Разработка параллельного линейного решателя для гидродинамического моделирования нефтегазовых месторождений на гибридных системах с графическими процессорами

Р.Р. Губайдуллин, Н.В. Репин, А.В. Юлдашев

Уфимский государственный авиационный технический университет

Для задачи гидродинамического моделирования нефтегазовых месторождений характерна высокая вычислительная ресурсоемкость. Настоящая работа нацелена на ускорение соответствующих расчетов посредством использования гибридных вычислительных систем с графическими процессорами, главным образом, на ускорение решения линейных систем, возникающих при численном моделировании многофазной фильтрации. Исследовано влияние формата хранения матриц на время выполнения на графическом процессоре базовых операций предобусловленного метода бисопряжённых градиентов со стабилизацией. Продемонстрирована производительность решения тестовых разреженных систем при использовании различных предобуславливателей и подходов к их распараллеливанию.

1. Введение

Численное моделирование многофазных фильтрационных потоков углеводородов в пористой среде является одной из практических задач геолого-гидродинамического моделирования, решаемых недропользователями в процессе разработки нефтегазовых месторождений. Данная вычислительная задача решается в рамках построения и адаптации трехмерных геолого-гидродинамических моделей к данным истории разработки, прогнозирования эффективности эксплуатации нефтегазовых месторождений, анализа неопределенности и рисков при планировании геолого-технических мероприятий, выбора технологии разработки низко проницаемых коллекторов и др.

Для полномасштабного гидродинамического моделирования процессов нефтегазодобычи традиционно используются высокопроизводительные вычислительные системы, а также специализированные программные комплексы, имеющие параллельные версии: Eclipse (Schlumberger), Tempest More (Roxar), tNavigator (Rock Flow Dynamics), «РН-КИМ» (Роснефть) и т.п. Как правило, львиную долю времени (до 90%) в процессе численного моделирования занимает блок решения разреженных систем линейных алгебраических уравнений (СЛАУ) [1], поэтому ключевую роль играет производительность и масштабируемость линейного решателя.

Вышеупомянутые симуляторы ориентированы на традиционные вычислительные платформы – многоядерные системы с общей либо распределенной памятью на базе центральных процессоров (Central Processing Unit, CPU) преимущественно архитектуры x86, в то время как сегодня широкое применение находят гибридные (гетерогенные) системы, в которых наряду с центральными процессорами (x86, ARM, Power) устанавливаются массивно-параллельные ускорители вычислений, в частности графические процессоры (Graphics Processing Unit, GPU) NVIDIA. Эффективное применение гибридных систем сопряжено с необходимостью переработки программного обеспечения путем переписывания либо оптимизации исходных кодов под конкретную вычислительную платформу, причем в некоторых случаях необходим переход на использование модифицированных либо альтернативных алгоритмов, обладающих необходимым ресурсом параллелизма для обеспечения полноценной загрузки большого количества ядер, имеющихся в гибридных системах. Например, GPU NVIDIA K20X содержит более 2 500 ядер, причем для его полноценной загрузки требуется породить десятки тысяч легковесных нитей, одновременно выполняющих множество независимых арифметических операций.

В данной статье представлены текущие результаты проекта по разработке специализированного параллельного линейного решателя СЛАУ для гидродинамического моделирования нефтегазовых месторождений на гибридных вычислительных системах, оснащенных GPU NVIDIA, а также традиционных многоядерных системах.

2. Алгоритмы решения СЛАУ и их программная реализация

Матрицы СЛАУ, возникающие в ходе численного решения уравнений многофазной фильтрации потоков углеводородов в пористой среде, являются сильно разрежёнными и плохо обусловленными, поэтому для их решения принято использовать итерационные методы подпространства Крылова, например, метод бисопряжённых градиентов со стабилизацией (BiCGStab) с различными предобуславливателями. Из-за высокой жесткости системы найти решение без предобуславливателя или с простейшими вариантами (например, Якоби) не удается, поскольку при большом числе итераций накопление вычислительной ошибки приводит к значительному искажению приближенного решения [2].

