Применение CUDA для распараллеливания трехмерной задачи фильтрации нефти

Т.Т. Бекибаев, Б.К. Асилбеков, У.К. Жапбасбаев, И.К. Бейсембетов, Б.К. Кенжалиев
Казахстанско-Британский технический университет, Алматы, Казахстан
e-mail: assilbekov@mail.ru

Аннотация

В данной работе проведено исследование применимости технологии CUDA для распараллеливания трехмерных расчетов нефтедобычи. Гидродинамические расчеты были проведены на искусственно созданной тестовой модели и модели одного из месторождений Западного Казахстана. Был разработан алгоритм параллельного расчета задачи вытеснения нефти водой с использованием технологии CUDA. Было получено максимальное ускорение расчетов в 60,4 раз на графическом процессоре GeForce GTX 560.

1. Введение. Перед нефтяными компаниями при разработке нефтегазовых месторождений стоит задача эффективной добычи углеводородного сырья. Для бурения новых скважин, выбора схемы добычи нефти и газа должны быть проведены исследования и компьютерные расчеты. Многие нефтяные компании на всех этапах эксплуатации месторождения, в обязательном порядке создают и анализируют модель нефтегазового месторождения с использованием современных программных средств. Золотое правило: можно извлечь нефть только один раз, а моделировать этот процесс можно сколько угодно раз! Таким образом, программные продукты, позволяющие рассчитать различные варианты разработки месторождения с учетом бурения новых скважин, вложения денежных средств на разработку запасы углеводородов и эксплуатации скважин и др. влияющих факторов, играют важную роль в нефтегазовой отрасли. Одним из важных звеньев гидродинамического моделирования являются проведение расчетов по прогнозированию добычи и адаптации к истории за короткое время. Влияние этого фактора усиливается с увеличением количеств расчетных блоков в модели месторождения. Выполнение быстрых расчетов с помощью известных коммерческих симуляторов, таких как Eclipse (Schlumberger), Tempest MORE (Roxar), VIP (Landmark), tNavigator (RF-Dynamics) и др. основано на распараллеливании расчетов на вычислительных ядрах высокопроизводительных кластеров и персональных компьютеров с использованием MPI и OpenMP. В данной работе рассматривается распараллеливание гидродинамических расчетов с использованием вычислительных ядер видеокарты персонального компьютера. Это является одним из основных отличий предлагаемого 3D гидродинамического симулятора от аналогов.

Долгое время графический процессор (GPU) использовался только по своему прямому назначению - для ускорения вычислений, связанных с визуализацией данных. С ростом мощности GPU у разработчиков программных обеспечений появилось желание переложить на них ряд задач, которые до этого момента выполнялись CPU.

Высокая вычислительная мощность GPU объясняется особенностями архитектуры. Если современные CPU содержат несколько ядер (на большинстве современных систем

от 2 до 4х), графический процессор изначально создавался как многоядерная структура, в которой количество ядер измеряется сотнями. Разница в архитектуре обусловливает и разницу в принципах работы. Если архитектура CPU предполагает последовательную обработку информации, то GPU исторически предназначался для обработки компьютерной графики, поэтому рассчитана на массивно параллельные вычисления. Каждая из этих двух архитектур имеет свои достоинства. CPU лучше работает с последовательными задачами. При большом объеме обрабатываемой информации очевидное преимущество имеет GPU [1].

На сегодняшний день разработаны различные библиотеки, позволяющие выполнять программный код на графических процессорах, и подход использование GPU для вычислений общего назначения получил очень широкое распространение в самых разных сферах, начиная от астрофизики и заканчивая расчетом финансовых рисков [2]-[5]. В данной работе исследуется вычислительные возможности GPU применительно задач нефтедобычи.

2. Постановка задачи. В данной работе рассматривается гидродинамическое моделирование режима разработки пласта, когда давление выше давления насыщения, что исключает выделение газовой фазы. В расчетах не учитывается изменение температуры пласта. Тогда течение флюидов в пористой среде описывается следующими уравнениями [5]-[7]:

$$\nabla \cdot [\lambda_o K (\nabla p_o - \gamma_o \nabla z)] = \frac{\partial}{\partial t}\left[\frac{\phi(1-S_w)}{B_o}\right] + q_o, \tag{1}$$

$$\nabla \cdot [\lambda_w K (\nabla p_o - \nabla p_{ow} - \gamma_w \nabla z)] = \frac{\partial}{\partial t}\left[\frac{\phi S_w}{B_w}\right] + q_w, \tag{2}$$

где K – абсолютная проницаемость пласта, λ_o, λ_w – параметры подвижности нефти и воды, ϕ – пористость, S_w – водонасыщенность, p_o – давление нефти, q_o, q_w - притоки нефти и воды к скважине, B_o, B_w - объемные коэффициенты нефти и воды, z - вертикальная координата, γ_o, γ_w удельные весы нефти и воды, t - время, p_{ow} - капиллярное давление. На границах расчетной области для давления и насыщенности ставятся условия непротекания.

3. Численный метод решения. Система уравнений (1)-(2) дискретизируется IMPE S-методом [6]-[8]. Значение давления в каждом расчетном блоке находится с помощью метода последовательной верхней релаксаций. Насыщенность определяется явным образом после вычисления поля давления. Затем пересчитываются капиллярное давление и межблочные проводимости, которые используются на следующем временном шаге.

4. Реализация на GPU с помощью CUDA. Для эффективного использования вычислительных возможностей графических процессоров необходимо было выявить в программе участки кода, требующих наибольших затрат времени, и адаптировать их с помощью технологии CUDA. При численном решении задач нефтедобычи практически всю часть времени затрачивается на работы с трехмерными массивами, так как они имеет огромные размеры. Именно работа с ячейками различных трехмерных массивов было распараллелена на GPU. Ниже на рисунке 1 показана блок-схема алгоритма программы на основе CUDA

CheckSchTime() – процедура, выполняющая проверку работу скважины (расписания и режимы, заданные пользователем). При изменении режима в константной памяти видеокарты происходит обновление данных, касающиеся скважины.

