А.В. Евсеев, Ф.Н. Ясинский

Ивановский государственный энергетический университет

О РЕШЕНИИ УРАВНЕНИЯ НАВЬЕ-СТОКСА В ПЕРЕМЕННЫХ «ФУНКЦИЯ ТОКА – ВИХРЬ» С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ С НЕСКОЛЬКИМИ ГРАФИЧЕСКИМИ УСКОРИТЕЛЯМИ

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

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

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

Кроме того, это уравнение можно свести к системе в переменных «Функция тока – Вихрь», которая может быть более удобной в некоторых задачах [1]. Если ограничиться двумерным случаем, то можно ввести следующие определения функции тока:

$$\begin{cases} u_1 = \dfrac{\partial \Psi}{\partial x_2}; \\ u_2 = -\dfrac{\partial \Psi}{\partial x_1} \end{cases} \tag{2}$$

и вихря:

$$\omega = \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}. \tag{3}$$

Соответственно уравнение Навье-Стокса преобразуется в следующую форму [1]:

$$\begin{cases} \dfrac{\partial \omega}{\partial t} + u_1 \dfrac{\partial \omega}{\partial x_1} + u_2 \dfrac{\partial \omega}{\partial x_2} = \upsilon \left(\dfrac{\partial^2 \omega}{\partial x_1^2} + \dfrac{\partial^2 \omega}{\partial x_2^2} \right); & (4) \\ \dfrac{\partial^2 \Psi}{\partial x_1^2} + \dfrac{\partial^2 \Psi}{\partial x_2^2} = -\omega. & (5) \end{cases}$$

На твёрдых и неподвижных поверхностях граничные условия задаются следующим образом:

$$u_1\big|_{\Gamma} = 0;$$
$$u_2\big|_{\Gamma} = 0;$$
$$\Psi\big|_{\Gamma} = \text{const};$$
$$\omega\big|_{\Gamma} = -\frac{\partial^2 \Psi}{\partial n^2}\bigg|_{\Gamma},$$

где n – нормаль к границе Г.

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 (2)–(3).

Также используемая модель предполагает, что кинематическая вязкость задана константой.

В процессе решения уравнения Навье-Стокса необходимо попутно решать уравнения Лапласа, Пуассона и множество трёхдиагональных нелинейных систем.

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

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

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, объединённых в потоковые мультипроцессоры, которые могут проводить вычисления параллельно.

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

Данные устройства хорошо себя показали при решении уравнения Пуассона [3], где использовалось одно устройство с 2 потоковыми мультипроцессорами. В то же время современные графические ускорители позволяют объединять их в общие вычислительные блоки по технологии Nvidia SLI или Nvidia Quadro SLI для объединения 2 или 4 графических карт.

Целью данного исследования было показать принципиальную возможность проведения гидродинамических вычислений на нескольких графических ускорителях. Упрощённая схема системы с двумя графическими ускорителями приведена на рисунке. 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 равно 240 CUDA-ядер в чипе × 2 чипа × 2 карты = 960 CUDA-ядер при суммарном объёме видеопамяти 3,5 Гбайт.

В системе CUDA вычисления на каждой графической карте разбиваются на блоки потоков. Каждый блок может содержать до 512 потоков, причём они могут быть упорядочены линейно или в двух- или трёхмерную 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.

Однако 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. Таким образом, при проектировании алгоритмов решения необходимо учитывать эти ограничения и сводить количество пересылок данных на стыках к минимуму.

Заметим, 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].

Сложности возникают при решении трёхдиагональных систем. В случае с несколькими графическими ускорителями общее поле решения равномерно распределено между картами. Как уже указывалось выше, в этом случае пересылки данных 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.

