СРАВНЕНИЕ РЕЗУЛЬТАТОВ ИССЛЕДОВАНИЯ ЭФФЕКТИВНОСТИ ПЕРЕУПОРЯДОЧЕННОГО МЕТОДА BICGSTAB НА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ «ЧЕБЫШЕВ» И «ЛОМОНОСОВ»

Б.И. Краснопольский

Одним из наиболее трудоемких этапов численного решения системы уравнений гидродинамики сеточными методами является решение уравнения неразрывности, сводящегося к уравнению Пуассона для давления. В большинстве случаев, этот этап оказывается доминирующим и занимает подавляющую часть времени (более 90 %). Для простых геометрий расчетных областей и ортогональных систем координат, а так же областей, отображаемых преобразованиями систем координат в прямоугольные области, разработаны прямые быстрые методы решения уравнения Пуассона (например, [1]). Однако, для сколь-нибудь сложных конфигураций расчетных областей, а так же неортогональных расчетных сеток данные методы оказываются неприменимыми. В таком случае достаточно популярным является подход, при котором используется итерационный метод градиентного типа, например метод BiCGStab [2], с предобуславливателем для ускорения сходимости. В качестве предобуславливателей долгое время использовалось частичное LU-разложение [3] с различной степенью заполненности. Однако, при всех своих плюсах (в первую очередь, простота и эффективность при последовательной реализации), данный метод обладает рядом значительных недостатков. Основные из них — это ухудшение эффективности предобуславливания по мере увеличения матрицы системы и высокая чувствительность к переупорядочиванию элементов матрицы. Последний аспект является критическим при параллельных вычислениях, поскольку эффективное распределение матрицы системы между вычислительными процессами также требует переупорядочивания элементов матрицы. Ввиду этого для решения уравнений эллиптического типа в произвольных областях набирают популярность многосеточные методы [4], которые могут использоваться как отдельно взятые решатели, так и эффективные предобуславливатели.

1. $x_0 =$ initial guess; $r_0 = b - Ax_0$
2. $\rho_0 = (r_0, r_0)$
3. $p_0 = r_0$

   for $j = 0, 1, \ldots$
4. $\quad v = Ap_j$
5. $\quad \alpha_j = \frac{\rho_j}{(v, r_0)}$
6. $\quad s_j = r_j - \alpha_j v$
7. $\quad t = As_j$
8. $\quad \omega_j = \frac{(t, s_j)}{(t, t)}$
9. $\quad x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j$; check for convergence
10. $\quad r_{j+1} = s_j - \omega_j t$
11. $\quad \rho_{j+1} = (r_{j+1}, r_0)$
12. $\quad \beta_j = \frac{\rho_{j+1}}{\rho_j} \frac{\alpha_j}{\omega_j}$
13. $\quad p_{j+1} = r_{j+1} + \beta_j (p_j - \omega_j v)$

    end

Рис. 1. Классический вариант метода BiCGStab.

Алгоритм классического метода BiCGStab [2] приведен на рис. 1. Каждая итерация этого метода включает в себя две операции умножения разреженной матрицы на вектор (и две операции предобуславливания для предобусловленного варианта метода), четыре скалярных произведения векторов и двенадцать операций сложения векторов/умножения на число. При традиционном подходе к распараллеливанию итерационных градиентных методов с сильно-разреженной матрицей, исходная матрица системы и вектора распределяются между вычислительными процессами построчно [5], и каждый из локальных фрагментов матрицы также разбивается на блоки пропорционально количеству строк, пришедшихся на каждый из вычислительных процессов. Такая схема распределения данных между вычислительными процессами позволяет эффективно организовать вычисление операций сложения векторов/умножения на число и умножение матрицы на вектор: операции с векторами выполняются локально, тогда как время на пересылку данных для вычисления произведения матрицы на вектор может быть эффективно «замаскировано» вычислениями с локальным фрагментом вектора. Наибольшие сложности возникают при распараллеливании скалярных произведений векторов, поскольку после вычисления скалярного произведения локального фрагмента вектора требуется взаимодействие всех процессов для обмена данными каждого вычислительного процесса с каждым. Такое взаимодействие между процессами может быть реализовано как одна коллективная операция или набор

