УДК 519.688

РЕАЛИЗАЦИЯ ЧИСЛЕННОГО РЕШЕНИЯ ТРЕХМЕРНОГО УРАВНЕНИЯ ЛАПЛАСА НА ПЛАТФОРМЕ CUDA

Гризан С.А.,
научный руководитель д-р техн. наук Легалов А. И.
Сибирский Федеральный Университет

В настоящее время наблюдается широкое использование суперкомпьютерных систем, построенных с использованием неоднородных (гетерогенных) архитектур. Основными вычислительными элементами в них являются специализированные графические ускорители (GPU), позволившие обойти кластерные как по пиковой, так и по реально достигнутой производительности. Ситуацию подтверждают последние публикации списков 500 лучших суперкомпьютеров мира, в которых системы на основе графических ускорителей занимают места в первой десятке.

Суть рассматриваемого в данной работе алгоритма заключается в численном решении эллиптического уравнения в постановке задачи Дирихле в некоторой трёхмерной области, сетка на которой задана в виде графа, каждый из узлов которого имеет не более N соседей.

Описывается опыт переноса на графические вычислители и платформу CUDA метода разложения матрицы СЛАУ на суперпозицию трёх вспомогательных матриц, который в дальнейшем упрощает процесс итерационного построения точного решения. Данный метод активно применяется в популярном пакете OpenFOAM, в результате чего он стал известен под сокращённым названием D-ILU.

После составления тестовой программы, решающей трёхмерное уравнение Лапласа методом Якоби на структурированной и неструктурированной сетках, было решено использовать неструктурированную, т.к. при её использовании возможно существенно уменьшить количество узлов на целевой модели, и тем самым сократив время вычислений.

При сравнении в обоих случаях использовалось равномерное разбиение параллелепипеда, однако в первом варианте данные представлялись как трёхмерный массив, а во втором как некоторый граф, каждой вершине которого соответствует значение моделируемой характеристики в заданном узле, а ребру — связь между соседними узлами. В случае неструктурированной сетки проводилось два типа тестов — оценка производительности при использовании «упорядоченного» графа, в котором все соседние вершины некоторого узла по возможности располагаются подряд в используемом для их хранения массиве, и «перемешенного» графа, где все вершины произвольным образом размешены в соответствующем массиве.

Стоит отменить оптимизации, произведенные при переносе логики работы со структурами данных. В случае неструктурированной сетки возможности по использованию разделяемой памяти фактически отсутствуют, так как для получения индекса соседней вершины требуется прочитать значение из вспомогательного массива индексов, что делает невозможным выделение некоторой подобласти. По этой причине пришлось задействовать такие альтернативные механизмы, как текстурная память и L1-кэш глобальной памяти.

В первом случае адресация массива со значениями происходила через специальные текстурные блоки графического ускорителя, кэширующие сразу некоторый участок памяти. Во втором предполагалось автоматическое использование кэша L1, который поддерживается в графических адаптерах, начиная с архитектуры Fermi, и который, по

заявлениям компании NVidia, позволяет достичь лучших результатов, чем при использовании текстурной памяти.

В связи с использованием достаточно удобного формата представления данных, проблемы обеспечения атомарности операций при переносе основной логики отсутствовали. Однако при реализации CUDA версии алгоритма инвертирования одной из вспомогательных матриц возникла необходимость его модификации: в изначальной реализации для этого использовался паттерн вычислений, известный под названием scan и заключающийся в последовательном обходе массива, для обработки i-го элемента которого требуются результаты обработки всех предыдущих элементов с индексами i - 1, i - 2, ..., 1, 0. Очевидно, что данный алгоритм в принципе не подходит для переноса на графический ускоритель, и потому пришлось заменить его итерационным аналогом, который строит приближённое значение искомой обратной. В результате была получена достаточно интересная реализация, в которой присутствуют внешние итерации для построения приближённого решения искомой величины, а также в каждой внешней введены внутренние итерации для инвертирования вспомогательной матрицы.

Подобная реализация требует не только большего количества вычислений, но и нахождения того порога точности при построении вспомогательной матрицы, который обеспечит достаточную скорость сходимости внешних итераций и минимальное суммарное время работы всего решателя. Было проведён ряд тестов, благодаря которым удалось определить, что 3-4 внутренние итерации гарантированно обеспечивают сходимость для всех тестовых данных, и при этом подобная реализация имеет максимальную производительность. Стоит отметить, что при этом в предложенном варианте потребовалось на 30% больше внешних итераций для достижения заданной точности.

