На правах рукописи

Богачев Кирилл Юрьевич

Эффективное решение задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси на параллельных ЭВМ

05.13.18 – Математическое моделирование, численные методы и комплексы программ

АВТОРЕФЕРАТ

диссертации на соискание ученой степени

доктора физико-математических наук

Москва – 2011

Работа выполнена на *механико-математическом факультете МГУ им. М.В.Ломоносова*.

Официальные оппоненты:

д.ф.-м.н.,
профессор *Амосов Андрей Авенирович*

д.ф.-м.н.,
профессор, *Василевский Юрий Викторович*

д.ф.-м.н.,
профессор, *Победря Борис Ефимович*

Ведущая организация:
Казанский (Приволжский) федеральный университет

Защита состоится «____» _____ 2012 г. в 11 часов на заседании диссертационного совета *Д 002.024.03* при *Институте прикладной математики им. М.В. Келдыша РАН*, расположенном по адресу: *Москва, Миусская пл., 4*

С диссертацией можно ознакомиться в библиотеке *Института прикладной математики им. М.В. Келдыша РАН*.

Автореферат разослан «____» _____ 2012 г.

Отзывы и замечания по автореферату в двух экземплярах, заверенные печатью, просьба высылать по вышеуказанному адресу на имя ученого секретаря диссертационного совета.

Ученый секретарь
диссертационного совета,

д.ф.-м.н. *Змитренко Н.В*

Общая характеристика работы

Актуальность работы Разрешающая способность сеточных данных и сложность математической модели, используемой при описании фильтрационных течений многофазной многокомпонентной смеси в нефтегазовых месторождениях, постоянно растут. Это неизбежно приводит к возрастанию времени расчета при численном решении задачи моделирования процессов разработки углеводородного сырья. При этом все компании-поставщики микропроцессоров повышают производительность своей продукции в основном за счет увеличения количества вычислительных ядер. Поэтому для достижения максимальной эффективности расчетов требуется разрабатывать методы решения, максимально использующие параллельность ЭВМ. Несмотря на большие усилия в этом направлении, из-за специфики задачи фильтрации, большинство программ для ее численного решения не показывают ускорения, более чем в 12–16 раз, по сравнению с временем работы последовательной программы, а начиная с 20–30 использованных логических процессоров время расчета начинает возрастать. Таким образом, решение задачи фильтрации оказалось в стороне от развития высокопроизводительных вычислений, где речь уже идет о сотнях и тысячах процессоров.

Цель диссертационной работы состоит в создании эффективного метода расчета и соответствующего комплекса программ для решения задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси на параллельных ЭВМ.

Для достижения поставленных целей были решены следующие задачи:

- разработан метод блочного хранения разреженных несимметричных матриц в памяти и блочные варианты построения предобуславливателей, основанных на неполном LU разложении;

- разработан параллельный алгоритм построения предобуславливателей, основанных на неполном LU разложении;
- разработан метод использования ресурсов ЭВМ с неоднородным доступом к памяти (NUMA);
- разработан метод построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере).

Научная новизна работы состоит в разработке метода построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере), позволившего достигать ускорения, более чем в 50 раз, по сравнению с временем работы последовательной программы, причем сокращение времени работы идет даже при использовании более 200 логических процессоров.

Практическая значимость. Результаты, изложенные в диссертации, могут быть использованы при численном решении задачи моделирования процессов разработки углеводородного сырья, а также при решении других систем уравнений в частных производных на параллельных ЭВМ.

На защиту выносятся следующие основные результаты и положения:

- математическая модель техногенной трещиноватости вблизи скважин;
- метод блочного хранения разреженных несимметричных матриц в памяти и блочные варианты построения предобуславливателей, основанных на неполном LU разложении;
- алгоритм распараллеливания построения предобуславливателей, основанных на неполном LU разложении;

- эффективная технология, включающая алгоритмы, структуры данных и комплекс программ, для построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере).

- комплекс программ tNavigator для решения задачи фильтрации на современных параллельных ЭВМ.

Апробация работы. Основные результаты диссертации докладывались на следующих конференциях:

- Уфимская международная математическая конференция "Теория функций, дифференциальные уравнения, вычислительная математика" (Уфа, 2007)

- 4-я международная конференция "Математические идеи П.Л. Чебышева и их приложения к современным проблемам естествознания" (Обнинск, 2008)

- Научная конференция "ЛОМОНОСОВСКИЕ ЧТЕНИЯ" – 2008 (Москва, 2008)

- Всероссийская конференция по вычислительной математике КВМ-2009 (Академгородок, Новосибирск, 2009)

- Научная конференция "ЛОМОНОСОВСКИЕ ЧТЕНИЯ" – 2009 (Москва, 2009)

- Научная конференция "ЛОМОНОСОВСКИЕ ЧТЕНИЯ" – 2010 (Москва, 2010)

- Конференция "Суперкомпьютеры в нефтегазовой отрасли" (Москва, 2010)

Также основные результаты диссертации докладывались на научно-исследовательских семинарах:

- Кафедры вычислительной математики механико-математического факультета МГУ им. М.В. Ломоносова
- Института математического моделирования РАН (ИММ РАН)
- Кафедры механики композиционных материалов механико-математического факультета МГУ им. М.В. Ломоносова
- Вычислительного центра РАН (ВЦ РАН)
- Института вычислительной математики РАН (ИВМ РАН)
- Института проблем безопасного развития атомной энергетики РАН (ИБРАЭ РАН)

Разработанный в диссертационной работе комплекс программ tNavigator свободно доступен для загрузки с сайта `rfdyn.ru`.

Публикации. Материалы диссертации опубликованы в 14 печатных работах, из них 4 монографии [A1, A2, A3, A4], 8 статей в рецензируемых журналах [A5, A6, A7, A8, A9, A10, A11, A12], 2 статьи в специализированных журналах [A13, A14].

Отметим также, что без предшествующих работ [1–15], посвященных эффективным методам решения жестких нелинейных уравнений в частных производных, и монографии [16], посвященной архитектурам операционных систем, эта работа тоже была бы невозможной.

Личный вклад автора. Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Подготовка к публикации полученных результатов проводилась совместно с соавторами, причем вклад диссертанта был определяющим.

Структура и объем диссертации. Диссертация состоит из введения, обзора литературы, 4 глав, заключения и библиографии. Общий объем диссертации 183 страницы, из них 156 страниц текста, включая 57 рисунков. Библиография включает 123 наименований на 17 страницах.

Содержание работы

Во Введении обоснована актуальность диссертационной работы, сформулирована цель и аргументирована научная новизна исследований, показана практическая значимость полученных результатов, представлены выносимые на защиту научные положения.

