Развитие параллельной версии прикладного пакета моделирования течения углеводородов в пласте NGT BOS

О.С. Борщук, И.Ф. Сайфуллин, М.Л. Хаит

В статье описан подход к созданию гибридной параллельной версии с использованием многоядерных архитектур. Приведены результаты многопоточной параллельной версии с использованием OpenMP, а также параллельной версии на основе одномерной декомпозиции по пространству с использованием библиотеки MPI. Кроме того, описаны возможности применения альтернативных параллельных архитектур от NVIDIA и AMD.

1. Введение

Современные технологии планирования, управления и контроля над разработкой нефтегазовых месторождений, направленные на оптимизацию процессов поиска, разведки и добычи, достижение максимального коэффициента извлечения нефти базируются на использовании гидродинамических моделей месторождений. Создание таких моделей и их практическое применение является сейчас обязательным требованием недропользователей при создании проектной документации и анализе разработки.

Современный этап развития вычислительной техники (появление многоядерных компьютеров и высокопроизводительных вычислительных кластерных систем) позволили поновому подойти к задаче моделирования процессов добычи углеводородов. Во-первых, появилась возможность рассмотрения сложных математических моделей, приводящих к вычислительным задачам большой размерности (с большим числом расчетных ячеек). Во-вторых, стало реальным проведение многовариантных расчетов с целью выбора наиболее оптимальных параметров модели.

К сожалению, до настоящего времени российские нефтяные компании, как правило, используют западные программные комплексы гидродинамического моделирования, которые разработаны ведущими фирмами: Schlumberger (Eclipse), LandMark (VIP), Roxar (TEMPEST MORE). Несмотря на все известные преимущества этих программных продуктов, ни один из них в полной мере не отвечает потребностям российского рынка. Другой отрицательной стороной западных программных комплексов является игнорирование ими передовых отечественных технологий и специфики сложившейся российской производственной практики разведки и разработки месторождений.

2. Постановка задачи

В основе моделирования многофазных фильтрационных потоков в пористой среде лежит система уравнений в частных производных, описывающая распределение давлений и насыщенностей фаз в пласте, дополненная соответствующими начальными и граничными условиями [1–4].

$$\frac{\partial}{\partial t}\left(\phi\rho_w S_w\right) - \nabla\left(\mathbf{K}\frac{k_{rw}}{\mu_w}\rho_w\left[\nabla\left(p + p_{cow}\right) - \rho_w g\nabla D\right]\right) = q_w, \tag{1}$$

$$\frac{\partial}{\partial t}\left(\phi\rho_o\left[1 - S_w\right]\right) - \nabla\left(\mathbf{K}\frac{k_{ro}}{\mu_o}\rho_o\left[\nabla p - \rho_o g\nabla D\right]\right) = q_o, \tag{2}$$

$$p(\mathbf{x},\, t) = \tilde{p}(t), \quad \mathbf{x} \in \partial\Omega, \tag{3}$$

$$S_w(\mathbf{x},\, t) = \tilde{S}_w(t), \quad \mathbf{x} \in \partial\Omega, \tag{4}$$

$$p(\mathbf{x},\, 0) = p^0(\mathbf{x}), \tag{5}$$

$$S_w(\mathbf{x},\, 0) = S_w^0(\mathbf{x}), \tag{6}$$

где t — время, $\phi = \phi(p)$ — пористость, S_w — насыщенность воды, $\rho_w = \rho_w(p,\, S_w)$, $\rho_o = \rho_o(p)$ — плотности воды и нефти,

$$\mathbf{K} = \begin{bmatrix} k_{xx} & 0 & 0 \\ 0 & k_{yy} & 0 \\ 0 & 0 & k_{zz} \end{bmatrix}$$

— диагональный тензор абсолютной проницаемости, $k_{rw} = k_{rw}(S_w)$, $k_{ro} = k_{ro}(S_w,)$ — относительные фазовые проницаемости воды и нефти, $\mu_w = \mu_w(p, S_w)$, $\mu_o = \mu_o(p)$ — вязкости воды и нефти, p — давление, $p_{cow} = p_{cow}(S_w)$ — капиллярное давление между нефтяной и водной фазой, g — ускорение свободного падения, D — глубина, Ω — область моделирования и q_w, q_o — массовые плотности источников воды и нефти.

Отметим, что уравнения (1, 2) представляют собой уравнения сохранения массы для воды и нефти, уравнения (3, 4) — граничные условия первого рода, а уравнения (5, 6) — начальные условия.

Отметим так же, что записанная 2-х фазная система может быть легко продолжена на трехфазный случай [1].

