УСКОРЕНИЕ РАСЧЕТОВ ЗАДАЧ ГРАВИТАЦИОННОЙ ГАЗОВОЙ ДИНАМИКИ С ПОМОЩЬЮ СПЕКТРАЛЬНОГО РЕШЕНИЯ УРАВНЕНИЯ ПУАССОНА НА GPU

Б.П. Рыбакин[1,2]

[1] *НИИСИ РАН*
[2] *МГУ им. М.В.Ломоносова*

Абстракт. Данная работа посвящена построению параллельного алгоритма решения уравнения Пуассона на многопроцессорных ЭВМ и GPU. Решение уравнения Пуассона необходимо для нахождения гравитационного потенциала, который используется в задачах гравитационной газовой динамики. Для моделирования трехмерных задач астрофизики необходимо использовать сетки с большим количеством вычислительных ячеек. Нахождение гравитационного потенциала в таких задачах занимает более 60% общего времени расчета. Поэтому ускорение вычислений с помощью графических процессоров позволяет уменьшить общее время вычислений. В работе были использованы технологии параллельного программирования OpenMP и CUDA.

Адаптивное уточнение расчетной сетки. Расчеты с помощью метода AMR позволяют улучшить качество расчетов и уменьшить время вычислений [1]. Качество расчетов улучшается за счет того, что мы измельчаем расчетную сетку в наиболее важных с точки зрения задачи областях — зонах больших градиентов, около границ и в других необходимых зонах. Размеры ячеек сетки в областях достаточно гладкого решения можно сделать значительно большими. Таким образом нет необходимости покрывать всю расчетную сетку ячейками небольшого размера, и поэтому снижение общего количества расчетных ячеек приводит к уменьшению общего времени расчета. Расчетная сетка строится с помощью AMR. После построения сетки уровня L_i объединяются в patch (прямоугольные заплаты). Таким образом получается иерархическая сетка, на которой решается система уравнений. Иерархия сетки зависит от задачи и часто получается достаточно сложной. В ней одновременно используются сетки различных уровней — начиная с самого «грубого» и до самого «тонкого». Каждый уровень представляет собой набор заплат различного размера. Разбиение каждого уровня, кроме нулевого, осуществляется динамически, учетом указанных выше критериев.

Уравнение Пуассона можно представить в виде:

$$\nabla^2\Phi(r) = 4\pi G\rho, \qquad (1)$$

Для задач, связанных с моделированием взрыва сверхновой звезды, формирования планетных систем из газопылевых облаков критерием уточнения сетки служит градиент плотности. Рассмотрим алгоритм построения вложенных сеток для коллапсирующей звезды. В трехмерной расчетной области выделяются области, в которых плотность превосходит некоторое заданное значение. В этой области каждая ячейка сетки нулевого уровня делится на 8 ячеек. Таким образом создается набор вложенных сеток уровня L_{i-1} из этого набора создается заплата, которая в трехмерном случае имеет вид параллелепипеда или куба. Такая заплата состоит как из ячеек, которые удовлетворяют выбранному критерию разбиения, так и не удовлетворяют ему. Это сделано для того, чтобы форма заплаты была достаточно простой. Такое разбиение позволяет организовать эффективное распараллеливание [2]. Каждый уровень представляет собой структуру, которая содержит всю необходимую информацию. При создании нового уровня создается новая структура, которая заполняется необходимой информацией. Для перехода между уровнями используются специальные подпрограммы перехода от уровня L_i к уровню L_{i-1}, при движении к точным сеткам или от уровня L_{i-1} к уровню L_i при переходе к более грубым сеткам.

Нахождение гравитационного потенциала осуществлялось с помощью методов Якоби, SOR, методом сопряженных градиентов и с помощью быстрого преобразования Фурье. Для проверки выбранных методов были проведены тестовые расчеты. В качестве теста была выбрана задача, приведенная в [3]. В ней задается функция $\Phi(r)$ и приведено аналитическое решение для потенциала. На Рис. 1 приведено аналитическое решение(слева), справа показано численное решение для 5 уровней вложенности сеток.

$$\Phi(r) = \begin{cases} 2\pi G\rho_0(R^2 - r^2/3) & \text{if} \quad r \le R \\ \frac{4}{3}\pi G\rho_0 R^3/r & \text{if} \quad r > R \end{cases},$$

$$\nabla\Phi(r) = \begin{cases} -\frac{4}{3}\pi G\rho_0 r & \text{if} \quad r \le R \\ -\frac{4}{3}\pi G\rho_0 R^3/r^2 & \text{if} \quad r > R \end{cases};$$

На Рис. 1 приведено аналитическое решение(слева), справа показано численное решение для 5 уровней вложенности сеток.

