

УДК 519.683.4

ТРЕХУРОВНЕВЫЙ MPI+NUMA+THREADS МЕТОД ПОСТРОЕНИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ РАСЧЕТА ЗАДАЧ ГИДРОДИНАМИЧЕСКОГО МОДЕЛИРОВАНИЯ НА КЛАСТЕРНЫХ СИСТЕМАХ С УЗЛАМИ С НЕОДНОРОДНЫМ ДОСТУПОМ К ПАМЯТИ

К. Ю. Богачев¹, Я. В. Жабицкий¹, А. А. Климовский¹,
А. Р. Миргасимов¹, А. Е. Семенко¹

Рассматривается параллельная реализация программы решения задачи трехфазной фильтрации вязкой сжимаемой жидкости на кластерных системах с распределенной памятью, узлы которых имеют разное время доступа процессоров к различным участкам своей памяти. Традиционным подходом построения параллельных программ на кластерных системах с распределенной памятью является использование библиотеки MPI. В силу специфики решаемой задачи приходится иметь дело с неструктурированной расчетной сеткой и моделированием динамически меняющегося набора скважин, проходящих через значительное число блоков сетки. Это вносит ряд проблем при использовании традиционного подхода: дисбаланс загруженности вычислительных узлов, увеличение объема пересылок между MPI-процессами, увеличение используемого объема оперативной памяти и др. Предлагается трехуровневый MPI+NUMA+Threads-метод построения параллельной программы, целью которого является решение указанных проблем. В основе метода лежит идея переноса архитектуры узла кластера (многоядерность и неоднородный доступ к общей памяти узла) на архитектуру параллельной программы. Выполнено сравнение программ, реализованных с использованием предложенного метода и традиционного подхода, с точки зрения скорости работы и объема используемой оперативной памяти. Приведены результаты численных экспериментов на большом количестве реальных задач.

Ключевые слова: высокопроизводительные вычисления, гибридные MPI-многопоточные программы, NUMA-системы, балансировка загруженности, задача фильтрации.

1. Введение. До 2007 г. каждый узел вычислительного кластера содержал не более двух процессорных ядер. Начиная с 2007 г. в списке наиболее мощных суперкомпьютеров Top500 стало появляться все больше кластерных систем с четырьмя и более процессорными ядрами на узел [1]. В 2008 г. были разработаны процессоры Intel Nehalem со встроенным контроллером памяти. Стало ясно, что это приведет к тому, что многопроцессорные вычислительные узлы кластерных систем на базе процессоров Intel станут NUMA-системами (Non-Uniform Memory Access). На текущий момент самым производительным в мире таким суперкомпьютером является SuperMUC, занимающий четвертую позицию в Top500. Его вычислительную основу составляют 9216 узлов, каждый из которых является NUMA-системой, состоящей из двух восьмиядерных процессоров Intel Xeon E5-2680.

Из сказанного следует, что архитектура отдельного вычислительного узла кластера значительно усложнилась — число ядер внутри одного процессора достигло 8 и более, а доступ к оперативной памяти стал неоднородным. Это не могло не привести к тому, что в некоторых трудоемких задачах традиционный метод составления параллельных программ с использованием лишь библиотеки MPI оказался не столь эффективным в случае их запуска на современных кластерных системах. В связи с этим стала расти популярность гибридной технологии MPI+OpenMP составления параллельных программ, которая нацелена на использование общей памяти в рамках одного узла кластера. Однако и у этой технологии имеются свои недостатки — отсутствие гибкости в управлении вычислительными процессами, в частности: на каких ядрах будут выполняться вычислительные потоки и как память программы будет распределена между NUMA-узлами. Кроме того, появляются существенные накладные расходы на создание/удаление вычислительных потоков.

¹ Московский государственный университет им. М. В. Ломоносова, механико-математический факультет, Ленинские горы, д. 1, 119899, Москва; К. Ю. Богачев, доцент, e-mail: bogachev@mech.math.msu.su; Я. В. Жабицкий, аспирант, e-mail: jjv@fromru.com; А. А. Климовский, аспирант, e-mail: arseny@klimovsky.ru; А. Р. Миргасимов, аспирант, e-mail: mirgasimov.almaz@gmail.com; А. Е. Семенко, аспирант, e-mail: a semenko@gmail.com

В настоящей статье представлен новый метод MPI+NUMA+Threads построения параллельного программного кода на кластерных системах, узлы которых имеют неоднородный доступ памяти. В его основе лежит отображение сложной архитектуры кластера вплоть до отдельного процессорного ядра на архитектуру самой программы. Три уровня иерархии кластерной системы (вычислительные узлы, NUMA-узлы и процессорные ядра) соответствуют трем различным технологиям построения параллельного кода (MPI, NUMA и Threads), что позволяет программе эффективно учитывать все особенности доступа к оперативной памяти используемой кластерной системы.