В зависимости от выбора схемы дискретизации системы дифференциальных уравнений по времени можно получить один из следующих методов:

- IMPES — схема, неявная по давлению и явная по насыщенностям, один неизвестный параметр на ячейку;
- Fully Implicit — полностью неявная схема, два – три неизвестных параметра на ячейку;
- AIM — адаптивный неявный метод, один – три неизвестных параметра на ячейку.

Используемая вычислительная схема базируется на пространственной дискретизации дифференциальных уравнений в рамках блочно-центрированной прямоугольной сетки с использованием метода конечного объема.

Для численного решения линеаризованной системы могут применяться различные методы решения систем линейных алгебраических уравнений (в частности, реализованные в стандартных библиотеках) [5, 6].

Размерность решаемых систем определяется как количеством рассматриваемых расчетных ячеек, так и количеством неизвестных в одной ячейке. Если 5 – 6 лет назад средний размер решаемой в процессе моделирования системы алгебраических уравнений составлял около 1,5 млн. неизвестных, то в настоящее время используются модели, размерность которых достигают 100 млн. неизвестных.

На данный момент завершена разработка:

- параллельной версии Fully Implicit-схемы моделирования фильтрационных течений на основе технологии MPI;
- параллельной версии Fully Implicit-схемы на основе технологии OpenMP.

3. OpenMP-версия Fully Implicit-схемы

Первым шагом стало создание параллельной версии Fully Implicit-схемы, ориентированной на SMP-системы. Причина этого выбора — смена ориентации развития современных процессоров. Рост тактовой частоты отдельных ядер фактически прекратился, прирост производительности обеспечивается наращиванием количества ядер в процессоре. Для использования дополнительных ядер приложение должно быть многопоточным. Применение технологии OpenMP позволило легко создать кросс-платформенный многопоточный код. Были подвергнуты модернизации алгоритмы расчета свойств ячеек, соединений, а также решатель СЛАУ. Для повышения эффективности распараллеливания был применен двухступенчатый предобуславливатель. На первом этапе предобуславливания полная система (включающая неизвестные по давлению и насыщенностям) приводится к системе на давление и решается с использованием алгебраических многосеточных методов (AMG) [8]. На втором этапе полученное приближение уточняется с помощью неполного LU-разложения. Так как первый шаг предобуславливания может быть эффективно распараллелен, а второй шаг (неполное LU-разложение, последовательный код) занимает менее 10% от общего времени решения, то время выполнения последовательного кода в параллельной программе значительно снижается (на 30% по сравнению со стандартным одноступенчатым предобуславливателем). В результате работы было распараллелено 80% программного кода (по времени его выполнения), соответственно по закону Амдаля максимальное теоретическое ускорение на двух потоках составило бы 1,6. На практике был достигнут максимум, равный 1,57. На двухпроцессорном узле кластера с независимыми шинами доступа к памяти были получены следующие результаты (Таблица 1).

Количество потоков	1	2	4	8
Ускорение	1	1,57	1,95	2,05

Таблица 1. Достигнутое ускорение на системе с двумя четырехядерными процессорами Intel Xeon 5345 при расчете двухфазной модели SPE10 (1 100 000 расчетных ячеек)

Заметим, что при параллельном выполнении двух и более потоков OpenMP-программы на многоядерных процессорах архитектуры Intel-AMD ускорение данного класса задач значительно падает из-за совместного использования общей шины доступа к памяти. Так, тестирование показало, что одновременное выполнение двух однопоточных программ с идентичными входными данными происходит в лучшем случае лишь в 1,6 раз быстрее, нежели их поочередное выполнение, а максимальное достигнутое ускорение при выполнении двух потоков OpenMP программы составляет 1,57. Полученные результаты показывают, что для данного класса задач узким местом является пропускная способность шины памяти. Если текущий вектор развития архитектуры Intel-AMD — кратное увеличение числа ядер в процессоре без соответствующего роста производительности шины процессор-память — сохранится, то реальная производительность, в отличие от пиковой, будет увеличиваться незначительно.

4. MPI-версия Fully Implicit-схемы

Во-первых, разработка параллельной версии для систем с распределенной памятью позволила дистанцироваться от проблем эффективности архитектуры и значительно ускорить расчет моделей. Во-вторых, применение MPI-версии при наличии достаточного количества вычислительных ресурсов открывает возможность расчета модели любого размера.

