УДК 519.673:544.4:004.41/42

О МОДЕЛИРОВАНИИ ТЕЧЕНИЙ, ОСЛОЖНЁННЫХ ФИЗИКО-ХИМИЧЕСКИМИ ПРОЦЕССАМИ, ПРИ БОЛЬШОМ ЧИСЛЕ РЕАГИРУЮЩИХ ВЕЩЕСТВ

А.В. Евсеев

Ивановский государственный энергетический университет
alexander.yevseyev@gmail.com

Аннотация

В статье даётся постановка задачи моделирования течений реагирующей среды с множеством участвующих веществ. В модели используется уравнения Навье-Стокса и динамики концентрации веществ, решаемые методом переменных направлений, а также уравнения химической кинетики, вычисляемые методом Гира. На каждом шаге интегрирования уравнения Навье-Стокса по времени необходимо вычислять уравнение Пуассона. Проводится сравнение систем параллельного программирования MPI и CUDA.

ON MODELLING REACTIVE PHYSICOCHEMICAL FLOWS WITH A LARGE NUMBER OF REACTANTS

The article describes the flow simulation with the large number of reactants. The mathematical model is based on Navier-Stokes equation, dynamic concentration of agents and chemical kinetics equations. The first equation is solved using the alternating directions method (ADI) and a Poisson equation on each time step. The second group by the ADI method and the last group of equations is computed using the Gear method. Parallel systems MPI and CUDA which were used to simulate the model are compared.

1 Введение

Исследование физико-химических течений при большом числе реагирующих веществ представляет значительный интерес во многих приложениях, например, при сжигании топлив, моделирование процессов происходящих при пожаре, распространение загрязнений, газодинамические и химические лазеры и многое другое.

Моделирование аэрогидродинамических процессов само по себе является нетривиальной задачей. Кроме того, эти процессы осложнены множественными реакциями большого количества веществ.

Таким образом, адекватное моделирование таких реагирующих сред возможно лишь при использовании многопроцессорной вычислительной техники, которая позволяет существенно сократить затраты машинного времени.

2 Модель движения реагирующей среды

2.1 Уравнение Навье-Стокса

Одним из основополагающих уравнений гидродинамики является уравнение Навье-Стокса, которое традиционно записывается в системе "Скорость – Давление" [1, 10]:

$$\begin{cases} \dfrac{\partial \vec{u}}{\partial t} = -(\vec{u}\cdot\nabla)\vec{u} + \upsilon\Delta\vec{u} - \dfrac{1}{\rho}\nabla P + \vec{F}; \\ \operatorname{div}\vec{u} = 0, \end{cases} \tag{1}$$

где $\vec{u}$ – вектор скорости, P – давление среды, $\vec{F}$ – ускорение макрочастиц среды, вызванное обменом количеством движения с соседними макрочастицами, υ – кинематическая вязкость.

Эти уравнения применимы как к ламинарным, так и к турбулентным движениям. В целях упрощения постановки задачи, мы не использовали какую-либо модель турбулентности, а лишь задали некоторое значение турбулентной вязкости.

Для плоских течений можно использовать запись исходных уравнений в переменных «Функция тока – Вихрь»:

$$\begin{cases} \dfrac{\partial\omega}{\partial t} + u_1\dfrac{\partial\omega}{\partial x_1} + u_2\dfrac{\partial\omega}{\partial x_2} = \upsilon\left(\dfrac{\partial^2\omega}{\partial x_1^2} + \dfrac{\partial^2\omega}{\partial x_2^2}\right); \\ \dfrac{\partial^2\Psi}{\partial x_1^2} + \dfrac{\partial^2\Psi}{\partial x_2^2} = -\omega. \end{cases} \tag{2}$$

При этом введены следующие определения функции тока:

$$\begin{cases} u_1 = \dfrac{\partial\Psi}{\partial x_2}; \\ u_2 = -\dfrac{\partial\Psi}{\partial x_1}, \end{cases} \tag{3}$$

и вихря:

$$\omega = \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}. \tag{4}$$

На твёрдых и неподвижных поверхностях граничные условия задаются следующим образом:

$$\begin{aligned} & u_1\big|_\Gamma = 0; \\ & u_2\big|_\Gamma = 0; \\ & \Psi\big|_\Gamma = \text{const}; \\ & \omega\big|_\Gamma = -\left.\frac{\partial^2\Psi}{\partial n^2}\right|_\Gamma, \end{aligned} \tag{5}$$

где n – нормаль к границе Г.

Константа для функции тока на стенках выбирается просто: на одной из граничных стенок функция тока равна нулю, на других функция тока подбирается таким образом, что разница функций между двумя границами равна объемному расходу среды между ними:

$$Q = \Psi\big|_{\Gamma_2} - \Psi\big|_{\Gamma_1}, \tag{6}$$

где $\Psi\big|_{\Gamma_1}$ – значение функции тока на границе Γ_1, $\Psi\big|_{\Gamma_2}$ – на границе Γ_2.

На входных и выходных отверстиях задаётся профиль скорости, а расчётные формулы для функции тока и вихря получаются либо интегрированием, либо дифференцированием в соответствии с их определениями (3)-(4).

2.2 Дискретизация задачи

Для нахождения поля скоростей в будущий момент времени, необходимо сначала найти поле вихря ω^{t+1}, на основе которого затем получаем поле функции тока Ψ^{t+1}. Далее по формулам (3) вычисляется новое поле векторов скоростей [1, 2].

При получении поля вихря необходимо решить ОДУ с конвективными членами, для которого в двумерном случае удобно применять метод переменных направлений [3].