2. Постановка задачи фильтрации. Рассматриваются стандартные уравнения многофазной изотермической модели черной нефти в трехмерной области [2]:

$$\begin{aligned} \frac{\partial}{\partial t}(\phi N_c) &= \operatorname{div} \sum_{l=o,w,g} x_{cl} \xi_l \left(\mathbf{k} \frac{k_{rl}}{\mu_l} (\nabla p_l - \gamma_l \nabla D) \right) + q_c, \quad c = 1, \dots, n_c, \\ p_o - p_g &= P_{c,og}, \quad p_o - p_w = P_{c,ow}, \quad S_w + S_o + S_g = 1, \\ N_w &= \frac{S_w}{B_w}, \quad N_o = \frac{S_o}{B_o} + R_{og} \frac{S_g}{B_g}, \quad N_g = \frac{S_g}{B_g} + R_{go} \frac{S_o}{B_o}, \end{aligned} \quad (1)$$

где n_c — количество компонентов в смеси; $R_{og} = R_{og}(p_g)$ — летучесть нефти в газовой фазе; $R_{go} = R_{go}(p_o)$ — растворимость газа в нефтяной фазе; $B_l = B_l(p_l)$ — коэффициент объемного расширения фазы l , где $l = o$ (oil), g (gas), w (water); $\phi = \phi(p_w, p_o, p_g, x, y, z)$ — пористость среды; $x_{cl} = x_{cl}(p_w, p_o, p_g, \mathbf{N})$ — молярная доля компонента c , $c = 1, \dots, n_c$, в фазе l ; $\mathbf{N} = (N_1, \dots, N_{n_c})$ — вектор молярных плотностей; $\xi_l = \xi_l(p_l, \mathbf{N})$ — молярная плотность фазы l ; $\mathbf{k} = \mathbf{k}(p_w, p_o, p_g, x, y, z)$ — тензор абсолютной проницаемости; $k_{rl} = k_{rl}(S_w, S_g)$ — относительная проницаемость фазы l ; $\mu_l = \mu_l(p_l)$ — вязкость фазы l ; $\gamma_l = \rho_l g$ — вертикальный градиент гидростатического давления в фазе l ; g — гравитационная постоянная; $D = D(x, y, z)$ — вектор глубины (сверху вниз); $\rho_l = \rho_l(p_l)$ — массовая плотность фазы l ; $P_{c,og} = P_{c,og}(S_g)$ — капиллярное давление в системе нефть–газ; $P_{c,ow} = P_{c,ow}(S_w)$ — капиллярное давление в системе вода–нефть; $q_c = q_c(p_w, p_o, p_g, \mathbf{N}, t, x, y, z)$ — источник компонента c (скважина).

Неизвестными в этой системе уравнений являются:

- 1) $N_c = N_c(t, x, y, z)$ — молярная плотность компонента c (для модели черной нефти компонентами служат вода, нефть и газ),
- 2) $S_l = S_l(t, x, y, z)$ — насыщенность фазы l ,
- 3) $p_l = p_l(t, x, y, z)$ — давление в фазе l .

Для этой системы уравнений задаются начальные условия, а также на внешней границе резервуара ставятся условия непротекания.

3. Численное решение задачи фильтрации. Для аппроксимации системы (1) по времени используется полностью неявная схема. Аппроксимацией по пространственным переменным методом конечных объемов на неравномерной сетке первоначальная задача сначала сводится к системе нелинейных алгебраических уравнений вида $F(\mathbf{p}, \mathbf{N}_1, \dots, \mathbf{N}_{n_c}) = 0$, где $\mathbf{p} = (p^i)$ и $\mathbf{N}_c = (N_c^i)$ — векторы значений давления и молярных плотностей в блоках сетки.

Для решения системы нелинейных уравнений $F(\mathbf{U}) = 0$, $\mathbf{U} \equiv (\mathbf{p}, \mathbf{N})$ используется стандартный метод Ньютона $\mathbf{U}^{m+1} = \mathbf{U}^m - \left(\frac{\partial F(\mathbf{U}^m)}{\partial \mathbf{U}} \right)^{-1} F(\mathbf{U}^m)$, где $\frac{\partial F(\mathbf{U}^m)}{\partial \mathbf{U}}$ — матрица $R^{n_c * K} \times R^{n_c * K}$ (здесь K — число блоков сетки).

На каждом шаге метода Ньютона необходимо решать систему с несимметричной матрицей $\frac{\partial F(\mathbf{U}^m)}{\partial \mathbf{U}}$, т.е. задача сводится к решению системы линейных уравнений

$$A\mathbf{x} = \mathbf{r}, \quad (2)$$

матрица которой представляет собой якобиан из метода Ньютона.