Ниже приведены результаты тестирования сходимости ускоренной версии рассмотренного решателя:

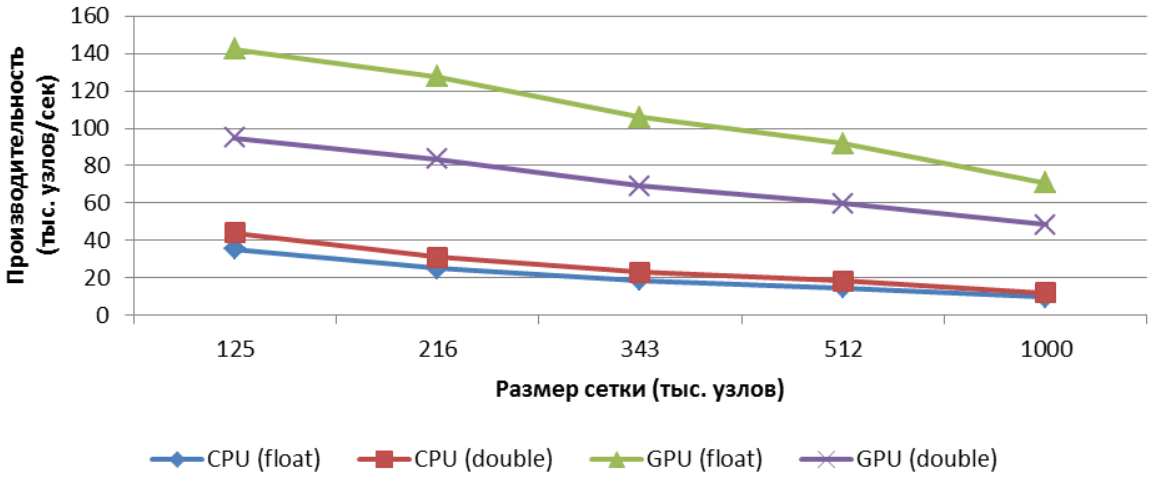

Рисунок 1. Скорость вычислений.

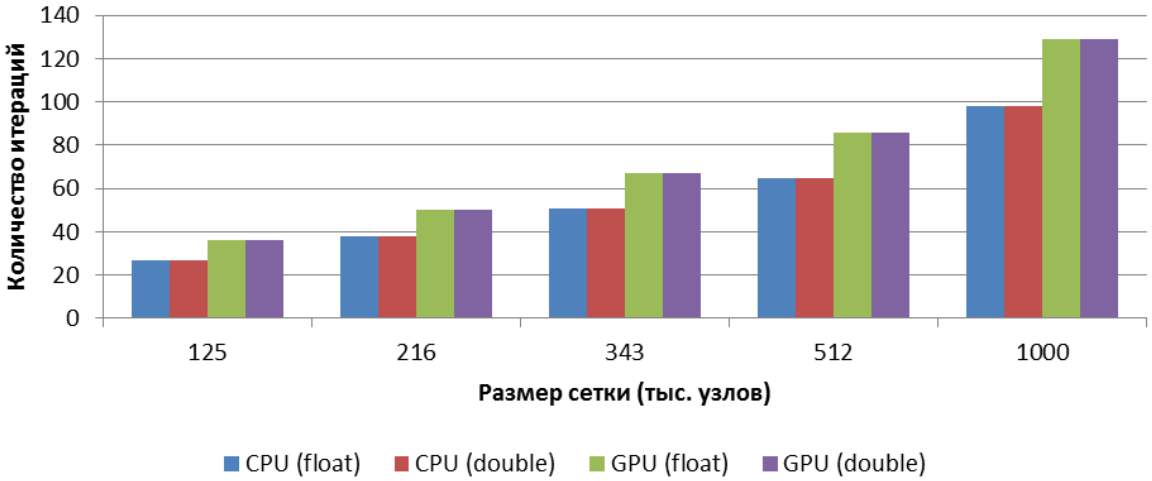

Рисунок 2. Скорость сходимости.

Как видно из графика, скорость работы алгоритма D-ILU с увеличением размера сеток заметно понижается как при проведении вычислений на графическом ускорителе, так и на центральном процессоре. Итоговое ускорение от портирования составило порядка 4-7 раз. Стоит отметить, что после проведения ряда работ по оптимизации пересылок данных между центральным и графическим ускорителем возможно повысить скорость работы GPU-версии более чем на 50% и сделать ускорение 6-12 кратным.