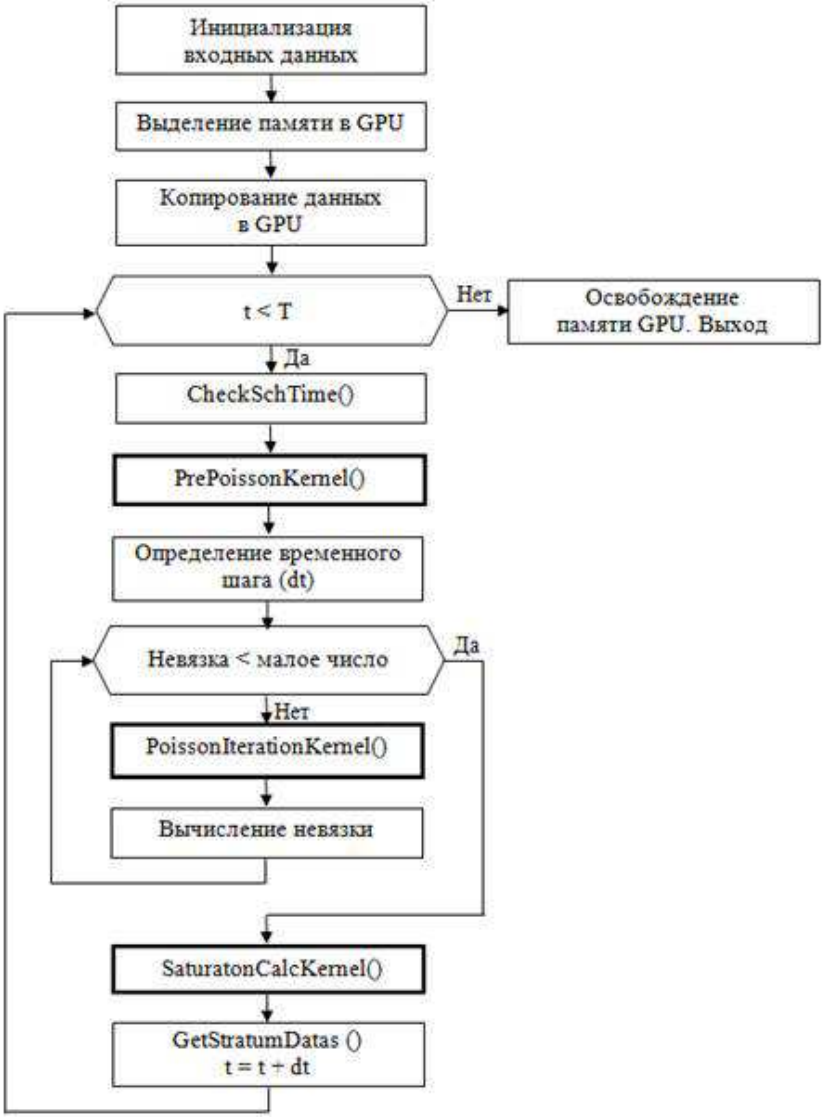

Рис. 1: Блок-схема алгоритма программы на основе CUDA

PrePoissonKernel() – процедура, запускаемая на GPU (ядро CUDA). Рассчитывает коэффициенты, необходимые для решения уравнения давления. Определяет временной шаг расчета.

PoissonIterationKernel() – ядро, выполняющий очередную итерацию Пуассона и пересчитывающее значение давления в каждой ячейки модели. А также процедура определяет невязки для каждого блока

SaturatonCalcKernel() – ядро, рассчитывающее в каждой ячейки значения водонасыщенности используя новые значения давления. А также определяются необходимые показатели скважины (дебит, забойное давление, обводненность и т.д.).

GetStratumDatas() – процедура, извлекающая поля распределений насыщенности, давления, температуры в пласте из DRAM видеокарты. Это занимает значительное время. Выполняется только на определенных временных шагах.

Кроме того, в *PrePoissonKernel()* и *PoissonIterationKernel()* происходит частичная параллельная редукция, где вычисляются необходимые величины для определения временного шага и невязки для каждого отдельного блока потоков. Окончательная редукция относительно блоков осуществляется последовательно на CPU

Одним из основных отличий между GPU и CPU является организация памяти и работа с ней. Так как архитектура CUDA имеет сложную структуру памяти, для того чтобы добиться наибольшего ускорения, необходимо четко распределить данные по видам памяти. Данные, связанные со значениями в узлах сетки, занимают большой объем и поэтому хранятся в виде трехмерных массивов в глобальной памяти (global memory). Каждая ячейка массива (i,j,k) хранит значение для соответствующей расчетной ячейки гидродинамической модели пласта. Эти данные можно разделить на три категории:

1. основные данные – давление, водонасыщенность, температура. Изменяются при каждой итерации по времени;

2. коэффициенты для уравнения Пуассона. Вычисляются в процедуре PrePoissonKernel() и используются для определения значения давления в каждой итерации PoissonIterationKernel();

3. неизменяемые данные – глубины, проводимости расчетных ячеек.

PVT таблица, таблица функции насыщенности занимают малый объем и хранятся в константной памяти (constant memory) для быстрого доступа. Эти таблицы многократно используются для определения значения давления и насыщенности для каждой расчетной ячейки пласта.

Данные по скважинам включают в себя значения дебит флюидов и жидкости, обводненностей, забойных давлений для каждой скважины. Хранятся в памяти хоста с блокировкой страниц (page-locked host memory) для чтения из CPU процедуры и записи в кернеле.

Данные по управлению скважинами включают в себя тип и значения режимов, а также данные по перфорации скважины. Хранятся в константной памяти. При изменении режима скважины, значения данных изменяются из хоста.

Разделяемая память (shared memory) и регистры используется для хранения различных промежуточных значений величин в кернеле. А также блоки потоков загружают часть данных из глобальной памяти в разделяемую, для более быстрого дальнейшего использования в текущем ядре.