В обзоре литературы приведена постановка задачи фильтрации, следующая классическим книгам [17], [18], [19], [20]. Система дифференциальных уравнений имеет вид

$$\begin{gathered}\frac{\partial}{\partial t}(\phi N_c) = \operatorname{div} \sum_{P=1}^{n_P} x_{c,P}\xi_P \boldsymbol{k}\frac{k_{rP}}{\mu_P}\Big(\nabla p + \nabla P_{cP} - \rho_P g \nabla D\Big) + q_c, \quad c = 1,\ldots,n_c, \\ \sum_{P=1}^{n_P} x_{c,P}\xi_P S_P = N_c, \quad c = 1,\ldots,n_c, \\ \sum_{P=1}^{n_P} S_P = 1.\end{gathered} \tag{1}$$

Здесь используются следующие обозначения для функций:

- $N_c = N_c(t,x,y,z)$ (*неизвестны*), $c = 1,\ldots,n_c$ – молярная плотность компонента c, здесь n_c – число компонент (2 или 3); часто также вместо номеров компонент $c = 1,\ldots,n_c$ используют обозначения $c = w,o,g$ по их именам (water, oil, gas),
- $p = p(t,x,y,z)$ (*неизвестно*) – давление (в фазе нефть),
- $S_P = S_P(t,x,y,z)$ (*неизвестны*), $P = 1,\ldots,n_P$ – насыщенность фазы P, здесь $n_P = n_c$ – число фаз (2 или 3); часто также вместо номеров фаз

$P = 1, \ldots, n_P$ используют обозначения $P = W, O, G$ по их именам (Water, Oil, Gas),

- $P_{cP} = P_{cP}(S_P)$ – капиллярное давление в системе нефть–фаза P (*известная* функция), для $P = O$ полагается тождественно равной 0,
- $\phi = \phi(p, x, y, z)$ – пористость (*известная* функция),
- $x_{c,P} = x_{c,P}(p, \boldsymbol{N})$ – молярная доля компонента c в фазе P (*известная* функция),
- $\xi_P = \xi_P(p, \boldsymbol{N})$ – молярная плотность фазы (*известная* функция),
- $\rho_P = \rho_P(p, \boldsymbol{N})$ – массовая плотность фазы (*известная* функция),
- $\boldsymbol{k} = \boldsymbol{k}(p, x, y, z)$ – тензор абсолютной проницаемости (*известная* функция),
- $k_{rP} = k_{rP}(S_P)$ – относительная фазовая проницаемость (*известная* функция),
- $\mu_P = \mu_P(p_P, \boldsymbol{N})$ – вязкость фазы (*известная* функция),
- $D = D(x, y, z)$ – вектор глубины (сверху вниз) (*известная* координатная функция), $g = \text{const}$ – *известная* гравитационная постоянная,
- $q_c = q_c(p, \boldsymbol{N}, t, x, y, z)$ – функция источников компонента c (нагнетательные и добывающие скважины, водонапорные горизонты) (*известная* функция).

Это система из $2n_c + 1$ уравнения относительно $n_c + n_P + 1$ неизвестных N_c, $c = 1, \ldots, n_c$, S_P, $P \in 1, \ldots, n_P$, p. В рассматриваемой так называемой “модели черной нефти” $n_P = n_c$.

На внешней границе резервуара ставятся условия непротекания. Начальные условия вычисляются либо по заданным значениям N_c, p, S_P, либо из условий гидростатического равновесия.

Для решения задачи (1) используется полностью неявная схема аппроксимации по времени и метод конечных объемов для дискретизации по пространству.

В первой главе предложена новая математическая модель техногенной трещиноватости, возникающей около скважин при проведении гидравлического разрыва пласта (ГРП) или при превышении давления закачки величины горного давления.

Технология ГРП применяется практически на 100% скважин в условиях низкопроницаемого коллектора (малой величины компонент тензора проницаемости $\boldsymbol{k}$ в (1)). Эта технология является основной при разработке сланцевого газа и сланцевой нефти. Однако, при увеличении степени детализации сеточных аппроксимаций все чаще возникает ситуация, когда размеры трещин начинают превышать размеры ячеек сетки. В этом случае традиционно применяемое при описании ГРП увеличение эффективности источников и стоков q_c уже не подходит. В этом случае обычно применяют локальное измельчение сетки вблизи скважин и формируют высокопроводящие каналы вдоль траектории трещины. Этот подход имеет существенные недостатки:

- движение жидкости или газа по трещине в этом случае подчиняется фильтрационному механизму, что не соответствует физике процесса, более похожего на течение в трубе;
- значительное увеличение гетерогенности тензора проницаемости приводит к сильному замедлению расчета.

В работе для более реалистичного моделирования динамики поведения трещин ГРП предлагается математическая модель, в которой трещина рассматри-

вается не как область расчетной сетки с повышенной проницаемостью, а как набор новых связей (участков перфораций) скважины с блоками, пересекаемыми траекторией трещины. Задавая эффективность работы этих виртуальных перфораций можно реалистично описать

- гидравлический характер движения жидкости или газа по трещине;
- раскрытие или схлопывание трещины при повышении или понижении давления в ней;
- влияние на течение по трещине закачанного при проведении ГРП пропанта;
- эффекты разрушения пропанта от времени и выноса его потоком жидкости или газа;
- эффекты увеличения гидродинамического сопротивления течению в трещине за счет осаждения высоковязких фракций.

В этой модели происходит значительное усложнение функции источников и стоков q_c в (1). Это сказывается на время вычислений и усложняет расчет на параллельных ЭВМ (см. ниже), однако, предложенная модель оказывается эффективнее традиционного подхода, что подтверждается численными экспериментами.

Эта математическая модель гидроразрыва пласта реализована в комплексе программ tNavigator, предложенном в работе. Она была опробована при расчете реальных месторождений Западной Сибири и была рекомендована к применению в нескольких добывающих подразделениях крупнейших Российских нефтяных компаний.

В первой главе также рассмотрены особенности получаемой при дискретизации задачи (1) системы линейных уравнений, влияющие на расчет на параллельных ЭВМ:

- Большой шаблон матрицы системы линейных алгебраических уравнений (СЛАУ) (т.е. количество ненулевых элементов в строке) приводит к большому количеству связей между между частями матрицы (или вычислительной области), распределенными на разные логические процессоры. Это приводит к большому объему данных, которыми должны обмениваться вычислительные процессы на каждой итерации.

- Большие объемы обрабатываемых на каждой итерации данных делают невозможным обработку последовательных частей алгоритма (если они есть) одним логическим процессором из-за недостатка оперативной памяти. Все данные должны быть распределены между работающими процессами, что приводит к большому объему обменов между ними.

- Из-за высокой жесткости задачи необходимо применять сложные предобуславливатели, многие из которых существенно последовательные по своей природе.

- Часто невозможно отнести к одному логическому процессору целиком набор блоков, через которые проходит

  – одна скважина,

  – несколько “пересекающихся” скважин,

  – одна группа скважин, связанная поверхностными трубопроводами,

  – один водонапорный горизонт,

  – одна трещина ГРП при ее моделировании через набор виртуальных перфораций.