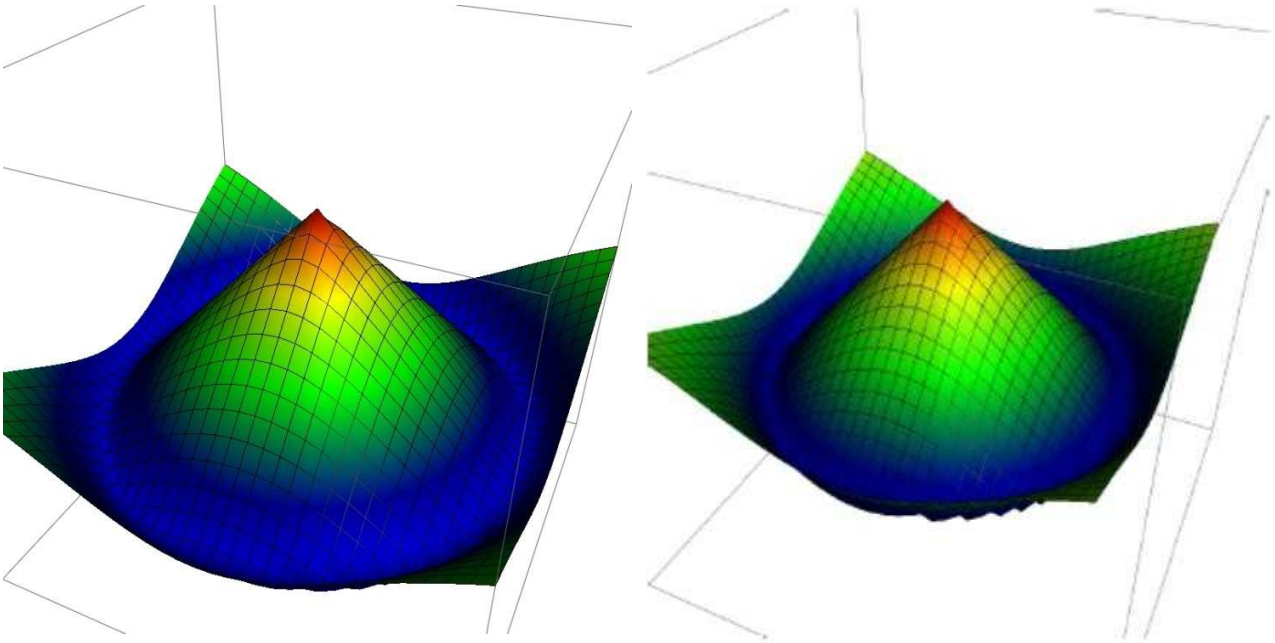

Рис. 1. Аналитическое решение тестовой задачи (слева) и численное решение (справа)

На Рис. 2 приведена разность аналитического и численного решений для сетки 32x32 (для одного уровня - слева) и сетки 512x512 для 5 уровней вложенности сеток (справа) [4].