Согласно этому методу, шаг по времени разбивается на 2 полушага в разных направлениях, которые решаются методом прогонки. Для прогонки по x_1 используется следующая расчётная формула:

$$\frac{\omega_{ij}^{t+1/2}-\omega_{ij}^{t}}{\tau}=-u_{1ij}\left(\frac{\omega_{i+1j}^{t+1/2}-\omega_{i-1j}^{t+1/2}}{2h_1}\right)-u_{2ij}\left(\frac{\omega_{ij+1}^{t}-\omega_{ij-1}^{t}}{2h_2}\right)+\frac{\upsilon}{h_1^2}\left(\omega_{i+1j}^{t+1/2}+\omega_{i-1j}^{t+1/2}-2\omega_{ij}^{t+1/2}\right)+$$
$$+\frac{\upsilon}{h_2^2}\left(\omega_{ij+1}^{t}+\omega_{ij-1}^{t}-2\omega_{ij}^{t}\right). \qquad (7)$$

Соответственно для прогонки в направлении x_2:

$$\frac{\omega_{ij}^{t+1}-\omega_{ij}^{t+1/2}}{\tau}=-u_{1ij}\left(\frac{\omega_{i+1j}^{t+1/2}-\omega_{i-1j}^{t+1/2}}{2h_1}\right)-u_{2ij}\left(\frac{\omega_{ij+1}^{t+1}-\omega_{ij-1}^{t+1}}{2h_2}\right)+\frac{\upsilon}{h_1^2}\left(\omega_{i+1j}^{t+1/2}+\omega_{i-1j}^{t+1/2}-2\omega_{ij}^{t+1/2}\right)+$$
$$+\frac{\upsilon}{h_2^2}\left(\omega_{ij+1}^{t+1}+\omega_{ij-1}^{t+1}-2\omega_{ij}^{t+1}\right), \qquad (8)$$

где

$$u_{1ij}=\frac{\Psi_{ij+1}-\Psi_{ij-1}}{2h_2},\quad u_{2ij}=\frac{\Psi_{i+1j}-\Psi_{i-1j}}{2h_1}. \qquad (9)$$

Получение поля функции тока подразумевает решение уравнения Пуассона.

2.3 Методы решения уравнения Пуассона

Существует множество методов решения уравнения Пуассона, отличающихся скоростью работы, точностью, применимостью, расширяемостью. Были рассмотрены следующие методы: метод верхней релаксации, Ричардсона, Дугласа-Писмана-Рекфорда, попеременно-треугольный метод и метод FACR.

Метод верхней релаксации является одним из наиболее простых методов [3]. Его итерационная схема не сложна, но, к сожалению, обладает невысокой точностью и малой скоростью сходимости. Его разностная схема имеет следующий вид:

$$\begin{cases} u_{ij}^{k+1}=(1-\Theta)u_{ij}^{k}+\dfrac{\Theta}{4}(u_{i-1,j}^{k}+u_{i+1,j}^{k}+u_{i,j-1}^{k}+u_{i,j+1}^{k}-h^2 f_{ij}), \\ i,j=1,\ldots,M-1, \\ u_{ij}^{k+1}\big|_{\Gamma}=0,\quad u_{ij}^{0}=0, \end{cases} \qquad (10)$$

где Θ – параметр релаксации (обычно $\Theta\cong 1.85$).

Метод Ричардсона является усложнением предыдущего метода за счёт введения итерационного временного параметра, основанного на полиномах Чебышева. Было показано, что при специальном выборе последовательности шагов $\tau_k (k = 1,2,\ldots,n)$, называемой циклом, метод будет сходиться к стационарному решению в O(M) раз быстрее, чем при итерациях с постоянным оптимальным шагом τ. Итерационная схема метода Ричардсона:

$$\begin{cases} \dfrac{u_{ij}^{k+1} - u_{ij}^{k}}{\tau_{\chi^{k+1}}} = \dfrac{1}{h^2}(u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k} - 4u_{ij}^{k}) - f_{ij}, \\ i, j = 1,\ldots,M-1, \\ u_{ij}^{k+1}\big|_{\Gamma} = 0, \\ u_{ij}^{0} = 0, \end{cases} \tag{11}$$

где $\chi^n = (\chi_1, \chi_2, \ldots, \chi_n)$ – некоторая перестановка членов исходной последовательности шагов, позволяющая достичь устойчивости.

Методы Дугласа-Писмана-Рекфорда и попеременно-треугольный являются расширением идей предыдущего метода, но в разных направлениях реализации. В методе Дугласа-Писмана-Рекфорда используется схема переменных направлений, решаемая с помощью алгоритма прогонки:

$$\begin{cases} \dfrac{\widetilde{u}_{ij} - u_{ij}^{k}}{\tau_{k+1}} = \dfrac{1}{2}\left(\dfrac{\widetilde{u}_{i-1,j} - 2\widetilde{u}_{ij} + \widetilde{u}_{i+1,j}}{h^2} + \dfrac{u_{i,j-1}^{k} - 2u_{ij}^{k} + u_{i,j+1}^{k}}{h^2} - f_{ij} \right), \\ \dfrac{u_{ij}^{k+1} - \widetilde{u}_{ij}}{\tau_{k+1}} = \dfrac{1}{2}\left(\dfrac{\widetilde{u}_{i-1,j} - 2\widetilde{u}_{ij} + \widetilde{u}_{i+1,j}}{h^2} + \dfrac{u_{i,j-1}^{k+1} - 2u_{ij}^{k+1} + u_{i,j+1}^{k+1}}{h^2} - f_{ij} \right), \\ i, j = 1,\ldots,M-1, \\ u_{ij}^{k+1}\big|_{\Gamma} = 0, \\ \widetilde{u}_{ij}\big|_{\Gamma} = 0, \\ u_{ij}^{0} = 0, \end{cases} \tag{12}$$

Попеременно-треугольный метод использует разделение общего дифференциального оператора на два более простых, которые можно решить последовательно с помощью так называемой схемы "бегущего счёта":

$$\begin{aligned} &(E + \omega R_1) w^{k+1/2} = Au^k - f, \\ &(E + \omega R_2) w^{k+1} = w^{k+1/2}, \\ &u^{k+1} = u^k + \tau^{k+1} w^{k+1}, \\ &R_1 u_{ij} = \frac{1}{h}\left(\frac{u_{ij} - u_{i-1,j}}{h} + \frac{u_{ij} - u_{i,j-1}}{h} \right), \\ &R_2 u_{ij} = \frac{1}{h}\left(\frac{u_{ij} - u_{i+1,j}}{h} + \frac{u_{ij} - u_{i,j+1}}{h} \right). \end{aligned} \tag{13}$$