коммуникаций «точка-точка» для каждого процесса с каждым. Второй вариант может быть предпочтительным, если алгоритмически имеется возможность организовать порядок вычислений так, что время на выполнение этих коммуникаций будет «замаскировано» другими вычислениями. В противном случае операция скалярного произведения векторов является точкой глобальной синхронизации всех вычислительных процессов, что нежелательно при реализации параллельной программы.

Анализируя классический алгоритм, представленный на рис. 1, можно обратить внимание, что последовательность вычислений в данном методе строго последовательна, то есть для выполнения вычислений *i*-й строки алгоритма, необходимо, чтобы были завершены вычисления всех предыдущих *i-1* строк. Таким образом, присутствие нескольких скалярных произведений векторов, которые не могут быть алгоритмически переставлены, а время на коммуникации для завершения этих операций «замаскировано» другими вычислениями, приводит к наличию трех точек глобальной синхронизации вычислений, и как следствие, к низкой эффективности параллельной реализации исходного варианта метода.

Известно несколько работ [6, 7], посвященных попыткам переформулировать исходный непредобусловленный метод BiCGStab с целью уменьшить количество операций скалярного произведения векторов, приводящих к глобальной синхронизации вычислительных процессов. В [6] предложен вариант метода с двумя глобальными точками синхронизации при вычислении скалярных произведений векторов, а в [7] позднее был предложен вариант с одной точкой синхронизации. Основываясь на идеях и подходах, изложенных в [6, 7], в [8, 9] для предобусловленного варианта метода была предложена модификация, которая позволяет избежать глобальной синхронизации всех вычислительных процессов на каждой итерации. Предложенный переупорядоченный метод имеет только четыре дополнительных операции сложения векторов/умножения на число по сравнению с исходным алгоритмом, в то время как IBiCGStab [7] требует два дополнительных скалярных произведения и шесть операций с векторами. Скорость сходимости модифицированного метода идентична скорости сходимости предобусловленного классического метода, поскольку в предложенной модификации изменена ***только*** последовательность вычисления некоторых векторов. Отличительной же чертой этого метода является полное отсутствие точек глобальной синхронизации процессов. Скалярные произведения векторов сгруппированы в три блока, причем для каждого из них имеется возможность «замаскировать» время обмена данными вычислением предобуславливания, которое, зачастую, является наиболее трудоемкой операцией на итерации. Как результат, это позволяет значительно улучшить эффективность параллельной реализации данного алгоритма.

Исследование эффективности предложенной в [8, 9] модификации алгоритма было проведено на вычислительных системах СКИФ МГУ «Чебышёв» и «Ломоносов». Для этого была разработана гибридная параллельная программа, реализующая предложенный метод с алгебраическим многосеточным методом в качестве предобуславливания (для построения предобуславливания использована свободно-распространяемая библиотека численных методов hypre [10]), с взаимодействием процессов между узлами через MPI и взаимодействием через POSIX Shared Memory для вычислительных процессов на одном узле. Для хранения исходной матрицы и промежуточных матриц интерполяции использованы специализированные форматы CRS и DMSR (например, [11]).

Обычно, для построения оптимального переупорядочивания применяются алгоритмы разбиения графов или гиперграфов, которые строятся по исходной матрице системы, подлежащей переупорядочиванию. Был проведен ряд исследований по изучению эффективности различных алгоритмов разбиения графов, реализованных в свободно-распространяемых библиотеках METIS, hMETIS, ParMETIS, Scotch, PaToH и ряде других. По результатам исследований были выбраны библиотека Scotch и Pt-Scotch для разбиения локального и распределенного графов соответственно.