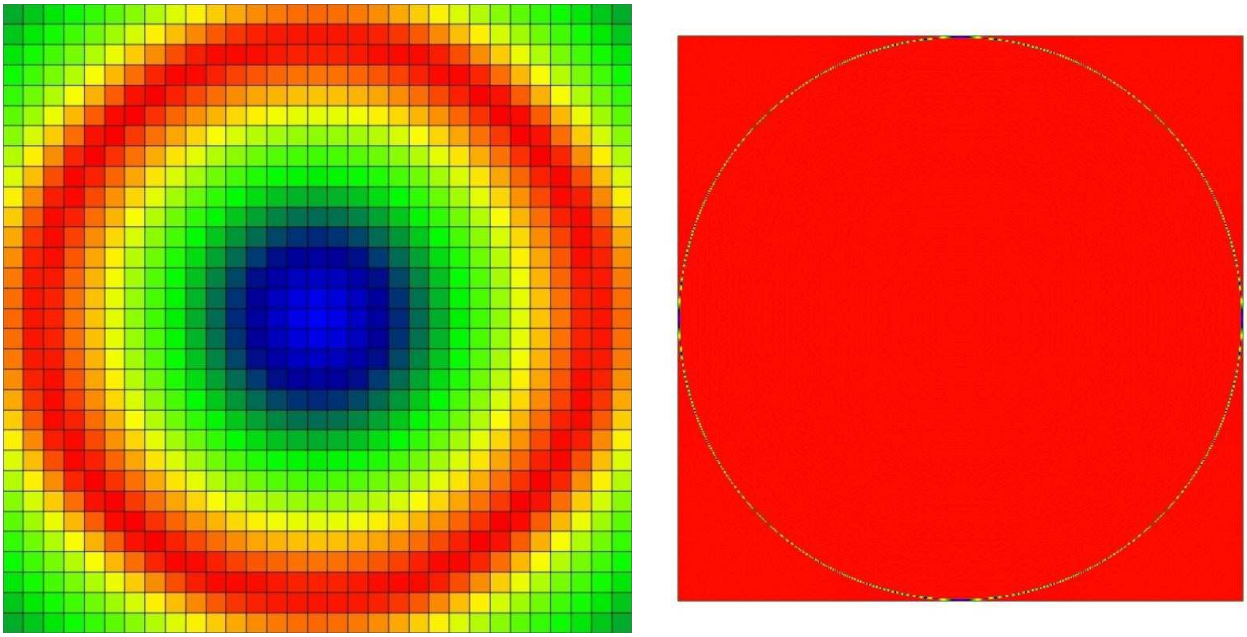

Рис. 2. Разность между аналитическим и численным решениями. Слева для сетки 32x32 и одним уровнем, справа для сетки 512x512 и 5 уровнями.

Ускорение решения с помощью OpenMP и CUDA.

Для ускорения решения было использовано распараллеливание с помощью технологии OpenMP. На Рис. 3. приведены результаты численных экспериментов по расчету трехмерного уравнения Пуассона на 5 вложенных уровнях сетки. Расчеты проводились на 2 процессорах Intel Xeon E2620, каждый процессор имеет 6 ядер и поддерживает технологию HT.

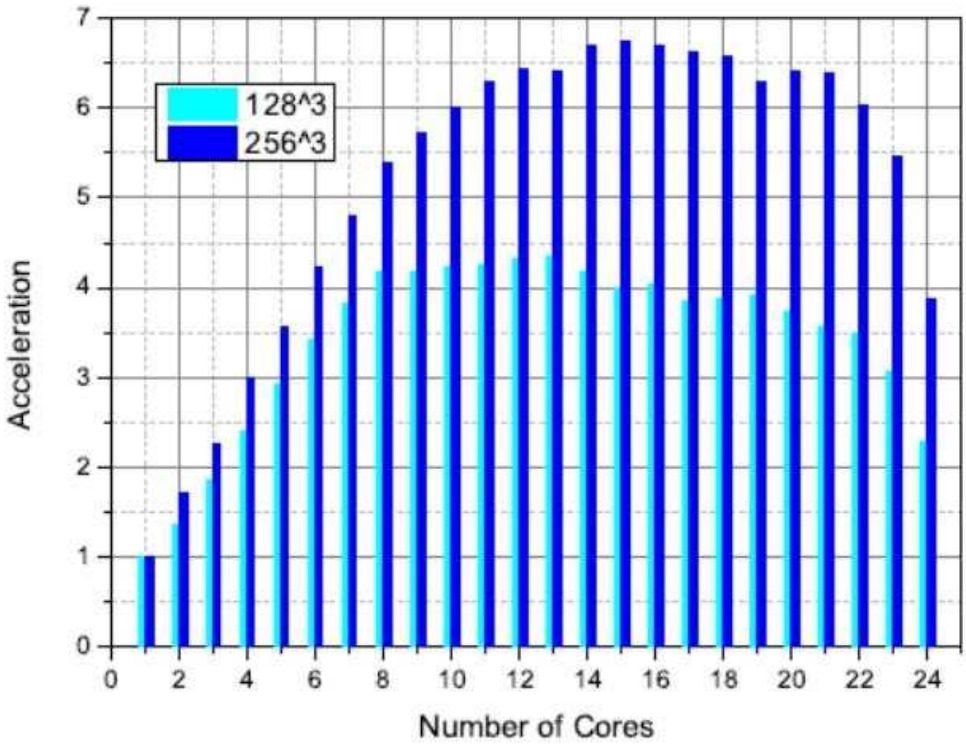

Рис. 3. Зависимость ускорения от количества ядер для двух сеток: 128x128x128 и 256x256x256