  Это приводит к тому, что алгебраические уравнения для определения параметров притока в(из) скважину или водонапорный горизонт придется

решать совместно на нескольких вовлеченных логических процессорах, передавая между ними всю необходимую информацию.

- Объем входных данных не определяет полностью сложность вычислительной работы. Получается, что при равномерном разделении входных данных между логическими процессорами вычислительная работа при этом делится неравномерно. Это происходит из-за отсутствия фильтрационных течений в ряде подобластей резервуара, причем для определения этих подобластей требуется произвести часть расчета.

Указанные причины вызывают быстрое насыщение параллельных алгоритмов, т.е. ситуации, когда добавление новых логических процессоров не приводит к сокращению времени расчета.

Результаты первой главы опубликованы в работах [A13], [A14].

Во второй главе рассматриваются блочные предобуславливатели класса неполного LU разложения (ILU(0), ILU(1), ILUT) для матриц, полученных в результате аппроксимации системы дифференциальных уравнений фильтрации. Воспользовавшись схожей структурой ненулевых элементов каждого уравнения системы для одной ячейки сетки и занумеровав неизвестные в системе таким образом, что сначала идут все компоненты векторов неизвестных, соответствующие первой базисной функции, затем — второй и т.д., можно представить матрицу Якобиана метода Ньютона как матрицу, составленную из блоков, размер которых равен числу решаемых дифференциальных уравнений (обычно 2 или 3). Такой подход требует изменения формата хранения разреженной матрицы MSR [21] и переформулировки алгоритмов ILU разложения в терминах блоков.

Блочные варианты алгоритмов LU и ILU ранее рассматривались в литературе, однако, они применялись к общим разреженным матрицам, а уравнения группировались в блоки либо для полного задействования кеш-памяти про-

цессора, либо для использования идей метода разделения области (см. [22], [23]).

Перечислим основные свойства построенного блочного представления:

- *Расширяется класс матриц*, для которых применимы ILU предобуславливатели. Действительно, при обычном (скалярном) ILU разложении в ходе разложения на диагонали матрицы не должно появляться нулей, а при блочном ILU разложении требования к матрице снижаются: диагональные блоки должны иметь ненулевой определитель.

- *Происходит избирательное увеличение ширины шаблона матрицы*. Так как блоки матрицы хранятся целиком (даже если там есть нули), то фактически обычный метод ILU разложения немного видоизменяется и в чем-то походит на ILU(p) метод. Численные эксперименты показывают, что уже ILU(1) не дает ускорения, в то время, как переход к блочному варианту дает выигрыш даже в числе итераций. Расширение ленты приводит к тому, что получаемое ILU разложение точнее приближает исходную матрицу, и, следовательно, улучшаются характеристики сходимости алгоритмов с использованием ILU разложения.

- *Появляется ускорение за счет использования кэш-памяти*. Вследствие того, что везде в блочном алгоритме мы оперируем блочными матрицами размерности $d = 2, 3$, происходит меньшее число обращений к данным на медленных носителях (оперативная память). В начале работы с новой строкой матрицы она вся попадает в кэш, и дальше идет работа уже только с кэш-памятью, а вследствие использования MSR формата хранения матрицы обеспечивается непрерывность обращения к памяти от первых до последних блоков матрицы (без «прыжков» через сегменты памяти).

- *Уменьшается требуемый объем памяти* для хранения матрицы из-за хранения списка номеров ненулевых элементов для блока, а не для каждой строки. Это также сказывается и на времени работы, так как повышается эффективность кеширования используемой памяти.

- *Увеличиваются возможности для масштабирования на параллельных ЭВМ* за счет увеличения использования кэша (процессоры реже обращаются к общей шине памяти).

Указанные свойства подтверждаются численными экспериментами с матрицами, полученные при дискретизации уравнений фильтрации с данными для реальных месторождений на разных шагах по времени в комплексе программ tNavigator, предложенном в работе, используя как классическую двухточечную аппроксимацию потока, так и многоточечную, приводящую к матрицам с гораздо более сложным шаблоном.

Результаты второй главы опубликованы в работах [A7], [A5].

В третьей главе рассматриваются параллельные блочные предобуславливатели класса ILU.

Алгоритмы неполного LU разложения являются одними из лучших предобуславливателей, но являются существенно последовательными по своей природе, и поэтому задаче построения их вариантов, эффективно работающих на параллельных ЭВМ, посвящено достаточно много работ. В основном, для распараллеливания предлагались те или иные способы обхода неизвестных в системе для того, чтобы уменьшить количество обменов между параллельно работающими процессами, либо введение на графе матрицы иерархической структуры аналогично с многосеточными методами и методами декомпозиции, см. [24], [25], [23], [26].

И. Е. Капорин и И. Н. Коньшин [27] при решении симметричных положительно-определенных СЛАУ методом сопряженных градиентов предложили

использовать для построения предобуславливателя метод перекрывающихся разбиений матрицы СЛАУ на блоки. К матрице применяется блочная версия двустороннего неполного обратного разложения Холецкого, а затем каждый из блоков заменяется на аппроксимацию — результат неполного разложения Холецкого. Полученный таким образом метод обладает лучшими характеристиками сходимости и параллелизма по сравнению с большинством других методов разбиения.

В диссертации рассмотрено решение несимметричных СЛАУ методом бисопряженных градиентов на параллельных ЭВМ, где предобуславливателем служит ILU-разложение, получающееся с помощью метода разбиения матрицы, предложенного И. Е. Капориным и И. Н. Коньшиным. Исследованы различные стратегии выбора перекрытий: выбор с фиксированным размером перекрытия (ILUG) и выбор элементов с помощью графа связности ненулевых элементов матрицы коэффициентов СЛАУ (ILUA); а также проведено их сравнение. Также проведено исследование применения предложенного метода в составе предобуславливателя CPR (Constrained Pressure Residual) для задачи фильтрации, где он сравнивался с алгебраическим многосеточным методом (AMG) и показал сравнимую эффективность работы на сложных матрицах.

Предложенный параллельный алгоритм блочного варианта построения предобуславливателей класса ILU (ILU(0), ILU(1), ILUT) хорошо сохраняет их качество и обеспечивает значительное ускорение по сравнению с последовательной версией при увеличении числа используемых логических процессоров. Проведены численные эксперименты с использованием различных матриц, полученных при дискретизации задач с реальными данными нефтяных месторождений в комплексе программ tNavigator, предложенном в работе, которые подтверждают хорошую масштабируемость предложенного алгоритма.

Результаты третьей главы опубликованы в работах [A8], [A6], [A10].

В четвертой главе рассмотрен гибридный MPI–многопоточный алгоритм распараллеливания решения задач фильтрации на ЭВМ с распределенной памятью (кластере).