В отличие от предыдущих итерационных методов FACR является прямым методом [4]. Он основывается на Быстром Преобразовании Фурье и прогонке. Метод быстр, но на-

кладывает дополнительные ограничения на решаемую область. Для нахождения решения необходимо пройти 3 этапа вычислений:

$$\widetilde{f}_{ki} = -\mathrm{Im}\left(\sum_{j=0}^{N-1} z_j e^{-\left(\sqrt{-1}\right)2\pi k\frac{j}{N}}\right),\quad \mathrm{k}=1,2,..,\mathrm{M}-1, \tag{14}$$

$$\sum_{k=1}^{M-1}\mu_{kj}\left(\frac{c_{ki+1}-2c_{ki}+c_{ki-1}}{h^2}\right)+\sum_{k=1}^{M-1}c_{ki}\left(\frac{\mu_{kj+1}-2\mu_{kj}+\mu_{kj-1}}{h^2}\right)=\sum_{k=1}^{M-1}\widetilde{f}_{ki}\mu_{kj}, \tag{15}$$

$$u_{ij} = -\mathrm{Im}\left(\sum_{k=0}^{N-1} z_k e^{-\left(\sqrt{-1}\right)2\pi j\frac{k}{N}}\right),\quad \mathrm{j}=1,2,..,\mathrm{M}-1. \tag{16}$$

2.4 Моделирование химических реакций

Чтобы корректно описывать движение реагирующей среды, расширим модель, добавив уравнение диффузии реагирующих веществ и уравнения химической кинетики. Первое уравнение позволит определить концентрации вещества в зависимости от координаты и времени, а уравнения химической кинетики позволят описать изменение концентрации веществ в результате проходящих химических реакций (прямая кинетическая задача) [1].

Уравнение диффузии:

$$\frac{\partial C_j}{\partial t}+\sum_{i=1}^{2}u_i\frac{\partial C_j}{\partial x_i}=D_{C_j}\sum_{i=1}^{2}\frac{\partial^2 C_j}{\partial x_i^2}+\frac{dC_j}{dt};\quad j=1,\ldots,N, \tag{17}$$

где C_j – концентрация j-го вещества, D_{C_j} – коэффициент диффузии j-го вещества, N – число веществ. Член $\frac{dC_j}{dt}$ определяет изменение концентрации j-го вещества в результате химических реакций.

Граничные условия на твёрдых стенках для концентраций C_j устанавливаются с учётов инертности этих поверхностей:

$$\left.\frac{\partial C_j}{dn}\right|_{\Gamma}=0. \tag{18}$$

На открытых границах используются мягкие граничные условия. Случай, когда границы не являются инертными, не рассматривался.

При моделировании химических реакций предположим, что имеется N веществ, участвующих в K реакциях, тогда система уравнений будет иметь вид [1, 5]:

$$\sum_{i=1}^{N}L_{ij}P_i\rightarrow\sum_{i=1}^{N}R_{ij}P_i,\ j=\overline{1,K}, \tag{19}$$

где P_i — символ i-го вещества; L_{ij} — стехиометрический коэффициент при i-м веществе в левой части j-й реакции; R_{ij} — стехиометрический коэффициент при i-м веществе в правой части j-й реакции. Матрицы $\mathbf{L} = [L_{ij}]$ и $\mathbf{R} = [R_{ij}]$ имеют размерность (N×K).

Система дифференциальных уравнений химической кинетики полностью определяется набором химических реакций. Это система обыкновенных дифференциальных уравнений первого порядка следующего вида:

$$\frac{dC_i}{d\,t} = \sum_{j=1}^{K} R_{ij} A_j \prod_{m=1}^{N} C_m^{L_{mj}} - \sum_{j=1}^{K} L_{ij} A_j \prod_{m=1}^{N} C_m^{L_{mj}} \; ; i = \overline{1, N}, \tag{20}$$

где A_j – константа скорости j-й реакции. При рассмотрении равновесных и квазиравновесных процессов ограничимся для оценки скоростей реакций уравнением Аррениуса:

$$A_j = Q_j T^{N_j} e^{\frac{-E_j}{RT}}, \tag{21}$$

в котором Q_j — предэкспонент, N_j — показатель степени в температурном множителе, T — температура проведения реакции (в градусах Кельвина), E_j — энергия активации (в ккал/моль), $R = 1{,}987 \cdot 10^{-3}$ ккал · град$^{-1}$· моль$^{-1}$.

Таким образом, объём вычислений сильно зависит от числа участвующих веществ и происходящих реакций. Параболические уравнения (10) решаются наподобие уравнения Навье-Стокса с помощью метода переменных направлений, однако присутствующий в нём член изменения концентраций в результате химических реакций необходимо вычислять перед каждым шагом по времени с использованием текущего значения τ. Система уравнений химической кинетики (12) достаточно часто является жёсткой, поэтому шаг интегрирования должен согласовываться с характерным временем наиболее быстрого процесса. В этом случае необходимо использовать жёстко устойчивые численные методы, такие как метод Гира.

2.5 Метод Гира

Метод Гира относится к классу так называемых многозначных методов. Также его часто называют методом Нордсика или метод на основе формул дифференцирования назад (ФДН-метод) в представлении Нордсика [5, 6].

Как и многошаговые, многозначный метод основан на получении нового значения функции с использованием аппроксимирующего полинома. Однако в отличие от них многозначный метод использует разложение решения в ряд Тейлора в некоторой точке интегрирования. Таким образом, коэффициентами полинома становятся приближения к производным от решения в точке t_{n-1}. Теперь цель интегрирования заключается в том, чтобы получить те же коэффициенты ряда Тейлора, но уже в следующей точке t_n. Для этой цели используется вектор Нордсика:

$$\mathbf{z} = (y, h \cdot y', \frac{h^2}{2} y'', \ldots, \frac{h^p}{p!} y^{(n)})^T, \tag{22}$$

где p – порядок метода.

