РЕАЛИЗАЦИЯ РАЗНОСТНОГО РЕШЕНИЯ УРАВНЕНИЙ МАКСВЕЛЛА НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ МЕТОДОМ ПИРАМИД

С.А. Малышева [1], Д.Л. Головашкин [1,2]

[1] Институт систем обработки изображений РАН, Самара, Россия
[2] Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет)

Работа посвящена развитию метода пирамид в приложении к решению системы уравнений Максвелла во временной области посредством конечных разностей (FDTD) и его реализации на графическом процессоре. Применение этого метода позволяет снять ограничение на объем памяти графического вычислительного устройства, существенное для FDTD.

Ключевые слова: FDTD, уравнения Максвелла, метод пирамид, GPU, CUDA.

Введение

Метод разностного решения уравнений Максвелла во временной области (FDTD) [1] широко используется в современных исследованиях при моделировании: распространения излучения [2], материалов с новыми свойствами [3], взаимодействия излучения с биологическими объектами [4] и во многих других приложениях.

Популярность метода обусловлена методической наглядностью (основан на замене производных разностными отношениями), обширностью сферы применения (строгая теория дифракции) и обилием готовых программных реализаций (множество коммерческих и свободно распространяемых пакетов для различных архитектур и операционных систем). Платой за универсальность являются высокие системные требования (к быстродействию процессора и объему оперативной памяти), в силу которых разработанный полвека назад метод начал активно применяться в вычислительной практике только сейчас.

Сталкиваясь с известными трудностями ("кремниевый тупик") развитие аппаратной базы вычислительной техники в последнее десятилетие связывается не с наращиванием тактовой частоты процессоров, а со специализированными архитектурными решениями, позволяющими значительно увеличить быстродействие операций определенного типа. Наиболее заметные успехи в этом направлении достигнуты при развитии архитектуры графических процессоров (GPU) и соответствующих программных инструментов (CUDA [5], OpenCL [6]), позволяющих многократно ускорить действия с векторами и матрицами.

Современные реализации FDTD-метода на GPU [7, 8], в основе которых лежит операция вычитания матриц, характеризуются повышением производительности на порядок по сравнению с вычислениями на центральном процессоре. Однако, ограниченный объем видеопамяти (2-6 Гб. по сравнению с 4-32 Гб. оперативной памяти) не позволяет в полной мере использовать преимущество GPU в быстродействии.

Преодоление упомянутого ограничения представляется авторам весьма актуальной задачей. Для ее решения в настоящей работе предлагается применить метод пирамид [9,10,11], основанный на снижении интенсивности коммуникаций между оперативной и

видеопамятью за счет дублирования арифметических операций (в частности, FDTD-метода).

1. Программная реализация FDTD-метода

Рассмотрим распространение плоской волны в трехмерной области свободного пространства, ограниченной поглощающими слоями CPML [1]. Для чего будем рассматривать уравнения Максвелла и разностную схему Yee для них [1], решение сеточных уравнений которой и является средством моделирования.

Зададим сеточную область размером $256\times256\times256$ отсчетов по пространству, ширину поглощающего слоя выберем равной 11 отсчетам. Для размещения такой задачи в памяти видеокарты необходимо 467 Мб.