Решению задачи фильтрации многокомпонентной смеси в пористой среде на параллельных ЭВМ посвящено большое количество работ: [28], [29], [30], [31], [32], [33], [34], [35], [36], [37]. Однако, вопрос остается открытым, так как имеющиеся алгоритмы насыщаются после 20–30 логических процессоров.

В диссертации рассмотрена эффективная технология, включающая алгоритмы, структуры данных и комплекс программ, для построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере). Предложенная технология реализована в комплексе программ tNavigator, предложенном в работе.

Основной мотивацией к исследованиям стали значительные изменения в строении вычислительных кластеров, изменения, которые по масштабам являются самыми большими за 20-летнюю историю применения параллельных вычислений на таких системах. Каждый узел современного кластера сам является большим многопроцессорным компьютером, причем с неоднородным доступом к памяти (NUMA). Не учет этих изменений при разработке программного обеспечения ведет к значительным потерям в производительности и слабому использованию ресурсов оборудования.

Во-первых, изменились два (или четыре) физических процессора, установленные в узле кластера: они стали многоядерными. Фактически внутри одного физического корпуса процессора расположены несколько процессоров, называемых ядрами, которые доступны операционной системе как «обычные» процессоры. В настоящий момент производятся варианты с 4, 6, 8, 10 и 12 ядрами и это количество постоянно растет, поскольку оно стало главным источником повышения производительности физического процессора из-за проблем с повышением его тактовой частоты. Более того, часто каждое ядро имеет по

два комплекта всех вычислительных блоков и для оптимального их заполнения вводят многопоточность (hyperthreading), позволяя одному физическому ядру исполнять два потока данных. Это удваивает количество «процессоров», доступных операционной системе и работающим приложениям. Для единообразия вводят понятие «логического процессора» — это «процессор» предоставляемый операционной системе и работающим приложениям, без относительно его физической природы.

Для программирования кластеров де-факто стандартом стал интерфейс MPI. Пока логических процессоров на узле было 2 (как всего пять лет назад), накладные расходы на обмен сообщениями между процессами, работающими на одном узле, были минимальны. Учет того, что некоторые процессы могут взаимодействовать между собой намного эффективнее, слишком усложнял и без того тяжелую в разработке MPI программу, не принося заметного ускорения в ее работе. Все изменилось, когда узлы стали состоять из более, чем десятка логических процессоров. Потери в производительности стали составлять уже разы, и это уже нельзя было не замечать. Поэтому возникла идея разработки гибридных приложений: взаимодействие между узлами осуществляется с помощью интерфейса MPI, а внутри узла программа работает, как на системе с общей памятью, используя интерфейс OpenMP или потоки исполнения, предоставляемые непосредственно операционной системой.

Выгоды от такого подхода очевидны. Рассмотрим, например, кластер из 20-ти узлов с 12-ю логическими процессорами в каждом, всего 240 процессоров. При традиционном подходе на этой системе надо было использовать 240 MPI процессов, что обычно уже сталкивается с проблемами масштабируемости задачи. При гибридном подходе надо использовать 20 MPI процессов (по числу узлов) и по 12 потоков исполнения на каждом узле.

Вторым значительным изменением стал новый способ общения процессоров с памятью. Появление многоядерных процессоров сделало главное до-

стоинство симметричных многопроцессорных систем – общую шину к памяти, ее главным недостатком. Эта шина является тем «бутылочным горлышком», через который не могут пройти данные, требуемые для загрузки работой все большего числа логических процессоров. Для решения этой проблемы подсистему оперативной памяти подключили непосредственно к физическим процессорам, а для эмуляции общей памяти процессоры связаны высокоскоростным каналом. Так возникает неоднородность в устройстве памяти: данные из «своей» памяти (подключенной непосредственно к нему) процессор берет с одной скоростью, а из «чужой» (подключенной к другому процессору) – с почти вдвое меньшей, так как данные проходят еще и по линии связи между процессорами.

Опишем *основные этапы работы* гибридного MPI–многопоточный приложения, призванного учесть указанные особенности. Описанная ниже схема реализована в комплексе программ tNavigator, предложенном в работе.

- *Запуск MPI процессов по числу узлов кластера*. Поскольку в дальнейшем из каждого MPI процесса будут запущены потоки исполнения по числу логических процессоров, то требуется использовать реализацию MPI-2, допускающую такую возможность.

- *Загрузка входных данных на каждом из узлов*. Поскольку канал ввода существенно менее производительный, чем единичное ядро узла, загрузка данных для всех логических процессоров на узле будет производиться одним из них. Это уменьшает как количество синхронизаций, так и трафик по каналу ввода. Это уже отличает гибридное приложение от традиционных MPI решений, где каждый из процессов должен считывать данные для себя (или получать от других процессов по коммуникационному каналу).

  При загрузке данных важно обеспечить, чтобы

– в памяти узла даже кратковременно не находились бы все данные задачи, а только предназначенная ему часть (иначе из-за большого объема данных они могут не поместиться в оперативной памяти узла),

– вычислительная работа между узлами была бы распределена равномерно.

Указанные требования для задачи фильтрации удовлетворить не просто из-за того, что по входным данным задачи нельзя без дополнительных вычислений определить объем работы для узла, а для проведения вычислений надо данные уже загрузить, т.е. разделить между узлами. Эта проблема не решена в имеющихся пакетах решения задачи фильтрации. В диссертации (см. ниже.) описана технология, позволяющая достигнуть оптимальной балансировки загруженности узлов без «пика» по требуемой алгоритму памяти.

- *Выделение памяти под данные на узле с учетом неоднородного доступа (NUMA)*. Загружаемые данные требуется разместить в оперативной памяти так, чтобы при последующем решении задачи минимизировать обращение каждого логического процессора к памяти «чужого» NUMA узла, см. ниже.

- *Построение графа связей MPI процессов* и определение порядка MPI пересылок, при котором все процессы могут обменяться данными со связанными с ними (в силу стандартного перекрытия и конструкции предобуславливателя) процессами. При этом все потоки, «составляющие» MPI процесс на одном узле, участвуют как одно целое, формируя объединенный узел графа. Это радикально:

  – уменьшает количество MPI обменов путем группировки транзак-

ций от всех потоков в одну,

- уменьшает объем MPI обменов из-за уменьшения размера зоны перекрытия (так как потоки исполнения на одном узле «просто» обращаются к общей памяти друг друга, не формируя интерфейсную зону).

Эти факторы являются одними из главных составляющих ускорения комплекса программ tNavigator, предложенного в работе.

Для задачи фильтрации этот этап приходится делать на каждом шаге по времени из-за постоянно растущего количества скважин. Дело здесь в том, что из-за наличия длинных горизонтальных стволов скважин (или больших водонапорных горизонтов) часто невозможно разделить задачу между узлами так, чтобы одна скважина (водонапорный горизонт) не попала в несколько MPI процессов. Это, приводит к

- изменению графа связей между процессами и необходимости его перестройки,
- изменению предобуславливателя,
- добавлению дополнительных MPI обменов в ходе решения уравнений для скважины.

