Параллельный предобуславливатель на основе степенного разложения обратной матрицы для решения разреженных линейных систем на графических процессорах*

Н.В. Репин, А.В. Юлдашев

Уфимский государственный авиационный технический университет

Рассмотрена применимость метода AIPS, аппроксимирующего обратную матрицу на основе степенного разложения в ряд Неймана, в рамках двухступенчатого предобуславливателя CPR. Предложен ориентированный на архитектуру CUDA параллельный алгоритм решения линейных систем с трехдиагональной матрицей, состоящей из независимых блоков различного размера. Показано, что реализация предложенного алгоритма может более чем в 2 раза превосходить по быстродействию функции решения трехдиагональных систем из библиотеки cuSPARSE. Проведено тестирование метода BiCGStab с предобуславливателем CPR-AIPS на современных GPU, показавшее приемлемую масштабируемость данного предобуславливателя, а также возможность ускорить решение линейных систем, характерных для задачи гидродинамического моделирования нефтегазовых месторождений, относительно CPR-AMG.

Ключевые слова: графические процессоры, итерационные методы, параллельные вычисления, предобуславливатели, разреженные матрицы, трехдиагональные системы.

1. Введение

Одним из актуальных трендов в суперкомпьютерной индустрии является применение наряду с центральными процессорами (CPU) массивно-параллельных процессорных устройств, используемых для ускорения вычислений. К примеру, в 28 редакции списка Топ50 мощнейших компьютерных систем России и стран СНГ [1] от 03.04.2018 г. 26 суперкомпьютеров (52 %) оснащены ускорителями, причем 18 из них графическими процессорами (GPU) NVIDIA Tesla.

В последнее время нами ведется разработка и реализация параллельных алгоритмов, ориентированных на архитектуру GPU, решения разреженных систем линейных алгебраических уравнений (СЛАУ), возникающих при дискретизации дифференциальных уравнений в частных производных, к примеру, в задаче моделирования многофазных фильтрационных потоков углеводородов в пористых средах [2]. Сначала [3] был реализован решатель, позволяющий выполнять как на CPU, так и на GPU решение разреженных СЛАУ итерационным методом бисопряженных градиентов со стабилизацией (BiCGStab) [4] с блочной модификацией неполного LU-разложения [5] (BILU(0)) в качестве предобуславливателя. Далее [6] была разработана кластерная версия решателя, позволяющая проводить расчеты на множестве GPU, распределенных по узлам вычислительного кластера. В ней, наряду с BILU(0), был реализован двухступенчатый предобуславливатель Constrained Pressure Residual (CPR) [7], а точнее его популярная версия, часто обозначаемая в литературе аббревиатурой CPR-AMG, в которой на первой ступени (локальный предобуславливатель) используется алгебраический многосеточный метод (AMG) [8], а на второй (глобальный предобуславливатель), как правило, неполное LU-разложение.

В данной работе рассматривается предобуславливатель CPR-AIPS, в котором в качестве альтернативы AMG на первой ступени используется метод AIPS, аппроксимирующий обратную матрицу на основе степенного разложения в ряд Неймана. Основной акцент делается на разработке ориентированного на архитектуру GPU параллельного алгоритма решения СЛАУ с трехдиагональной матрицей специального вида (состоящей из множества независимых блоков различного размера), эффективная программная реализация которого и метода AIPS в целом обеспечивает высокое быстродействие предобуславливателя CPR-AIPS.

* Работа выполнена при частичной финансовой поддержке Министерства образования и науки РФ, государственное задание № 1.3103.2017/4.6.

2. Теоретическая часть

2.1 Метод аппроксимации обратной матрицы на основе степенного разложения

Большой потенциал распараллеливания имеют предобуславливатели на основе аппроксимации обратной матрицы [9], в том числе путем разложения в ряд Неймана [10]:

$$(E - B)^{-1} = E + B + B^2 + B^3 + \cdots,$$

где E – единичная матрица. Соответственно обратная матрица $A^{-1} = (E - B)^{-1}$ может быть аппроксимирована некоторым количеством членов ряда Неймана.

Наряду с приведенным представлением матрицы A рассматриваются и другие (см., например, [10] и ссылки в работе), в том числе в виде разности матриц общего вида: $A = P - R$. Если же представить A в виде

$$A = P + R = P(E + P^{-1}R) = P(E - (-1)(P^{-1}R)),$$

то с использованием разложения в ряд Неймана можно получить следующую формулу для построения предобуславливателя, аппроксимирующего обратную матрицу при заданном N:

$$A^{-1} \approx M_N^{-1} = \left[E + \sum_{k=1}^{N}(-1)^k (P^{-1}R)^k\right]P^{-1}. \quad (1)$$

Малое значение параметра N способствует более быстрому вычислению M_N^{-1}, однако, может не обеспечить необходимую скорость сходимости итерационного процесса решения СЛАУ ввиду низкой точности аппроксимации обратной матрицы. При увеличении N предобуславливатель становится более эффективным в плане обеспечения скорости сходимости, но высокая трудоемкость его построения может привести к замедлению решения СЛАУ в целом. Хотя в экспериментальной части данной работы N было положено равным 10, выработка подхода к выбору оптимального значения данного параметра является предметом дальнейшего исследования.

Что касается структуры матрицы P, в [5] рассматривается использование диагональной либо блочно-диагональной части матрицы A, но существуют и альтернативные варианты. Например, в [11] обратные матрицы аппроксимируются с помощью разложения в ряд Неймана для решения в режиме реального времени СЛАУ малой размерности и в качестве P предложено использование трехдиагональной части A.

