УДК 519.673

Использование системы с несколькими ускорителями CUDA для решения уравнения Навье-Стокса в переменных «функция тока – вихрь»

Ясинский Ф.Н., д-р физ.-мат. наук, Евсеев А.В., асп.

Исследуется принципиальная возможность моделирования движения вязкой несжимаемой жидкости, описываемого уравнением Навье-Стокса в переменных «функция тока – вихрь», в системе с несколькими ускорителями CUDA.

Ключевые слова: граничные условия, функции тока и вихря, графические ускорители, система CUDA, трехдиагональные системы, транспонированное общее поле данных.

Using CUDA Accelerators System for Decision of Navier–Stokes Equation in «Stream Function – Curl» Variable

F.N. Yasinskiy, Doctor of Physics and Mathematics, A.V. Evseev, Post-Graduate Student

The article studies the possibility of modelling the viscous incompressible fluid motion described by the Navier-Stokes equations in the stream function from, using multiGPU CUDA system.

Key words: numerical simulation, CFD, Navier-Stokes equation, Poisson equation, Laplace equation, sweep method, CUDA, SLI, many accelerators, multiprocessor, parallel algorithm.

Одним из основополагающих уравнений гидродинамики является уравнение Навье-Стокса, которое традиционно записывается в системе «скорость – давление» [1]:

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

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

Если ограничиться двумерным случаем, то можно ввести следующие определения:

- функции тока

$$\begin{cases} u_1 = \dfrac{\partial \Psi}{\partial x_2}, \\ u_2 = -\dfrac{\partial \Psi}{\partial x_1}; \end{cases} \quad (2)$$

- функции вихря

$$\omega = \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}. \quad (3)$$

Соответственно, уравнение Навье-Стокса (1) преобразуется в следующую форму в переменных «функция тока – вихрь» [1]:

$$\frac{\partial \omega}{\partial t} + u_1\frac{\partial \omega}{\partial x_1} + u_2\frac{\partial \omega}{\partial x_2} = \upsilon\left(\frac{\partial^2\omega}{\partial x_1^2} + \frac{\partial^2\omega}{\partial x_2^2}\right), \quad (4)$$

$$\frac{\partial^2\Psi}{\partial x_1^2} + \frac{\partial^2\Psi}{\partial x_2^2} = -\omega. \quad (5)$$

На твердых и неподвижных поверхностях граничные условия задаются следующим образом:

$$u_1|_\Gamma = 0;$$

$$u_2|_\Gamma = 0;$$

$$\Psi|_\Gamma = \text{const};$$

$$\omega|_\Gamma = -\left.\frac{\partial^2\Psi}{\partial n^2}\right|_\Gamma,$$

где n – нормаль к границе Γ.

На входных и выходных отверстиях задается профиль скорости, а расчетные формулы для функции тока и вихря получаются либо интегрированием, либо дифференцированием в соответствии с их определениями (2)–(3).

Также используемая модель предполагает, что кинематическая вязкость задана константой и не меняется в течение всего вычислительного процесса.

При решении уравнения Навье-Стокса необходимо также решать уравнения Лапласа, Пуассона и множество трехдиагональных нелинейных систем.

Уравнения Пуассона и Лапласа в вычислительном смысле являются «тяжелыми», так как обычно они интегрируются с помощью некоторого итерационного метода и содержат большое количество операций на каждой итерации.

Хотя решение трехдиагональной системы является достаточно несложным и быстрым действием, на каждой итерации алгоритма решения уравнения Навье-Стокса содержится N^2 (N – количество узлов сетки) таких операций.

В настоящее время одним из вариантов ускорения вычислительного процесса является использование графических ускорителей, представляющих собой массивно-параллельные процессоры с общей памятью. В отличие от центрального процессора с несколькими ядрами, один графический

процессор содержит несколько сотен ядер, объединенных в потоковые мультипроцессоры, которые могут проводить вычисления параллельно.

Наиболее развитой на данный момент технологией является система CUDA, предложенная компанией Nvidia в 2007 г. [7, 8]. 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 каждый потоковый мультипроцессор имеет некоторый объем разделяемой памяти, которая является быстрой общей памятью для всех потоков одного блока (рис. 1). Каждый потоковый мультипроцессор состоит из 8 ядер, называемых CUDA-ядрами. Наиболее современные графические ускорители содержат до 60 потоковых мультипроцессоров или 480 CUDA-ядер.

Подобные устройства были использованы ранее для решения уравнения Пуассона и хорошо себя зарекомендовали [3]. Поэтому значимым является исследование возможности объединения 2-х или 4-х графических карт по технологии Nvidia SLI или Nvidia Quadro SLI, предоставляемой современными графическими ускорителями.