Одним из наиболее часто используемых предобуславливателей, применяемых при численном решении уравнений многофазной фильтрации углеводородов, является неполное LU-разложение без заполнения (с нулевым заполнением) — ILU(0) [3], также используются и другие предобуславливатели класса ILU [4]. Более эффективным с точки зрения сходимости итерационного метода считается специализированный двухступенчатый предобуславливатель CPR и его модификации [2,5,6]. В рамках CPR выделяется локальный предобуславливатель, в качестве которого часто используется алгебраический многосеточный метод (Algebraic Multigrid Method, AMG) [7], а также глобальный предобуславливатель, к примеру, ILU(0). В нашей работе локальный предобуславливатель строится на основе классического алгоритма алгебраического многосеточного метода, где в качестве сглаживателя используется блочный метод Якоби, для решения задачи на грубой сетке – полное LU-разложение.

Алгоритмы построения предобуславливателя ILU(0) и решения сопутствующих систем с нижне- и верхнетреугольными матрицами в классической постановке обладают незначительным ресурсом параллелизма для СЛАУ с сильно разреженными матрицами. Существует несколько различных подходов к распараллеливанию предобуславливателей класса ILU. Одним из таких подходов является разделение матрицы на уровни, содержащие строки матрицы, которые могут обрабатываться параллельно (level scheduling) [8]. Подобный подход реализован, например, в библиотеке NVIDIA cuSPARSE для GPU [9, 10]. Альтернативой является использование блочных модификаций предобуславливателей класса ILU: предобуславливатель строится в виде блоков, которые могут обрабатываться параллельно. К ним можно отнести метод Капорина-Коньшина разбиения матрицы на блоки с перекрытиями, который может быть применён и для линейных систем с несимметричными матрицами [11, 12], а также более простой подход – построение предобуславливателя с помощью блочного метода Якоби на основе блочно-диагональной части исходной матрицы (Block-Jacobi) [3].

В нашем решателе заложены различные подходы к распараллеливанию ILU(0).

1) На CPU используется блочный метод Якоби, обеспечивающий возможность параллельного исполнения операций построения и применения неполного LU-разложения на множестве процессорных ядер.
2) На одном GPU средствами библиотеки cuSPARSE реализуется разделение строк матрицы на уровни и их дальнейшая параллельная поуровневая обработка.
3) На одном или нескольких GPU (в рамках CPR) предлагается использовать комбинированный подход: процедура разделения на уровни применяется к блочно-диагональной части исходной матрицы с произвольно заданным количеством блоков (в нашей работе от 64 до 256) [13];
4) Для распараллеливания ILU(0) как самостоятельного предобуславливателя на нескольких GPU также предлагается использование комбинированного подхода, но число блоков выбирается равным количеству задействованных GPU. Таким образом, каждый GPU получает один диагональный блок исходной матрицы и обрабатывает его параллельно посредством разделения на уровни.

CPU-версия решателя является многопоточной (OpenMP) и базируется на функциях математической библиотеки Intel Math Kernel Library (MKL). Для хранения матриц используется формат CSR, т.к. другие форматы в функциях MKL, связанных с ILU(0), не поддерживаются. Реализован итерационный метод BiCGStab с предобуславливателем ILU(0), распараллеливание которого проводится как отмечено выше в пункте 1.

GPU-версия решателя разработана средствами CUDA, OpenMP, а также базируется на функциях математических библиотек NVIDIA cuBLAS и cuSPARSE из состава CUDA Toolkit и свободной для некоммерческого использования библиотеки AmgX. Реализован итерационный метод BiCGStab с предобуславливателями ILU(0) и CPR. Причем распараллеливание ILU(0) осуществляется в соответствии с приведенными выше пунктами 2-4 в зависимости от конфигурации запуска. Операции построения и применения AMG на данный момент выполняются только на одном GPU, т.к. параллельная версия классического алгоритма алгебраического многосеточного метода для систем с несколькими GPU, реализованная в библиотеке AmgX, может быть задействована только из MPI-программ. Для хранения разреженных матриц в памяти поддерживается как универсальный формат CSR (Compressed Sparse Row), так и формат BSR (Block compressed Sparse Row), предназначенный для хранения разреженных матриц с блочной структурой. В целях обеспечения максимальной производительности GPU-версии линейного решателя далее в разделе 3.1 исследуется влияние формата хранения матриц на время выполнения на GPU базовых вычислительных операций.