Однако итерационные формулы, полученные непосредственно из такого представления полинома, оказались нестабильными. Для решения этой проблемы Нордсик предложил ввести p коэффициентов, что при их правильном подборе позволяет добиться стабильности.

Важными особенностями многозначного метода Гира является его возможность на каждом шаге контроля ошибки интегрирования и соответственно изменять порядок метода и шаг интегрирования по времени.

Метод Гира работает по принципу «предиктор-корректор», поскольку в начале каждого шага по времени строится предиктор на основе матрицы векторов Нордсика:

$$\widetilde{\mathbf{Z}}_n = \mathbf{B}(q)\mathbf{Z}_{n-1}, \tag{23}$$

$$\mathbf{B}(q) = \begin{bmatrix} \dfrac{j!}{i!(j-i)!}, & i \le j \\ 0, & i > j \end{bmatrix}, \quad i, j = 0,1,\ldots,p, \tag{24}$$

$$\mathbf{Z}_n = \left(\begin{pmatrix} y_1 \\ hy_1' \\ \vdots \\ \dfrac{h^p}{p!} y_1^{(p)} \end{pmatrix} \begin{pmatrix} y_2 \\ hy_2' \\ \vdots \\ \dfrac{h^p}{p!} y_2^{(p)} \end{pmatrix} \cdots \begin{pmatrix} y_N \\ hy_N' \\ \vdots \\ \dfrac{h^p}{p!} y_N^{(p)} \end{pmatrix} \right) = \left(\mathbf{z}_{n_1}, \mathbf{z}_{n_2}, \ldots, \mathbf{z}_{n_N} \right). \tag{25}$$

Затем с помощью метода Ньютона вычисляется корректор:

$$\mathbf{r} = \mathbf{y}_n - \widetilde{\mathbf{y}}_n = \begin{pmatrix} y_1 - \widetilde{y}_1 \\ y_2 - \widetilde{y}_2 \\ \vdots \\ y_N - \widetilde{y}_N \end{pmatrix}, \tag{26}$$

с последующим контролем ошибки интегрирования:

$$\mathbf{e}_n(p) = -\frac{\mathbf{r}}{l_1} \left(1 + \prod_{i=2}^{p} \frac{t_n - t_{n-i}}{t_{n-1} - t_{n-i}} \right). \tag{27}$$

Если шаг признан успешным, то происходит обновление текущих значений шага:

$$\mathbf{Z}_n[j] = \widetilde{\mathbf{Z}}_n[j] + \mathbf{r} \cdot l_j, \quad j = 1, \ldots, p. \tag{28}$$

Среди общедоступных библиотек, реализующих алгоритм метода Гира, одной из самых лучших и широко используемых является библиотека CVODE из пакета SUNDIALS [12], разработанной в Ливерморской национальной лаборатории США. Она легка в использовании, написана на языке C, легко расширяема и позволяет использовать собственные стратегии и методы решения отдельных этапов общего алгоритма, а также базовых единиц хранения информации (векторов значений) и операций с ними.

3 Ускорение вычислений

3.1 Построение параллельных схем с использованием MPI

В рассматриваемой проблеме существует два класса вычислительных задач, которые требуют распараллеливания. Это алгоритм прогонки, используемый при решении различных систем уравнений, и уравнение Пуассона [8, 9].

Интерфейс параллельного программирования MPI предназначен для систем с разделённой памятью, и, соответственно, главные вопросы, которые необходимо решить при построении параллельных алгоритмов, это способ разбиения задачи между процессорами и определение эффективного взаимодействия между ними при передачах данных.

Замечательным свойством решения уравнения Пуассона является то, что при любом числе измерений с использованием геометрического параллелизма исходную задачу можно разбить на ряд аналогичных задач меньшего объёма. Подобное разбиение легко реализуется и требует в общем случае пересылок данных только на граничных слоях.

В методе верхней релаксации в дополнение к геометрическому параллелизму необходимо использовать ещё и конвейерный, поскольку вычисления в нём представляют последовательный пробег по всем узлам сетки. Поэтому в этом случае процессоры последовательно включаются в вычисления, что вносит дополнительные задержки и трудности с проверкой сходимости метода.

Метод Ричардсона имеет более привлекательное распараллеливание, поскольку в нём все процессоры включаются в вычисления все вместе и требуют только пересылок данных на границах областей, которые организуются эффективным образом.

Наибольшие затруднения при построении параллельных алгоритмов вызывают методы FACR и Дугласа-Писмана-Рекфорда, а также при решении уравнения Навье-Стокса для нахождения вихря ω и уравнения диффузии веществ.

Эти методы содержат части, которые могут вычисляться абсолютно независимо и параллельно, а потому эффективно, как, например, прогонка в направлении параллельном плоскости разбиения задачи между процессорами. Проблема в том, что они содержат также и прогонку в направлении перпендикулярном этой плоскости, как показано на рис.1 слева. В этом случае получается, что необходимо провести вычисления в обоих направлениях на всех процессорах, и на каждом переходе приходится производить пересылки промежуточных данных. Таким образом, этот участок алгоритма становится "бутылочным горлышком", ограничивающим общую производительность методов. В связи с этим был разработан оригинальный способ, основанный на конвейерном параллелизме для минимизации задержек в работе алгоритма прогонки в направлении перпендикулярном направлению разбиения задачи на области, как показано на рис.1 справа.

