ИСПОЛЬЗОВАНИЕ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И ГРАФИЧЕСКИХ УСКОРИТЕЛЕЙ ДЛЯ МОДЕЛИРОВАНИЯ ЗАДАЧ ГАЗОДИНАМИКИ

Б.П. Рыбакин, Л.И. Стамов

В данной работе предложены параллельные реализации двумерной нелинейной разностной схемы второго порядка точности с ограничением общей вариации (TVD). Проведено тестирование полученных алгоритмов на ряде тестовых задач. Проведено сравнение производительности при использовании различных технологий параллельного программирования на задачах газовой динамики. Предложенные схемы могут быть использованы для расчетов газодинамических задач на многопроцессорных ЭВМ и графических ускорителях.

Введение. Многие современные задачи могут быть описаны с помощью уравнений газодинамики и требуют, для своего решения, применения высокоточных численных методов, которые в свою очередь весьма громоздки с вычислительной точки зрения. Данный факт можно обойти, используя достижения в современной вычислительной технике, а именно, развитие многоядерных и многопроцессорных компьютерных систем и появление графических ускорителей, ориентированных только на высокопроизводительные вычисления. В связи с этим возникает необходимость изменения как существующих численных алгоритмов решения прикладных задач, так и создание новых с учетом особенностей всех разработанных в настоящее время архитектур вычислительных средств.

В данной работе в качестве основы использована двумерная центрированная TVD *(Total Variation Diminution)* схема второго порядка точности, предложенная Jiang и Tadmor [1], которая является расширением одномерной схемы предиктор-корректор Nessyahu и Tadmor [2]. Данный тип схем значительно проще схем, использующих инварианты Римана, и, в тоже время, он обладает неплохой точностью, гораздо лучшей, чем другие схемы данного класса и достаточной для решения большого числа задач. Тестирование данной схемы осуществлялось с помощью ряда одномерных и двумерных тестовых задач, рассмотренных в работах Toro [3] и Liska и Wendroff [4].

В качестве методов параллельного программирования были использованы открытый стандарт для распараллеливания программ OpenMP [5,6] и технология CUDA [7]. OpenMP (*Open Multi-Processing*) представляет собой набор директив, процедур и функций, предназначенных для программирования многопоточных программ на многопроцессорных вычислительных системах с общей памятью. Все вычисления проводятся на центральных процессорах CPU (*Central Processing Unit*). В OpenMP параллельные вычисления реализуются с помощью многопоточности, в которой главный поток (*master*) создает набор подчиненных потоков (*slave*) и задача распределяется между всеми данными потоками. Данные потоки выполняются параллельно на вычислительной системе с несколькими процессорами, при этом не обязательно, чтобы число потоков совпадало с числом процессоров. Технология CUDA (*Compute Unified Device Architecture*), разработанная компанией Nvidia, представляет собой язык программирования со своим компилятором и библиотеками для программирования графических процессоров GPU (*Graphics Processing Unit*) фирмы Nvidia. В настоящее время архитектура графических процессоров такова, что они, работая по принципу SIMT (*Single Instruction Multiple Thread*), позволяют одновременно обрабатывать большое количество данных (не обязательно графических), и тем самым проводить численные расчеты больших объемов. На графической карте создается несколько тысяч параллельных (конкурирующих) потоков, которые объединяются в потоковые блоки (одно-, дву-, трехмерные). Каждый блок выполняется на своем потоковом мультипроцессоре (*Streaming multiprocessor*) – одной из составных частей графического процессора. Мультипроцессор состоит из потоковых процессоров (*Streaming Processor*), на которых уже выполняется минимальная часть блока — варп (*warp*), величина которой отличается для различных графических процессоров. Каждый блок и поток в блоке имеют свой идентификационный номер, с помощью которого и можно указать, что данный поток должен сделать. Данные номера можно узнать с помощью встроенных переменных ThreadIdx и BlockIdx. При этом все операции на GPU проводятся с данными, расположенными на GPU, которые необходимо скопировать в память GPU по сравнительно медленной шине PCI-E. Кроме того, память GPU ограничена по объему. Это создают дополнительные трудности при программировании графических процессоров, которые должны обязательно учитываться при проведении вычислений на них [8]. Пример распараллеливания циклов с помощью этих двух архитектур приведен на Рис. 1. Если код программы, написанной с помощью директив OpenMP весьма схож с кодом обычной программы, то код, написанный для вычисления на графическом ускорителе существенно отличается из-за другой архитектуры и принципа работы.