Одним из важных аспектов построения алгоритма был выбор варианта разделения расчетной области между потоками. Для этого нужно было определить размерность (2D или 3D) и размеры блока потоков. Стоял выбор между трехмерным и двумерным разделением сетки (рисунок 2). При трехмерном разделении один поток обрабатывает одну ячейку модели, при двухмерном – столбец ячеек вдоль оси Z. Индексы ячеек (ijk) вычисляются относительно номера блоков и номера потока.

Обращение к данным при расчете значений каждой ячейки необходимы значения в соседних ячейках (рисунок 3а). Например, итерационная формула уравнения Пуассона выглядит так:

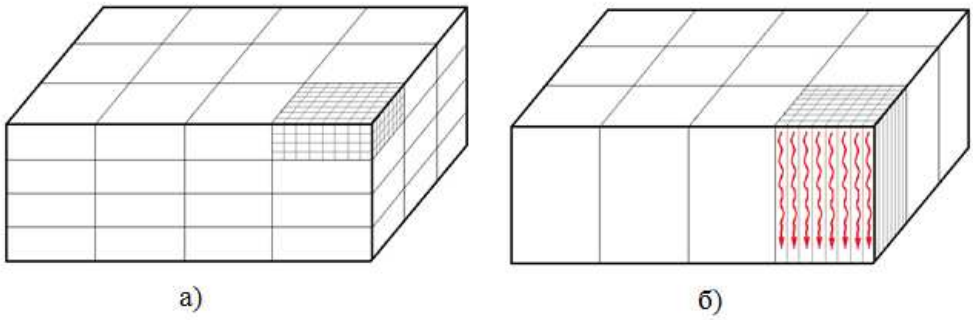

Рис. 2: Разбиение расчетных ячеек на блоки потоков в 3D (а) и 2D (б) случае

$$p_{ijk}^{l+1} = Ap_{i+1jk}^{l} + Bp_{i-1jk}^{l+1} + Cp_{ij+1k}^{l} + Dp_{ij-1k}^{l-1} + Ep_{ijk+1}^{l} + Fp_{ijk-1}^{l+1} + RHS$$

Из-за особенности обращения к данным, для расчета в двумерной области размером $NX \times NY$ необходимо дополнительно загрузить боковые данный, т.е. всего *NX*NY+2*NX +2*NY* значений (рисунок 3б), для трехмерной области размером $NX \times NY \times NZ$ необходимо *NX*NY*NZ+2* NX*NY+2*NX*NZ+2*NY*NZ* значений (рисунок 3в).

Очевидное на первый взгляд трехмерное разделение (рисунок 2а) оказалось менее эффективным. Такое разделение не позволяет содержать в блоке большое количество расчетных ячеек (ограничение по количеству потоков), что ведет к малому размеру 3D блока потоков и, как следствие, к большей долей "боковых" загрузок из глобальной памяти. Например, при блоке потоков размером 8×8×8 "боковые" загрузки занимают $\frac{6*8*8}{8*8*8} = 66{,}66\%$. В то время при двумерном блоке потоков (рисунок 3б) размером 16×16 такие загрузки занимают $\frac{4*16}{16*16} = 25\%$, а в случае 32×16 - $\frac{2*32+2*16}{32*16} = 18{,}75\%$. Лишние "боковые" загрузки – дорогостоящие обращения к глобальной памяти и ветвление в коде, при котором одни потоки загружают "боковые" части, а другие простаивают. Кроме того, такое двумерное разбиение на блоки потоков дает возможность эффективно использовать shared memory. Поэтому использование двумерное деление предпочтительней.

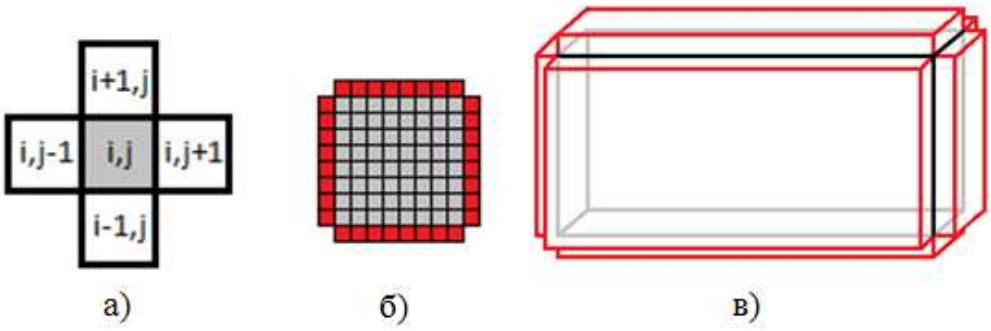

Рис. 3: Дополнительные расчетные ячейки (красные) для блока потоков

Если сравнить размеры блоков потоков 16×16, 32×16 и 32×8, то следует отметить, что при выделении глобальной памяти необходимо выравнивать массивы, чтобы выполнялось условие объединения запросов к памяти (coalescing) при обращении полуварпов к памяти. Если запускать блок потоков размером 32×16 или 32×8, то необходимо больше отступа для выравнивания, чем при 16×16. А также обладая меньшими размерами, блок потоков 16×16 лучше масштабируется под различные размеры расчетной сетки вдоль оси X (либо Y). Однако, блок потоков 32×16 или 32×8 лучше подходит для видеокарт с compute capability 2.x (вычислительной возможностью), так как в этой архитектуре запросы к глобальной памяти осуществляется целым варпом размером 32 потока. Тогда запросы такого варпа будут происходить за одну транзакцию, а при 16×16 за две, то есть доступ к данным в DRAM будет выполняться быстрее.

Однако, также следует учитывать занятость потокового мультипроцессора. В данной работе экспериментальным путем было установлено, что при вычислительной возможности 2.x видеокарты программа дает наибольшее ускорение, когда из 48 варпов, которые могут запускаться на одном мультипроцессоре, активны лишь 32 варпа. Другими словами, оптимальная занятость мультипроцессора равняется 67% при различных размерах блока потоков, т.е в данном случае максимальная занятость не является эффективным. Это объясняется лучшим скрытием латентности обращения к глобальной памяти и большим количеством регистров на поток для промежуточных вычислений по сравнению с максимальной занятостью. Ниже на рисунке 4 приведено сравнение ускорений программы на GPU при разной занятости мультипроцессора и при различных размерах блоков.