Целью предлагаемого исследования было показать принципиальную возможность проведения гидродинамических вычислений на нескольких графических ускорителях CUDA. Упрощенная схема системы с двумя графическими ускорителями приведена на рис. 1. В качестве тестовой аппаратуры выступал суперкомпьютер Ивановского института ГПС МЧС России с двумя графическими ускорителями GTX 295, каждый из которых имеет по два чипа на плате. Общее количество ядер составляет: 240 CUDA-ядер в чипе $\times$ 2 чипа $\times$ 2 карты = 960 CUDA-ядер при суммарном объеме видеопамяти 3,5 Гбайт.

Система CUDA требует, чтобы расчетная задача разбивалась на блоки до 512 вычислительных потоков максимум в каждом. Потоки в блоке могут быть упорядочены линейно либо в двух- или трехмерную сетку. 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.

Кроме того, в системе CUDA с несколькими графическими ускорителями присутствует одно важное архитектурное ограничение – не существует иного способа обмена данными между чипами, кроме как через ОЗУ компьютера в той части программы, которая выполняется на ЦПУ. Таким образом, при проектировании алгоритмов решения необходимо учитывать эти ограничения и сводить количество пересылок данных на стыках к минимуму.

Можно заметить, что в данном случае можно рассматривать систему с несколькими графическими ускорителями CUDA как систему с разделённой памятью. Подобная аппроксимация позволяет применять методы, рассчитанные на системы параллельного программирования MPI. Так, например, решение уравнения Пуассона можно разделить на отдельные независимые участки со своими граничными условиями, обновляемыми на каждой итерации [2, 4].

В то же время эти независимые участки можно решать стандартными способами на каждой графической карте в отдельности [3].

Так же, как и в случае с системами MPI, нетривиальным является решение трехдиагональных систем, когда данные распределены между вычислительными процессорами. Как уже указывалось выше, в этом случае пересылки данных между графическими ускорителями необходимо осуществлять через центральный процессор и ОЗУ, следовательно, эффективные параллельные алгоритмы, например параллельная циклическая редукция [3, 6], становятся крайне неэффективными, если прогонку необходимо провести на данных, находящихся на разных картах.

Таким образом, чтобы уменьшить количество пересылок данных между картами на данном этапе решения уравнения Навье-Стокса, было принято решение перестраивать данные так, чтобы прогонка всегда осуществлялась в пределах одной карты. Это позволяет использовать один общий алгоритм решения метода прогонки [3] с разным набором данных.

В двумерном случае решение уравнения Навье-Стокса предполагает вычисление прогонки для всей области данных в двух направлениях. Пусть данные будут разделены между картами полосками по строкам. В этом случае первая прогонка будет проходить на каждой карте независимо. Для второй прогонки необходимо подготовить данные, а именно, транспонировать их. Транспонирование также может быть эффективно выполнено с использованием графических ускорителей, поскольку каждый из них может повернуть свой участок данных в своей памяти, а затем они передают свои транспонированные участки в ОЗУ, где центральный процессор записывает их вертикальными полосками. В итоге получается транспонированное общее поле данных, которое затем снова разделяется на горизонтальные полоски и передается на карты, где снова проходит прогонка, и так далее (рис. 2).