- *Вычисление матрицы системы* выполняется параллельно и независимо всеми логическими процессорами системы из-за обычного перекрытия (хранения данных в блоках сетки хотя и из другого процесса, но связанными с блоками сетки на рассматриваемом узле). Отметим, что в гибридной реализации здесь тоже есть экономия в оперативной памяти из-за отсутствия дублирования информации между потоками на одном узле.

- *Итерационный алгоритм решения и предобуславливатель*. Построение предобуславливателя было подробно рассмотрено в главе 3, а выбор оптимального итерационного алгоритма решения для гибридного процесса решения рассмотрен в главе 4, см. ниже.

В разделе 4.3 рассмотрен метод *балансировки загруженности узлов кластера при расчете задачи фильтрации*.

Для построения аппроксимации системы дифференциальных уравнений фильтрации вводится сетка, учитывающая геологическое строение месторождения (разломы, выклинивания). Далее будем называть упомянутую сетку исходной. Для решения задачи используются только данные для ячеек сетки, имеющих поровый объем, больше определенного порогового значения. Такие ячейки сетки называются активными. Назовем расчетной сеткой совокупность активных ячеек исходной сетки. Данные в неактивных ячейках во время расчета не хранятся, но используются для построения расчетной сетки (например, пороговое значение для определения активности ячейки может зависеть от среднего порового объема по всем ячейкам).

Стандартный подход к разделению задачи между узлами кластера состоит в геометрическом разрезании области на части по числу вычислительных узлов. Этому разрезанию области соответствуют разделение исходной и расчетной сеток. В каждом MPI-процессе загружаются данные и выполняется вычислительная работа для ячеек сетки, соответствующих его части области. Из-за особенностей геологического строения месторождения часто невозможно разделить область на части так, чтобы в каждой части одинаково было количество ячеек исходной и расчетной сеток. Все имеющиеся пакеты расчета задачи фильтрации разделяют *исходную* сетку на одинаковые по количеству ячеек исходной сетки части, что идеально балансирует загруженность узлов кластера на этапе чтения входных данных. Однако, *расчетная* сетка при таком

геометрическом разделении области оказывается распределена между узлами очень неоднородно. Может даже получится так, что в некоторых частях исходной сетки отсутствуют блоки расчетной сетки, и соответствующему узлу кластера вообще не достается вычислительной работы. В этом случае большинство программных пакетов решения задачи фильтрации останавливают свою работу.

Предлагается разделить загрузку данных исходной сетки и построение распределенной по узлам расчетной сетки на три этапа:

1. Исходная сетка делится равномерно между узлами с перекрытиями (как в традиционном подходе). На каждом узле осуществляется построение части расчетной сетки с вычислением графа связей с ячейками в других MPI-процессах.

2. Осуществляется сборка карты соответствия ячеек расчетной и исходной сеток на каждом узле и на основании этой информации осуществляется вычисление номеров ячеек, которые попадут на каждый из узлов на следующем этапе.

3. Загрузка данных исходной сетки, с сохранением на каждом узле кластера данных только для ячеек расчетной сетки, распределенных на данный узел.

Подход проверен в коплексе tNavigator на задаче, содержащей 232 миллиона ячеек исходной сетки и 74 миллиона ячеек расчетной. Результаты тестирования показывают, что предложенный алгоритм балансировки загруженности позволяет эффективно рассчитывать очень большие задачи фильтрации на кластерных системах, обеспечивая равномерность как выделенной оперативной памяти, так и вычислительной нагрузки даже при сложном взаимном расположении исходной и расчетной сеток. Алгоритм использует из входных дан-

ных только геометрическую информацию, что позволяет распределить данные на ранних этапах их загрузки.

В разделе 4.4 рассмотрен метод *оптимизации решения задачи фильтрации для многопроцессорных систем с общей неоднородной памятью*, который проще всего продемонстрировать на основной операции итерационных алгоритмов решения СЛАУ — параллельном умножении матрицы на вектор.

В традиционной программной реализации память под матрицы и вектор выделяется в главном потоке. На NUMA системе это приводит к тому, что физически матрица и вектор будут располагаться в локальной памяти одного NUMA узла, что увеличивает как минимум в два раза время доступа к ним потоков, работающих на другом узле.

Покажем, как можно оптимизировать программный код для NUMA систем, не изменив сам численный алгоритм, и получить ускорение $1.5-2$ раза.

Допустим, что размер матрицы равен $K = M \cdot T$, где T – число потоков, матрица хранится в памяти по строкам. Поток с номером i вычисляет результат произведения строк матрицы с номерами $M \cdot (i-1)+1, ..., M \cdot i$ на вектор. Будем говорить, что эти строки матрицы принадлежат i-тому потоку. Таким образом, матрица делится на T непрерывных частей, каждая из которых обрабатывается собственным потоком.

Ключевая идея оптимизации заключается в реализации двух условий, которые позволят минимизировать число обращений к нелокальным участкам памяти:

- каждый из вычислительных потоков работает на фиксированном узле NUMA системы,
- те части матрицы и вектора, с которыми вычислительный поток оперирует большую часть времени, физически находятся в локальной памяти этого узла.

Для выполнения первого условия в каждом из вычислительных потоков сразу после его создания сделаем системный вызов, который привяжет этот *i*-тый поток к *i*-тому логическому процессору.

Для удовлетворения второго условия с минимальным переписыванием программы воспользуемся особенностью механизма выделения памяти в операционных системах, поддерживающих NUMA. В системах Linux и Windows 7 при выделении памяти посредством функции malloc отображение затребованной виртуальной памяти на физическую строится не сразу. Когда поток обращается к блоку виртуальной памяти, для которого еще не построено отображение, происходит страничный промах (page fault), который инициирует создание отображения запрошенного блока виртуальной памяти на локальную память того узла NUMA системы, с которого был произведен запрос.

В традиционном подходе выделения памяти в многопоточных приложениях главный поток запрашивает память у операционной системы и инициализирует ее, скажем, нулями. На NUMA системе это приводит к тому, что вся запрошенная память выделится в локальной памяти узла, на котором работал главный поток. Остальные потоки в это время простаивают, дожидаясь завершения этой операции.

Модифицируем этот подход, чтобы его можно было эффективно использовать на NUMA системе. Будем выполнять операцию выделения памяти в два этапа. На первом этапе главный поток запрашивает у операционной системы память требуемого объема. Затем следует точка синхронизации, после которой указатель на затребованную память можно использовать во всех потоках. На втором этапе выделенная память параллельно инициализируется всеми потоками, причем каждый из потоков инициализирует только свою часть запрошенной памяти.