Это обеспечивается использованием принципа параллелизма по данным, что позволяет провести пространственную декомпозицию расчетной области. Декомпозиция производит-

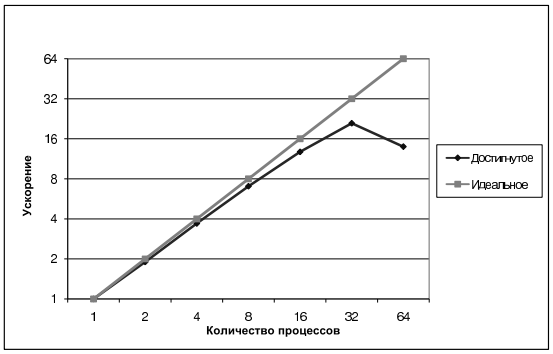

Рис. 1. Результаты MPI-версии

ся таким образом, чтобы количество активных ячеек на каждом процессе было примерно равным, с перекрытием сеток на соседних процессах для осуществления межпроцессорных обменов. К настоящему моменту реализована одномерная модель декомпозиции. Данный этап определяет эффективность всего дальнейшего параллельного счета, так как неравномерное разбиение исходной сетки влечет дисбаланс вычислительной нагрузки на узлы многопроцессорной системы. Далее запускается эволюционный процесс, на каждом шаге которого параллельно выполняется формирование матрицы системы линейных алгебраических уравнений, решение системы итерационным методом, вычисление новых приближений, а также пересылка рассчитанных значений между соседними процессами. Рассмотрим более внимательно каждый из вычислительных блоков.

Линейная система, получаемая в результате применения метода конечных объемов, имеет на декартовой сетке не более семи ненулевых элементов в каждой строке. В последовательной программе на формирование матрицы системы уходит около 10 – 20% всего времени счета. В параллельной версии программы каждый процесс формирует свою часть глобальной матрицы. Формат хранения матриц в параллельной версии отличается от такового в последовательной версии. Данные, относящиеся ко внутренним ячейкам, и данные по ячейкам, граничащим с соседними областями декомпозиции, хранятся и нумеруются отдельно. Кроме того, хранится дополнительная служебная информация, определяющая схему обмена данными между всеми процессами. Данный подход позволяет минимизировать объем пересылаемых данных, обрабатывать внутренние данные независимо и ускорять обработку данных, присланных с внешних процессов. Решение линейной системы производится методом обобщенных минимальных невязок (GMRES) с применением двухступенчатого предобуславливателя. Первая ступень предобуславливателя довольно хорошо поддается распараллеливанию [9, 10]. Как было отмечено ранее, алгоритм ILU(0) является существенно последовательным, поэтому была использована алгоритмически неэквивалентная параллельная модификация, практически не ухудшающая скорость сходимости.

5. Результаты MPI-версии

Для тестирования была выбрана двухфазная модель с сильно анизотропным распределением пористости и проницаемости породы. Общее число расчетных ячеек около 4 500 000. Модель содержит четыре добывающие и одну нагнетательную скважины.

Тестирование проводилось на кластере Уфимского государственного авиационного технического университета. Результаты ускорения показаны на (Рис. 1).

На данном тесте наблюдается близкий к линейному рост ускорения до 32 процессов, дальнейшее ухудшение объясняется критическим уменьшением расчетной области для каждого процесса с сохранением объема пересылок. Данная проблема может быть решена реа-

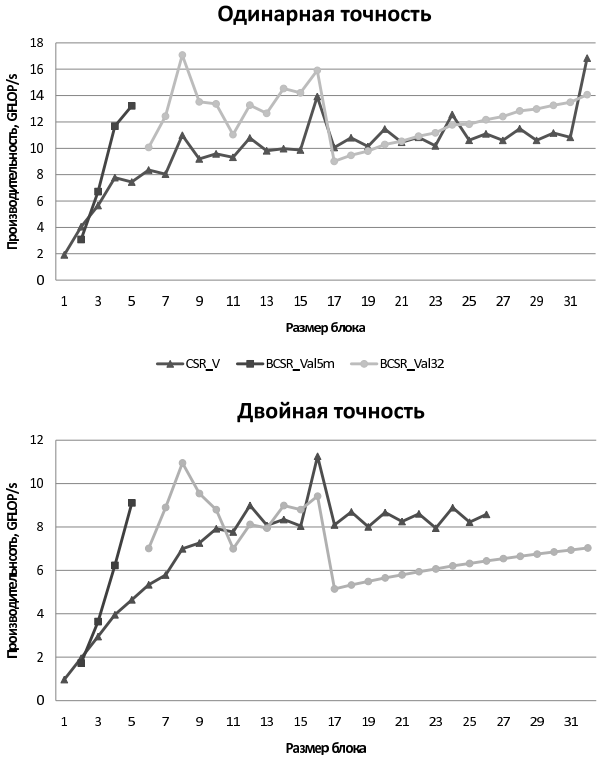

Рис. 2. Производительность методов умножения разреженной матрицы на вектор

лизацией двух- и трехмерной декомпозиции расчетной области и более совершенными механизмами балансировки нагрузки.