3. Экспериментальная часть

В таблице 1 рассмотрены характеристики тестовых разреженных матриц СЛАУ, полученных при гидродинамическом моделировании реальных нефтегазовых месторождений. Отметим, что тестовые матрицы получены при решении уравнений трехфазной фильтрации, дискретизированных по времени с использованием полностью неявной разностной схемы, и для них характерна группировка ненулевых элементов в блоки размером 3x3, что учитывается при использовании формата хранения BSR. Особенности структуры подобных матриц более подробно рассмотрены в [14]. Данные матрицы будут использованы далее для оценки различных аспектов эффективности решения СЛАУ на CPU и GPU.

Таблица 1. Характеристики тестовых матриц

Матрица	Размерность	Кол-во ненулевых элементов	Среднее количество элементов в строке
kslv	872 547	15 397 533	17,65
imsh	1 500 000	55 815 624	37,21
imms	2 304 102	42 775 866	18,57
krrv	4 320 921	85 471 137	19,78
fdrv	6 610 263	118 000 000	17,85

Приведенные далее результаты получены на гибридной вычислительной системе, содержащей два CPU Intel Xeon E5-2670 и два GPU NVIDIA K20x, работающих в режиме ECC Off. В ходе тестирования использовались следующие параметры: условие остановки итерационного процесса – достижение относительной невязкой величины $\varepsilon = 1e - 6$; начальное приближение – нулевой вектор; расчёты проводились с двойной точностью.

3.1 Исследование влияния формата хранения матриц на время выполнения на GPU базовых операций предобусловленного метода BiCGStab

К наиболее трудоемким (базовым) операциям метода BiCGStab с предобуславливателем ILU(0) относятся следующие:

- построение предобуславливателя ILU(0);
- решение треугольных систем;
- умножение матрицы на вектор.

Ранее в нашем решателе как в CPU-, так и в GPU-версиях для хранения разреженных матриц использовался популярный формат CSR, который поддерживается библиотеками MKL и cuSPARSE во всех указанных выше операциях. С выходом CUDA Toolkit версии 6.0 в библиотеке cuSPARSE появилась поддержка формата BSR в функциях, связанных с использованием неполного LU-разложения, причем для операции умножения разреженной матрицы на вектор

соответствующая поддержка имелась и ранее. В связи с этим в GPU-версии решателя в качестве одного из основных форматов хранения матриц был поддержан формат BSR. В данном разделе приводятся результаты исследования влияния используемого формата хранения матриц на производительность выполнения на GPU упомянутых выше базовых операций.

Необходимо отметить, что на данный момент библиотека cuSPARSE содержит две версии функций, реализующих построение ILU(0) и решение треугольных систем, для разреженных матриц в формате CSR. Например, **csrilu0* соответствует первой версии функции построения ILU(0), а **csrilu02* – второй версии. С учетом того, что для формата BSR первые версии функций отсутствуют, далее в сравнении используются только вторые версии функций и для формата CSR.

В таблице 2 приведены времена выполнения базовых операций при работе с различными форматами хранении матриц. Видно, что на операциях, связанных с ILU(0) наибольшая производительность достигается при использовании формата BSR: время построения ILU(0) меньше в 2,3-2,6 раза, а время решения треугольных систем меньше в 1,2-1,5 раза. В то же время операция умножения разреженной матрицы на вектор выполняется быстрее при использовании формата CSR, соответствующее ускорение составляет от 2 до 3 раз.

Таблица 2. Время выполнения на GPU базовых операций метода BiCGStab с предобуславливателем ILU(0)

Матрица	Время построения ILU(0), с		Время решения треугольных систем, с		Время умножения матрицы на вектор, с	
	CSR	BSR	CSR	BSR	CSR	BSR
kslv	0,078	0,03	0,0181	0,0135	0,0019	0,0046
imsh	0,14	0,06	0,03	0,02	0,003	0,008
imms	0,21	0,09	0,046	0,034	0,005	0,012
krrv	0,4	0,17	0,088	0,065	0,01	0,02
fdrv	0,6	0,25	0,12	0,097	0,01	0,03

В связи с полученными результатами в нашем решателе используются оба формата хранения матриц: CSR для хранения матрицы СЛАУ, а BSR для хранения неполного LU-разложения.