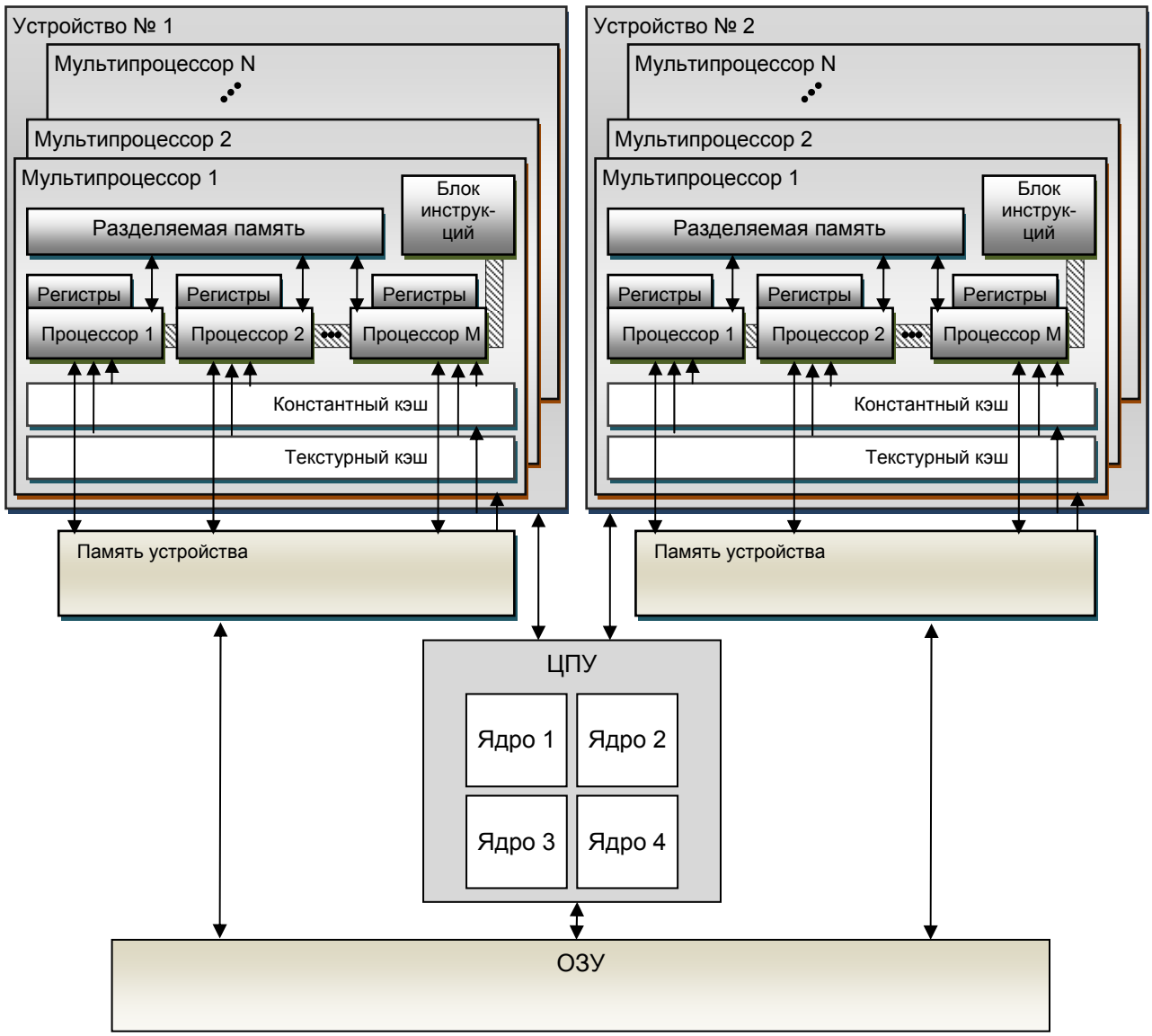

Рис. 1. Архитектура CUDA

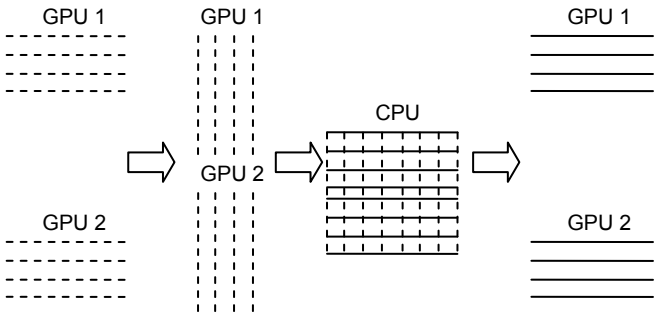

Рис. 2. Архитектура CUDA

Таким образом, предложенный алгоритм позволяет снизить накладные расходы на пересылках данных, поскольку пересылки больших объемов памяти более выгодны, чем множество маленьких пересылок данных.

Список литературы

1.**Алгоритмы** и программы для многопроцессорных суперкомпьютеров / В.В. Пекунов, С.Г. Сидоров, Л.П. Чернышева и др.; Иван. гос. энерг. ун-т. – Иваново: ИГЭУ, 2007.

2.**Евсеев А.В.** Методы решения уравнения Пуассона. – Иваново: ИГТА, 2009.

3.**Евсеев А.В.** Вопросы распараллеливания уравнения Пуассона и сравнение эффективности различных вариантов // Высокие технологии, исследования, промышленность. Т. 3: Сб. тр. IX Междунар. науч.-практич. конф. «Исследование, разработка и применение высоких технологий в промышленности». – СПб.: изд-во Политехн. ун-та, 2010. – С. 46–52.

4.**Евсеев А.В., Ясинский Ф.Н.** Распараллеливание методов решения уравнения Пуассона // Высокопроизводительные параллельные вычисления на кластерных системах: мат-лы IX Междунар. конф.-семинара. – Владимир: изд-во ВГУ, 2009. – С. 166.

5.**Самарский А.А.** Введение в численные методы. – М.: Наука, 1982.

6.**Zhang Y., Cohen J., Owens J.D.** Fast tridiagonal solvers on the GPU // Principles and Practice of Parallel Programming. – 2010. – P.127–136.

7.**http://cuda.cs.msu.su/** – специализированный курс «Архитектура и программирование массивно-параллельных

вычислительных систем» на основе технологии CUDA в МГУ им. Ломоносова.

8. **http://developer.nvidia.com/object/gpucomputing.html** – NVIDIA GPU Computing Developer.

Ясинский Федор Николаевич,
ГОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
доктор физико-математических наук, профессор кафедры высокопроизводительных вычислительных систем,
телефон (4932) 26-98-29.

Евсеев Александр Владимирович,
ГОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
аспирант кафедры высокопроизводительных вычислительных систем,
телефон (4932) 26-98-29.