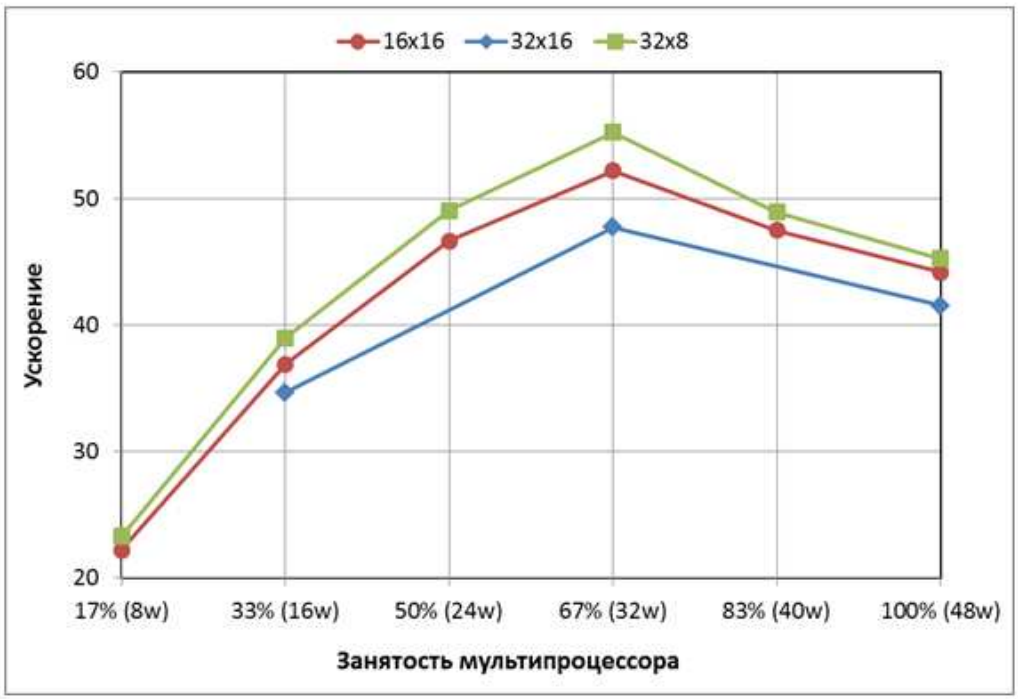

Рис. 4: Ускорение расчетов при различной занятости мультипроцессора и блоков потоков

5. Проведение численных экспериментов. Результаты и их анализы. Расчеты проводились на тестовой модели месторождения (Случай 1) с целью выявления влияния общего количества ячеек в модели, а также соотношения количеств ячеек NX, NY и NZ по осям X, Y и Z соответственно, при постоянном общем количестве, на ускорение. А также проведен расчет на примере месторождения Молдабек Восточный (Случай 2). Ниже приведены описания этих моделей.

Для расчетов использовался центральный процессор Intel Core i7 2600 с тактовой частотой 3.40 ГГц и частотой DRAM - 667МГц, и графическое устройство GeForce GTX 560, которая имеет следующие характеристики: тактовая частота 1620 МГц, частота памяти - 2004 МГц, вычислительная возможность - 2.1, число мультипроцессоров 7 (336 ядер).

Случай 1. Пласт считается однородным. Проницаемость по X, Y и Z – 2900 мД, пористость – 33%, начальное пластовое давление – 100 бар, водонасыщенность - 20 %. Неоднородность характеристик пласта по пространству не влияет на производительность программы. Модель содержит две скважины - нагнетательная и добывающая, которые расположены на противоположных углах (рисунок 5). Ниже на рисунке 6 показана зависимость ускорения от "соотношения размеров сетки" модели, т.е. соотношения количеств расчетных ячеек по осям X, Y и Z при постоянном общем количестве. Использовался блок потоков 16×16. В данном случае во всех экспериментах количество ячеек равно 921600. Величина NZ меняется от 1 до 3600 и $NX*NY*NZ$=921600.

На ускорение расчета влияют два фактора.

Первое, это эффективность использования всех мультипроцессоров (суммарная загруженность мультипроцессоров блоками). В рассматриваемом GPU имеются 7 потоковых мультипроцессоров (streaming multiprocessors или SM) и каждый может обрабатывать по 4 блока потоков, т.е за один некий временной цикл в GPU рассчитываются 28 блоков. Поэтому, важно, чтобы в разбиении модели на блоки потоков 16×16 их количество делилось на 28. Например, модель размером NX=96, NY=96, разбивается на 6*6=36 блоков потоков, тогда на первом шаге работают все 7 мультипроцессора, рассчитывая 28 блоков, на втором шаге оставшиеся 8 блоков будут выполнять только 2 SM, а остальные 5 будут простаивать. Поэтому эффективность использования всех SM будет $\frac{36}{2*28} = 64\%$. Очевидно,чем выше этот коэффициент, чем выше ускорение, полученное на GPU.

Если количество блоков потоков намного меньше 28, то только немногие SM будут использованы на первом шаге, т.е. конвейер из 7 SM будет работать не на полную эффективность. И, как следствие, ускорение будет значительно снижаться при малых значениях NX и NY. Стоит отметить, что делимость числа блоков потоков на 28 играет важную роль только при малом NX, NY. При больших значениях NX, NY (т.е. при большом количестве блоков потоков) данное свойство не существенно влияет на ускорение, так как будет множество шагов обработки 28 блоков, а незанятыми SM могут остаться только на последнем шаге, и в целом эффективность использования всех мультипроцессоров будет стремиться к 100%.