Реализация численных схем проводилась на языке Fortran 95 с помощью компилятора PGI Visual Fortran 11.4 фирмы Portland Group [9], который поддерживает как стандарт OpenMP, так и технологию CUDA.

Все расчеты проводились на процессоре Intel Core i5-460M и графической видеокарте фирмы Nvidia GeForce GT 420M, с объемом памяти 1Gb. Данная графическая карта сделана по новой технологии Fermi,

которая предоставляет новые возможности для программирования графических процессоров. Центральный процессор (CPU) имеет два физических ядра и поддерживает технологию Hyper-Threading. В данной технологии каждое физическое ядро определяется как два логических, что, при определенных условиях, позволяет увеличить производительность процессора.

```
!$OMP PARALLEL PRIVATE(i,j)
!$OMP & SHARED(A,B,C,K,NX,NY)

!$OMP DO
do i=1,NX
   do j=1,NY
      A(i,j)=B(i,j)+K*C(i,j)
   enddo
enddo
!$OMP END DO
!$OMP ENDPARALLEL
```

```
i = (blockIdx%x-1)*blockDim%x + threadIdx%x
j = (blockIdx%y-1)*blockDim%y + threadIdx%y

if (i>=1 .AND. i<=NX) then
   if (j>=1 .AND. j<=NY) then
      A(i,j)=B(i,j)+K*C(i,j)
   endif
endif
```

Рис. 1. Пример фрагмента программы, написанной с помощью директив OpenMP (слева) и CUDA (справа).

Постановка задачи. Расмотрим задачу взаимодействия сферической ударной волны с сферической полостью, заполненной газом пониженной плотности. Данная задача представляет собой практический интерес и, кроме того, позволяет оценить устойчивость разностных схем и проверить работоспособность параллельных программ программ [10].

Система уравнений газовой динамики может быть записана в следующем виде:

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho \mathbf{v}) = 0,$$

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\,\mathbf{v} + \frac{1}{\rho}\nabla p = 0,$$

$$\frac{\partial E}{\partial t} + \operatorname{div}\{(E+p)\,\mathbf{v}\} = 0,$$

где ρ - плотность, $\mathbf{v}$ - вектор скорости, p - давление, $E = \rho\left(\frac{1}{2}\mathbf{v}^2 + e\right)$ - полная энергия, e - удельная внутренняя энергия. Данная система может быть дополнена уравнением состояния для идеального газа:

$$p = \rho\,(\gamma - 1)\,e,$$

где γ - показатель адиабаты.

Задача взаимодействия ударной волны с областью пониженной плотности рассматривается на квадрате *S*: $\{0 \le x \le 10,\ 0 \le y \le 10\}$. Область *S* в начальный момент времени равномерно заполнена покоящимся газом, плотность которого $\rho_0 = 1$ и давление $p_0 = 1$. В центре данной квадратной области, в точке $X_1(5,5)$, задается ударная волна сферической формы с радиусом $R_1 = 1$, плотностью и давлением в центре $\rho_1 = 2$ и $p_1 = 10$ соответственно. В точке $X_2(2.5, 2.5)$ задается сферическая полость пониженной плотности радиуса $R_2 = 0.5$ с плотностью $\rho_2 = 0.1$. Показатель адиабаты для данного случая был выбран $\gamma = 1.4$. При рассмотрении временной эволюции в данной задаче возникает распространяющаяся от центра области сферическая ударная волна, которая взаимодействует с областью пониженной плотности. Результат данного взаимодействия представлен на рис. 2.