Система линейных уравнений (2) решается методом BiCGStab (Biconjugate Gradient Stabilized) [3] с использованием предобусловливателя класса ILU (Incomplete LU) [4, 5]. В работе [8] было проведено сравнение различных итерационных методов решения линейных систем на матрицах, возникающих в данной задаче. Это сравнение показало, что метод BiCGStab работает быстрее и масштабируется лучше, чем другие итерационные методы (например, ORTHOMIN, QGMRES или DQGMRES).

4. Целевая вычислительная система. На рис. 1 схематично изображен кластер с 10 вычислительными узлами, каждый из которых является NUMA-системой, состоящей из двух четырехъядерных

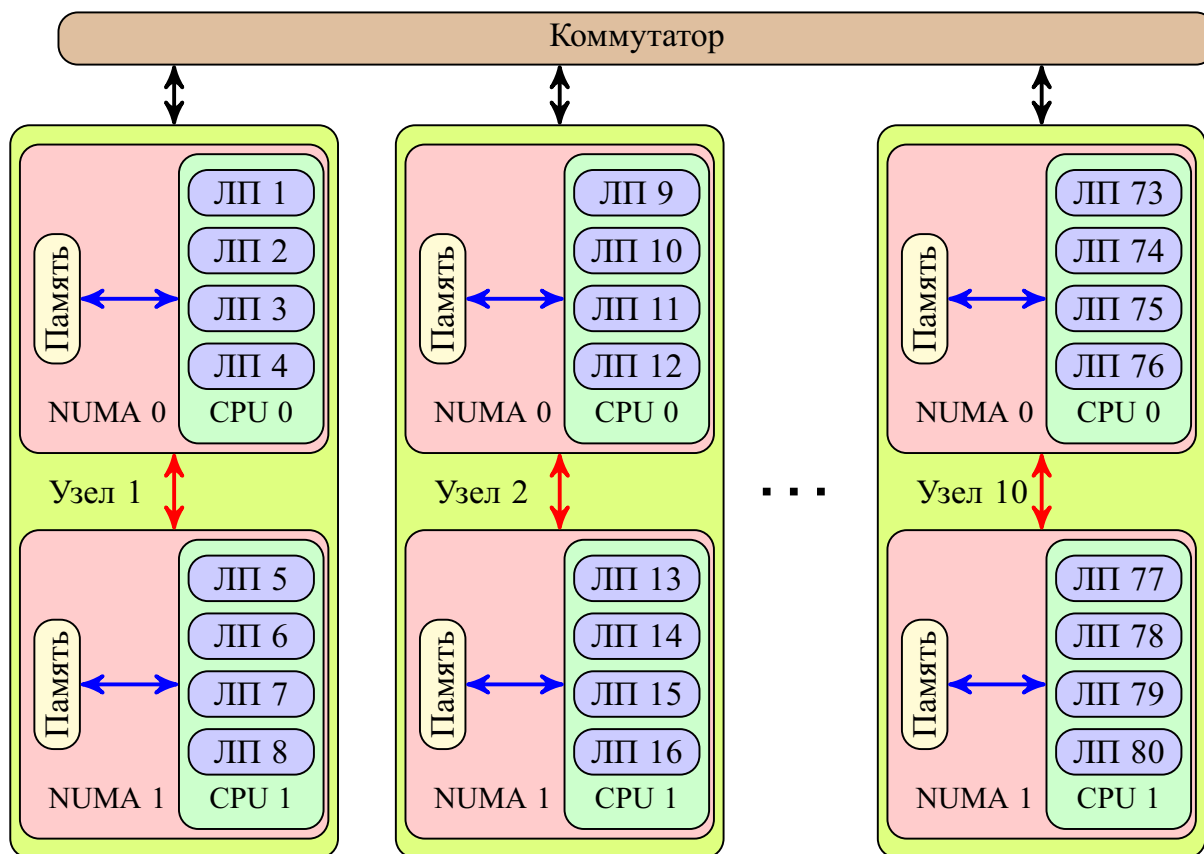


Рис. 1. Кластерная система, узлы которой имеют неоднородный доступ к памяти

процессоров. Скорость передачи данных существенно зависит от того, между какими компонентами происходит обмен:

- процессор–память: 64 ГБ/с (DDR3-2000, 4 канала);
- NUMA–NUMA: 25.6 ГБ/с (QPI);
- узел–коммутатор: 5 ГБ/с (QDR Infiniband).