Тестовыми задачами для исследуемого метода выбраны:

1. задача Неймана для уравнения Пуассона с постоянной правой частью в кубической области на равномерной сетке;
2. задача Неймана для уравнения Пуассона для давления с постоянной правой частью в расчетной области, представляющей собой канал с препятствием на стенке в виде куба (задача обтекания куба турбулентным потоком [12]; 5.45 млн. ячеек).

При исследовании масштабируемости метода для завершения итерационного процесса, во избежание некоторых флуктуаций количества итераций до выполнения соответствующего условия, использовано ограничение на количество итераций метода, тогда как для практических расчетов критерием завершения итерационного процесса задано условие:

$$\frac{\|x_{j+1}-x_j\|}{\|x_j\|}<10^{-6}.$$

В качестве характеристики эффективности параллельной программы использовано ускорение времени решения задачи, определяемое как отношение времени решения задачи на одном вычислительном ядре (процессоре) ко времени решения на заданном количестве вычислительных ядер (процессоров):

$$P_N=\frac{T_1}{T_N}.$$

Первоначально, была реализована параллельная программа с взаимодействием вычислительных процессов только посредством MPI. На рис. 2 представлены результаты исследования ускорения времени работы программы на тестовой задаче в кубической расчетной области размером 8 млн. ячеек (количество ядер в легенде графика соответствует количеству вычислительных процессов, запущенных на каждом из узлов вычислительной системы). Следует отметить относительно слабые результаты для большого количества вычислительных процессов на узлах. Согласно приведенных значений, выигрыш от использования всех восьми вычислительных ядер на каждом узле при количестве используемых узлов до 16 оказывается крайне незначительным в сравнении с 4 ядрами, а при большем количестве используемых узлов он становится лишь отрицательным. Максимальное ускорение в такой конфигурации не превышает 35 раз. Эффективность использования 4 ядер с каждого узла оказывается несколько лучшей, однако насыщение также достигается на отметке 48 узлов (196 вычислительных ядер) и ускорении в 66 раз. Наиболее удачной оказывается конфигурация, при которой задействованы лишь два вычислительных ядра с каждого узла: на малом количестве узлов преимущества от использования большего количества ядер оказываются незначительными, зато насыщение коммуникационной сети, соединяющей вычислительные узлы, наступает значительно позже. В такой конфигурации получено ускорение до 140 раз на 112 вычислительных узлах (224 ядра), причем до этой отметки зависимость ускорения от количества узлов носит практически линейный характер. Для каждого из вариантов оптимальным оказывается использование порядка 200-250 вычислительных процессов, что эквивалентно локальной области около 30-35 тыс. строк исходной матрицы, приходящейся на один вычислительный процесс.