Результаты исследования. Как следует из полученных результатов, при взаимодействии ударной волны с полостью пониженной плотности образуется симметричные вихри (Рис. 2). Такой результат свидетельствует о надежности и пригодности рассматриваемой схемы для решения задач такого рода.

Полученный алгоритм распараллелен с помощью директив OpenMP и протестирован на одной, двух и четырех нитях. Результаты времени счета для данного и других методов при различных размерах сетки представлены на Рис. 3 и Рис. 4. Как видно из полученных данных, при использовании двух потоков наблюдается прирост производительности, в среднем в 1.5 раза по сравнению с одним потоком (или одним ядром). При создании четырех потоков прирост незначителен, а для некоторых размеров сеток отсутствует. Данный факт можно объяснить тем, что в вычислительной системе присутствуют только два действительных (физических) ядра, остальные два только логические. Такая технология не всегда дает такое же увеличение производительности, как при наличии четырех действительных ядер, и не всегда положительно влияет на

производительность программ. Время расчета программы без использования каких-

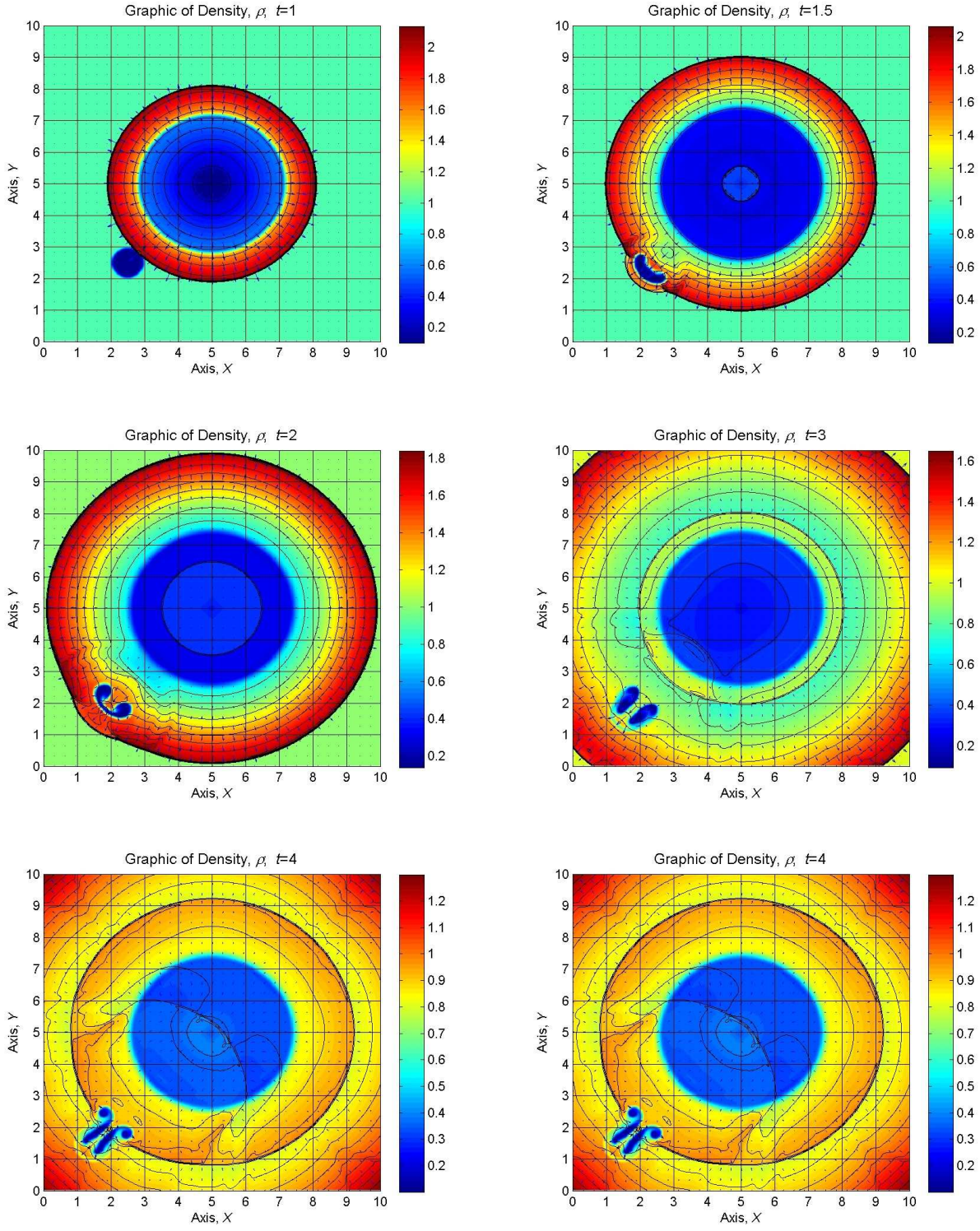

Рис. 2. Временная эволюция задачи взаимодействия ударной волны с полостью пониженной плотности для сетки 512×512 ячеек.

либо средств распараллеливания практически идентично времени расчета одной нити при использовании технологии OpenMP, поэтому данное время не рассматривается отдельно. Следует отметить, что при некоторых размерах сетки программа без использования распараллеливания работает немного быстрее, чем для одной нити в технологии OpenMP. Это может быть связано со спецификой работы OpenMP.

При расчете на графическом ускорителе с помощью технологии CUDA наблюдается (рис. 3,4) существенный прирост производительности, который увеличивается при возрастании числа ячеек в сетке. При уменьшении числа ячеек наблюдается обратный процесс — время расчетов увеличивается по сравнению с

временем расчетов на центральном процессоре. Таким образом, при малых размерах области, выгоднее в качестве средства счета использовать центральный процессор, так как время на пересылку данных между графическим ускорителем и процессором больше времени счета. При увеличении размеров сетки время расчета на центральном процессоре резко возрастает, при этом для больших размеров сетки начинает проявляется преимущество архитектуры графического процессора. На задачах большой размерности время счета заметно сокращается.