Отметим, что наблюдается практически линейный рост ускорения до 12 ядер сетки 256x256x256. Для сетки 128x128x128 линейный рост наблюдается до 8 ядер. Это связано с одной стороны с объемом вычислений, а с другой стороны тем, что использовалась технология HyperThreading [6-8].

Рассмотрим различные варианты решения уравнения Пуассона на графических процессорах. Разобьем расчетную область на отдельные двумерные блоки. Размеры любого блока выбираются из следующих соображений:

1. Каждый блок включает в себя расчетную сетку с перекрытием, то есть он окаймляется с каждой стороны несколькими рядами из соседних расчетных областей. Количество этих рядов выбирается в зависимости от разностного шаблона.
2. Размеры блоков выбираются таким образом, чтобы доступ к памяти был выровненным (coalescing).
3. Размеры блоки данных выбирается так, чтобы избегать конфликтов данных для распределения в shared memory и не превзойти доступного размера shared memory.

Реализация решения, и в том числе выбор размеров блоков зависит от графического устройства и версии программного обеспечения. Для GPU на основе архитектуры Keple или Fermi, например карт Tesla K20 и M2090, максимальное значение block_Xxblock_Y равно 1024, а для карт предыдущего поколения – 512. Решение уравнения Пуассона осуществлялось с помощью метода Якоби на компактном девятиточечном шаблоне [5]. Мы размещаем каждый блок в разделяемую память (shared memory), тогда все потоки из блока могут обращаться к любому элементу этого фрагмента. Так как мы делаем перекрытие блоков с соседними, то общая часть загружается два раза — один раз текущим блоком потоков,и второй раз соседним блоком. В зависимости от размеров блоков 8 x 16, 8 x 32, и т. д. мы загружаем из глобальной памяти уже загруженные области, что снижает общую производительность. Более выгодно организовать блоки размерами 4 x 64 по оси, вдоль которой производятся вычисления. При этом мы избегаем перекрытия при расчете на таком шаблоне для сетки 64x64x64. Каждый warp для такого фрагмента читает из глобальной памяти соседние данные, поэтому достигается хорошее объединение операций доступа к глобальной памяти.

Если проанализировать этот вариант с опцией компилятора -Mcuda=ptxinfo мы увидим, что оно потребляет 14 регистров и задействует 1152 байта разделяемой памяти. На Tesla K20x

при полной занятости количество регистров на один поток не должно превышать 32 (65536 регистров/2048 потоков на одном мультипроцессоре). Далее, 1152 байта разделяемой памяти в одном блоке потоков, умноженные на 16 (максимальное число блоков потоков на одном мультипроцессоре), помещаются в 46 КБ разделяемой памяти, доступной мультипроцессору. В данном случае ядро будет работать с полной занятостью.

Решение уравнения Пуассона, которое осуществлялось на девятиточечном шаблоне с помощью итерационного алгоритма. Вычисляются новые значения в ячейке i,j,k. После проведения N итераций вычисляется максимальная невязка между соседними итерациями и на основе ее значения принимается решение об остановке. Вычисления прекращаются также после заданного максимального числа итераций. Для того, чтобы избежать работы с медленной глобальной памятью, разместим наши данные в общей памяти.