3.2 Оценка эффективности решения СЛАУ на CPU и GPU методом BiCGStab с предобуславливателем ILU(0)

В данном разделе приводится сравнительная оценка эффективности решения тестовых СЛАУ на 1-16 ядрах CPU и 1-2 GPU. Отметим, что распараллеливание операций построения и применения ILU(0) для CPU производится с помощью блочного метода Якоби: матрица, на основе которой строится ILU(0), разбивается на диагональные блоки (по числу задействованных процессорных ядер), далее обрабатываемые независимо. Аналогично производится разделение работы между несколькими GPU. Данный подход приводит к утрате части ненулевых элементов в матрице предобуславливателя с увеличением количества блоков, что негативно влияет на сходимость итерационного метода. Это можно заметить в таблице 3: количество итераций при расчете на 16 ядрах CPU больше в 1,04-3,24 раза относительно последовательного расчета.

Таблица 3. Количество итераций метода BiCGStab с блочно-диагональным предобуславливателем ILU(0) при решении СЛАУ на CPU и GPU

Матрица	Количество итераций						
	1 ядро CPU	2 ядра CPU	4 ядра CPU	8 ядер CPU	16 ядер CPU	1 GPU	2 GPU
kslv	19	19	37,5	36,5	61,5	19	19
imsh	1,5	1,5	1,5	1,5	1,5	1,5	1,5
imms	20,5	21	20,5	20	23	20,5	21
krrv	24,5	24	24,5	25	25	24,5	24
fdrv	4	4	4	4	4	4	4

Время решения СЛАУ, включающее в себя время на построение предобуславливателя и итерации метода BiCGStab, на 1-16 ядрах CPU и 1-2 GPU приведено в таблице 4. Максимальное ускорение параллельных вычислений на CPU составляет от 2,14 до 6,7 раз в зависимости от матрицы СЛАУ, причем минимальные ускорения наблюдаются на матрицах, для которых характерен рост числа итераций с увеличением количества задействованных ядер CPU. Ускорение при расчете на 2 GPU относительно 1 GPU составляет 1,61-1,73 раза.

Таблица 4. Время решения СЛАУ на CPU и GPU методом BiCGStab с блочно-диагональным предобуславливателем ILU(0)

Матрица	Время решения СЛАУ, с						
	1 ядро CPU	2 ядра CPU	4 ядра CPU	8 ядер CPU	16 ядер CPU	1 GPU	2 GPU
kslv	3,51	2	2,14	1,64	2,69	0,65	0,41
imsh	1,81	0,93	0,51	0,33	0,27	0,17	0,10
imms	10,27	5,65	3,35	2,49	2,72	1,81	1,12
krrv	23,33	12,37	7,63	5,9	5,58	4,04	2,34
fdrv	9,36	4,92	2,8	1,92	1,63	1,21	0,71

Целесообразность использования GPU подтверждается снижением времени решения СЛАУ на 2 GPU в 2,23-4,03 раза относительного минимального времени расчета на 2 CPU.