Скорость доступа к оперативной памяти другого NUMA-узла ограничивается пропускной способностью высокоскоростной шины между процессорами (QPI для процессоров Intel, HyperTransport для процессоров AMD). Естественно, что при проектировании параллельной программы, которая будет исполняться на такой системе, для достижения максимальной эффективности необходимо учитывать как наличие общего доступа к оперативной памяти на вычислительных узлах, так и его неоднородность.

5. Традиционный MPI-метод и трехуровневый MPI+NUMA+Threads-метод построения параллельных программ. Рассмотрим кластерную систему с распределенной памятью, состоящую из N расчетных узлов. Каждый узел является NUMA-системой и содержит M NUMA-узлов со своим контроллером и банком памяти; каждый NUMA-узел состоит из P процессорных ядер.

Для параллельной программы, построенной с использованием лишь технологии MPI, при полной загрузке всех процессорных ядер будет характерным то, что число процессов равно общему числу процессорных ядер $N * M * P$, причем каждый процесс привязан к своему процессорному ядру.

Для параллельной программы, построенной согласно трехуровневому MPI+NUMA+Threads-методу и работающей на данной системе, будет характерным следующее:

- минимальное число MPI-процессов равно N , по одному на расчетный узел;
- максимально эффективное число потоков (threads) в каждом MPI-процессе равно $M * P$, по одному на каждое процессорное ядро;
- внутри каждого MPI-процесса используется технология оптимизации программного кода под NUMA-систему [6]; потоки разбиваются на M групп и работают на фиксированных NUMA-узлах; участки памяти программы, к которым поток обращается чаще всего, выделяются на том же NUMA-узле, где работает поток.

6. Преимущества трехуровневого MPI+NUMA+Threads-метода. Для программы, решаю-

щей поставленную задачу, требуется оперативная память для хранения различных структур данных. Среди этих структур можно выделить

- расчетную сетку;
- набор массивов для хранения различных свойств смеси и их частных производных в блоках сетки; такой набор зависит от набора физических явлений, которые заложены в модели: давление, насыщенности и плотности отдельных фаз, молярные плотности компонентов, масштабирование концевых точек фазовых кривых и др.;
- внутренние буферы для работы линейного солвера;
- структуры для решения уравнений скважин;
- структуры для управления процессом расчета, выведения отчетных данных и т.п.

В последовательной версии программы доминирует объем оперативной памяти A_1 , используемый для хранения расчетной сетки, массивов свойств и их частных производных, а также буферов линейного солвера. В этом случае наблюдается хорошая масштабируемость при увеличении числа процессов — объем этой памяти, требуемый процессу, уменьшается почти пропорционально числу MPI-процессов (за исключением границ расчетных сеток и перекрытий, используемых в предобуславливателе при решении линейных систем).

Второй объем оперативной памяти A_2 выделяется под прочие структуры, необходимые для решения уравнений скважин, управления процессом расчета, выведения отчетных данных и др. В этом случае наблюдается плохая масштабируемость или же ее отсутствие. Это связано с тем, что построение программы с целью уменьшения потребления памяти невозможно или очень трудозатратно либо приведет к уменьшению скорости работы. Таким образом, при увеличении числа MPI-процессов доля выделенной памяти под прочие структуры растет и может стать доминирующей.

Допустим, что мы запускаем программу на N узлах кластера по M MPI-процессов на узел. Тогда объем требуемой оперативной памяти на узел можно грубо оценить как $A_1/N + A_2 * M$. Поскольку в трехуровневой архитектуре программы $M = 1$, то это позволяет использовать на $A_2 * (M - 1)$ меньше оперативной памяти на узел, чем использует традиционная MPI-программа.

Не все вычисления могут быть эффективно распараллелены между MPI-процессами. Например, в поставленной задаче присутствуют такие объекты, как скважины (рис. 2). Все блоки сетки, через которые проходит траектория скважины, становятся связанными между собой. Это вносит в матрицу линейной системы дополнительные ненулевые элементы, для вычисления которых необходимо знать характеристики скважины, получаемые после решения уравнения скважин. Скважина может проходить через блоки разных процессов, поэтому каждый из этих процессов должен решить уравнение этой скважины (передать по MPI найденное решение слишком трудозатратно). Внутри же одного процесса вычислительные потоки могут поделить эту работу между собой, не пересекаясь. Каждый поток решит уравнения своей части скважин и сохранит ответ в общую память.

Использование минимально возможного числа MPI-процессов, равного числу узлов кластера, уменьшает общий размер границы сеток, что позволяет снизить объем передаваемых по MPI данных. Это влияет не только на время передачи данных, но и предотвращает преждевременное переполнение внутренних MPI-буферов при увеличении числа используемых узлов кластера.

