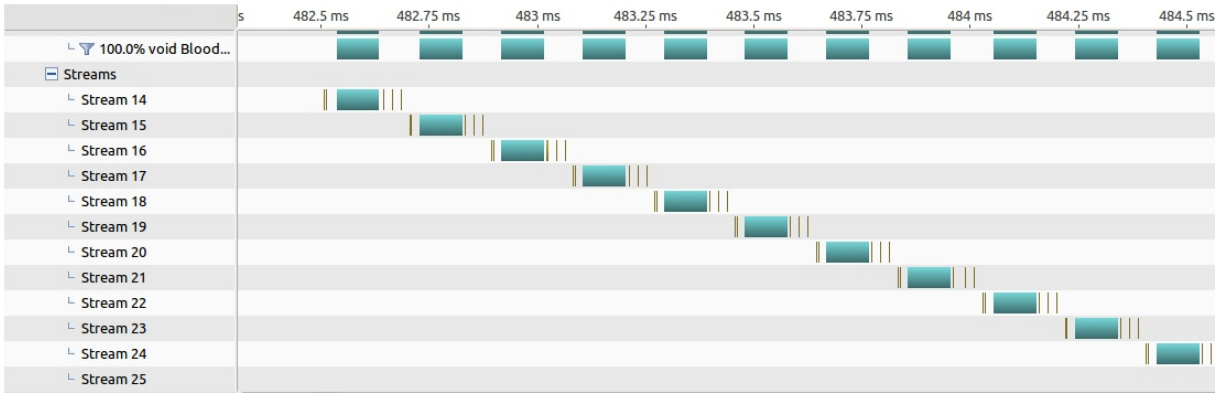

Рис. 2.4: Выполнение функций методом 1

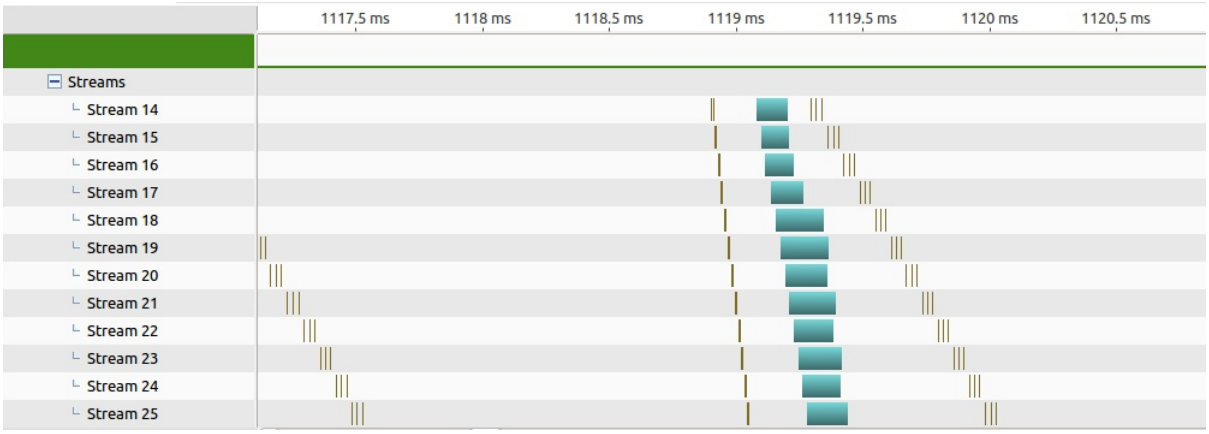

Рис. 2.5: Выполнение функций методом 2

В связи с особенностями CUDA необходимость вычисления значе-

ний в узлах, отвечающих выходам из системы сосудов в рассматриваемой модели сети, методом Ньютона сильно ухудшала производительность полученной программы. Для решения данной проблемы при помощи Maple было получено точное решение данной системы, которое и применялось в данной программе.

Были проведены расчеты для различного числа узлов сетки n, а также различного числа ядер OpenMP. Как можно видеть из графиков, представленных на рисунках 2.6–2.9, полученные результаты незначительно отличаются от результатов, представленных в [2]. Данные расхождения объясняются различием методов, рассмотренных в этой работе, а также различиями в шаге разбиений, выбранных для расчетов. На рисунке 2.10 представлены графики ускорения и эффективности в зависимости от числа узлов сетки. Как можно видеть — при увеличении числа узлов возрастает как ускорение, так и эффективность, при этом наибольшей эффективностью обладает программа, выполняемая с применением двух потоков OpenMP.