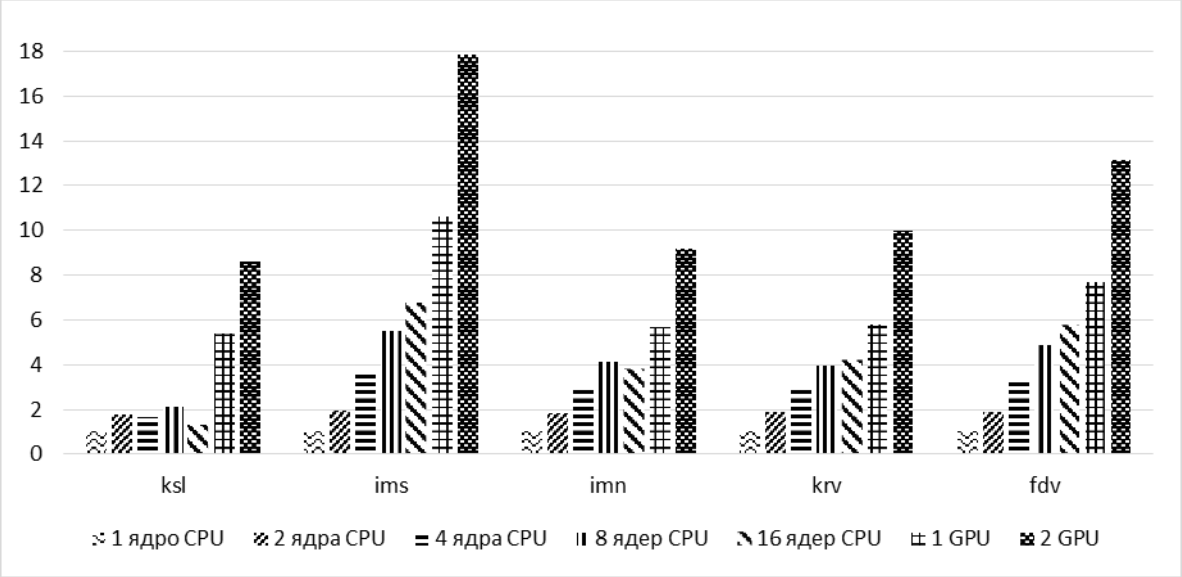

Рис. 1. Ускорение решения СЛАУ на CPU и GPU методом BiCGStab с блочно-диагональным предобуславливателем ILU(0)

Ускорение, которое можно получить при параллельном решении различных тестовых СЛАУ методом BiCGStab с блочно-диагональным предобуславливателем ILU(0), относительно времени последовательного расчета на одном ядре CPU продемонстрировано на рисунке 1.

3.3 Оценка эффективности решения СЛАУ на GPU методом BiCGStab с предобуславливателем CPR

Таблица 5 иллюстрирует улучшение сходимости итерационного метода при использовании предобуславливателя CPR: например, при решении СЛАУ с матрицей krrv число итераций может быть снижено до 7 раз. При этом возможно без ухудшения сходимости применение в рамках CPR комбинированного подхода к параллельному построению неполного LU-разложения без заполнения (ILU(0)*), который предполагает построение предобуславливателя в соответствии с алгоритмом разделения на уровни на базе блочно-диагональной части исходной матрицы с заданным количеством блоков.

Таблица 5. Количество итераций метода BiCGStab с различными предобуславливателями при решении СЛАУ на GPU

Матрица	Количество итераций						
	ILU(0)		CPR (AMG + ILU(0))		CPR* (AMG + ILU(0)*)		
	1 GPU	2 GPU	1 GPU	2 GPU	1 GPU	2 GPU	Количество блоков
kslv	19	19	5,5	5,5	5	5	256
imsh	1,5	1,5	1,5	1,5	1,5	1,5	64
imms	20,5	21	10,5	10,5	10,5	10,5	64
krrv	24,5	24	3,5	3,5	3,5	3,5	128
fdrv	4	4	1,5	1,5	1,5	1,5	256

В таблице 6 приведены времена решения СЛАУ на 1-2 GPU при использовании различных предобуславливателей: ILU(0), CPR и CPR*, в рамках которого реализуется комбинированный подход при работе с ILU(0) для извлечения дополнительного параллелизма. Видно, что снижение количества итераций, отмеченное выше при использовании CPR, преимущественно приводит к снижению времени, затрачиваемого на 1 GPU на решение СЛАУ (построение предобуславливателя и итерационный процесс). Исключением является матрица imsh, для которой использование CPR приводит к замедлению решения СЛАУ, т.к. построение предобуславливателя CPR является достаточно трудоемкой операцией, в то время как для обеспечения хорошей сходимости метода (1,5 итерации) достаточно предобуславливателя ILU(0).

Таблица 6. Время решения СЛАУ на GPU методом BiCGStab с различными предобуславливателями

Матрица	Время решения СЛАУ, с						
	ILU(0)		CPR (AMG + ILU(0))		CPR* (AMG + ILU(0)*)		
	1 GPU	2 GPU	1 GPU	2 GPU	1 GPU	2 GPU	Число блоков
kslv	0,65	0,41	0,36	0,29	0,30	0,25	256
imsh	0,17	0,10	0,33	0,26	0,32	0,25	64
imms	1,81	1,12	1,44	1,07	1,39	1,05	64
krrv	4,04	2,34	1,22	0,92	1,16	0,89	128
fdrv	1,21	0,71	1,13	0,87	1,03	0,83	256

Ускорение решения СЛАУ с предобуславливателями CPR и CPR* на 2 GPU относительно 1 GPU составляет от 1,2 до 1,34 раза. Относительно низкая масштабируемость связана с тем, что значительную часть от общего времени решения СЛАУ (не менее 24%) при расчете на 1 GPU занимает классический алгебраический многосеточный метод, реализация которого в библиотеке AmgX не позволяет загрузить несколько GPU из многопоточной программы. Применение комбинированного подхода к распараллеливанию ILU(0) в рамках CPR позволяет снизить время решения СЛАУ на 3-16% для одного GPU и 3-12% для двух GPU.