Если на узле кластера работает лишь один MPI-процесс, то в момент исполнения последовательной части программы загружено лишь одно процессорное ядро, так как работает только один поток, а остальные ждут завершения его работы. В этом случае технология TurboBoost (автоматическое повышение частоты одного ядра при простаивании других) позволяет ускорить работу программы. Если на узле кластера работает несколько MPI-процессов (по одному на каждое ядро), то технология TurboBoost не может быть использована.

Код параллельной программы, написанный с использованием трехуровневого MPI+NUMA+Threads-метода, можно скомпилировать, отключив уровень MPI, и получить эффективную параллельную программу для запуска на многоядерной рабочей станции. Добавив к расчетному ядру программы дополни-

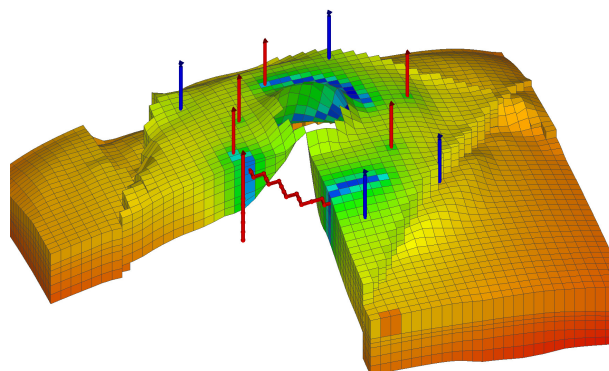


Рис. 2. Добывающие (красные) и нагнетательные (синие) скважины

тельные потоки для визуализации, можно получить интерактивный графический интерфейс, в котором можно наблюдать и контролировать процесс расчета. Потоки графического интерфейса разделяют адресное пространство с расчетными потоками. Это экономит объем оперативной памяти и избавляет от пересылок больших визуализируемых объектов, например карты давления во всех блоках.

Таким образом, преимущества трехуровневой архитектуры программы перед традиционной MPI-реализацией заключаются в следующем:

- уменьшается требование к объему оперативной памяти узла кластера;
- появляются гибкие возможности распараллеливания вычислений на уровне потоков за счет общего доступа ко всей памяти процесса;
- уменьшается объем пересылок данных между MPI-процессами и снижается нагрузка на внутренние MPI-буферы;
- эффективно используется технология TurboBoost;
- универсальность программного кода, т.е. появляется возможность компиляции различных версий программы под различные целевые вычислительные системы (кластерные системы или рабочие станции с общей памятью).

7. Практическое применение трехуровневого MPI+NUMA+Threads-метода построения параллельных программ. Рассмотрим применение предложенного метода к реализации параллельной программы численного решения задачи (1) фильтрации вязкой сжимаемой многофазной смеси в анизотропной пористой среде. Программа рассчитана на запуск на кластерных системах с распределенной памятью, узлы которых могут являться NUMA-системами. Программа реализована в рамках гидродинамического пакета моделирования tNavigator.

Для построения аппроксимации системы дифференциальных уравнений фильтрации (1) вводится исходная сетка, учитывающая геологическое строение месторождения (разломы, выклинивания). Для решения задачи используются только данные для ячеек сетки, имеющих объем, который может занимать жидкость (так называемый поровый объем), больше определенного порогового значения. Такие ячейки сетки называются активными. Совокупность активных ячеек образует глобальную расчетную сетку.

Для организации параллельных вычислений на системах с распределенной памятью все активные ячейки сетки распределяются между MPI-процессами. Далее в каждом MPI-процессе заводится локальная расчетная сетка, которая содержит

- 1) свои ячейки, т.е. ячейки, отведенные данному процессу,
- 2) граничные ячейки, т.е. ячейки, соседние со своими.

Для построения локальных расчетных сеток используется модификация метода, описанного в работе [7]. В этом методе сначала определяется, какие ячейки будут активными, а какие нет. Для этого исходная сетка разбивается между MPI-процессами, каждый из которых вычисляет набор активных ячеек в своей части исходной сетки. Далее информация об активных ячейках собирается в каждом MPI-процессе. С целью балансировки загрузки узлов кластера во время расчета локальные расчетные сетки строятся таким образом, чтобы число своих ячеек у разных MPI-процессов было примерно одинаково.