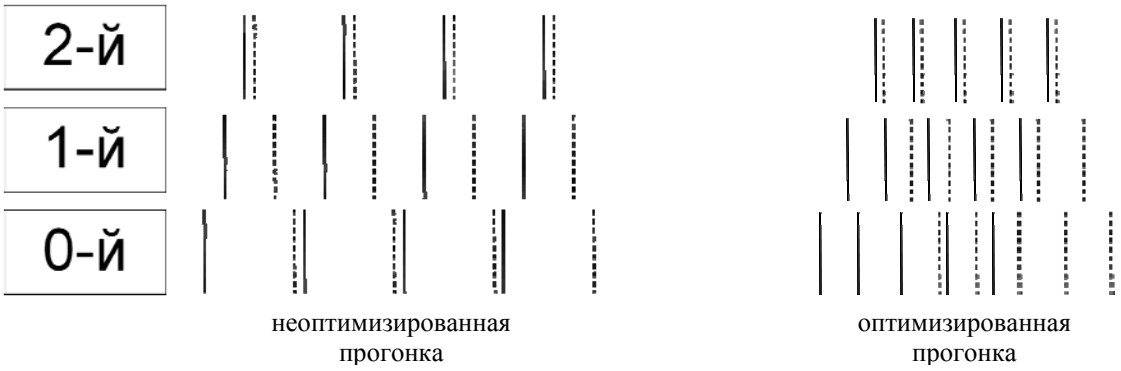

Рис.1. Прогонка в направлении, перпендикулярном плоскости разбиения задачи между процессорами

3.3 Построение параллельных схем с использованием CUDA

В сфере научных вычислений всё большую популярность набирают вычисления с использованием графических ускорителей. Современные видеоакселераторы представляют собой массивно-параллельные процессоры с общей памятью. В отличие от центрального процессора с несколькими ядрами, один графический процессор содержит несколько сотен ядер, которые могут проводить вычисления параллельно.

Наиболее развитой на данный момент технологией является система CUDA предложенная компанией Nvidia в 2007 году [13]. В данной технологии вычисления производятся множеством блоков, состоящих из некоторого числа обрабатывающих потоков. В отличие от MPI, Nvidia CUDA представляет собой систему с общей памятью. Однако каждое обращение к общей памяти приводит к существенным задержкам, во время которых вычисления потоком не проводятся. Чтобы избежать подобных задержек каждое ядро имеет некоторый объём разделяемой памяти, которая является быстрой общей памятью для всех потоков одного блока (рис.2).

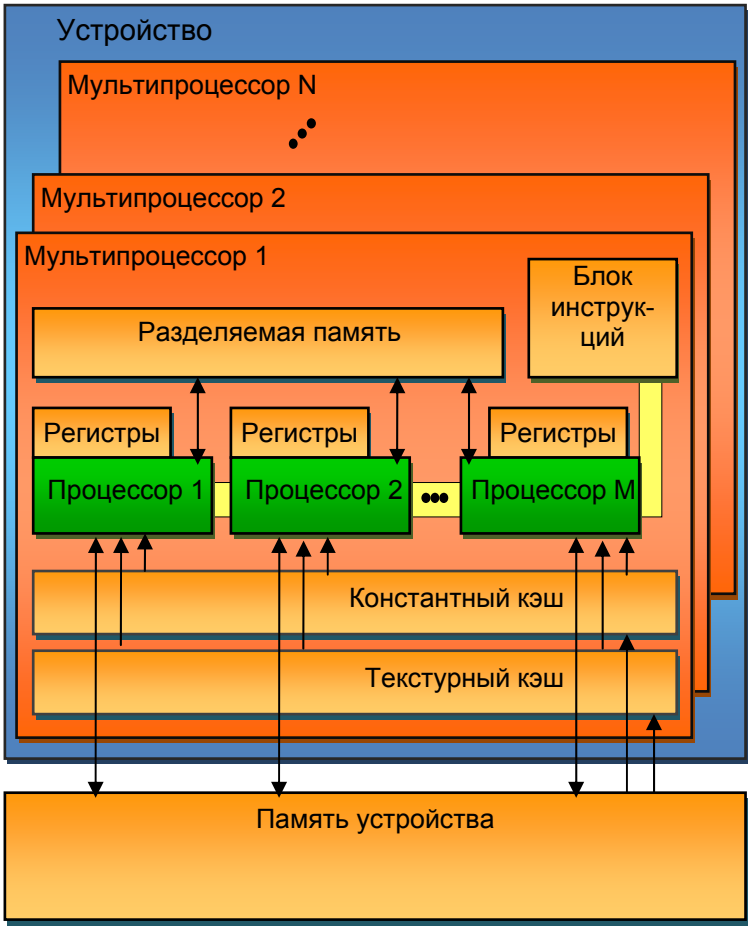

Рис.2. Архитектура CUDA

Главной задачей при построении параллельных алгоритмов для Nvidia CUDA становится эффективное разделение расчётных данных задачи между блоками и потоками [10].

Таким образом, как и при использовании MPI, необходимо использовать геометрический параллелизм, когда каждому блоку назначается свой сектор исходных данных, а каждый поток выполняет вычисления только с одним элементом данных. Но, в отличие от MPI, граничные условия становятся существенной проблемой, т.к. они необходимы для вычислений, но не должны вычисляться по общему алгоритму. Следовательно, требуется, чтобы каждый расчётный блок использовал разделяемую память с учётом граничных условий и обновлял их на каждой итерации.

Отличительной особенностью CUDA является то, что в ней нет возможности осуществить глобальную синхронизацию вычислений между блоками иным способом, кроме как закончив вычисления во всех блоках. В связи с этим каждый этап вычислительного алгоритма может использовать собственное, оптимальное для него разделение задачи между блоками.

С целью упрощения параллельного алгоритма метода Ричардсона было принято решение использовать ускоритель для вычисления одной итерации, поскольку это не только решает проблему граничных условий, но и позволяет использовать эффективный алгоритм для проверки сходимости решения на всём наборе данных.

В итоге, чтобы вычисления проводились наиболее оптимально, каждый блок содержит 256 потоков в виде сетки 16x16, что, по используемым ресурсам ускорителя, позволяет выполнять по 3 блока на каждом ядре одновременно. Однако это накладывает ограничения на входные данные. Количество внутренних точек в общей расчётной области должно быть не меньше 16 и быть степенью двойки.