В работе [12] формула (1) используется для построения предобуславливателя LSPS, применяемого в задаче гидродинамического моделирования нефтегазовых месторождений. В предобуславливателе LSPS матрица P конструируется так, чтобы она включала существенные элементы матрицы A и оставалась легко обратимой. Наряду с более общим подходом к выбору P метод LSPS предусматривает нестандартное упорядочивание неизвестных в решаемой СЛАУ исходя из физических свойств модели нефтегазового месторождения, что по всей видимости приводит к более высокой скорости сходимости итерационного процесса. Авторы метода LSPS предлагают использовать его как глобальный предобуславливатель, а также в качестве первой ступени предобуславливателя CPR и демонстрируют идеальную масштабируемость итерационного метода решения СЛАУ с соответствующими предобуславливателями на 240 процессорах вычислительного кластера.

Вдохновившись превосходной параллельной эффективностью метода LSPS мы реализовали предобуславливатель, называемый здесь и далее методом аппроксимации обратной матрицы на основе степенного разложения (Approximation of Inverse by Power Series, AIPS), который строится по формуле (1). В качестве P в данной работе берется трехдиагональная часть матрицы A.

Отметим, что организовать применение степенного разложения обратной матрицы в качестве предобуславливателя в рамках итерационных методов решения СЛАУ можно несколькими способами, например, явно и неявно.

1. Построить матрицы P и R, вычислить P^{-1} и явно получить M_N^{-1}. При этом применение предобуславливателя в итерационном процессе сведется к одной операции матрично-векторного умножения.

2. Перед началом итерационного процесса ограничиться построением матриц P^{-1} и R, далее на этапе применения предобуславливателя на каждой итерации вычислять произведение M_N^{-1} на вектор путем выполнения последовательности матрично-векторных операций в соответствии с (1).

К минусам первого способа относятся трудоемкость процедуры нахождения обратной матрицы (P^{-1}), а также рост потребления памяти в связи с увеличением заполненности матрицы в ходе вычисления M_N^{-1}. Поэтому, в работе реализован только второй (неявный) способ.

2.2 Применение метода аппроксимации обратной матрицы на основе степенного разложения в рамках двухступенчатого предобуславливателя CPR

Одним из наиболее эффективных предобуславливателей при решении СЛАУ, возникающих в рамках задачи гидродинамического моделирования нефтегазовых месторождений в результате дискретизации уравнений многофазной фильтрации по времени с использованием полностью неявной разностной схемы, считается специализированный двухступенчатый предобуславливатель CPR и его модификации.

Ранее [6] нами был проведен анализ масштабируемости на двух узлах вычислительного кластера, оснащенных суммарно восемью GPU NVIDIA Tesla M40, метода BiCGStab с предобуславливателем CPR-AMG, в котором на первой ступени использовался классический алгоритм AMG, а на второй – BILU(0). Было показано, что эффективность распараллеливания решателя на 8 GPU составляет не более 50 %, причем основным узким местом, препятствующим достижению более высокой масштабируемости на нескольких GPU, является многопроцессорная реализация классического алгоритма AMG.

В данной работе рассматривается предобуславливатель CPR-AIPS, в котором в качестве альтернативы AMG на первой ступени используется описанный выше метод аппроксимации обратной матрицы. Разложение по формуле (1) строится для подматрицы на давление (A_p), которая формируется из исходной матрицы СЛАУ с помощью алгоритма метода CPR в предположении о малом изменении насыщенностей в расчетных ячейках.

При использовании метода AIPS в рамках CPR в качестве матрицы P предлагается задействовать трехдиагональную часть A_p. Это позволяет в рамках неявного способа применения AIPS свести вычисление выражений вида $z = P^{-1}b$ к решению систем $Pz = b$ с трехдиагональной матрицей. Следовательно, для повышения быстродействия предобуславливателя CPR-AIPS на GPU необходим параллельный алгоритм, рассчитанный на многократное решение СЛАУ с неизменной трехдиагональной матрицей, сформированной из подматрицы на давление.

2.3 Параллельный алгоритм решения линейных систем с трехдиагональной матрицей, состоящей из множества независимых блоков различного размера

Ключевым прямым методом решения СЛАУ с трехдиагональной матрицей общего вида является метод прогонки [13], обладающий крайне низким ресурсом параллелизма. В то же время существует множество подходов к распараллеливанию решения трехдиагональных СЛАУ [14], в том числе параллельные алгоритмы, подходящие для исполнения на GPU. Так в популярной библиотеке cuSPARSE [15] из состава CUDA Toolkit имеются функции *cusparse<t>gtsv* и *cusparse<t>gtsv2,* основанные на параллельном алгоритме SPIKE [16], а также функции *cusparse<t>gtsv_nopivot* и *cusparse<t>gtsv2_nopivot*, в которых используются алгоритмы циклической редукции [17].

Отметим, что при поочередном решении СЛАУ с неизменной трехдиагональной матрицей выгоднее использовать вторые версии функций: *cusparse<t>gtsv2* и *cusparse<t>gtsv2_nopivot,* которые предусматривают использование буфера, предварительно подготовленного с помощью функций *cusparse<t>gtsv2_bufferSizeExt* и *cusparse<t>gtsv2_nopivot_bufferSizeExt* соответственно. Это позволяет устранить избыточные задержки на создание буферов при многократном поочередном вызове функций, реализующих решение трехдиагональных СЛАУ.