Применим метод пирамид с одномерной декомпозицией [12] и проведем разбиение исходной сеточной области по одному выбранному направлению. Ниже приведен авторский код на CUDA C, описывающий алгоритм Yee из [1]:

```
__host__ void raschetGPU_pyramid_1d()
{
 ...
 //Число запускаемых блоков
 int  SIZEx  =  ((Imax-1)%BLOCK_DIMx==0)  ?  (Imax-1)/BLOCK_DIMx  :  (Imax-
 1)/BLOCK_DIMx+1;
 int  SIZEy  =  ((Jmax-1)%BLOCK_DIMy==0)  ?  (Jmax-1)/BLOCK_DIMy  :  (Jmax-
 1)/BLOCK_DIMy+1;
 dim3 gridSize = dim3(SIZEx,SIZEy, 1);
 dim3 blockSize = dim3(BLOCK_DIMx,BLOCK_DIMy, BLOCK_DIMz);

 create_temp_fields();//Копирование пересекаю- //щихся областей во временный
 буфер
  int countPyramidsK = ceil((Kmax-1) / (pyramidBaseLengthK + 0.0f));
  // Число пирамид
  int currentTime = 1;//Текущий временной шаг
  int timeOfPass =((nmax)%PASS==0)? nmax/PASS : nmax/PASS+1;
 // Число проходов
 //Цикл по проходам
 for (int pass = 1; pass <= PASS; pass++)
 {
  int currentBaseLengthK; //ширина по верх-//него основания текущей пирамиды
  int leftOffsetK;     //ширина левого перекрытия // нижнего основания пира-
  миды
  int rightOffsetK;    //ширина правого перекры-
  //тия нижнего основания пирамиды
  int fullBaseLengthK;    //ширина нижнего //основания текущей пирамиды
  int leftPyramidBorderK; //индекс левой грани-
  //цы основания пирамиды
  int rightPyramidBorderK;//индекс правой гра-//ницы основания пирамиды
  int durationOfTimePass = timeOfPass * pass; //временной шаг, соответствую-
  щий окончанию //текущего прохода

  //Копируем во временный буфер начальные зна-//чения полей для прохода
  copyFields(...);

  //Цикл по пирамидам
  for(int pyramidIdK = 0; pyramidIdK < countPyramidsK; pyramidIdK++)
  {
   int  startPyramidBasePositionK  =  pyramidIdK  *  pyramidBaseLengthK;
   //начальный индекс //верхнего основания пирамиды в сетке
```

```
    getOffsets(pyramidIdK,      pyramidBaseLengthK,      &currentBaseLengthK,
    &leftOffsetK, &rightOffsetK);
    getBorders(currentBaseLengthK,          leftOffsetK,          rightOffsetK,
    &fullBaseLengthK, &leftPyramidBorderK, &rightPyramidBorderK);

    //Копирование полей на видеокарту
    create_arrays_on_GPU(Imax,Jmax,fullBaseLengthK);
    copy_temp_arrays_to_GPU(0,0,leftPyramidBorderK,Imax,Jmax,fullBaseLengthK)
    ;
    copy_constant_to_device();
    //Цикл по времени внутри прохода
    for (int n = currentTime; n <= durationOfTimePass && n <=nmax; n++)
    {
     //Ядро для пересчета компонент магнитного //поля на GPU
         kernelH_pyramid1<<<gridSize,blockSize>>>(...);
     cudaEventSynchronize(syncEvent);
     //Ядро для пересчета компонент электриче-//ского поля на GPU
     kenelE_pyramid1<<<gridSize,blockSize>>>(...);
     cudaEventSynchronize(syncEvent);
    }

    //Копирование данных из видеопамяти
    copy_arrays_from_GPU_with_offset(currentBaseLengthK,  startPyramidBasePo-
    sitionK, leftOffsetK);
    delete_arrays_on_GPU();
   }
   currentTime = durationOfTimePass +1;
  }
  ...
}
```

Сеточные подобласти, соответствующие нижним основаниям пирамид и содержащие начальные данные для каждого прохода, пересекаются, поэтому перед началом очередного прохода необходимо скопировать эти области во временный буфер (реализуется процедурой create_temp_fields()), чтобы значения не были изменены при копировании из видеопамяти значений, рассчитанных при обработке соседней пирамиды.

2. Постановка вычислительных экспериментов

Вычислительные эксперименты по определению длительности вычислений проводились на видеокарте NVIDIA GeForce GTX 660 Ti (табл. 7) и процессоре Intel Core 2 Duo E8500 (табл. 8). Исследование велось в операционной системе Ubuntu.

Исследуя условия эффективности применения метода пирамид проведем серию вычислительных экспериментов, ограничив объем используемой видеопамяти 245 Мб, что позволит построить до 3-х пирамид с разной шириной основания и высотой. Целью экспериментов будет определение зависимости ускорения вычислений от данных параметров.

На рис. 1 приведены результаты измерений длительности вычислений.

Как показали результаты экспериментов, минимальное время вычислений достигается при соблюдении баланса использования памяти устройства и числа пересылок. Это получается (рис. 1) при использовании пирамид с шириной основания 64 и высотой 20 отсчетов при общем количестве слоев по времени в 256. При дальнейшем увеличении высоты пирамиды время расчета начинает увеличиваться за счет увеличения объема дублирующихся данных, что иллюстрируется U-образной зависимостью времени вычислений от высоты пирамиды.