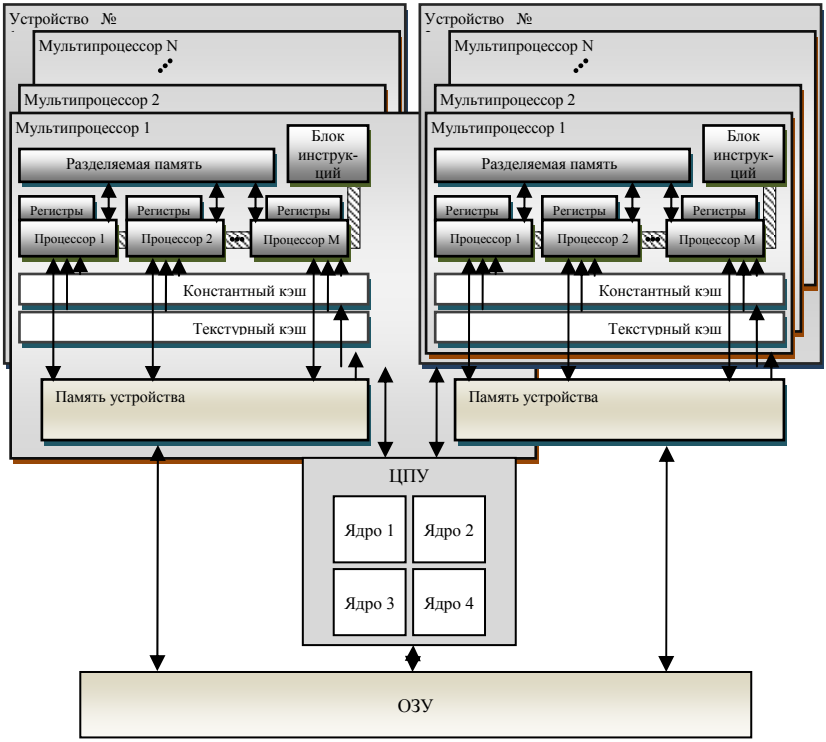

Рис. Архитектура системы с двумя устройствами CUDA

Поэтому, 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 уравнение 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. Транспо-

нирование 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 их записывает вертикальными полосками. В итоге получается транспонированное общее поле данных, которое затем снова разделяется на горизонтальные полоски и передаётся на карты, где снова проходит прогонка и т.д.

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

Список литературы

1. Алгоритмы и программы для многопроцессорных суперкомпьютеров / В.В. Пекунов [и др.]. – Иваново: Изд-во Ивановск. гос. энерг. ун-та, 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. Исследование, разработка и применение высоких технологий в промышленности: сб. тр. XI Междунар. науч.-практ. конф. – 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: материалы XI Междунар. науч.-практ. конф. – Владимир: Изд-во Владимирск. гос. ун-та, 2009. – С. 166.

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

6. 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.

7. Специализированный курс «Архитектура и программирование массивно-параллельных вычислительных систем» на основе технологии CUDA в МГУ им. Ломоносова. – URL: http://cuda.cs.msu.su.

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

С.А. Ермаков, Е.Б. Замятина, А.А. Козлов

Пермский государственный университет

ОПТИМИЗАЦИЯ РАСПРЕДЕЛЕННОГО ИМИТАЦИОННОГО ЭКСПЕРИМЕНТА ПО ВРЕМЕНИ И ПО НАДЕЖНОСТИ

Метод имитационного моделирования широко используется в качестве метода исследования сложных динамических систем в самых разных областях знаний: в бизнесе, производстве, здравоохранении, при проектировании компьютерных систем, летательных аппаратов и т.д. Системы имитационного моделирования часто используются в качестве ключевого компонента в системах поддержки принятия решения. Именно поэтому необходимо, чтобы результаты моделирования были достоверны, а следовательно, и процесс проведения имитационного эксперимента должен быть *надежным*, защищенным от сбоев, которые могут привести к потере важной информации. Поскольку объектом моделирования являются сложные системы, то и время моделирования может быть весьма и весьма значительным. По этой причине возникает настоятельная необходимость в *сокращении времени* имитационного прогона.

Сокращение времени имитационного прогона возможно, если при его проведении существует доступ к ресурсам сразу