Рассмотрим структуру трехдиагональной части матрицы линейной системы, получаемой в результате применения метода конечных объемов на декартовой сетке размерности *Nx*Ny*Nz* в

случае полностью неявной разностной схемы, к примеру, при дискретизации уравнений фильтрации углеводородов в пористых средах. Предположим, что в трехдиагональную часть матрицы СЛАУ будут входить коэффициенты, связывающие соседние ячейки по оси *Oz*. Тогда трехдиагональная часть матрицы СЛАУ будет состоять из *Nx*Ny* независимых квадратных блоков *Nz*Nz*. Данная особенность структуры трехдиагональной части матрицы СЛАУ позволяет вместо решения одной полноразмерной трехдиагональной СЛАУ проводить параллельное решение *Nx*Ny* трехдиагональных СЛАУ меньшей размерности, что обеспечивает существенный ресурс параллелизма. К примеру, при размерностях сеточной модели *Nx=Ny=Nz=100* получаем *10 000* блоков, каждый из которых может независимо обрабатываться с помощью метода прогонки. Рисунок 1 иллюстрирует данную идею на примере сетки с размерностями *Nx=Ny=Nz=2*.

Рис. 1. Пример регулярной блочности в трехдиагональной части матрицы линейной системы

В терминах архитектуры CUDA [18] для параллельной обработки можно задействовать *Nx*Ny* нитей, каждая из которых будет решать с помощью метода прогонки свою независимую часть трехдиагональной СЛАУ, размещенной в глобальной памяти GPU в виде трех массивов *l*, *d* и *u*, хранящих элементы нижней, главной и верхней диагонали соответственно. В «лобовой» реализации параллельного алгоритма решения множества независимых трехдиагональных СЛАУ нити, в рамках реализации концепции SIMT (Single Instruction Multiple Threads), будут одновременно обращаться к элементам, расположенным в памяти с шагом *Nz* (рис. 2), что является неэффективным шаблоном доступа к глобальной памяти GPU.

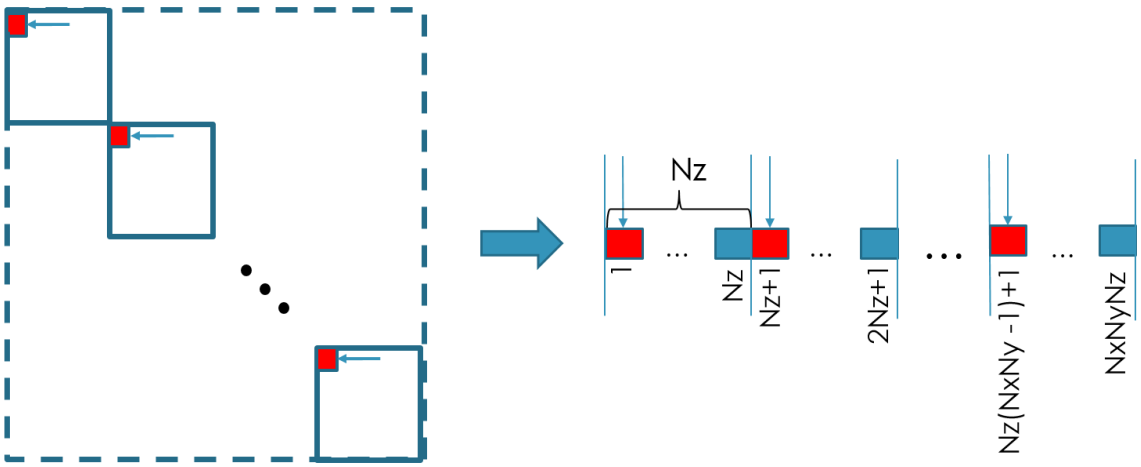

Рис. 2. Неэффективный шаблон доступа к глобальной памяти на примере главной диагонали

В целях ускорения доступа к глобальной памяти в литературе [19] предлагается выполнить предварительную перестановку строк трехдиагональной матрицы, обеспечивающую объединение запросов к глобальной памяти GPU от синхронно выполняющихся нитей, сгруппированных в один варп (от англ. warp) [18]. Для этого элементы каждой из диагоналей, расположенные с шагом *Nz*, переносятся в отдельный блок и располагаются поочередно друг за другом. В ре-

зультате (рис. 3) нити обращаются к данным, расположенным в глобальной памяти в рамках компактного сегмента, что обеспечивает более высокую скорость выборки данных из памяти.

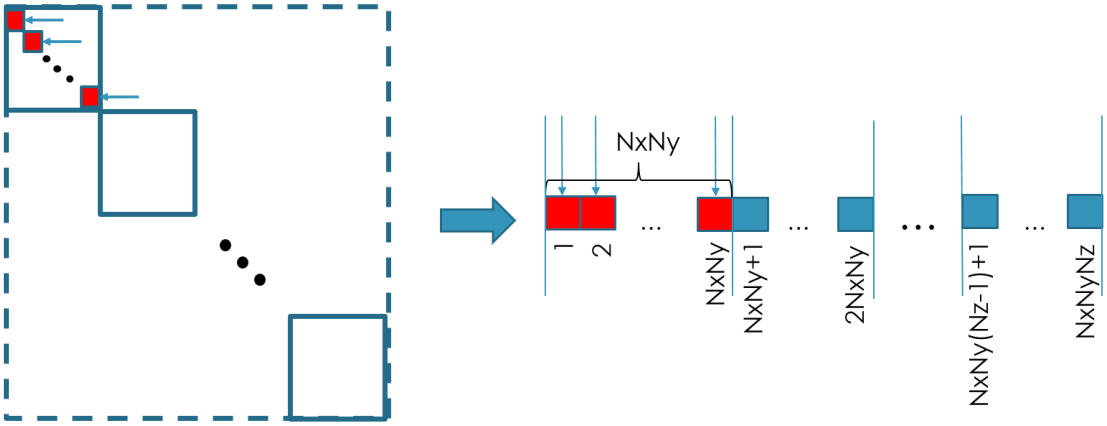

Рис. 3. Эффективный шаблон доступа к глобальной памяти на примере главной диагонали

На практике трехдиагональная часть исходной матрицы СЛАУ может состоять из независимых блоков различного размера. К примеру, при построении сеточных моделей реальных нефтегазовых месторождений значительное количество ячеек может попасть в разряд неактивных, которые исключаются из расчетов. На рисунке 4 проиллюстрирована ситуация, когда исключение из расчета двух ячеек приводит к появлению в трехдиагональной части матрицы СЛАУ наряду с двумя блоками 2*2 двух блоков 1*1.