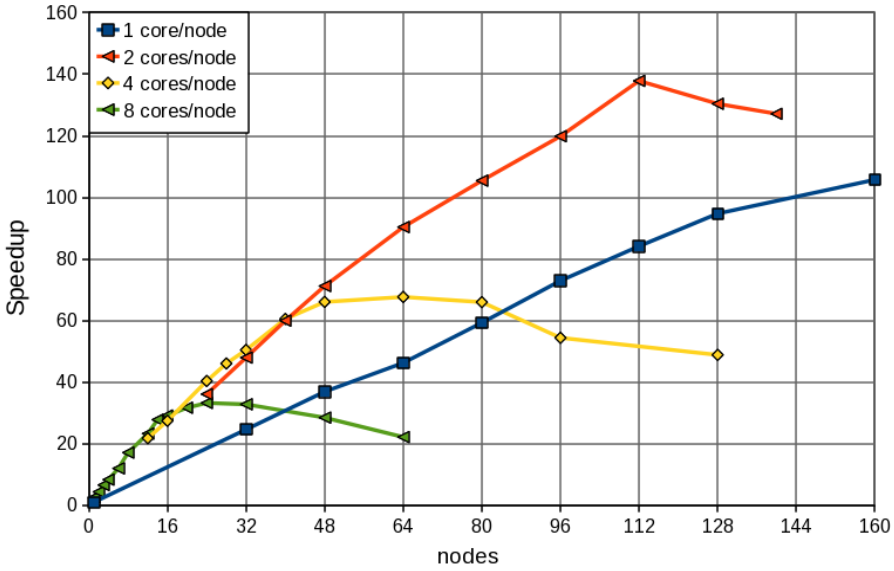

Рис. 2. Ускорение времени решения системы уравнений в зависимости от количества узлов и вычислительных процессов на каждом из узлов (hypre 2.4.0b, «Чебышев»).

После анализа первых результатов тестирования, было принято решение реализовать гибридную модель, заменив обмены между процессами внутри узлов через MPI на обмены через общую память вычислительного узла, что потенциально должно было улучшить ситуацию с использованием всех имеющихся вычислительных ядер в узлах и уменьшить нагрузку на коммуникационную сеть. Вместе с тем, можно отметить, что полученные результаты исследования эффективности программы и метода продемонстрировали высокий потенциал предложенной в [8, 9] модификации метода.

Такая версия программы была протестирована на задаче для расчетной области обтекания куба [12]. Результаты соответствующих исследований приведены на рис. 3 (количество тредов в легенде графика соответствует количеству MPI-процессов внутри узла, взаимодействие между которыми реализовано через общую память).

Сравнивая рис. 2 и 3, можно отметить, что для второй тестовой задачи выигрыш от использования всех имеющихся восьми ядер с каждого узла при использовании только MPI-обменов (кривая на графике для одного треда) оказывается еще меньшим, чем в предыдущем случае, и ускорение не превышает 20 раз, что можно объяснить уменьшением локальной области, приходящейся на каждый вычислительный процесс до 28 тыс. строк матрицы. Вместе с тем, представленный график наглядно демонстрирует преимущество гибридного подхода к реализации алгоритма, который позволяет существенно снизить нагрузку на коммуникационную сеть кластера и добиваться эффективной утилизации всех имеющихся процессорных ядер на узлах. Пиковое ускорение для данной тестовой задачи составило 156 раз (0.77 секунды) в сравнении с последовательно исполняемой программой (120.1 секунды) при использовании 224 узлов (1792 ядра). Локальная область данных

исходной матрицы, приходящейся на каждый процесс, составила примерно 3 тысячи строк, что является крайне малым показателем для таких задач.

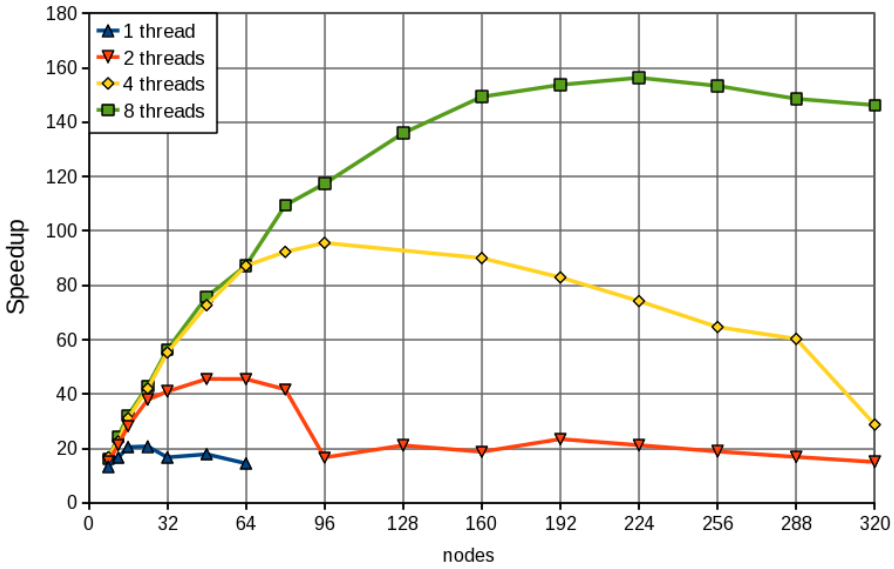

Рис. 3. Ускорение времени решения системы уравнений в зависимости от количества узлов и вычислительных процессов, объединенных через общую память на каждом из узлов (hypre 2.4.0b, «Чебышев»).