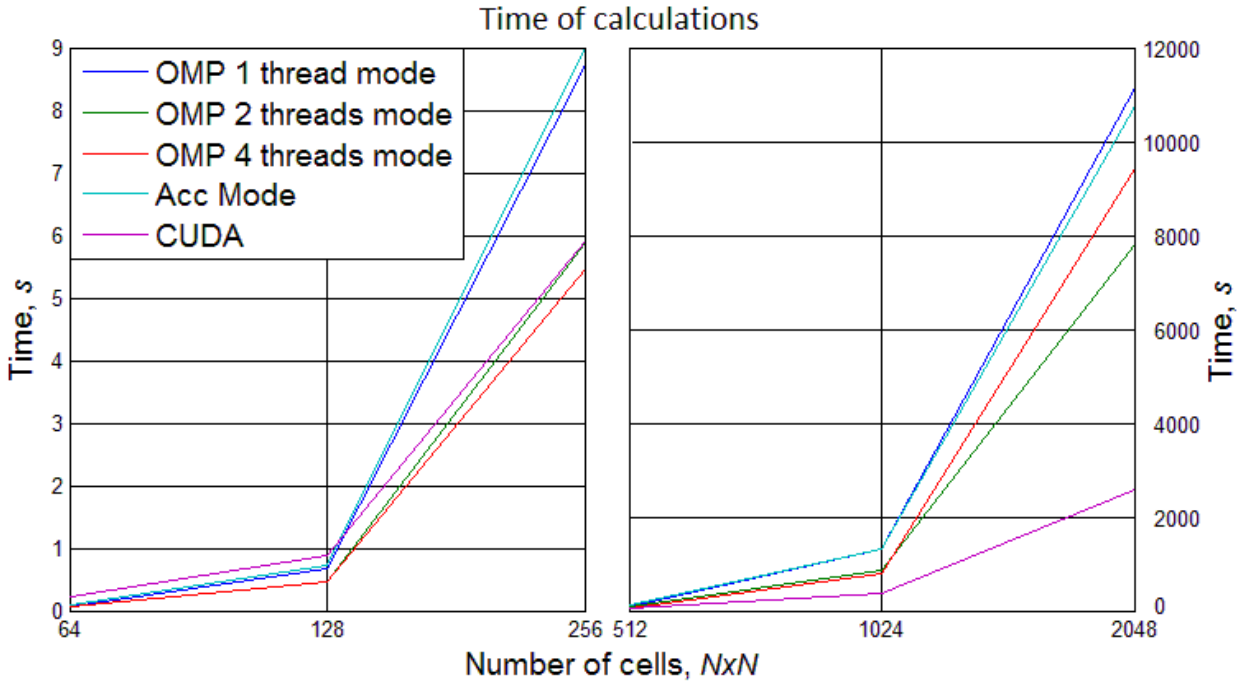

Рис. 3. График времени выполнения для задачи взаимодействии сферической ударной волны с областью пониженной плотности при использовании различных методов распараллеливания в зависимости от размера используемой сетки.

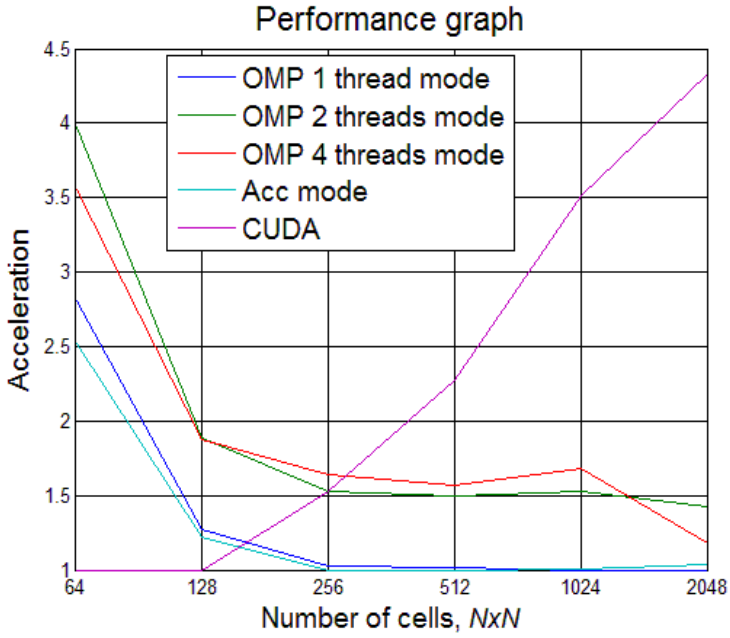

Рис. 4. График ускорения для задачи взаимодействия сферической ударной волны с областью пониженной плотности при использовании различных методов распараллеливания в зависимости от размера используемой сетки.

Рассмотренный алгоритм также протестирован с помощью встроенных средств распараллеливания на

графическом ускорителе, предоставляемых компилятором PGI Fortran (рис. 3,4). К сожалению, несмотря на появление ускорения для самой большой из рассмотренных сеток, заметного прироста производительности на данной задаче с помощью встроенных директив получить не удалось. Данный факт может быть объяснен сложностью рассматриваемой задачи с точки зрения пересылки информации на графический процессор и тем, что программист не может в должной мере контролировать процессы пересылки информации и проведения вычислений на графической карте. Стоит отметить, что несмотря на отсутствие эффективности для данной задачи, в некоторых других, более простых, с точки зрения архитектуры графических процессоров, задачах можно быстро, не углубляясь в основы программирования графических карт, получить существенный прирост производительности, например, в задачах обработки матриц больших размерностей [8]. Однако, при программировании графических процессоров «вручную», производительность будет существенно больше.

Выводы. В данной работе были предложены и изучены несколько параллельных реализаций центрированной TVD схемы второго порядка точности, полученные с помощью различных технологий параллельного программирования, для решения различных задач газовой динамики на многопроцессорных вычислительных системах и графических ускорителях, а также рассмотрено влияние данных технологий и архитектур на производительность на примере задачи взаимодействия сферической ударной волны с областью пониженной плотности. Приведены результаты расчетов, полученных с помощью центрального процессора (CPU) и графической видеокарты (GPU). Показана эффективность распараллеливания данной задачи с помощью технологии OpenMP на многопроцессорных вычислительных системах и с помощью технологии CUDA на графических ускорителях.