Второй фактор, влияющий на ускорение, связан с величиной NZ. Дело в том, что для расчета значения в ячейках модели на слое по Z необходимы ее значения ячеек на слоях $(K-1)$ и $(K+1)$, для этого вводятся боковые слои сверху и снизу. При переходе на расчет следующего слоя, слой $(K+1)$ становиться K, а K-ый – $(K-1)$,

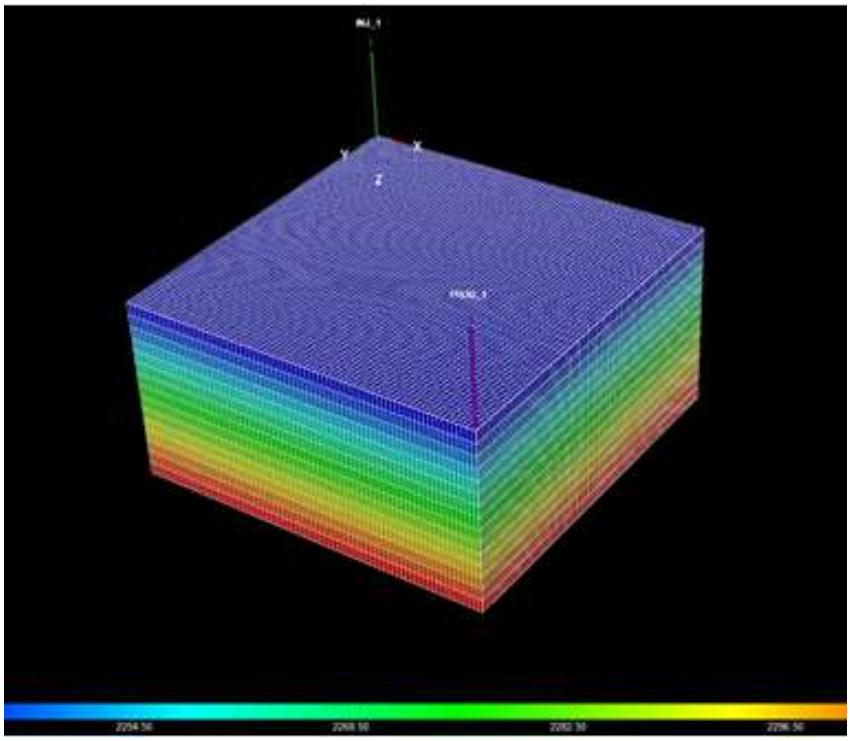

Рис. 5: Тестовая модель для случая 1

Рис. 6: Зависимость ускорения от соотношения размеров сетки

сохраняя их в разделяемой памяти. Т.е. нет необходимости загружать каждый раз по три слоя, достаточно загрузить только $(K+1)$ слой, а остальные уже сохранены с прошлых итераций по . В сумме загружаются $NZ+2$ слоев с глобальной памяти. Известно что, транзакции с DRAM медленно выполняются на графических процессорах. Поэтому критично количество обращений к DRAM. При малых значениях NZ, два "лишних" обращения существенно по сравнению с самим NZ. Например, для расчета $NZ=1$, требуется загрузка 3 слоев, для $NZ=2$ – 4 слоя, а для $NZ=100$ – 102 слоя и соотношения рассчитанных слоев к загруженным будет 33%, 50% и 98% соответственно. Очевидно, чем выше этот показатель, тем выше будет ускорение. Это и объясняет меньшее ускорение моделей с малым NZ.

Итак, при одинаковом количестве ячеек сетки, модели со сбалансированными размерами NX, NY, NZ имеют самое больше ускорение. Ускорение значительно снижается, когда величины NX, NY слишком малы.

На рисунке 7 показана зависимость ускорения от общего количества расчетных ячеек в гидродинамической модели при соотношении NX=NY=$\frac{16}{3}$NZ. Общее число расчетных блоков менялся от 96000 до 6144000. Использовался блок потоков 16×16.

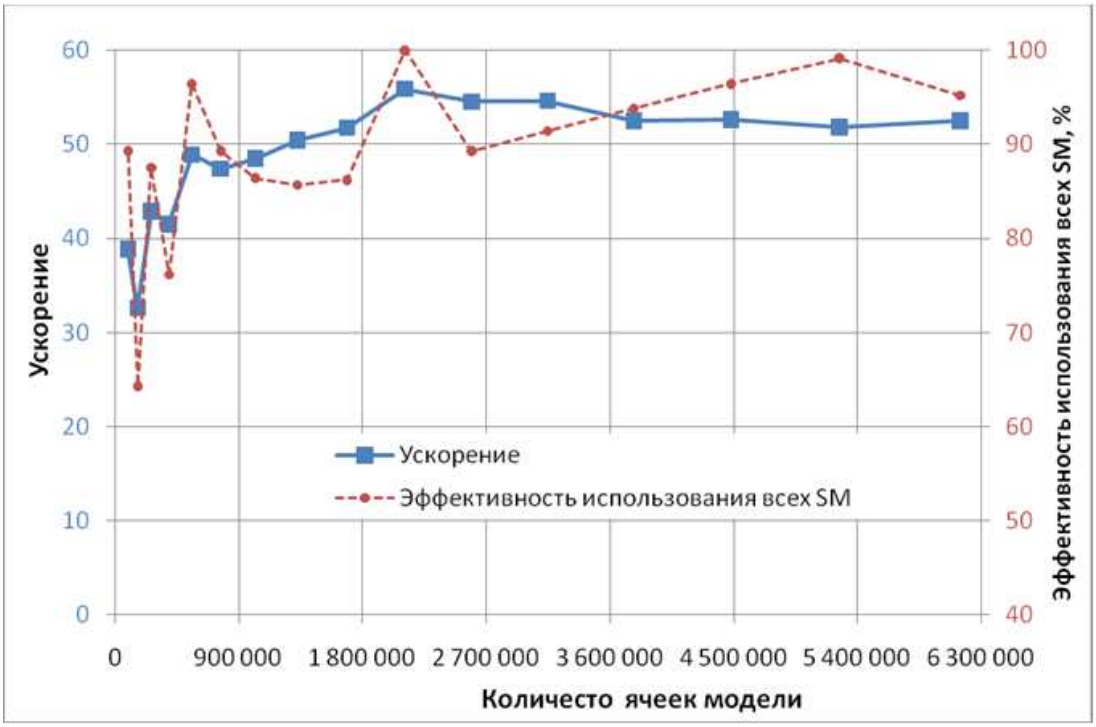

Рис. 7: Зависимость ускорения от общего количества расчетных ячеек

Из графика видно, что с увеличением общего числа расчетных ячеек ускорение растет и после двух миллионов стабилизируется на уровне 52-55 раз. Небольшая пилообразность графика, особенно в начале, объясняется различной суммарной загруженностью мультипроцессоров блоками (о чем говорилось ранее). Меньшее ускорение при малом количестве ячеек объясняется меньшей долей параллельной части программы