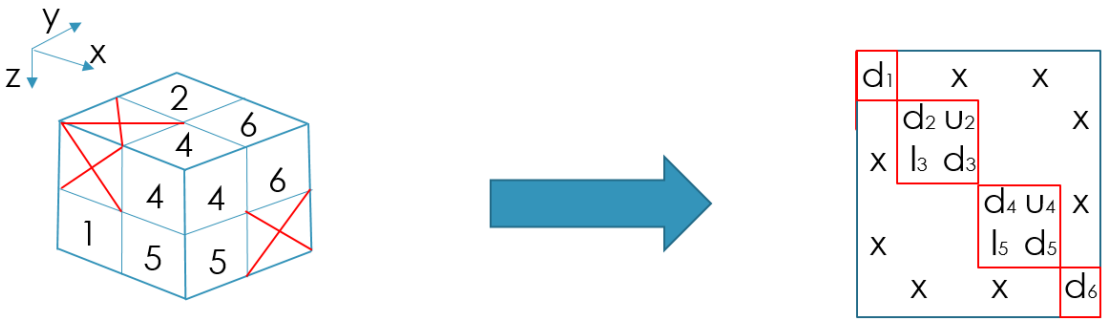

Рис. 4. Пример нерегулярной блочности в трехдиагональной части матрицы линейной системы

В таблице 1 содержатся характеристики тестовых матриц, полученных при решении уравнений трехфазной фильтрации, дискретизированных по времени с помощью полностью неявной разностной схемы, которые будут задействованы далее в экспериментальной части. В таблице 2 приведены характеристики трехдиагональных матриц, сформированных из подматриц на давление, которые в свою очередь получены в результате применения метода CPR при решении СЛАУ с матрицами из таблицы 1.

Таблица 1. Характеристики тестовых матриц СЛАУ

Название матрицы	Размерность матрицы	Количество ненулевых элементов	Среднее количество ненулевых элементов в строке
imsh	1 500 000	55 815 624	37,210
immn	2 304 102	42 859 314	18,601
krrv	4 320 921	85 471 137	19,781
mmnt	5 637 747	109 595 799	19,440
fdrv	6 610 263	118 221 633	17,885
kmms	8 630 895	167 332 329	19,388
lkms	13 665 705	254 102 823	18,594

Таблица 2. Характеристики тестовых трехдиагональных матриц

Название матрицы	Размерность матрицы	Минимальный размер блока	Максимальный размер блока	Количество независимых блоков
imsh3	500 000	10	10	50 000
immn3	768 034	1	34	221 402
krrv3	1 440 307	1	39	233 866
mmnt3	1 879 429	1	45	281 352
fdrv3	2 203 421	1	55	633 202
kmms3	2 876 965	1	121	389 199
lkms3	4 555 235	1	27	1 167 013

Из таблицы 2 можно с одной стороны сделать вывод о наличии в трехдиагональных матрицах большого количества блоков, которые могут обрабатываться независимо, обеспечивая существенный ресурс параллелизма, а с другой стороны о значительном разбросе в размерах блоков, составляющих трехдиагональные матрицы, что порождает необходимость балансировки нагрузки между параллельно работающими нитями.

Чтобы сбалансировать вычисления на GPU предлагается провести предварительную сортировку блоков матрицы согласно их размерности, после чего распределить блоки одинакового размера между нитями с последовательно идущими индексами. Это обеспечивает возможность в достаточной степени сбалансированного выполнения независимых друг от друга прогонок в каждом конкретном варпе и блоке нитей.

В результате мы приходим к следующему алгоритму решения линейных систем с трехдиагональной матрицей, имеющей нерегулярную блочную структуру.

Алгоритм 1.

1. *Этап анализа структуры матрицы.*

1.1 Провести анализ структуры трехдиагональной матрицы для выявления количества независимых блоков, их размеров и местоположения.

2. *Этап подготовки матрицы.*

2.1 Отсортировать блоки матрицы согласно их размерности и выделить группы блоков одинаковой размерности в целях балансировки нагрузки на этапе решения.

2.2 Провести перестановку строк матрицы внутри каждой группы блоков одинаковой размерности для более производительного доступа к памяти на этапе решения.

3. *Этап решения трехдиагональной линейной системы.*

3.1 Провести перестановку элементов вектора правой части согласно перестановкам строк матрицы, проведенным на шагах 2.1 и 2.2.

3.2 Параллельно применить метод прогонки для каждого блока матрицы.

3.3 Провести обратную перестановку элементов вектора решения.