Предложенный метод инициализации памяти в два этапа эффективен по двум причинам. Во-первых, та часть памяти, к которой поток будет обращать-

ся чаще всего, выделяется в локальной памяти узла NUMA системы, на котором он работает. Во-вторых, ускоряется операция ее инициализации за счет параллелизма в работе памяти. Минусом этого метода является наличие дополнительной точки синхронизации потоков.

Эти модификации затрагивают ту часть программного кода, которую можно назвать подготовительной стадией алгоритма — создание вычислительных потоков, заполнение матрицы и вектора. Тем самым, алгоритмически существенная часть программного кода, ответственная за непосредственное перемножение матрицы на вектор, остается нетронутой.

Используя эти идеи, представленный метод оптимизации кода для NUMA систем можно применить к любой многопоточной программе, в которой вычислительные потоки большую часть времени проводят, обращаясь к своей части всей выделенной процессу памяти. Получаемый выигрыш зависит от доли операций с памятью в общем времени работы программы. Результаты численных экспериментов в комплексе tNavigator показывают, что при использовании всех физических ядер параллельной ЭВМ оптимизированное для NUMA систем приложение работает в $1.5 - 2$ раза быстрее, чем его исходная версия.

В разделе 4.5 рассмотрен вопрос о *выборе оптимального итерационного метода решения систем линейных уравнений в задачах фильтрации*. Для гибридной MPI-многопоточной программы нахождения решения задачи фильтрации выбор оптимального итерационного метода для решения систем линейных уравнений с разреженной матрицей является непростой задачей. Слишком много параметров влияют на суммарную производительность:

- количество точек синхронизации (когда все работающие процессы и(или) потоки должны обменяться информацией);
- объем пересылаемых данных между процессами;

- объем оперативной памяти, требуемый алгоритму (поскольку оперативная память – намного более медленный ресурс, чем процессор);
- адаптивность алгоритма – его способность подстраиваться под входную матрицу, поскольку решается много разных систем (на каждом шаге по времени и на каждой итерации метода Ньютона) и выбирать набор параметров, оптимальный для каждой из них, невозможно.

При этом производительность надо измерять при разном количестве использованных узлов и логических процессоров на них, чтобы оценить масштабируемость алгоритмов. При таком количестве влияющих факторов теоретическое сравнение алгоритмов вряд ли возможно. Поэтому было проведено практическое сравнение трех основных алгоритмов решения систем линейных уравнений с несимметричной разреженной матрицей: BCGS, ORTHOMIN, GMRES (последний в двух вариантах — QGMRES, DQGMRES) (см. [21]) по скорости работы и ускорению на параллельных ЭВМ: на SMP-системах, на системах с распределенной памятью и на гибридных системах (системах с распределенной памятью, состоящих из SMP-узлов). Проведено также исследование масштабируемости каждого из алгоритмов (время работы алгоритма сравнивается с временем работы его же на другой конфигурации узлов/потоков) и сравнение алгоритмов между собой (время работы алгоритма сравнивается с временем работы другого алгоритма на той же конфигурации узлов/потоков). Все алгоритмы были реализованы в комплексе tNavigator с максимальными оптимизациями для параллельных ЭВМ, включая учет неоднородности доступа к памяти (NUMA). В качестве предобуславливателя использовался параллельный вариант предобуславливателя ILU(0) из главы 3.

Для гибридного MPI-многопоточного распараллеливания векторы и матрица делились поровну между всеми потоками исполнения. При вычислении скалярного произведения, линейной комбинации векторов или произведения

матрицы и вектора каждый поток считал свою часть. Далее, в случае вычисления скалярного произведения, результат получался посредством общей MPI-синхронизации. Перед каждой операцией умножения вектора на матрицу и применения предобуславливателя происходил MPI-обмен, в ходе которого каждый узел получал элементы вектора, необходимые для подсчета его части в соответствующей операции.

В качестве тестовых были выбраны несимметричные разреженные матрицы, полученные при аппроксимации задачи фильтрации с данными реальных нефтяных месторождений в комплексе tNavigator. Было проведено тестирование алгоритмов BCGS, ORTHOMIN, QGMRES, DQGMRES на параллельных ЭВМ при использовании от 1-го до 192-х потоков исполнения. В среднем лучше всего как с точки зрения масштабируемости, так и абсолютного времени решения, проявил себя алгоритм BCGS.

В разделе 4.6 приведены *численные эксперименты* для комплекса программ tNavigator в целом. Напомним кратко, какие шаги были описаны ранее для того, чтобы максимально эффективно решать систему линейных уравнений с матрицей, получающейся при аппроксимации задачи фильтрации на каждом шаге по времени и на каждой итерации метода Ньютона:

- выбран оптимальный метод хранения матрицы;
- выбран оптимальный предобуславливатель;
- выбран оптимальный способ распараллеливания предобуславливателя;
- выбран оптимальный итерационный алгоритм для гибридного MPI-многопоточного метода решения;
- выбран оптимальный способ балансировки загруженности узлов;

- выбрана оптимальная технология учета неоднородности доступа к памяти на узле.

Оптимальность выбора для каждого из этапов подтверждена численными экспериментами с предложенным комплексом программ tNavigator. Теперь необходимо проверить, какой же эффект дает описанная оптимизация для программы решения задачи фильтрации в целом. Результаты ускорения работы комплекса программ tNavigator на наборе данных реального месторождения «среднего» размера (2.4 миллиона ячеек сетки, 7.2 миллиона неизвестных в СЛАУ), по отношению к времени работы последовательной версии на той же системе:

- на 1-процессорной системе (4 физических ядра, 8 логических процессоров): 3.5 раза;
- на 2-процессорной системе (8 физических ядер, 16 логических процессоров): 8 раз;
- на 4-процессорной системе (32 физических ядра, 64 логических процессоров): 21 раз;
- на 20-узловом кластере (240 физических ядер, 240 логических процессоров): 51 раз.

Ускорение работы на наборе данных реального месторождения «большого» размера (*Самотлорское месторождение*, 4.7 млн. ячеек сетки, 13 тысяч скважин, 40 лет истории разработки, 14.1 млн. неизвестных в СЛАУ) на 16-узловом кластере (192 физических ядра, 192 логических процессоров): 41 раз.

В разделе 4.7 приведено *сравнение результатов с другими пакетами решения задачи фильтрации*, поскольку, во-первых, хорошее ускорение решения задачи фильтрации требует проверки, того, что оно не было достигнуто

за счет качества получаемого решения, а во-вторых, для сложной гибридной MPI-многопоточной программы tNavigator объемом свыше 60 мегабайт исходного текста на языке C++, остро стоит вопрос о проверке корректности ее работы.

Для ряда наборов данных реальных месторождений, продемонстрировано хорошее совпадение результатов расчетов при значительном сокращении времени расчета.

Программа tNavigator свободно доступна для загрузки с сайта `rfdyn.ru`.

Результаты четвертой главы опубликованы в работах [A11], [A9], [A12].