(расчет различных значений в узлах трёхмерной сетки модели) по сравнению с последовательной (пересчет режимов и проверка ограничений скважин, запуск кернелов, вывод результатов и т.д.).

Максимальное ускорение в 60,4 раз было получено с использованием блоков потоков 32×8 на моделях размером 2 млн. и более ячеек.

Необходимо упомянуть о тех факторах, которые отрицательно сказываются на ускорении при текущей реализации параллельной программы:

Деление с остатком NX и NY на размеры блока потоков (16×16, 32×16, 32×8). При невыполнении этого условия NX и NY будет округляться в большую сторону, и тем самым будут появляться "полупустые" блоки, в которых некоторые потоки не будут задействованы в расчете. Например, при $NX \times NY = 48 \times 48$ будет 9 блоков, при 48×49 будет уже 12 (так же как и в случае 48×64). Однако, при больших значениях NX, NY этот фактор становится не столь важным.

Частая смена режимов работы скважин. При смене режима необходимо обновить данные по скважине в памяти GPU. Поэтому частая смена режимов работы увеличивает объем передачи данных с ОЗУ на GPU память.

Наличие неактивных ячеек. При работе с неактивной ячейкой поток простаивает. Варп с простаивающими потоками выполняется за такое же время, как и полностью загруженный варп. В то время как в последовательной реализации, неактивные ячейки уменьшают количество итерации при пробеге по узлам трехмерной сетки, что уменьшает время последовательного расчета.

Случай 2. В этом случае рассматривается модель месторождения Молдабек Восточный (рисунок 8). Модель содержит 91476 (36×77×33) расчетных ячеек. Зависимости объемного коэффициента B_w,вязкости воды μ_w и пористости ϕ от давления задаются в виде:

$$B_w = 1,02[1 + c_w(p_w - 20) + 0,5c_w^2(p_w - 20)^2]^{-1},$$
$$B_w\mu_w = 1,029[1 + c_\mu(p_w - 20) + 0,5c_\mu^2(p_w - 20)^2]^{-1},$$
$$\phi = \phi_0[1 + c_\phi(p_0 - 39) + 0,5c_\phi^2(p_0 - 39)^2],$$

где $c_w = 3,25 \times 10^{-6}$ бар$^{-1}$, $c_\mu = 2,39 \times 10^{-6}$ бар$^{-1}$,$c_\phi = 5 \times 10^{-5}$ бар$^{-1}$.

Плотности нефти и воды в стандартных условиях равны 889,5 и 1000,1 кг/м^3, соответственно. Проницаемость по оси x изменяется в диапазоне от 514,49 мД до 4134,70 мД, по оси y – от 503,2 мД до 4240,1 мД и по z – от 53,0 мД до 420,9 мД. Пористость изменяется в диапазоне 17-40%. Распределение начальной водонасыщенности показано на рисунке 9. Зависимости фазовых проницаемостей и капиллярного давления от насыщенности воды приведены на рисунке 10, а на рисунке 11 представлены зависимости объемного коэффициента B_0 и вязкости μ_0 нефти от давления.

Проведённый расчет состоял из 100 итерации по времени, что суммарно составило 450 дней. Время работы параллельной версии программы составило 5.3 секунд, и дало ускорение в 33 раза по сравнению последовательной версии программы. Результаты расчетов двух версии были полностью идентичны. Меньшее ускорение по сравнению с максимальным добившимся ускорением в 60,4 раз, объясняется малым размером сеточной модели месторождения Молдабек Восточный и как следствие, неполной нагрузкой всех потоковых мультипроцессоров GPU.