Отметим, что при решении СЛАУ итерационным методом BiCGStab с предобуславливателем CPR-AIPS этапы 1 и 2 предложенного алгоритма достаточно выполнить только один раз до начала итерационного процесса (в последующих вызовах решателя СЛАУ при отсутствии изменений структуры блочности в трехдиагональной матрице шаг 1 можно вообще исключить), а шаги 3.1-3.3 необходимо выполнять многократно для поочередного решения трехдиагональных СЛАУ с различными правыми частями в рамках применения метода AIPS, аппроксимирующего обратную матрицу на основе степенного разложения по формуле (1). Также, учитывая то, что в рамках применения предобуславливателя AIPS этап 3 многократно повторяется с неизменной трехдиагональной матрицей, целесообразно на шаге 3.2 применять вместо классического повторный вариант алгоритма прогонки, в котором переиспользуются однократно предвычисленные коэффициенты [20].

3. Практическая часть

В данном разделе приведены результаты тестирования рассмотренных в теоретической части методов и алгоритмов на двух вычислительных платформах, оснащенных различными серверными GPU NVIDIA Tesla:

1) на 1 GPU P100 исследовано быстродействие предложенного параллельного алгоритма решения линейных систем с трехдиагональной матрицей, имеющей нерегулярную блочную структуру, относительно алгоритмов решения трехдиагональных систем общего вида, реализованных в библиотеке cuSPARSE;
2) на 1 GPU P100 и V100 проведена сравнительная оценка эффективности предобуславливателей CPR-AMG и CPR-AIPS;
3) на 4 GPU V100 исследована масштабируемость процедуры решения разреженных СЛАУ итерационным методом BiCGStab с предобуславливателем CPR-AIPS.

Для реализации первого эксперимента были задействованы трехдиагональные матрицы, характеристики которых приведены в таблице 2, а для второго и третьего экспериментов – разреженные матрицы, описанные в таблице 1.

Расчеты с использованием GPU P100 были проведены на одном из узлов вычислительного кластера УГАТУ [21] с 2 x CPU Intel Gold 6126 и 2 x GPU NVIDIA Tesla P100, работающего под управлением CentOS 6. Для проведения расчетов на GPU V100 была задействована облачная платформа Amazon AWS p3.8xlarge со следующими характеристиками: ОС RHEL 7, 32 x vCPU Intel Xeon E5-2686 v4 и 4 x GPU NVIDIA Tesla V100 SXM2. На обеих вычислительных платформах были установлены идентичные средства параллельного программирования: CUDA Toolkit 9.1, библиотека AmgX v.2.0.0.130-opensource, OpenMPI v.1.8.8.

Параметры тестирования: условие остановки итерационного процесса – достижение относительной невязкой величины $\varepsilon = 10^{-4}$; начальное приближение – нулевой вектор; расчеты проводились с двойной точностью.

3.1 Исследование быстродействия параллельного алгоритма решения линейных систем с трехдиагональной матрицей

В таблице 3 приведены осредненные времена решения СЛАУ с тестовыми матрицами из таблицы 2, полученные в ходе многократного решения трехдиагональных линейных систем в рамках итерационного процесса метода BiCGStab с предобуславливателем CPR-AIPS. Решение СЛАУ производилось средствами функций *cusparseDgtsv2* и *cusparseDgtsv2_nopivot* из библиотеки cuSPARSE, а также реализации параллельного алгоритма, описанного в разделе 2.3.

Таблица 3. Время решения тестовых трехдиагональных СЛАУ с использованием различных алгоритмов

Алгоритм	Матрица	imsh3	immn3	krrv3	mmnt3	fdrv3	kmms3	lkms3
gtsv2	Время решения, мс	0,323	0,469	0,753	0,976	1,209	1,459	2,207
gtsv2_nopivot		0,386	0,578	1,055	1,374	1,586	2,046	3,228
Алгоритм 1		0,119	0,212	0,409	0,530	0,568	0,824	1,152
	Время подготовки, мс	0,130	0,190	0,400	0,550	0,710	0,960	1,760

Из таблицы 3 следует, что среди рассмотренных функций из библиотеки cuSPARSE наибольшее быстродействие демонстрирует *cusparseDgtsv2*. В то же время решение СЛАУ с использованием предложенного нами алгоритма без учета временных затрат на выполнение вспомогательных этапов производится в среднем быстрее в 2,7 и 2,1 раза относительно *cusparseDgtsv2_nopivot* и *cusparseDgtsv2* соответственно. Детализацию полученных ускорений по матрицам иллюстрирует рисунок 5. Наибольшее ускорение достигается на матрице imsh3, состоящей из 50 000 независимых блоков одинакового размера, что обеспечивает идеальную балансировку нагрузки между нитями при выполнении шага 3.2 предложенного алгоритма.

Несмотря на то, что с учетом этапа подготовки матрицы на большинстве тестовых СЛАУ наш алгоритм отрабатывает медленнее чем *cusparseDgtsv2*, его преимущество проявляется при

необходимости выполнения процедуры решения СЛАУ с неизменной трехдиагональной матрицей хотя бы два раза. А с учетом того, что в рамках итерационного процесса метода BiCGStab с предобуславливателем CPR-AIPS процедура решения может повторяться десятки и даже сотни раз, время, затрачиваемое на подготовку, становится относительно мало.