4. Заключение

Продемонстрированная в работе производительность решения СЛАУ, возникающих в ходе численного решения уравнений многофазной фильтрации потоков углеводородов в пористой среде, позволяет говорить о возможности применения GPU NVIDIA в рамках решения задачи гидродинамического моделирования нефтегазовых месторождений. В целях повышения масштабируемости решателя на системах с несколькими GPU в перспективе планируется разработка MPI-версии, которая обеспечит возможность параллельной работы алгебраического многосеточного метода на нескольких GPU.

Литература

1. Борщук О. С. О модификации двухступенчатого метода предобуславливания при численном решении задачи многофазной фильтрации вязкой сжимаемой жидкости в пористой среде // Вестник УГАТУ, 2009. Т. 12, № 1, с. 146–150.
2. Богачев К.Ю. Эффективное решение задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси на параллельных ЭВМ: дис. ... доктора физико-математических наук: 05.13.18 М., 2012. – 201 С.
3. Saad Y. Iterative methods for sparse linear systems. 2nd ed. Philadelphia, PA, USA: SIAM, 2003. — 528 p.
4. Богачев К. Ю., Жабицкий Я. В. Блочные предобусловливатели класса ILU для задач фильтрации многокомпонентной смеси в пористой среде // Вестник МГУ. Серия 1, Математика. Механика, 2009, № 5, с. 19–25.
5. Wallis J. R., Kendall R.P., Little T. E. Constrained residual acceleration of conjugate residual methods // SPE 13536, 1985, p. 415–428.
6. Богачев К. Ю., Горелов И. Г. Применение параллельного предобуславливателя CPR к задаче фильтрации вязкой сжимаемой жидкости в пористой среде // Вычислительные методы и программирование: НИВЦ МГУ, 2008. Т. 9, с. 184–190.
7. Ruge J. W., St¨uben K. Algebraic multigrid (AMG) // Multigrid Methods / S. F. McCormick (ed.) — Philadelphia, PA, USA: SIAM, 1987. Vol. 3, p. 73–130.
8. Saad Y., Li R. GPU-accelerated preconditioned iterative linear solvers // The Journal of Supercomputing, February, 2013. Vol. 63, no. 2, p. 443–466.
9. Naumov M. Parallel Solution of Sparse Triangular Linear Systems in the Preconditioned Iterative Methods on the GPU: Technical Report NVR-2011-001 NVIDIA, June, 2011.
10. Naumov M. Incomplete-LU and Cholesky Preconditioned Iterative Methods Using CUSPARSE and CUBLAS: Technical Report NVR-2012-003 NVIDIA, May, 2012.
11. Богачев К. Ю., Богатый А. С., Лапин А. Р. Использование графических ускорителей и вычислительных сопроцессоров при решении задачи фильтрации // Вычислительные методы и программирование: НИВЦ МГУ, 2013. Т. 14, с. 357–361.
12. Капорин И. Е., Коньшин И. Н. Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки // Журнал вычислительной математики и математической физики, 2000. Т. 41, № 4, с. 515–528.
13. Ахметшин Р. А., Газизов И. И., Юлдашев А. В. Комбинированный подход к построению параллельного предобуславливателя для решения задачи фильтрации углеводородов в пористой среде на графических процессорах.
URL: http://2014.nscf.ru/TesisAll/6_Supercomputerniy_enginiring/08_215_YuldashevAV.pdf (дата обращения: 15.06.2015).
14. Газизов И.И., Юлдашев А.В. Исследование эффективности распараллеливания решения некоторых разреженных СЛАУ на многоядерных процессорах // Научный сервис в сети Интернет: все грани параллелизма: Труды Международной суперкомпьютерной конференции (23-28 сентября 2013 г., г. Новороссийск). – М.: Изд-во МГУ, 2013. – С. 150-157. (Электронное издание)

Development of parallel linear solver for reservoir simulation on hybrid computing systems with GPUs

Arthur Yuldashev, Ratmir Gubaidullin and Nikita Repin

Keywords: graphics processors, sparse linear systems, multi-and many-core systems, parallel computing, reservoir simulation

Problem of hydrodynamic modeling of oil and gas fields is characterized by high consumption of computing resources. The present work aims to accelerate reservoir simulations through the use of hybrid computing systems with graphics processors, mainly to accelerate linear solver for numerical simulations of multiphase filtration.