Ниже приведен фрагмент кода kernel, который проводит вычисления с использованием общей памяти:

```
i = (blockIdx%x-1)* blockDim%x + threadIdx%x
j = (blockIdx%y-1)* blockDim%y + threadIdx%y
 ishared = threadIdx%x
 jshared = threadIdx%y
      if (i > 1 .and. j > 1)  dev(ishared-1, jshared-1) = a(i-1, j-1)
      if (i > 1 .and. j < ny .and. jshared >= nBlock_Y-2) then
 dev(ishared-1, jshared+1) = a(i-1, j+1)
    endif
    if (i < nx .and. j > 1 .and. ishared >= nBlock _X-2) then
 dev(ishared+1,jshared-1) = a(i+1,j-1)
    endif
    if (i < nx .and. j < ny .and.  ishared >= nBlock _X-2 .and. jshared >= nBlock _Y-2) then
 dev(ishared+1,jshared+1) = a(i+1,j+1)
    endif
 call syncthreads()
u_d(i,j) = 0.2_fp_kind * (74 dev(ishared,js-1) + dev(ishared-1,jshared) +  &
dev(ishared+1,jshared) + dev(ishared,jshared+1))  + 76 + 0.05_fp_kind * &
(dev(ishared-1,jshared-1) + dev(ishared+1,jshared-1) + &
dev(ishared-1,jshared+1) + dev(ishared+1,jshared+1))
```

Можно переписать этот код с использованием текстурной памяти, отличие будет состоять в том, что в правой части будут ссылки на текстуры, которые вычислялись на предыдущей итерации. Проведенные расчеты показали, что для GPU с архитектурой Кеплер разница во времени выполнения варианта с глобальной памятью в 2.1 раза медленнее, чем вариант с общей памятью, использование текстурной памяти приводит к ускорению в 2.21 раза по сравнению с глобальной памятью.

Решение уравнения Пуассона с помощью быстрого преобразования Фурье из библиотеки CUFFT.

В CUDA существует несколько библиотек, использование которых позволяет облегчить программирование и ускорить выполнение программ. Воспользуемся библиотекой CUFFT, которая содержит подпрограмму быстрого преобразования Фурье. Эта библиотека может преобразовывать массивы, которые содержат до 2^{27} чисел одинарной точности и столько же чисел двойной точности. Если размер массива можно представить в виде произведения $2^a3^b5^c7^d$ эта библиотека порождает высокооптимизированные ядра. Для работы с CUFFT необходимо сделать следующие шаги:

1. **Создать план.** Нужно выполнить операции выделения памяти и инициализации, которые необходимы библиотеке. Для этого применяются разные функции: cufftPlan1d, cufftPlan2d или cufftPlan3d, они используются в зависимости от размерности исходных данных В одномерном случае библиотека CUFFT может преобразовывать сразу несколько массивов, для чего в функции cufftPlan1d предусмотрен аргумент batch.
2. **Выполнить план.** После создания плана, для вычисления входной последовательности БПФ можно вызывать различные функции. Если размерность данных не изменяется, то функцию можно вызывать несколько раз, не создавая нового плана. В случае преобразования комплексные-комплексные нужно еще задать направление: CUFFT_FORWARD для преобразования из физического пространства в пространство Фурье или CUFFT_INVERSE для обратного преобразования. Для преобразований вещественные-комплексные неявно подразумевается направление (CUFFT_FORWARD), а для преобразований комплексные-вещественные – направление CUFFT_INVERSE. В последнем случае

также предполагается, что исходные данные эрмитовы (для гарантии существования обратного преобразования, дающего вещественные значения).

3. **Удалить план.** На этом шаге освобождаются ресурсы, выделенные при создании плана. Функция называется cufftDestroy.

Для решения уравнения Пуассона применим спектральный метод в виде быстрого преобразования Фурье. Применяя преобразование Фурье к двумерному уравнению Пуассона, мы получаем систему алгебраических уравнений для каждой пары волновых чисел *kx*, *ky*:

$$-\left(k_x^2 + k_y^2\right)\hat{u}_{k_x,k_y} = \hat{\rho}_{k_x,k_y}$$

Коэффициенты Фурье $u_{kx,ky}$ можно найти из этого простого уравнения, а применив обратное преобразование Фурье, мы получим решение в физическом пространстве [5]. Таким образом, шаги решения двумерного уравнения Пуассона спектральным методом выглядят так.