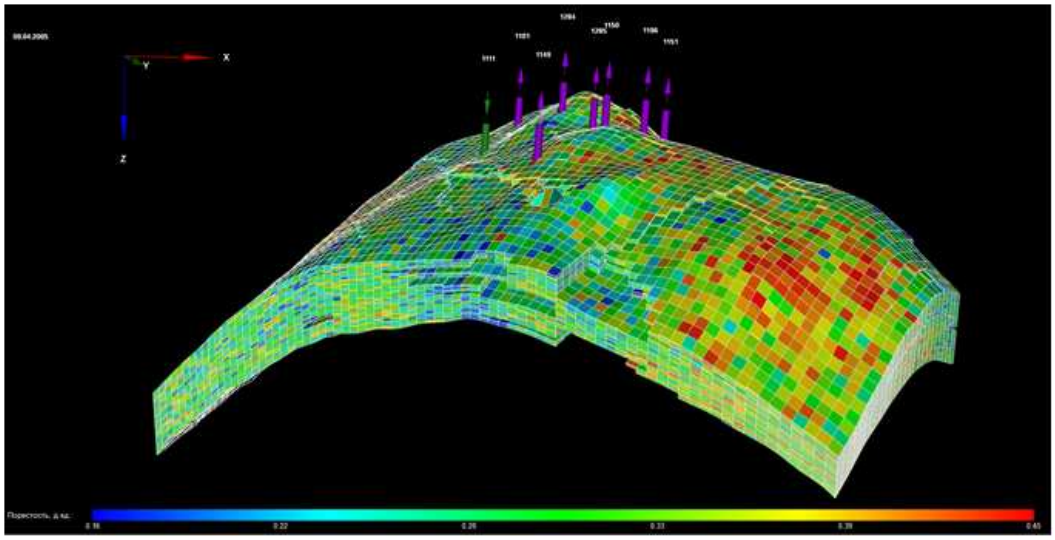

Рис. 8: Распределение пористости месторождения Молдабек Восточный

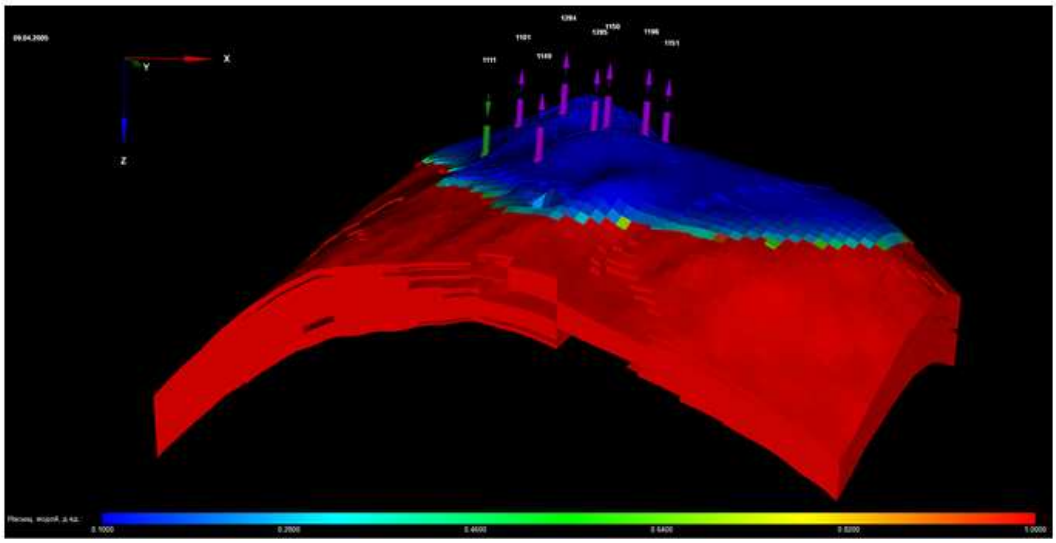

Рис. 9: Распределение начальной водонасыщенности

Заключение

В данной работе осуществлён перенос и оптимизация программы гидродинамического моделирования с последовательной версии на GPU с помощью технологии CUDA. Для выполнения параллельных расчетов использовалась видеокарта GeForce GTX 560, последовательные расчеты проводились на центральном процессоре Intel Core i7-2600.

Процесс переноса состоял из выделения частей алгоритма, которые могут выполняться с максимальной степенью параллелизма, и перенос этих участков на GPU. Для этих

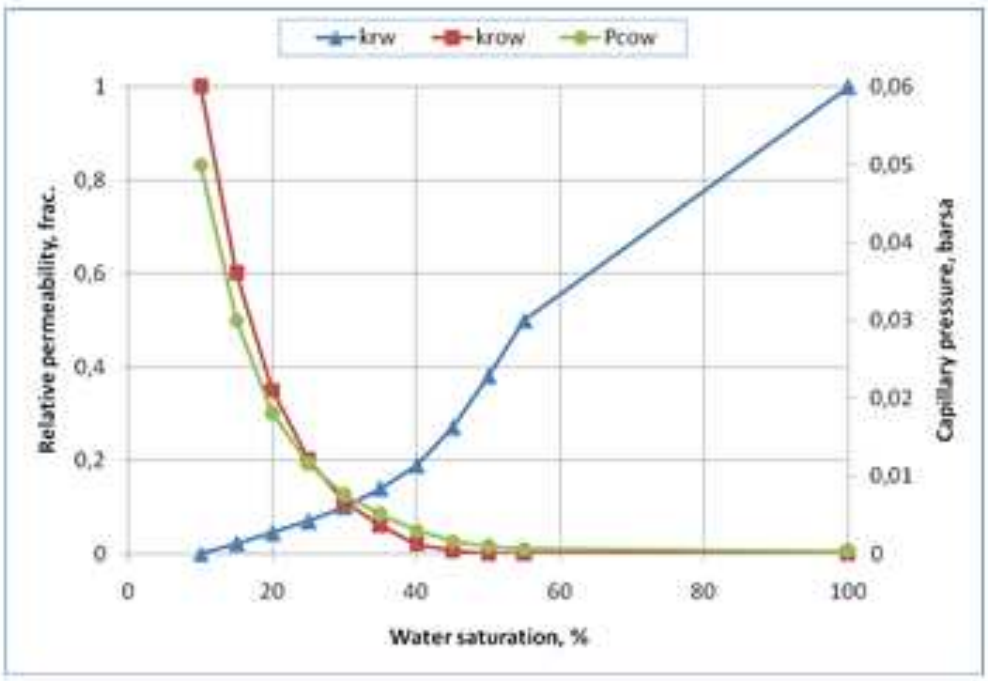

Рис. 10: Относительные фазовые проницаемости и капиллярное давление для слаучая 2

целей выбраны те участки кода, которые связаны вычислениями значений во всех узлах трехмерной сетки модели, так как в последовательной программе пробег по этим узлам занимает основное время, особенно при больших размерах модели. Различные входные данные расчета были поделены на различные классы памяти архитектуры CUDA, в зависимости от их объема, частоты и необходимости использования. Вся модель была поделена на регионы блоков потоков, каждый поток, которого рассчитывал ячейки с индексами с (i,j,1) до (i,j,NZ). Было выявлено, что эффективный по времени размер блоков потоков – 32×8. Однако блок 16×16 лучше масштабируется под различные размеры сетки и лишь немного уступает по производительности. Поэтому для дальнейших расчетов использовался блок 16×16.