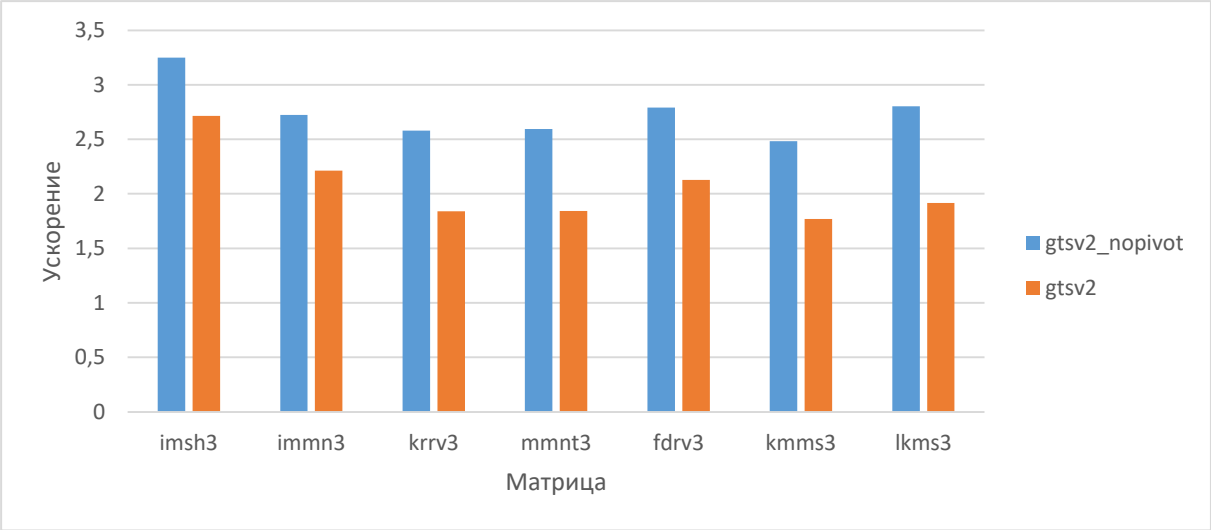

Рис. 5. Ускорение решения трехдиагональных систем с использованием предложенного алгоритма (без учета вспомогательных этапов) относительно функций cusparseDgtsv2_nopivot и cusparseDgtsv2

3.2 Сравнительная оценка эффективности предобуславливателей CPR-AMG и CPR-AIPS

В таблицах 4 и 5 представлены результаты работы на GPU NVIDIA Tesla P100 и V100 итерационного метода BiCGStab с предобуславливателями CPR-AMG и CPR-AIPS.

Таблица 4. BiCGStab с предобуславливателями CPR-AMG и CPR-AIPS на P100

Метод	Матрица	imsh	immn	krrv	mmnt	fdrv	kmms	lkms
CPR-AMG	Время решения, с	0,135	0,180	0,285	0,360	0,314	0,678	0,885
	Число итераций	1	2,5	2	2	1	3,5	2,5
CPR-AIPS	Время решения, с	0,070	0,120	0,189	0,205	0,182	0,738	1,409
	Число итераций	1	2,5	2	1,5	1	5	6,5

Таблица 5. BiCGStab с предобуславливателями CPR-AMG и CPR-AIPS на V100

Метод	Матрица	imsh	immn	krrv	mmnt	fdrv	kmms	lkms
CPR-AMG	Время решения, с	0,113	0,144	0,202	0,252	0,232	0,437	0,568
	Число итераций	1	2,5	2	2	1	3,5	2,5
CPR-AIPS	Время решения, с	0,043	0,075	0,102	0,108	0,100	0,396	0,772
	Число итераций	1	2,5	2	1,5	1	5	6,5

Проведенные эксперименты показали, что реализованный в данной работе предобуславливатель CPR-AIPS не уступает по обеспечению скорости сходимости итерационного процесса предобуславливателю CPR-AMG на большинстве тестовых СЛАУ. Ухудшение скорости сходимости наблюдается только на двух матрицах, имеющих наибольшую размерность: около 10^7.

CPR-AIPS позволяет ускорить относительно CPR-AMG решение 5 из 7 тестовых СЛАУ при расчете на P100 и 6 из 7 – на V100. Причем в среднем при помощи CPR-AIPS время решения СЛАУ снижается относительно CPR-AMG в 1,4 раза на P100 и в 1,9 раз на V100.

Рисунок 6 иллюстрирует повышение производительности решения СЛАУ за счет перехода с P100 на V100. Видно, что расчеты в большей степени ускоряются при использовании CPR-AIPS (в среднем в 1,8 раза) нежели CPR-AMG (в среднем в 1,4 раза). Анализ профилей выпол-

нения программ показал, что причина относительно низкого ускорения CPR-AMG на V100 относительно P100 кроется в процедуре построения иерархии уровней AMG, которая занимает более 50% от общего времени решения СЛАУ и практически не ускоряется при переходе с P100 на V100.