Также с использованием Nvidia CUDA был построен параллельный алгоритм метода FACR. При построении его параллельной схемы отпадает необходимость в специальном разделении общей памяти на блоки. Данный метод состоит из трёх основных этапов: 2-х преобразований Фурье и решения систем трёхдиагональных уравнений. Преобразования Фурье осуществляются с помощью библиотеки CUFFT, поставляемой вместе с CUDA SDK [3]. Трехдиагональные системы, в последовательном варианте обычно решаемые с использованием алгоритма прогонки, вычисляются с помощью параллельной циклической редукции, так как данный алгоритм эффективно может быть реализован для массивно-параллельных систем с общей памятью [4]. Указанные этапы требуют подготовки данных для обработки и их преобразование после вычислений. Эти части могут быть выполнены без специального разделения на области.

Как уже говорилось выше, в библиотеке CVODE, реализующей метод Гира, можно использовать собственную реализацию векторных операций. А поскольку эти операции представляют основную вычислительную и, что самое важное, массовую нагрузку, то именно они определяют общую производительность метода. Использование CUDA и библиотеки CUBLAS, также поставляемой вместе с CUDA Toolkit, позволяет легко распараллелить эти операции.

3.3 Сравнение параллельных схем

Испытания проводились на суперкомпьютере МВС-100 для MPI и на графическом ускорителе GeForce GTX 275 для CUDA. Все испытания проводились с требуемой точностью решения 10^{-7} последовательно для двумерных квадратных сеток с количеством интервалов 32, 64, 128, 256, 512 и 1024. Для наглядности, представим результаты испытаний в виде графиков.

При моделировании течений, основная нагрузка ложится на вычисление уравнения Пуассона. Его производительность в первую очередь определяет общую скорость вычислений.

Результаты испытаний на МВС-100 для сетки 256x256 элементов приведены на рис.3. Из графиков видно как масштабируется задача с увеличением числа задействованных процессоров. Кроме того, можно заметить, что, как и было указано, метод Ричардсона имеет очень эффективный параллельный алгоритм, который по производительности становится сравнимым с методом FACR.

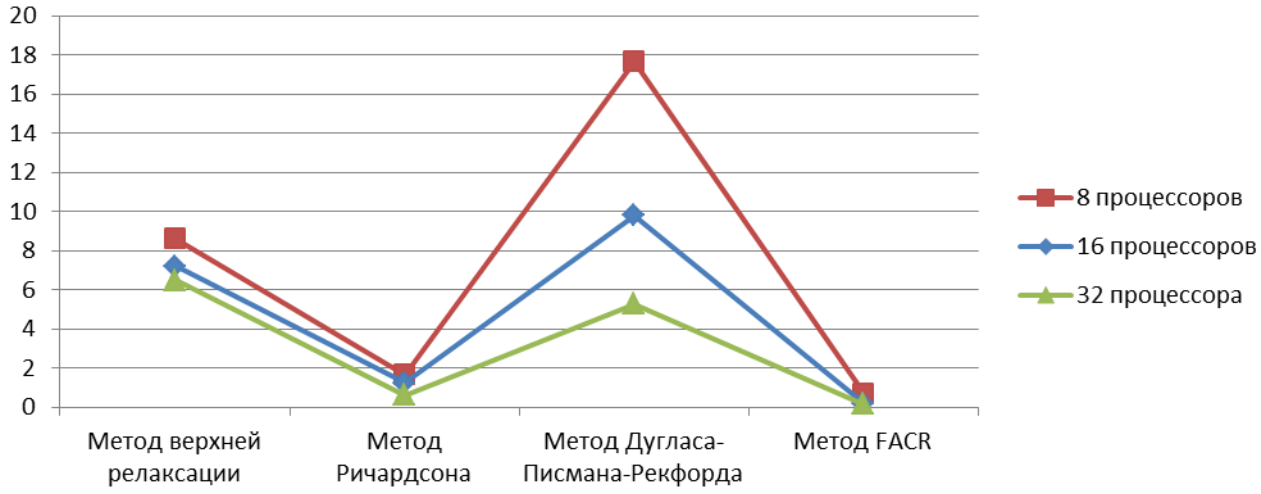

Рис.3. Результаты испытаний на параллельной машине (MPI)

На рис.4 представлено сравнение двух методов (Ричардсона и FACR) на различных вычислительных архитектурах.