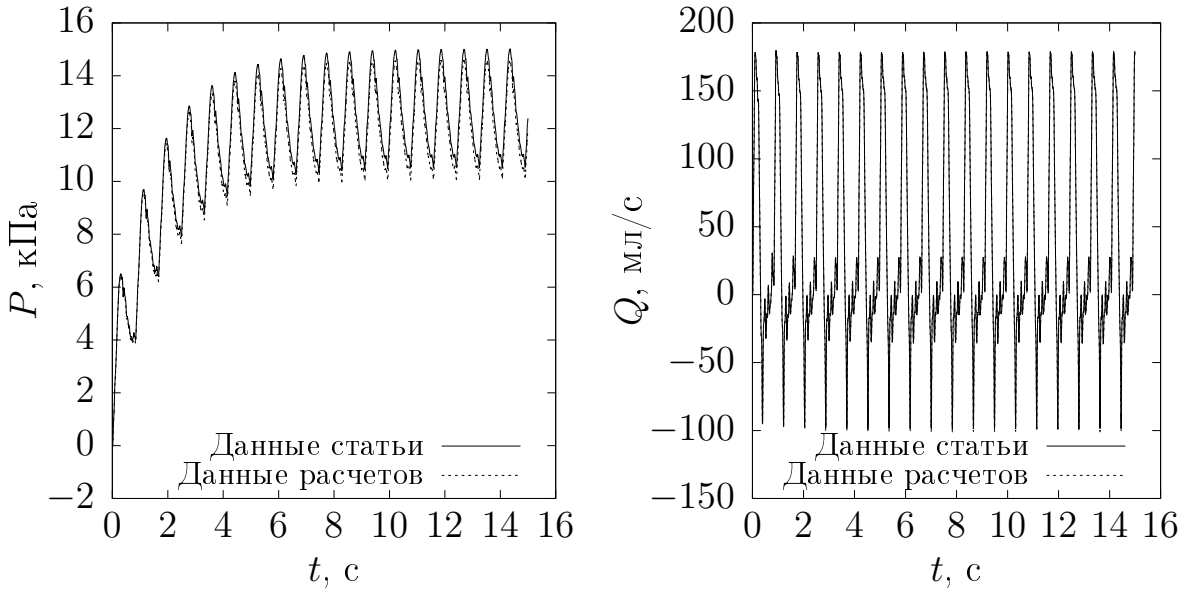

Рис. 2.6: Значения давления и потока в середине сосуда №7

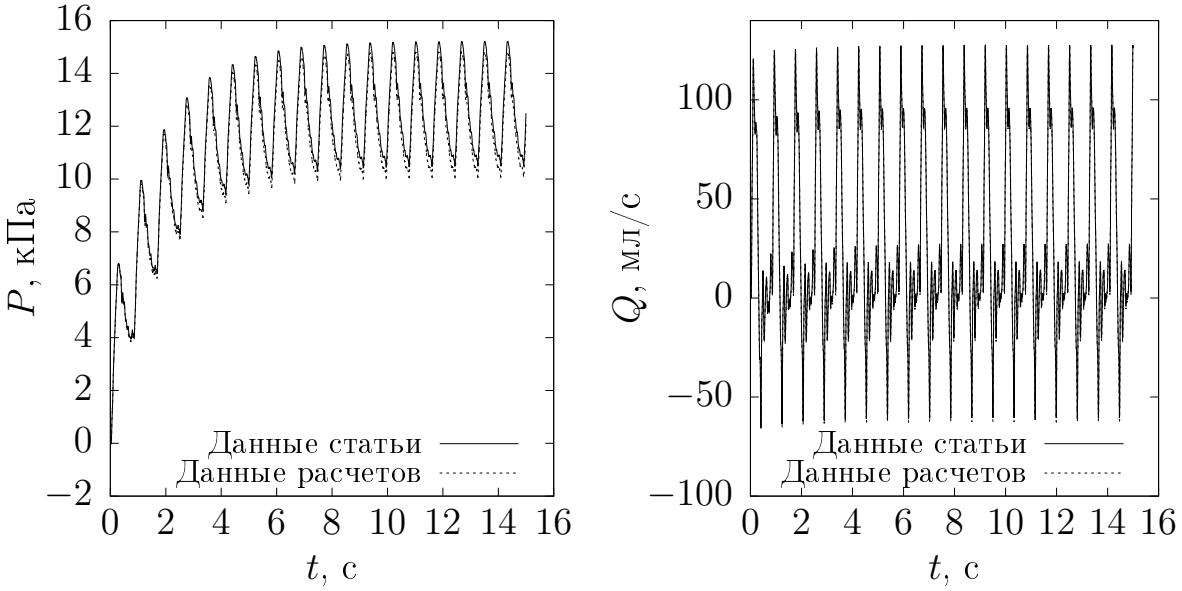

Рис. 2.7: Значения давления и потока в середине сосуда №10

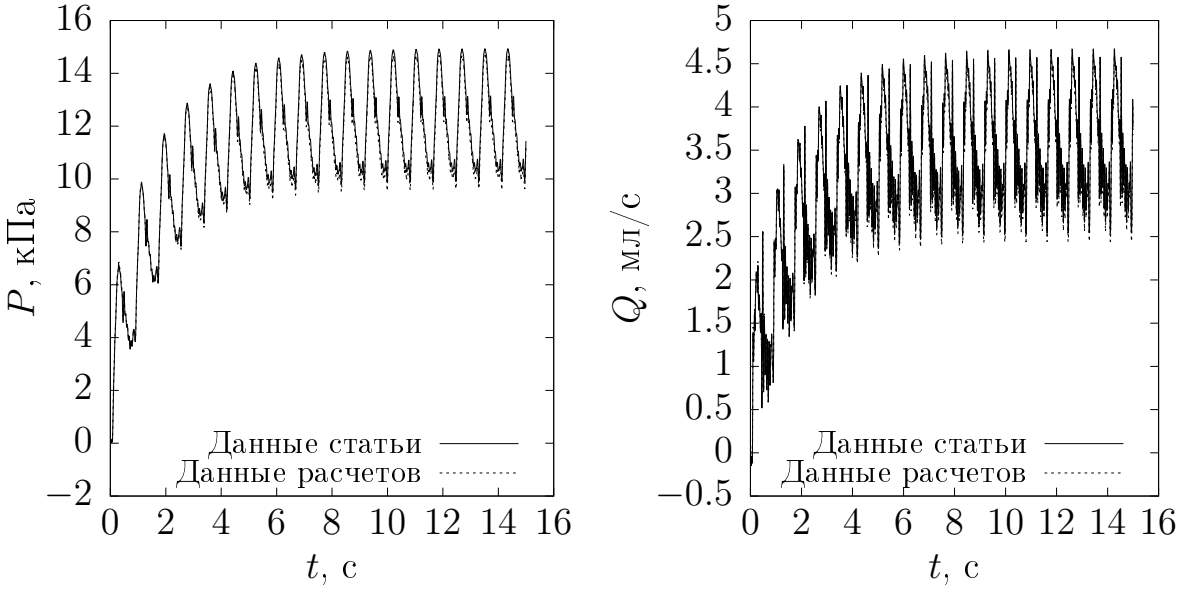

Рис. 2.8: Значения давления и потока в середине сосуда №29

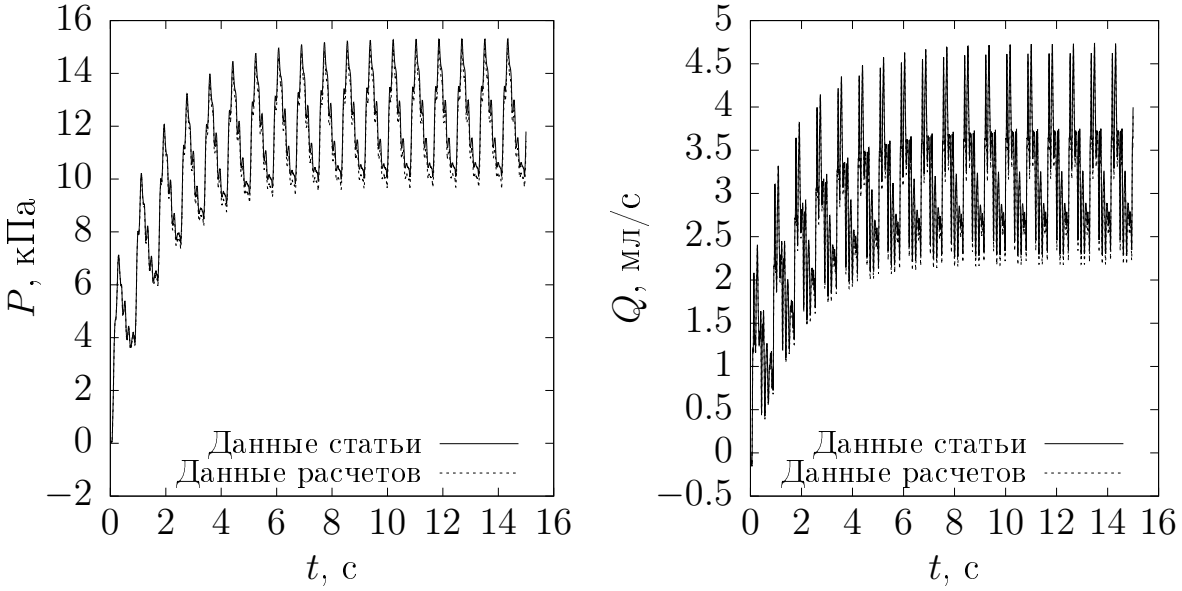

Рис. 2.9: Значения давления и потока в середине сосуда №34

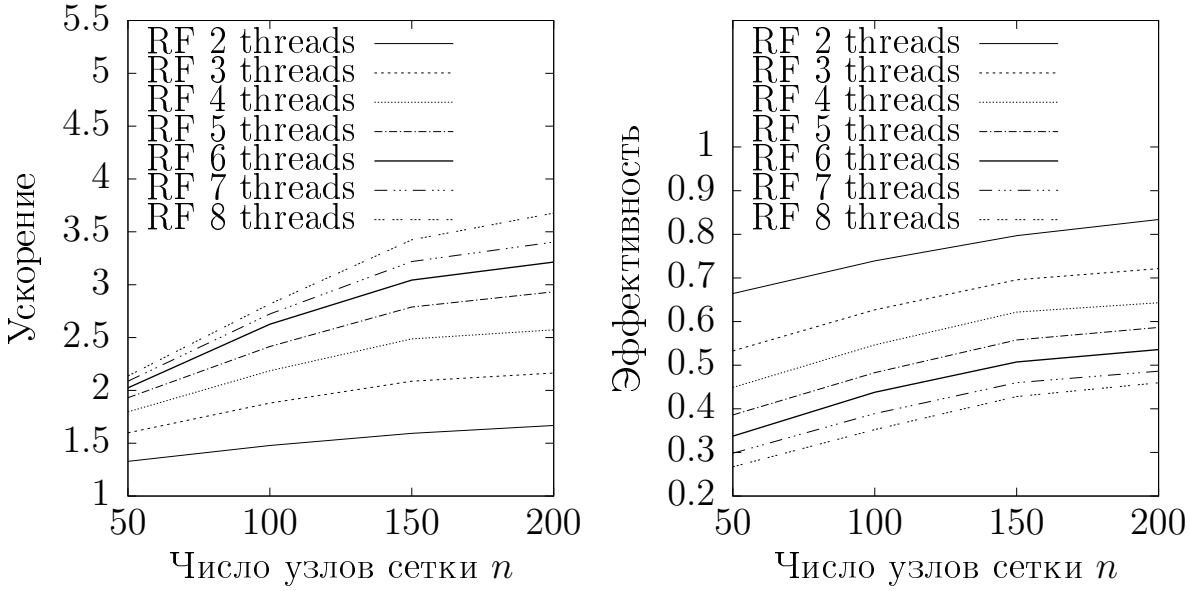

Рис. 2.10: Ускорение и эффективность для схемы Мак-Кормака в зависимости от числа узлов сетки и потоков OpenMP

Результаты и выводы

В данной главе были предложены и реализованы методы применения высокопроизводительных вычислений к различным задачам. Так, в применении к уравнению переноса были предложены два различных метода разбиения задачи и были получены значительные показатели прироста производительности для каждого из них при реализации данного алгоритма

с применением высокопроизводительных вычислений. Для задачи, моделирующей кровеносную систему также был предложен алгоритм решения, который был реализован для различных численных схем. Были проанализированы результаты вычислений по данным схемам для упрощенной модели из 12 сосудов и на основе данных результатов был сделан вывод, что двухшаговые схемы позволяют добиться наибольшего прироста производительности при применении высокопроизводительных систем. Для схемы сосудов, более близкой к реальной модели сердечно-сосудистой системы был реализован алгоритм с применением двухшаговой схемы и были получены результаты, согласующиеся с результатами, представленными в [2]. Таким образом можно сделать вывод, что применение систем высокопроизводительных вычислений позволяет достигнуть значительного прироста производительности при высокой точности полученных результатов.