В Заключении сформулированы основные результаты диссертации.

Основные результаты диссертации:

- Математическая модель техногенной трещиноватости вблизи скважин.
- Метод блочного хранения разреженных несимметричных матриц в памяти и блочные варианты построения предобуславливателей, основанных на неполном LU разложении.
- Алгоритм распараллеливания построения предобуславливателей, основанных на неполном LU разложении.
- Эффективная технология для построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью, включающая в себя:
  - технологию построения гибридного трехуровневого вычислительного процесса решения задачи фильтрации;
  - технологию балансировки загруженности узлов вычислительного кластера при расчете задачи фильтрации;

  - технологию оптимизации решения задачи фильтрации для многопроцессорных систем с общей неоднородной памятью.

- Комплекс программ tNavigator для решения задачи фильтрации на современных параллельных ЭВМ, реализующий описанные в работе методы и алгоритмы.

Для достижения наилучшей эффективности в программном комплексе tNavigator были реализованы и протестированы на большом количестве наборов входных данных и разнообразных конфигураций кластерных систем:

- 3 алгоритма построения предобуславливателей класса ILU (ILU(0), ILU(1), ILUT);

- 2 способа разделения матрицы при распараллеливании предобуславливателей класса ILU (ILUG и ILUA);

- 4 алгоритма решения систем линейных уравнений с разреженной несимметричной матрицей (ORTHOMIN, QGMRES, DQGMRES, BCGS).

По результатам тестирования был выбран наилучший для задачи фильтрации алгоритм.

Для тестирования программного комплекса tNavigator в целом проведена серия экспериментов:

- на разнообразных параллельных ЭВМ, от 8-ми до 240 логических процессоров, с разными частотами оперативной памяти, — для определения характеристик масштабируемости;

- на разнообразных наборах данных для реальных месторождений — для определения согласованности расчета с другими программными комплексами решения задачи фильтрации.

Численные эксперименты показали хорошую эффективность программного комплекса tNavigator, реализующего описанный в работе метод решения задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси, при хорошем согласовании с результатами работы других программ решения той же задачи.

Список публикаций

A1. К. Ю. Богачев. Практикум на ЭВМ. Методы решения линейных систем и нахождения собственных значений. 2 изд. Москва: Механико-математический ф-т МГУ им. М.В.Ломоносова, 1999. 200 с.

A2. К. Ю. Богачев. Практикум на ЭВМ. Методы приближения функций. 3 изд. Москва: Механико-математический ф-т МГУ им. М.В.Ломоносова, 2002. 200 с.

A3. К. Ю. Богачев. Основы параллельных вычислений. Москва: ЦПИ при механико-математическом ф-те МГУ им. М.В.Ломоносова, 2002. 352 с.

A4. К. Ю. Богачев. Основы параллельного программирования. Москва: Бином, 2003. 342 с. ISBN: 5-94774-037-0.

A5. К. Ю. Богачев, Н. С. Мельниченко. О пространственной аппроксимации методом подсеток для задачи фильтрации вязкой сжимаемой жидкости в пористой среде // Вычислительные методы и программирование. 2008. Т. 9. С. 191–199.

A6. К. Ю. Богачев, И. Г. Горелов. Применение параллельного предобуславливателя CPR к задаче фильтации вязкой сжимаемой жидкости в пористой среде // Вычислительные методы и программирование. 2008. Т. 9. С. 184–190.

A7. К. Ю. Богачев, Я. В. Жабицкий. Блочные предобуславливатели класса ILU для задач фильтрации многокомпонентной смеси в пористой среде // Вестник Моск. ун-та, Сер. 1, Математика, Механика. 2009. Т. 5. С. 19–25.

A8. К. Ю. Богачев, Я. В. Жабицкий. Метод Капорина-Коньшина параллельной реализации блочных предобусловливатей для несимметричных матриц в задачах фильтрации многокомпонентной смеси в пористой среде // Вестник Моск. ун-та, Сер. 1, Математика, Механика. 2010. Т. 1. С. 46–52.

A9. К. Ю. Богачев, А. Р. Миргасимов. Об оптимизации вычислительных приложений для многопроцессорных систем с общей неоднородной памятью // Вычислительные методы и программирование. 2010. Т. 11. С. 198–209.

A10. К. Ю. Богачев, М. Ю. Михалева, И. Г. Горелов. Алгебраический многоуровневый метод AMG: сравнение с методом BICGSTAB+ILU и использование в составе метода CPR // Вестник Моск. ун-та, Сер. 1, Математика, Механика. 2010. Т. 4. С. 24–28.

A11. К. Ю. Богачев, А. А. Климовский, А. Р. Миргасимов, А. Е. Семенко. Балансировка загруженности узлов кластера при расчете задачи фильтрации // Вычислительные методы и программирование. 2011. Т. 12. С. 70–73.

A12. К. Ю. Богачев, Я. В. Жабицкий, А. А. Климовский и др. Сравнение итерационных методов решения разреженных систем линейных уравнений в задачах фильтрации на вычислительных системах с распределенной памятью // Вычислительные методы и программирование. 2011. Т. 12. С. 74–76.

A13. Bogachev K. Y., Shelkov V. New Realistic Hydraulic and Technogenic Fracture Modeling Approach in Full-Scale Dynamic Models // SPE paper 138071. Presented at the Russian Oil and Gas Conference and Exhibition, Moscow, Russia. 2010. 6 pp.

A14. Bogachev K. Y., A.Eydunov D., Robinson T. et al. A New Approach to Numerical Simulation of Fluid Flow in Fractured Shale Gas Reservoirs // SPE paper 147021. Presented at the Canadian Unconventional Resources Conference, Alberta, Canada. 2011. 6 pp.

Цитированная литература

1. К. Ю. Богачев. Итерационные методы решения квазилинейных эллиптических задач в областях сложной формы // ДАН. 1992. Т. 322, № 4. С. 641–645.

2. Bogachev K. Y. Iterative methods of solving main boundary value problems for second-order quasilinear elliptic equations in complexly shaped domains // Rus. J. Numer. Analysis Math. Model. 1992. Vol. 7, no. 4. Pp. 281–298.

3. К. Ю. Богачев. Итерационный метод решения смешанных задач для квазилинейных эллиптических уравнений в областях сложной формы // ДАН. 1995. Т. 340, № 6. С. 727–730.

4. Bogachev K. Y. Iterative method of solving quasilinear elliptic equations with rapidly varying coefficients singulary depending on two parameters // Rus. J. Numer. Analysis Math. Model. 1995. Vol. 10, no. 1. Pp. 9–32.

5. К. Ю. Богачев. Обоснование метода фиктивных областей решения смешанных краевых задач для квазилинейных эллиптических уравнений //

Вестник Московского Университета, Серия 1, Математика, Механика. 1996. № 3. С. 16–23.