Из полученных результатов можно сделать вывод, что эффективность использования графических карт гораздо выше, чем многопроцессорных систем с небольшим числом вычислительных узлов, и их использование дает существенный прирост производительности. Рост производительности также существенно зависит от числа процессоров и использования специализированных графических ускорителей.

Существенным фактором, влияющим на производительность, является учет особенностей используемых архитектур вычислительной техники и ее характеристики. Так, необходимо учитывать тот факт, что скорость передачи данных на графическую карту гораздо медленнее, чем скорость расчетов, проводимых на ней. Таким образом, данные необходимо подготовить и передать на видеокарту перед началом вычислений и сократить, по мере возможности, передачу информации между центральным и графическим процессорами. Также необходимо учитывать ограниченность объема памяти у графической карты. При невозможности разместить всю передаваемую информацию в памяти GPU, или при неправильной организации передачи данных, время расчетов резко возрастает.

Существует множество других факторов, которые необходимо учитывать при проведении расчетов на графических ускорителях, например, то, каким образом осуществляется доступ к памяти внутри видеокарты, как передается информация, насколько вычислительная задача подходит для рассматриваемой архитектуры (так как для некоторых задач можно получить обратный эффект — расчет будет проводиться существенно медленнее, чем без использования данных средств распараллеливания) и так далее.

Можно получить различные реализаций рассматриваемого алгоритма вычислительной задачи на конкретной архитектуре вычислительной техники, при этом каждая реализация алгоритма будет по своему учитывать особенности работы с вычислительным узлом. Различная запись одного и того же алгоритма решения может как увеличить производительность, так и уменьшить ее.

К увеличению производительности также приводит использование комбинированных архитектур, которые сочетают возможности многоядерных процессоров с возможностями графических ускорителей и используют различные технологии параллельного программирования. Использование гибридных систем, содержащих большое число многоядерных процессоров и мощных вычислительные графических карт позволяет существенно сократить время вычислений большинства существующих задач.

ЛИТЕРАТУРА:

1. G.-S. Jiang and E. Tadmor. Nonoscillatory central schemes for multidimensional hyperbolic conservation laws, SIAM J. SCI. COMPUT., **19** (1998), pp. 1892-1917.
2. H. Nessyahu, E. Tadmor. Non-oscillatory central differencing for hyperbolic conservation laws. J. Comp. Phys., **87** (1990), pp.408-463.
3. E. Toro. Riemann Solvers and Numerical Methods for Fluid Dynamics. Springer-Verlag, Berlin, Heidelberg, 1997.
4. R. Liska, B. Wendroff. Comparison of several difference schemes on 1d and 2d test problems for the Euler equations. SIAM J. SCI. COMPUT., **25** (2003), pp. 995-1017.
5. OpenMP Application Program Interface, Version 3.0, May 2008. http://openmp.org.
6. А.С. Антонов. Параллельное программирование с использованием технологии OpenMP. -М: Издательство Московского Университета, НИВЦ, 2009.
7. NVIDIA CUDA™. NVIDIA CUDA C Programming Guide. Version 3.1.1. NVIDIA Corporation, 2010. http://www.nvidia.ru/object/cuda.

8. Б.П. Рыбакин. Параллельное программирование для графических ускорителей. -М.: НИИСИ РАН, 2011.
9. CUDA Fortran Programming Guide and Reference. The Portland Group, Release 2011. http://www.pgroup.com/resources/cudafortran.htm.
10. Б.П. Рыбакин. Параллельная трехмерная TVD схема для решения задач гравитационной газовой динамики // Материалы международной научной конференции «Параллельные вычислительные технологии ПаВТ-2009», г. Нижний-Новгород, 30 марта - 3 апреля 2009 г., стр. 673-679.