Заключение

Таким образом в выпускной квалификационной работе были получены следующие результаты:

- Проведено исследование численных схем, применяемых для решения задач для уравнений гемодинамики.
- Были разработаны параллельные алгоритмы решения задач и проведен их анализ.
- Данные алгоритмы были реализованы для проведения расчетов с использованием высокопроизводительных систем на основе CPU (с использованием OpenMP) и GPU (с использованием CUDA), а также изучены сложности, возникающие при реализации параллельных алгоритмов.

По полученным результатам можно сделать следующие выводы:

- Как можно видеть из полученных результатов — рассмотренные методы позволяют достичь значительного ускорения решения подобных задач, однако существуют различные трудности, от сложности решения которых будет зависеть целесообразность применения подобных методов.
- Применение CUDA требует осуществления копирования данных в память графического процессора, а в связи с тем, что операции копирования требует расходов программного времени на их запуск, большое их количество может привести к тому, что применение CUDA не даст прироста производительности.
- Стоит учитывать особенности высокопроизводительных систем при решении вопроса о целесообразности их применения. Так, полученные результаты говорят о том, что CUDA наиболее хорошо подходит

для задач с большим числом узлов и отсутствием необходимости копировать данные между шагами вычислений, в то время как OpenMP позволяет добиться прироста производительности на простых задачах с низким числом узлов.

Список литературы

1. Федеральная служба государственной статистики: сайт // URL: https://rosstat.gov.ru/folder/12781 (дата обращения: 03.04.2021)

2. Matthys K.S., Alastruey J., Peiró J., Khir A. W., Segers P., Verdonck P. R., Parker K. H., Sherwin S. J.. Pulse wave propagation in a model human arterial network: assessment of 1-D numerical simulations against in vitro measurements // J Biomech. 2007. 40(15), P. 3476–3486

3. Boileau E., Nithiarasu P., Blanco P. J., Müller L. O., Fossan F. E., Hellevik L. R., Donders W. P., Huberts W., Willemet M., Alastruey J. A benchmark study of numerical schemes for one-dimensional arterial blood flow modelling // Int J Numer Method Biomed Eng. 2015. 31(10)

4. Физиология человека: учебник под ред. Смирнова В.М. М.: Медицина, 2002. 608 с.

5. Каро К., Педли Т., Шротер Р., Сид У. Механика кровообращения. Пер. с англ. М.: Мир, 1981. 624 с.

6. Валландер С. В. Лекции по гидроаэромеханике. Учеб. пособие. Л.: Ленингр. ун-т, 1978. 296 с.

7. Formaggia L, Lamponi D and Quarteroni A. One-dimensional models for blood flow in arteries // J. of Eng. Math. 2003. 47, P. 251–276

8. Xiao N., Alastruey J., Alberto Figueroa C. A systematic comparison between 1-D and 3-D hemodynamics in compliant arterial models // Int J Numer Method Biomed Eng. 2014. 30(2), P. 204–231

9. Токарева, С. А. Прикладная газовая динамика. Численные методы решения гиперболических систем уравнений : учебное пособие СПб.: Лань, 2019. 244 с.

10. Миньков Л.Л., Шрагер Э.Р. Основные подходы к численному решению одномерных уравнений газовой динамики : Учебное пособие Томск: STT, 2016. 136 с.

11. Антонов А.С., Параллельное программирование с использованием технологии OpenMP: Учебное пособие М.: Изд-во МГУ, 2009. 77 с.

12. Chapman B., Jost G., van der Pas R. Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation). Cambridge: The MIT Press., 2007. 353 с.

13. Боресков А. В., Харламов А. А., Марковский Н. Д., Микушин Д. Н., Мортиков Е. В., Мыльцев А. А., Сахарных Н. А., Фролов В. А. Параллельные вычисления на GPU. Архитектура и программная модель CUDA М.: Московский государственный университет имени М.В. Ломоносова, 2015. 336 с.

14. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах. Введение в программирование графических процессоров М.: ДМК Пресс, 2013. 232 с.

15. Agdestein S., Valen-Sendstad K., Diem A. Artery.FE: An implementation of the 1D blood flow equations in FEniCS // Journal of Open Source Software. 2018. 3(32)

16. Diem, A.K., Bressloff, N.W. VaMpy: A Python Package to Solve 1D Blood Flow Problems // Journal of Open Research Software. 2017. 5(1), P. 17