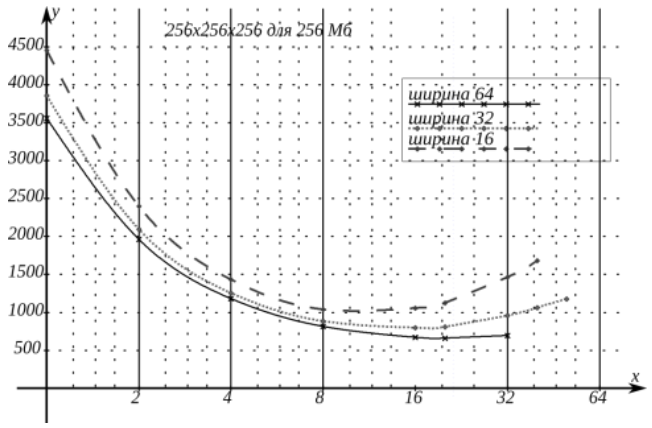

Рис. 1. Зависимость времени расчета y(мс) от высоты x(число отсчетов) пирамиды для различной ширины основания

Заключение

Представленная реализация FDTD с использованием метода пирамид позволяет преодолеть ограничение на объем памяти графического процессора и использовать преимущество GPU в быстродействии за счет снижения интенсивности коммуникаций между оперативной и видеопамятью за счет дублирования арифметических операций.

При реализации вычислений по методу пирамид важно соблюдать баланс между объемом пересылок и задействованным объемом памяти: определяя ширину основания пирамиды при фиксированной высоте, необходимо выбирать максимально допустимую ширину основания, так как это позволяет более оптимально использовать память устройства и сокращает число пересылок. Расчеты, произведенные с использованием пирамид одинаковой высоты, имеют близкие значения ускорения, что также подчеркивает значительность затрат на пересылку данных по сравнению с затратами на непосредственный расчет.

Благодарности

Работа выполнена при поддержке гранта РФФИ 14-07-00291-а.

Литература

1. Taflove A. Computational Electrodynamics: The Finite-Difference Time-Domain Method/ A.Taflove, S. Hagness. - Boston: Arthech House Publishers (Thrid Edition), 2005. - 1006 p.
2. Котляр, В.В. Фотонные струи, сформированные квадратными микроступеньками / В.В. Котляр, С.С. Стафеев, А.Ю. Фельдман // Компьютерная оптика. - 2014. - Т. 38, № 1. - С. 72-80.
3. Тиранов, А. Д. Коллективное спонтанное излучение в волноводе с близким к нулю показателем преломления / А. Д. Тиранов, А. А. Калачёв // Известия РАН, серия физическая. - 2014. - Т. 78, № 3. - С. 271–275.
4. Перов, С. Ю. Теоретическая и экспериментальная дозиметрия в оценке биологического действия электромагнитных полей носимых радиостанций. Сообщение 1. Плоские фантомы / С. Ю. Перов, Е. В. Богачева // Радиационная биология. Радиоэкология. - 2014. - Т. 54, № 1. - С. 57–61.
5. Основы работы с технологией CUDA. / А.В. Боресков, А.А. Харламов. - М.: ДМК Пресс, 2010. - 232 с.
6. OpenCL – The open standard for parallel programming of heterogeneous systems / URL: http://www.khronos.org/opencl/.
7. B-CALM - Belgium California Light Machine / URL: http://b-calm.sourceforge.net/

8. FDTD solver // Acceleware: [сайт]. URL: http://www.acceleware.com/fdtd-solvers.
9. The parallel execution of DO loops / L. Lamport // Communications of the ACM. - 1974. - V. 17 (2). - P. 83-93.
10. Вальковский, В.А. Параллельное выполнение циклов. Метод пирамид / В.А. Вальковский // Кибернетика. - 1983. - №5. - С. 51-55.
11. Головашкин, Д.Л. Решение сеточных уравнений на графических вычислительных устройствах. Метод пирамид / Д.Л. Головашкин, А.В. Кочуров // Современные проблемы прикладной математики и механики: теория, эксперимент и практика [Электронный ресурс] / Международная конференция, посвященная 90-летию со дня рождения академика Н.Н. Яненко, Новосибирск, Россия, 30 мая – 4 июня 2011 г. - Новосибирск, ИВТ СО РАН, 2011, № гос. регистрации – 0321101160, Режим доступа: http://conf.nsc.ru/files/conferences/niknik-90/fulltext/37858/ 46076/kochurov_final.pdf, свободный.
12. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media / K.S. Yee // IEEE Trans. Antennas Propag. - 1966. - AP-14. - P. 302-307.