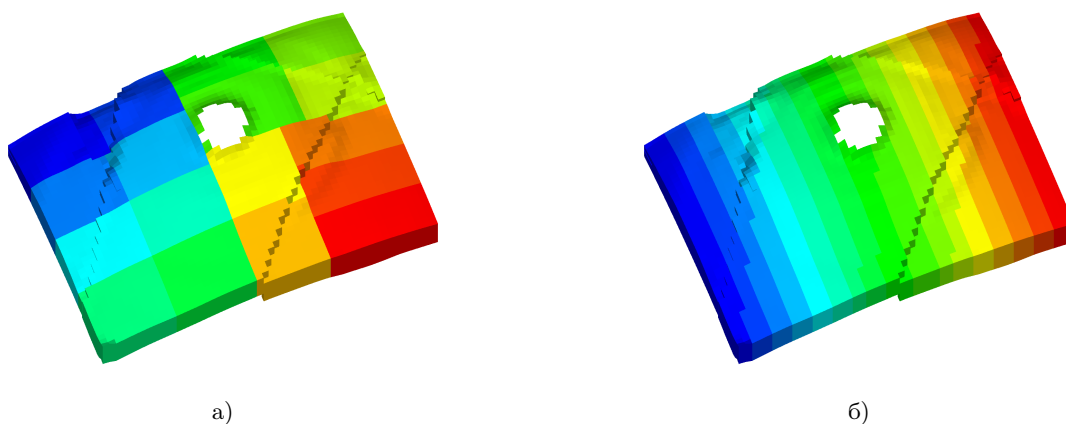


Рис. 3. Разбиение своих ячеек расчетной сетки между 17 MPI-процессами: а) с уменьшенной площадью границы локальных сеток, б) без модификаций

Цель модификации метода заключается в том, чтобы уменьшить число граничных ячеек в локальных

расчетных сетках (рис. 3). Это крайне важно, потому что при таком подходе уменьшается не только объем вычислений в каждом MPI-процессе, но и уменьшается объем пересылок данных между MPI-процессами во время решения системы линейных уравнений (2).

Практически все время работы программы уходит на исполнение следующих модулей:

- вычисление физических свойств смеси во всех блоках локальной сетки по известным давлению и молярным плотностям,
- построение матрицы A линейной системы (2),
- решение линейной системы (2).

Всю оставшуюся вычислительную работу, не входящую в эти модули, в первом приближении можно считать плохо распараллеливаемой.

Для параллельного вычисления физических свойств смеси все блоки сетки разбиваются между потоками MPI-процесса, каждый поток обрабатывает свою часть блоков. Накладными расходами является подсчет свойств смеси в граничных блоках, поэтому уменьшение общего числа граничных блоков, что и происходит при использовании MPI+NUMA+Threads-метода, ведет к ускорению работы данного модуля программы.

Далее, каждый процесс строит часть матрицы A , соответствующую своим ячейкам сетки. Свои ячейки сетки разбиваются между потоками, каждый из которых вычисляет свои строки матрицы. На данном этапе накладных расходов нет.

Линейная система (2) решается итерационным методом BiCGStab с использованием предобусловливателя класса ILU и его параллельной реализацией по методу Капорина–Коньшина [4, 5].

Для построения и применения предобусловливателя в каждом MPI-процессе необходимо

- к своим строкам матрицы присоединить дополнительные строки;
- к своим элементам правой части добавить элементы, соответствующие присоединенным строкам матрицы.

Эти дополнительные строки и элементы необходимо получать из других процессов. Уменьшение общего числа процессов за счет использования потоков снижает объем пересылок между MPI-процессами, таким образом ускоряя решение линейной системы.

Кластер	Узлы	Процессоры	Ядра	Память
Rock Flow Dynamics	20	2 x Intel Xeon X5650 2.66GHz	2 x 6	24 Gb
Суперкомпьютер “Ломоносов”	4160	2 x Intel Xeon X5570 2.93GHz	2 x 4	12 Gb

8. Вычислительные эксперименты. Проводились расчеты гидродинамических моделей реальных месторождений с различным числом активных ячеек с помощью двух версий программы (см. таблицу и рис. 4–7).