- Находятся коэффициенты Фурье $\rho\hat{}_{kx,ky}$ функции ρ с помощью двумерного БПФ.
- Находим коэффициенты Фурье $u_{kx,ky}$ функции u по формуле $-\rho\hat{}_{kx,ky}/(k_{x2} + k_{y2})$. Решение не определено при $k_x = k_y = 0$, что соответствует неизвестной константе c, которая должна быть задана. Это очевидно следует из того факта, что если u – решение уравнения Пуассона, то $u + c$ – тоже решение. Для задач с граничным условием Дирихле эту константу легко найти.
- Выполняем обратное двумерное БПФ $u_{kx,ky}$ для перехода в физическое пространство и используем граничные условия.

Фрагмент программы решения уравнения Пуассона с помощью спектрального метода выглядит так:

```
! Преобразуем вещественный массив в комплексный
    call real2complex1D<<<64,128>>>(rinput_d,cinput_d,N,M)
! Выполняем преобразование  с двойной точностью на месте
  if (fp_kind == doublePrecision) call cufftExecZ2Z(plan, cinput_d, cinput_d, CUFFT_FORWARD)
! Вызваем ядро решения уравнения Пуассона в пространстве Фурье
    call solve_poisson<<<grid,Block>>>(cinput_d,kx_d,ky_d,N,M)
! Выполняем обратное преобразование на месте
            if (fp_kind == doublePrecision) call cufftExecZ2Z(plan, cinput_d, cinput_d,
CUFFT_INVERSE)
```

Для процессора Kepler K20x решение Пуассона спектральным методом на сетке 1024x1024 занимает около 6 мс:

```
Уравнение Пуассона на сетке  : 1024x1024
Затрачено времени (мс)       : 5.749312
Ошибка по норме L            : 2.3077315062211532E-005
Ошибка по норме L2           : 5.6284773431400125E-009
```

Выводы: Приведены результаты работ по построению параллельных алгоритмов и программ решения уравнения Пуассона на вложенных сетках. Построены решения для многопроцессорных ЭВМ и графических ускорителей с помощью технологий OpenMP и CUDA. Для решения на графических ускорителях был использован спектральный метод быстрого преобразования Фурье из библиотеки CUFFT. Решение этим методом на сетке 1024x1024 работает значительно быстрее, чем на 24 ядрах процессора Xeon 2620 и быстрее, чем с помощью метода Якоби, реализованного на GPU.

В дальнейшем планируется построить программу решения уравнений гравитационной газовой динамики, которая использует асинхронную работу центрального процессора и графического ускорителя.

Данная работа была выполнена при поддержке Программы № 14 Президиума РАН.

ЛИТЕРАТУРА:

1. D.S. Balsara, C.D. Norton. Highly parallel structure adaptive mesh refinement using parallel language-based approaches. Elsevier, Parallel Computing (27), 2001, 37-70 pp.
2. Б.П. Рыбакин "Параллельная трехмерная TVD схема для решения задач гравитационной газовой динамики" // ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ (ПаВТ2009), Нижний Новгород, 30 марта - 3 апреля 2009, стр.673-679.
3. James M. Stone, Michael L. Norman. ZEUS-2D: A Radiation Magnetohydrodynamics Code for Astrophysical Flows in Two Space Dimensions. I. The Hydrodynamic Algorithms and Tests. The Astrophysical Journal Supplement Series, 80:753-790, 1992 June.
4. B. Rybakin, P. Bogatencov, G. Secrieru, N. Iluha. Using Adaptive Mesh Refinement Strategy for Numerically Solving of Gas Dynamics Problems on Multicore Computers. Springer, Modeling and Optimization in Science and Technologies Volume 2, 2014, pp 123-130

5. Gregory Ruetsch and Massimiliano Fatica. CUDA Fortran for Scientists and Engineers. Best Practices for Efficient CUDA Fortran Programming. AMSTERDAM, 364 pp.
6. B. Rybakin. Modeling of III-D Problems of Gas Dynamics on Multiprocessing Computers and GPU. Elsevier, Computers & Fluids, Volume 80, 10 July 2013, 403–407.
7. B. Rybakin, L.Stamov, E.Egorova. Accelerated Solution of Problems of Combustion GasDynamics on GPUs. Computers and Fluids, Elsevier, V. 90, February 2014, 164–171.
8. Rybakin, V. Goryachev. The supersonic shock wave interaction with low-density gas bubble. Acta Astronautica, Elsevier, 94, February 2014, 749-753