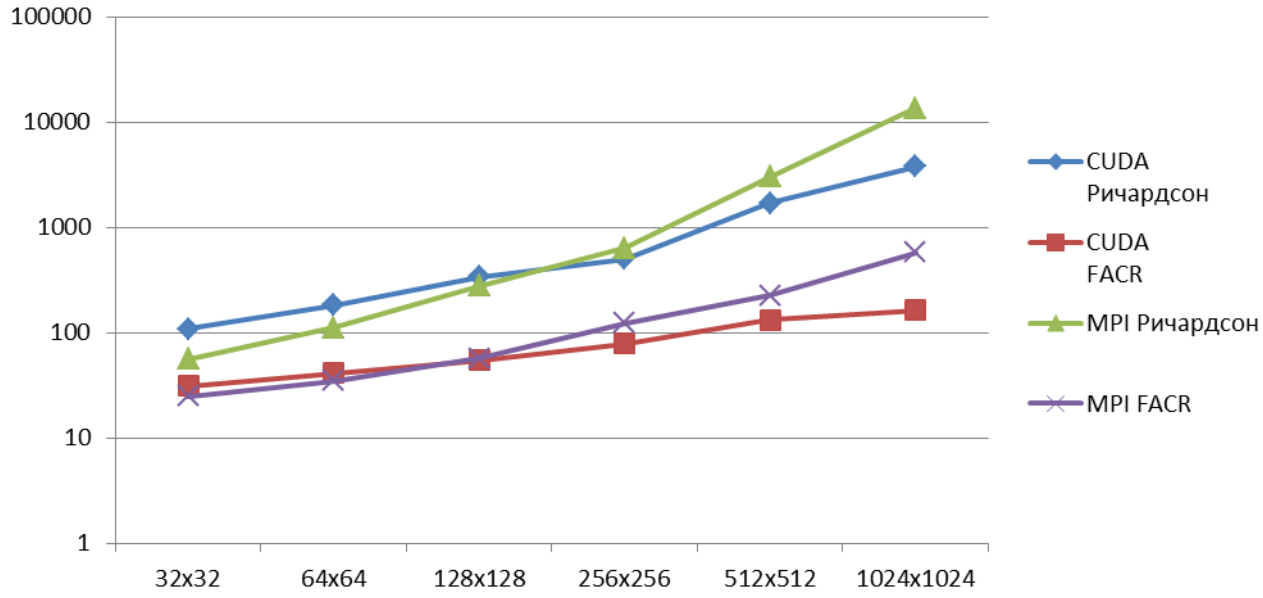

Рис.4. Сравнение методов на различных платформах

Графики на рис.4 подтверждают основной тезис, который произносится разработчиками CUDA, что для полного раскрытия потенциала графических ускорителей необходим очень большой объём данных, чтобы наиболее максимально использовать возможности оборудования. Соответственно, при переходе к сетке 256x256 элементов, методы с использованием Nvidia CUDA начинают выигрывать в производительности и чем больше данных, тем более существенным становится выигрыш.

Этот тезис становится ещё более важным в аспекте перехода к трёхмерному моделированию. В этом случае резко возрастает количество узлов сетки для каждой расчётной переменной. Это позволяет по максимум загрузить работой CUDA устройства.

Кроме того, увеличиваются и требования к памяти. Так, например, при моделировании уравнения Навье-Стокса в системе «Скорость – Давление» в кубе с числом узлов по одной грани равным 512 узлам, требуется 512^3 * (3 проекции скорости + давление + массовые силы) * 8 байт = 5368709120 байт = 5 Гбайт. А если к этому добавить ещё переменные турбулентной модели, то требования у памяти будут ещё более жёсткими. Большинство CUDA устройств имеет объём памяти от 1-го до 2-х Гбайт, что подразумевает использование систем с несколькими ускорителями CUDA (MultiGPU) или даже кластера таких систем [11]. В этом случае, целесообразным является применение комбинации технологий параллельного программирования:

- MPI – для программирования обмена данными между узлами кластерной системы;
- OpenMP – для создания многопоточного окружения на каждом узле кластера, необходимого для MultiGPU систем;
- CUDA – для непосредственного вычисления.

В подобном гетерогенном кластере, главной задачей будет выделение унифицированных вычислительных ядер CUDA, независимых от данных. За пересылки и распределение данных должны отвечать MPI и OpenMP.

При вычислении метода Гира с использованием библиотеки CVODE были протестированы три доступные реализации векторных операций: последовательная (serial), парал-

лельная с использованием MPI и CUDA. Результаты сравнения производительности векторных операций представлены на рис. 5. Как видно из графиков, CUDA-реализация векторных операций начинают выигрывать в производительности при увеличении размерности векторов, характеризующих количество веществ и реакций между ними. Таким образом, это позволит строить химические модели многокомпонентных смесей со сложными реакциями между ними, и их число может измеряться тысячами и сотнями тысяч.

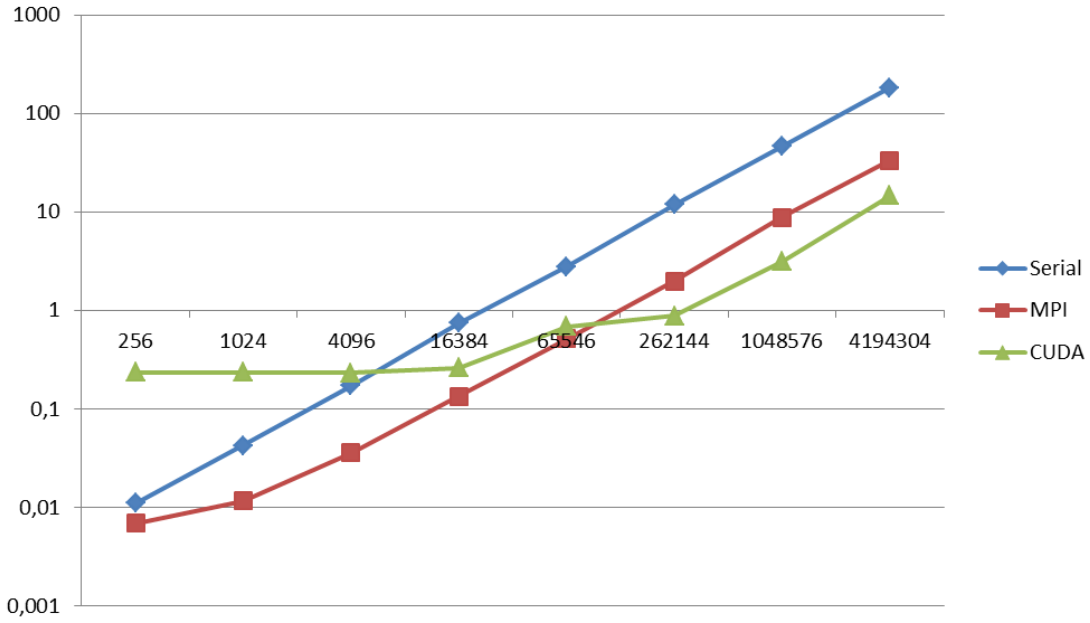

Рис.5. Сравнение векторных операций на различных платформах

В тоже время решение трёхдиагональных систем, как видно из графиков на рис. 6, происходит быстрее, чем на CPU уже даже при размерах систем в 128 элементов. Это обусловлено тем, что при использовании CUDA становится возможным решать 128 систем одновременно, а не последовательно, как на обычном процессоре. Кроме того, допустим, что каждый мультипроцессор графического ускорителя может вычислять 128 систем одновременно, а с учётом того, что на любой графической карте имеется N таких мультипроцессоров, то в целом одна карта может производить вычисления $N \cdot 128$ систем.