Была проведены две серии расчетов на тестовой модели: с различными соотношениями NX, NY, NZ относительно друг друга и с разным числом ячеек, а также проведен расчет на модели реального месторождения Молдабек Восточный. Анализируя результаты расчетов, выявив основные факторы, влияющие на производительность, можно прогнозировать ускорение модели по ее размерности (рисунки 6, 7). Расчеты показали, что наибольшее ускорение на графических процессорах достигается на больших моделях размером сеток от 2 млн. ячеек. Отрицательно влияет на ускорение частая смена режимов работы скважин и наличие неактивных ячеек. Существенно ухудшает ускорение неполная загрузка мультипроцессоров GPU, что происходит при расчете моделей малой размерности NX, NY. Максимальное полученное ускорение составило 60,4 раз с использованием GeForce GTX 560. Справедливо ожидать, что на более производитель-

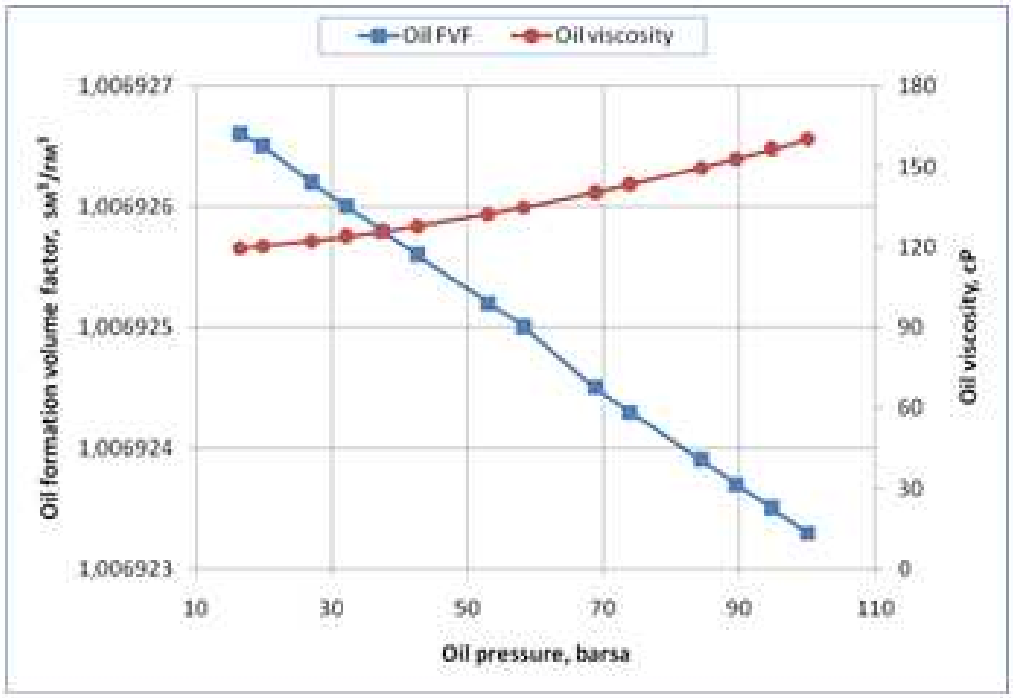

Рис. 11: Объемный коэффициент и вязкость нефти для случая 2

ных графических процессорах серии Tesla, имеющих больше вычислительных ядер и большую пропускную способность памяти, будет достигнуто большее ускорение.

Итак, алгоритм IMPES метода отлично подходить под архитектуру CUDA. Было получено значительное ускорение расчетов фильтрации нефти – от 25 до 60,4 раз. Использование GPU в качестве сопроцессора в гидродинамических симуляторах может составить конкурентоспособную альтернативу применению высокопроизводительных кластеров на базе CPU.

Список литературы

[1] *http://www.ot.ru/press20110215.html*

[2] *Buatois L., Caumon G., Levy B.* Concurrent number cruncher: a GPU implementation of a general sparse linear solver // International Journal of Parallel, Emergent and Distributed Systems. Vol. 24, Issue 3, June 2009. pp 205-223.

[3] *http://www.seismiccity.com/Technologies.html*

[4] *http://www.hanweckassoc.com/*

[5] *http://www.ks.uiuc.edu/Research/namd/*

[6] *Aziz K., Settari A.:* Petroleum Reservoir Simulation. New York: Elsevier, 1979. 406 p.

[7] *Chen Z., Huan G., Ma Yu.:* Computational Methods for Multiphase Flows in Porous Media. // Philadelphia: SIAM, 2006. 549 p.

[8] *Fanchi J.R.:* Principles of Applied Reservoir Simulation. // Second Edition. Houston: Gulf professional publishing, 2001.357 p.

T.T. Bekibaev, B.K. Assilbekov, U.K. Zhapbasbayev, I.K. Beisembetov, B.K. Kenzhaliev, *Application of CUDA for parallelization three-dimensional problem of oil recovery*, The Bulletin of KazNU, ser. math., mech., inf. 2012, №1(72), 65 – 78

In this work research of applicability of the CUDA technology for a parallelization of three-dimensional calculations of oil production is carried out. Hydrodynamic calculations were carried out on artificially created test model and model of one of fields of the Western Kazakhstan. The algorithm of parallel calculation of the water flooding problem using CUDA technology was developed. The maximum acceleration of calculations in 60,4 times on the GPU GeForce GTX 560 was achieved.

Т.Т. Бекибаев, Б.К. Асилбеков, У.К. Жапбасбаев, И.К. Бейсембетов, Б.К. Кенжалиев, *Үш өлшемді мұнайды сүзгілеу есебін параллельдеу үшін CUDA технологиясын пайдалану*, ҚазҰУ хабаршысы, мат., мех., инф. сериясы 2012, №1(72), 65 – 78

Бұл жұмыста CUDA технологиясын үшөлшемді мұнайды өндіру есебін параллельдеуде қолданылу мүмкіндігі зерттелген. Гидродинамикалық есептеулер мұнай кен орнының тестілік және Батыс Қазақстанның бір кен орнының нобайының мәліметтері бойынша жүргізілген. CUDA технологиясын пайдалана отырып мұнайды сумен ығыстыру параллельдік есебінің алгоритмі жасалды. GeForce GTX 560 графикалық процессорын пайдалана отырып, қойылған есептің орындалу уақыты 60,4 есеге қысқартылды.