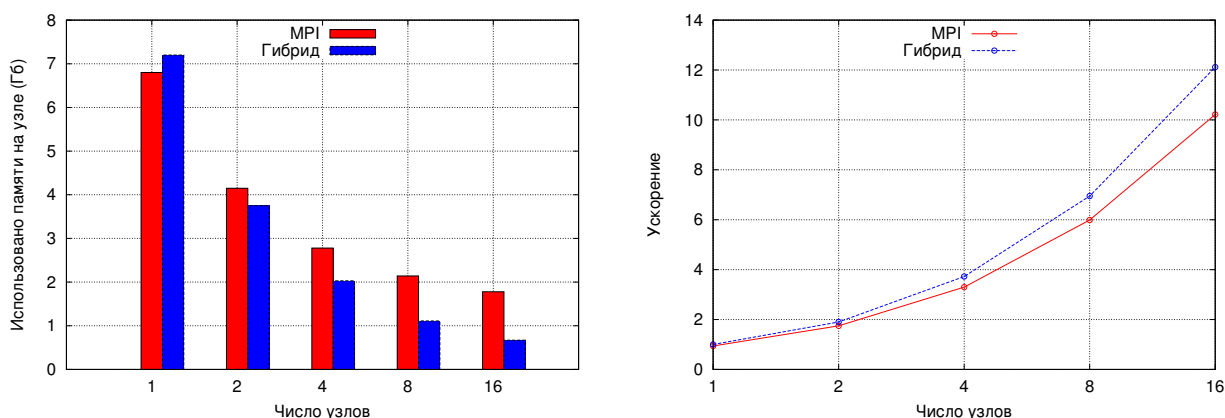


Рис. 4. Кластер Rock Flow Dynamics, число активных ячеек в модели 2418 989

В первой версии был использован традиционный подход к распараллеливанию вычислений на системах с распределенной памятью — использование лишь библиотеки MPI. Во второй версии был использован предложенный трехуровневый гибридный MPI+NUMA+Threads-метод построения параллельных программ. Эффективность подходов к программной реализации сравнивалась по двум параметрам:

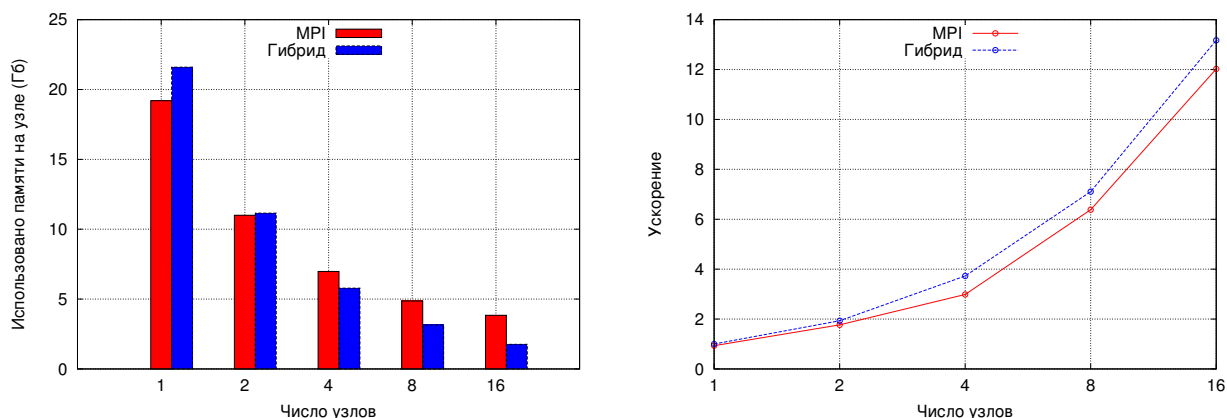


Рис. 5. Кластер Rock Flow Dynamics, число активных ячеек в модели 7 256 967

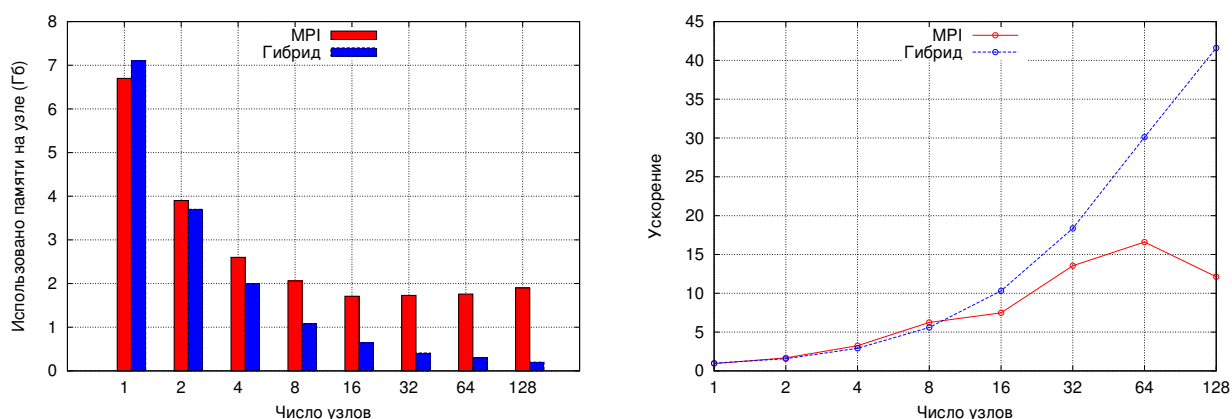


Рис. 6. Суперкомпьютер “Ломоносов”, число активных ячеек в модели 2 418 989

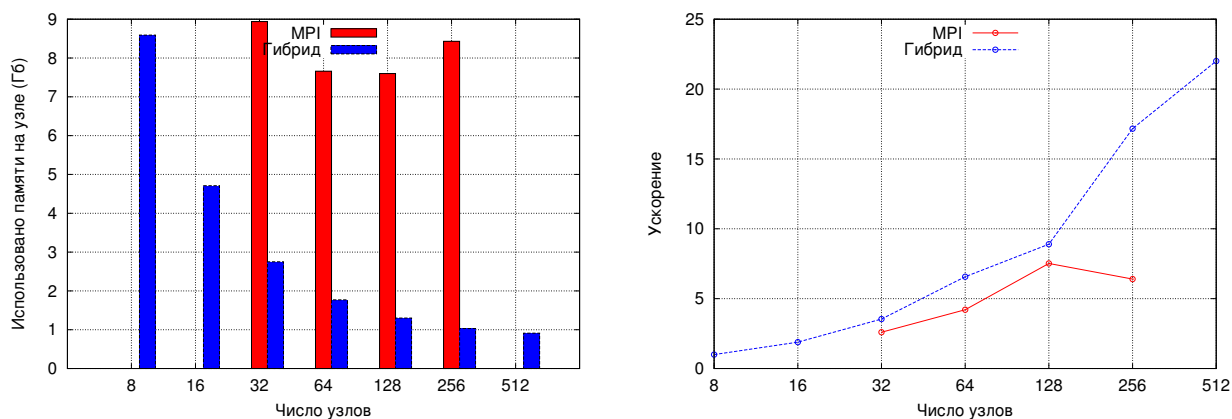


Рис. 7. Суперкомпьютер “Ломоносов”, число активных ячеек в модели 21 770 901

требуемый для расчета общий объем оперативной памяти и общее время расчета. При проведении экспериментов был использован суперкомпьютер “Ломоносов”, установленный в Научно-исследовательском вычислительном центре МГУ имени М. В. Ломоносова.

Тестовые запуски показывают, что гибридной реализации программы для работы требуется гораздо меньший объем оперативной памяти, чем в случае MPI-реализации. При использовании большого числа вычислительных узлов кластера MPI-реализация программы показывает спад коэффициента ускорения (и даже начинает замедляться) гораздо раньше, чем гибридная реализация программы.

9. Заключение. Разработан трехуровневый MPI+NUMA+Threads-метод составления параллельных программ. Этот метод был реализован в рамках гидродинамического пакета моделирования tNavigator. Программа, реализованная с использованием данного метода, была протестирована на реальных моделях

нефтяных месторождений на суперкомпьютере “Ломоносов” с использованием до 512 вычислительных узлов. Проведено сравнение производительности программ, реализованных при помощи данного метода и традиционного MPI-метода. Трехуровневый MPI+NUMA+Threads-метод показал лучшую скорость, масштабируемость и более низкие требования к оперативной памяти.

СПИСОК ЛИТЕРАТУРЫ

1. TOP500 Supercomputer Sites (<http://top500.org>).
2. *Chen Z., Huan G., Ma Y.* Computational methods for multiphase flows in porous media. Philadelphia: SIAM, 2006.
3. *Saad Y.* Iterative methods for sparse linear systems. Philadelphia: SIAM, 2003.
4. *Богачев К.Ю., Жабицкий Я.В.* Блочные предобусловливатели класса ILU для задач фильтрации многокомпонентной смеси в пористой среде // Вестн. Моск. ун-та. Сер. 1. Матем. Механ. 2009. № 5. 19–25.
5. *Богачев К.Ю., Жабицкий Я.В.* Метод Капорина–Коньшина параллельной реализации блочных предобусловливателей для несимметричных матриц в задачах фильтрации многокомпонентной смеси в пористой среде // Вестн. Моск. ун-та. Сер. 1. Матем. Механ. 2010. № 1. 46–52.
6. *Богачев К.Ю., Миргасимов А.Р.* Об оптимизации вычислительных приложений для многопроцессорных систем с общей неоднородной памятью // Вычислительные методы и программирование. 2010. **11**, раздел 1. 193–197 (<http://num-meth.srcc.msu.ru/>).
7. *Богачев К.Ю., Климовский А.А., Миргасимов А.Р., Семенко А.Е.* Балансировка загрузки узлов кластера при расчете задачи фильтрации // Вычислительные методы и программирование. 2011. **12**, раздел 1. 70–73 (<http://num-meth.srcc.msu.ru/>).
8. *Богачев К.Ю., Жабицкий Я.В., Климовский А.А., Миргасимов А.Р., Семенко А.Е.* Сравнение итерационных методов решения разреженных систем линейных уравнений в задачах фильтрации на вычислительных системах с распределенной памятью // Вычислительные методы и программирование. 2011. **12**, раздел 1. 74–76 (<http://num-meth.srcc.msu.ru/>).

Поступила в редакцию
27.05.2013