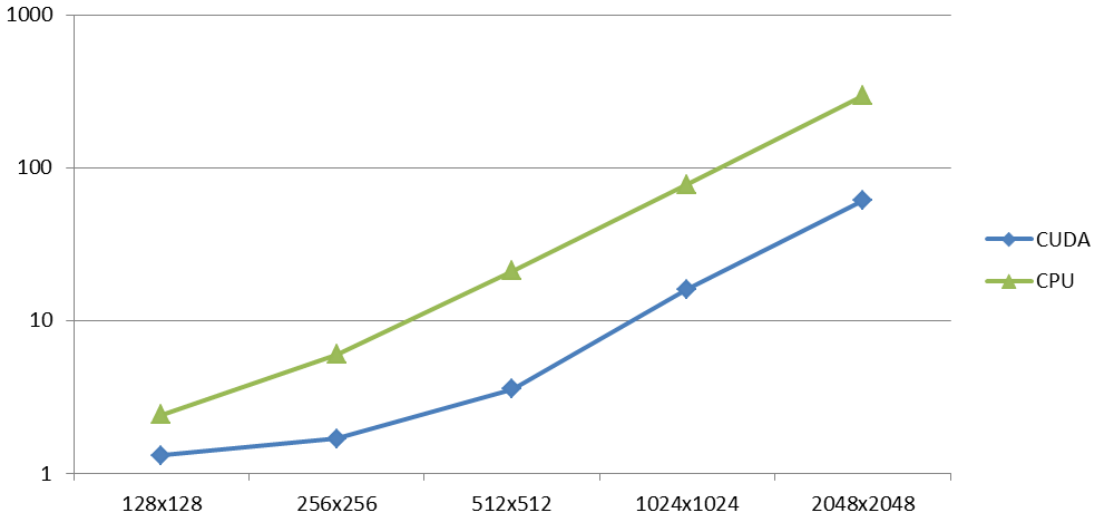

Рис.6. Сравнение решения трёхдиагональных систем

6 Заключение

В заключение необходимо отметить большой потенциал использования графических ускорителей в научных вычислениях. Обладая небольшими размерами, они предоставляют производительность небольшого суперкомпьютера. Конечно, у них имеются некоторые ограничения в плане более сложной организации параллельного процесса, по сравнению с традиционными системами, но с каждым выпуском новой аппаратной архитектуры и новых выпусков CUDA Toolkit, эти ограничения становятся всё менее существенными.

Моделирование движения среды, осложнённой множеством химических реакций большого числа веществ, изначально предполагает наличие больших расчётных сеток для более детального изучения. Кроме того, число веществ и реакций между ними может исчисляться десятками тысяч. В этом случае использование графических ускорителей становится оправданным, поскольку позволяет их полностью загрузить вычислениями. Однако на очень больших сетках или при переходе к моделированию в трёх измерениях, становится очевидным и необходимым использование системы с несколькими графическими ускорителями или даже кластера таких систем.

Предложенная методика позволяет заметно сократить накладные расходы на рассмотрение одного варианта течения и при поиске наилучшего инженерного решения, например, в области лазерных технологий, существенно увеличить число исследуемых вариантов. Также в некоторых случаях становится возможным моделирование процесса в режиме реального времени с возможностью интерактивного взаимодействия.

Литература

1. Балаев Э.Ф. Численные методы и параллельные вычисления для задач механики жидкости, газа и плазмы / Э.Ф.Балаев, Н.В.Нуждин, В.В.Пекунов, С.Г.Сидоров, Л.П.Чернышева, И.Ф.Ясинский, Ф.Н.Ясинский. – Иваново: ИГЭУ, 2003.
2. Пекунов В.В. Алгоритмы и программы для многопроцессорных суперкомпьютеров / В.В. Пекунов, С.Г. Сидоров, Л.П. Чернышева [и др.]. – Иваново: ИГЭУ, 2007.
3. Самарский А.А. Введение в численные методы / А.А. Самарский. – М.: Наука, 1982.
4. Хокни Р. Численное моделирование методом частиц : [пер. с англ.] / Р. Хокни, Дж. Иствуд. – М.: Мир, 1987.
5. Хайрер Э. Решение обыкновенных дифференциальных уравнений. Жесткие и дифференциально-алгебраические задачи : [пер. с англ.] / Э. Хайрер, Г. Ваннер. – М.: Мир, 1999.
6. Полак Л.С. Вычислительные методы в химической кинетике / Л.С. Полак, М.Я. Гольденберг, А.А. Левицкий. – М.: Наука, 1984.
7. Zhang Y. Fast tridiagonal solvers on the GPU / Y. Zhang, J. Cohen, J. D. Owens // Principles and Practice of Parallel Programming, 2010. P.127–136.
8. Евсеев А.В. Вопросы распараллеливания уравнения Пуассона и сравнение эффективности различных вариантов / А. В. Евсеев // Высокие технологии, исследования, промышленность. Т.3 : сборник трудов Девятой международной научно-практической конференции "Исследование, разработка и применение высоких технологий в промышленности". - СПб.: Изд-во Политехн. ун-та, 2010. С. 46-52.
9. Евсеев А.В. Распараллеливание методов решения уравнения Пуассона / А.В. Евсеев, Ф.Н. Ясинский // Высокопроизводительные параллельные вычисления на кластерных системах : Материалы Девятой международной конференции-семинара. – Владимир: Изд-во ВГУ, 2009. С. 166.
10. Ясинский Ф.Н., Евсеев, А.В. О решении уравнения Навье-Стокса в переменных "функция тока-вихрь" на многопроцессорной вычислительной машине с использованием системы CUDA // Вестник ИГЭУ. – 2010. – Вып.3. – С. 73-75.
11. Евсеев А.В. О решении уравнения Навье-Стокса в переменных «Функция тока – Вихрь» с использованием системы с несколькими графическими ускорителями / А.В. Евсеев, Ф.Н. Ясин-

ский // Т.1 : сборник трудов Десятой международной конференции «Высокопроизводительные параллельные вычисления на кластерных системах». – Пермь: Изд-во ПГТУ, 2010. С. 245-251.
12. SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers). https://computation.llnl.gov/casc/sundials/main.html
13. NVIDIA GPU Computing Developer. http://developer.nvidia.com/object/gpucomputing.html

Статья поступила в редакцию 17 марта 2011 г.