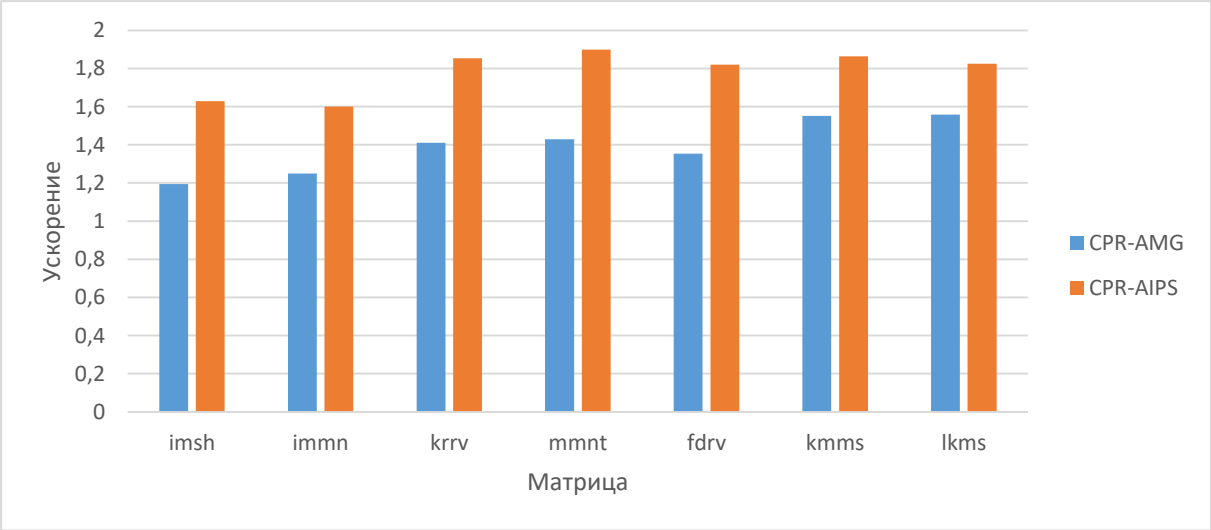

Рис. 6. Ускорение решения СЛАУ методом BiCGStab с предобуславливателями CPR-AMG и CPR-AIPS на V100 относительно P100

Таким образом, реализованный предобуславливатель CPR-AIPS за счет более легковесного относительно CPR-AMG локального предобуславливателя демонстрирует высокое быстродействие на современном GPU NVIDIA Tesla на базе архитектуре Volta и позволяет снизить относительно CPR-AMG время решения подавляющего большинства тестовых СЛАУ.

3.3. Масштабируемость метода BiCGStab с предобуславливателем CPR-AIPS

В таблице 6 представлены результаты работы метода BiCGStab с предобуславливателем CPR-AIPS на 1-4 GPU NVIDIA Tesla V100.

Таблица 6. Масштабируемость метода BiCGStab с предобуславливателем CPR-AIPS

Матрица	Количество GPU	1	2	4
imsh	Время решения, с.	0,043	0,031	0,028
	Число итераций	1	1	1
immn	Время решения, с.	0,075	0,054	0,047
	Число итераций	2,5	2,5	2,5
krrv	Время решения, с.	0,102	0,061	0,048
	Число итераций	2	2	2
mmnt	Время решения, с.	0,108	0,063	0,048
	Число итераций	1,5	1,5	1,5
fdrv	Время решения, с.	0,100	0,061	0,046
	Число итераций	1	1	1
kmms	Время решения, с.	0,396	0,224	0,157
	Число итераций	5	5	4,5
lkms	Время решения, с.	0,772	0,397	0,215
	Число итераций	6,5	6,5	6,5

Положительный момент заключается в том, что увеличение количества задействованных GPU не снижает скорость сходимости итерационного процесса, чему способствует естествен-

ное эквивалентное распараллеливание метода AIPS. Для наглядности ускорение решения СЛАУ на 2 и 4 GPU относительно 1 GPU представлено на рисунке 7.

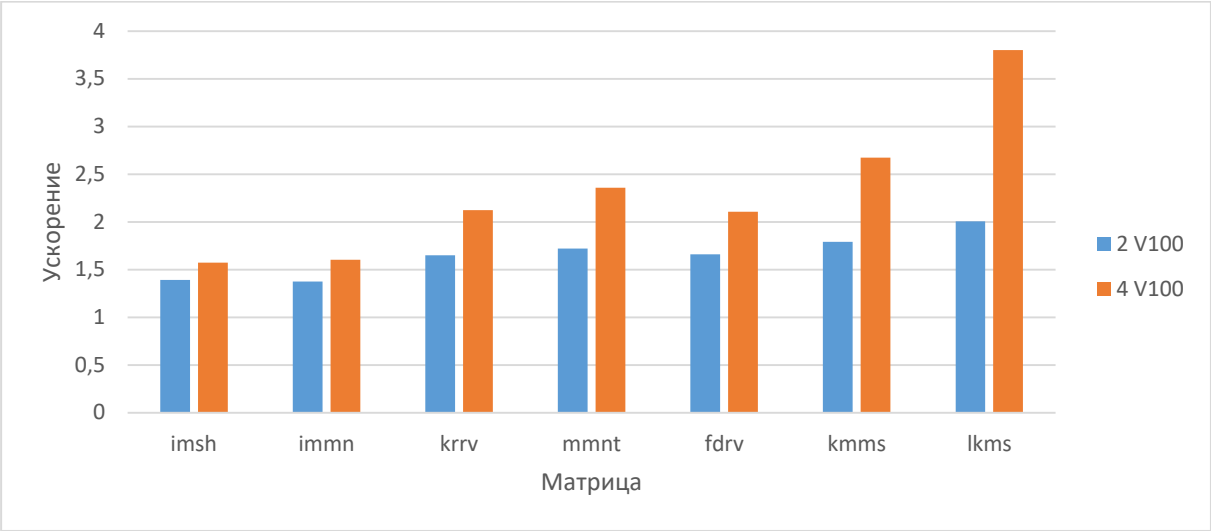

Рис. 7. Ускорение решения СЛАУ (BiCGStab + CPR-AIPS) на 2 и 4 GPU относительно 1 GPU.

Наилучшие показатели масштабируемости достигаются при решении СЛАУ большей размерности. Это вполне ожидаемый эффект, так как при декомпозиции СЛАУ меньшей размерности не обеспечивается достаточный ресурс параллелизма для загрузки нескольких GPU.

4. Заключение

В работе рассмотрена применимость метода AIPS, аппроксимирующего обратную матрицу на основе степенного разложения в ряд Неймана, в рамках двухступенчатого предобуславливателя CPR. Предложен ориентированный на архитектуру CUDA параллельный алгоритм решения линейных систем с трехдиагональной матрицей специального вида: состоящей из независимых блоков различного размера.