6. Н. С. Бахвалов, К. Ю. Богачев, М. Э. Эглит. Вычисление эффективных упругих модулей для несжимаемого пористого материала // Механика композиционных материалов. 1996. Т. 32, № 5. С. 579–587.

7. Н. С. Бахвалов, К. Ю. Богачев, Maitre J. F. Эффективный алгоритм решения жестких эллиптических задач с приложениями к методу фиктивных областей // Ж. вычисл. матем. и матем. физ. 1999. Т. 39, № 6. С. 919–931.

8. Bogachev K. Y. On Algorithm for Efficient Solving Stiff Elliptic Problems with Large Parameters // Rus. J. Numer. Analysis Math. Model. 1999. Vol. 14, no. 6. Pp. 479–493.

9. Bogachev K. Y. On Algorithm for Efficient Solving Multiparametrical Stiff Elliptic Problems // Proceedings of the ENUMATH-99, Jyvaskula, Finland. 1999. Pp. 52–53.

10. К. Ю. Богачев. Эффективные алгоритмы решения эллиптических задач с большими параметрами // Труды Математического центра имени Н.И. Лобачевского. 1996. Т. 2. С. 3–44.

11. К. Ю. Богачев. Эффективный алгоритм решения эллиптических задач с большими параметрами // Ж. вычисл. матем. и матем. физ. 2000. Т. 40, № 3. С. 402–415.

12. Bogachev K. Y. On new method of parabolic approximation of the nonstationary Navier-Stokes equations // Proceedings of the ENUMATH-99, Luglio, Italy. 2001. Pp. 201–202.

13. Bogachev K. Y. Efficient Algorithms for Stiff Elliptic Problems with Large Parameters // Rus. J. Numer. Analysis Math. Model. 2002. Vol. 17, no. 4. Pp. 347–366.

14. Н. С. Бахвалов, К. Ю. Богачев, М. Э. Эглит. Исследование эффективных уравнений с дисперсией, описывающих распространение волн в неоднородных тонких стержнях // ДАН. 2002. Т. 387, № 6. С. 749–753.

15. Bogachev K. Y., Kobelkov G. On new method of parabolic approximation of the nonstationary Navier-Stokes equations // Proceedings of the CFD 2003 Conference, Moscow, Russia. Elsevier, 2004. Pp. 163–172.

16. К. Ю. Богачев. Операционные системы реального времени. Москва: ЦПИ при механико-математическом ф-те МГУ им. М.В.Ломоносова, 2001. 200 с.

17. Peaceman D. W. Fundamentals of Numerical Reservoir Simulation. Elsevier Science Ltd, 1977. 176 pp. ISBN: 0-44441-578-5.

18. Aziz K., Settari A. Petroleum Reservoir Simulation. London: Applied Science Publishers, 1979. 497 pp.

19. Ertekin T., Abou-Kassem J. H., King G. R. Basic Applied Reservoir Simulation. Society of Petroleum Engineers, 2001. 421 pp. ISBN: 1-55563-089-8.

20. Chen Z., Huan G., Ma Y. Computational Methods for Multiphase Flows in Porous Media. SIAM, 2006. 561 pp. ISBN: 0-89871-606-3.

21. Saad Y. Iterative methods for sparse linear systems. Second edition. Philadelphia, PA, USA: SIAM Press, 2003. Pp. xviii + 528. ISBN: 0-89871-534-2.

22. Saad Y., Zhang J. BILUM: Block Versions of Multielimination and Multilevel

ILU Preconditioner for General Sparse Linear Systems // SIAM Journal on Scientific Computing. 1999. — November. Vol. 20, no. 6. Pp. 2103–2121.

23. Shen C., Zhang J. Parallel two level block ILU preconditioning techniques for solving large sparse linear systems // Parallel Computing. 2002. — October. Vol. 28, no. 10. Pp. 1451–1475.

24. Basermann A. Parallel block ILUT/ILDLT preconditioning for sparse eigenproblems and sparse linear systems // Numerical linear algebra with applications. 2000. — October/December. Vol. 7, no. 7–8. Pp. 635–648.

25. Magolu monga Made M., van der Vorst H. A. A generalized domain decomposition paradigm for parallel incomplete LU factorization preconditionings // Future Generation Computer Systems. 2001. — June. Vol. 17, no. 8. Pp. 925–932.

26. Hénon P., Saad Y. A Parallel Multistage ILU Factorization Based on a Hierarchical Graph Decomposition // SIAM Journal on Scientific Computing. 2006. — January. Vol. 28, no. 6. Pp. 2266–2293.

27. И. Е. Капорин, И. Н. Коньшин. Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки // Журнал вычислительной математики и математической физики. 2001. Т. 41, № 4. С. 515–528.

28. Wallis J., Foster J., Kendall R. A New Parallel Iterative Linear Solution Method for Large-Scale Reservoir Simulation // SPE paper 21209. Presented at the SPE Reservoir Simulation Symposium, Anaheim, California, USA. 1991. 10 pp.

29. Burrows R., Ponting D., Wood L. Paralell Reservoir Simulation with Nested Factorisation // Proceedings of the 5th European Conference on the Mathematics of Oil Reecovery. Leoben, Austria. 1996.

30. Shiralkar G., Stephenson R., Joubert W. et al. Falcon: A Production Quality Distributed Memory Reservoir Simulator // SPE paper 37975. Presented at the SPE Reservoir Simulation Symposium, Dallas, Texas, USA. 1997. 9 pp.

31. Dogru A., Li K., Sunaidi H. et al. A Massively Parallel Reservoir Simulator for Large Scale Reservoir Simulation // SPE paper 51886. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 1999. 28 pp.

32. Lacroix S., Vassilevski Y. V., Wheeler M. F. Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS) // Numerical linear algebra with applications. 2001. Vol. 8. Pp. 537–549.

33. Magras J.-F., Quandalle P. High-Performance Reservoir Simulation With Parallel ATHOS // SPE paper 66342. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2001. 8 pp.

34. Collins D. A., Grabenstetter J. E., Sammon P. H. A Shared-Memory Paralell Black-Oil Simulator with a Parallel ILU Linear Solver // SPE paper 79713. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2003. 8 pp.

35. DeBaun D., Byer T., Childs P. et al. An Extensible Architecture for Next Generation Scalable Parallel Reservoir Simulation // SPE paper 93274. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2005. 13 pp.

36. Hammersley R., Ponting D. Solving Linear Equations In Reservoir Simulation Using Multigrid Methods // SPE paper 115017. Presented at the SPE Russian Oil and Gas Technical Conference and Exhibition, Moscow, Russia. 2008. 9 pp.

37. Dogru A. H., Fung L. S. K., Middya U. et al. An Next-Generation Parallel Reservoir Simulator for Giant Reservoirs // SPE paper 119272. Presented at the SPE Reservoir Simulation Symposium, Woodlands, Texas, USA. 2009. 29 pp.