6. Опыт применения расчетов на GPU

Операция умножения разреженной матрицы на вектор является основной для итерационных методов решения систем линейных уравнений, применяемых в задачах моделирования. Как правило, производительность этой операции определяет производительность всего линейного решателя. Поэтому именно этот алгоритм был выбран для проведения опыта использования расчетов на GPU с использованием технология CUDA фирмы NVidia. Задачей эксперимента стала эффективная реализация умножения матрицы, хранящейся в формате Block CSR [11] с размерами блока до 32 на вектор соответствующей длины с одинарной и двойной точностью.В качестве отправной точки был использован исходный код приложения spmv [12], тестирующего производительность методов умножения разреженных матриц, хранящихся в различных форматах (в том числе в CSR), на вектор, в рамках технологии CUDA .

Особенностью архитектуры является сравнительно большое число возможных реализаций алгоритма, производительность которых зачастую можно оценить лишь на практике. В результате эксперимента были разработаны несколько реализаций, по-разному эффективных для различных размеров блоков матрицы. Ниже приведено сравнение производительности двух наиболее удачных разработанных реализаций (BCSR_Val5m для блоков от 2 до 5 и BCSR_Val32 для блоков от 6 до 32) и реализации CSR_V, предложенной в spmv и работающей с неблочным форматом CSR.

Тест проводился на базе матрицы с 28 561 строками и 173 567 ненулевыми элементами (6,07 ненулевых элемента в строке), полученной в ходе расчета реального месторождения. Под ненулевым элементом подразумевается блок соответствующего размера. По-

лученные результаты показывают, что производительность рассматриваемых реализаций в значительной степени зависит от размера блока матрицы. Абсолютные значения производительности достигают 17 GFlop/s в одинарной и 11 GFlop/s в двойной точности и превосходят существующую реализацию CSR_V при размерах блоков от 3 до 10. Необходимо учитывать, что для выполнения алгоритма на GPU необходимо сначала скопировать на устройство матрицу, а после умножения забрать результаты. Однако при использовании итерационных решателей возникает ситуация, когда одну и ту же матрицу необходимо умножать на различные вектора. В этом случае, по предварительным расчетам, реализация алгоритма умножения разреженной матрицы на вектор становится эффективной, если понадобится не менее 5 – 10 таких умножений.

В целом использование технологии CUDA, при условии переноса на GPU линейного решателя целиком, является весьма перспективным.

7. Заключение

Представленные в работе результаты являются промежуточными. Дальнейшее развитие связано во-первых с реализацией MPI-версии всего программного кода, во-вторых с объединением MPI версии и OpenMP версии, в-третьих реализация возможности выполнения линейного решателя на GPU.

Литература

1. Азиз Х., Сеттари Э. Математическое моделирование пластовых систем - М.: Недра, 1982. - 408 с.
2. Ertekin T., Aboy-Kassem J.H., King G.R. Basic applied reservoir simulation - Richardson, Texas, 2001. - 406 p.
3. Economides M.J., Nolte K.G. Reservoir Simulation - Prentice Hall, Eglewood Cliffs, New Jersey 07632, 1989. - 430 p.
4. Баренблатт Г.И., Ентов В.М., Рыжик В.М Движение жидкостей и газов в пористых пластах - М.: Недра, 1984. - 208 с.
5. Деммель Дж., Вычислительная линейная алгебра. Теория и приложения - М.: Мир, 2001. - 430 с.
6. Saad Y. Iterative methods for sparse linear systems - PWS Publishing Co., Boston, 1996.
7. Tuminaro R.S. Official Aztec User's Guide: Version 2.1 - USA, 1999.
8. Stuben K., Trottenberg U., Osterlee C. Algebraic Multigrid (AMG): An Introduction with Applications - Academic Press, New York, 2000.
9. Yang U.M., Bruaset A.M., Bjorstad P., Tveito A.(Eds.) Parallel Algebraic Multigrid Methods — High Performance Preconditioners // Numerical Solution of Partial Differential Equations on Parallel Computers -Vol. 51 - Springer Berlin Heidelberg, 2006 -P. 209–236.
10. Sterck H.D., Yang U.M., Jeffrey J.H. Reducing complexity in parallel algebraic multigrid preconditioners // SIAM Journal on Matrix Analysis and Applications -Vol. 27 - USA, 2006 -P. 1019–1039.
11. http://www.cs.utk.edu/~dongarra/etemplates/node375.html
12. Nathan Bell and Michael Garland. Efficient Sparse Matrix-Vector Multiplication on CUDA. NVIDIA Technical Report NVR-2008-004, December, 2008