Показано, что реализация предложенного алгоритма может более чем в 2 раза превосходить по быстродействию функции *cusparseDgtsv2_nopivot* и *cusparseDgtsv2* решения трехдиагональных линейных систем общего вида из библиотеки cuSPARSE. На гибридных вычислительных системах с GPU NVIDIA Tesla P100 и V100 проведена сравнительная оценка эффективности предобуславливателей CPR-AIPS и CPR-AMG, которая показала, что CPR-AIPS позволяет снизить относительно CPR-AMG время решения большинства тестовых СЛАУ, полученных при решении уравнений трехфазной фильтрации. Также отмечена приемлемая масштабируемость процедуры решения СЛАУ методом BiCGStab с предобуславливателем CPR-AIPS: ускорение на двух GPU V100 составляет от 1,4 до 2 раз, на четырех – от 1,6 до 3,8 раза относительно одного GPU.

Литература

1. Топ50. URL: http://top50.supercomputers.ru (дата обращения: 15.04.2018).
2. Х. Азиз, Э. Сеттари. Математическое моделирование пластовых систем. – 2-е изд., стер. – М.: Институт компьютерных исследований, 2004. 411 с.
3. И. И. Газизов, А. В. Юлдашев. Разработка параллельного линейного решателя для задачи гидродинамического моделирования нефтегазовых месторождений // Эвристические алгоритмы и распределённые вычисления. Самара: 2014. Том 1, № 1 (2014). С. 88–96.
4. AlgoWiki: Открытая энциклопедия свойств алгоритмов. Стабилизированный метод бисопряженных градиентов (BiCGStab). URL: http://algowiki-

project.org/ru/Стабилизированный_метод_бисопряженных_градиентов_(BiCGStab) (дата обращения: 31.05.2018).

5. Saad Y. Iterative methods for sparse linear systems. 2nd ed. SIAM, 2003. 528 p.
6. Р.Р. Губайдуллин, Н.В. Репин, Р.Ф. Сайфутдинов, А.В. Юлдашев. Специализированный решатель разреженных систем линейных алгебраических уравнений на вычислительных кластерах, оснащенных графическими процессорами // Суперкомпьютерные дни в России: труды международной конференции (26-27 сентября 2016 г., г. Москва). – М.: Изд-во МГУ, 2016. С. 673–682.
7. Wallis J. R., Kendall R.P., Little T. E. Constrained residual acceleration of conjugate residual methods // SPE 1353. 1985. P. 415–428.
8. Ruge J. W., Stuben K. Algebraic multigrid (AMG) // Multigrid Methods / S. F. McCormick (ed.) Philadelphia, PA, USA: SIAM, 1987. Vol. 3, p. 73–130.
9. Н.С. Недожогин, С.П. Копысов, А.К. Новиков. Параллельное формирование предобуславливателя на основе обращения Шермана-Моррисона // Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции (31 марта – 2 апреля 2015 г., г. Екатеринбург). Челябинск: Издательский центр ЮУрГУ, 2015. – С. 436–441. ISBN 978-5-696-04663-1 (Электронное издание).
10. Chen K. Matrix Preconditioning Techniques and Applications (Cambridge. Monographs on Applied and Computational Mathematics), Cambridge University. Press, July 2005.
11. Prabhu H., Edfors O., Rodrigues J., Liu L., Rusek F. Hardware Efficient Approximative Matrix Inversion for Linear Pre-coding in Massive MIMO // IEEE International Symposium on Circuits and Systems (ISCAS), 2014, Melbourne, Australia, 2014/06/01. P. 1700–1703.
12. Fung L. S. K., Dogru A. H. Parallel Unstructured Solver Methods for Complex Giant Reservoir Simulation // SPE 106237, proceedings of the SPE Reservoir Simulation Symposium, Houston, Texas, 26-28 February 2007.
13. AlgoWiki: Открытая энциклопедия свойств алгоритмов. Прогонка, точечный вариант. URL: http://algowiki-project.org/ru/Прогонка,_точечный_вариант (дата обращения: 15.04.2018).
14. Фролов А.В. Ещё один метод распараллеливания прогонки с использованием ассоциативности операций // Суперкомпьютерные дни в России: Труды международной конференции (28-29 сентября 2015 г., г. Москва). – М.: Изд-во МГУ, 2015. С. 151–162.
15. cuSPARSE. URL: https://docs.nvidia.com/cuda/cusparse/ (дата обращения: 15.04.2018).
16. Li-Wen Chang, John A. Stratton, Hee-Seok Kim, and Wen-mei W. Hwu. A Scalable, Numerically Stable, High-performance Tridiagonal Solver using GPUs // The International Conference for High Performance Computing, Networking Storage and Analysis, 2012.
17. Хокни Р., Дэюессхоуп К. Параллельные ЭВМ. Архитектура, программирование и алгоритмы. М.: Радио и связь, 1986. 392 с.
18. Боресков А.В. и др. Параллельные вычисления на GPU. Архитектура и программная модель CUDA. Учебное пособие // М.: МГУ. 2012. 336 с.
19. Endre Laszlo, Mike Giles, and Jeremy Appleyard. Manycore algorithms for batch scalar and block tridiagonal solvers. ACM Trans. Math. Softw. 42, 4, Article 31 (June 2016) 36 p.
20. AlgoWiki: Открытая энциклопедия свойств алгоритмов. Классическая монотонная прогонка, повторный вариант. URL: http://algowiki-project.org/ru/Классическая_монотонная_прогонка,_повторный_вариант (дата обращения: 15.04.2018).
21. Суперкомпьютер УГАТУ. URL: http://www.ugatu.su/supercomputer/ (дата обращения: 15.04.2018).