На рис. 4 приведен ряд результатов исследования масштабируемости решателя для тестовых матриц дискретного аналога уравнения Пуассона в кубической области с различным размером пространственного шага расчетной сетки[1]. Расчеты проведены для матриц размером 8, 27, 42.9 и 64 млн. строк. В данном случае приведены результаты ускорения, нормированные на время расчета на 32 вычислительных ядрах (4 узла), поскольку оперативной памяти одного вычислительного узла для решения таких задач оказалось недостаточно.

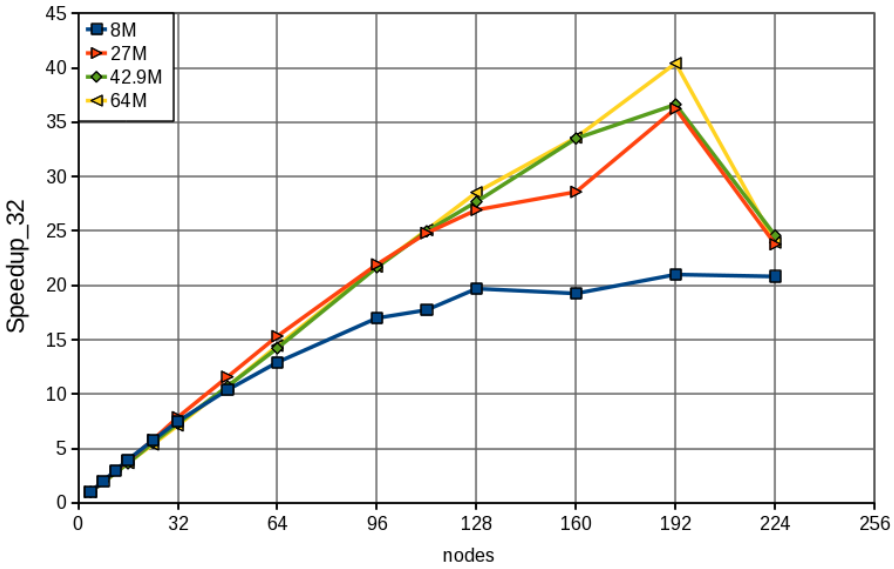

Рис. 4. Ускорение времени решения системы уравнений в зависимости от количества узлов и размера тестовых матриц относительно времени расчета на 4 узлах (32 вычислительных ядра; hypre 2.6.0b, «Чебышев»).

Для тестовых матриц 27 млн. строк и более наблюдается близкая к линейной масштабируемость решателя вплоть до 192 узлов. Поведение кривых при увеличении количества использованных узлов до 224 несколько странно, и, вероятно, вызвано особенностями работы аппаратной платформы во время проведения тестов. При увеличении размера тестовой матрицы наблюдается некоторое улучшение показателей масштабируемости решателя, что представляется вполне ожидаемым благодаря увеличению размеров локальных областей данных каждого из процессов.

[1]Здесь и далее в расчетах использована обновленная версия библиотеки hypre 2.6.0b, в которой был расширен набор параметров для построения алгебраического многосеточного предобуславливателя.

Основываясь на практически линейном поведении кривых ускорения на малом количестве узлов и приняв за исходные результаты масштабируемости для матрицы размером 8 млн. строк, можно получить оценочные результаты ускорения для использованных матриц относительно последовательно исполняемых вариантов. Для этого коэффициент ускорения был определен как

$$P_N = \frac{T_1}{T_{32}} \frac{T_{32}}{T_N} = K_{32} \frac{T_{32}}{T_N},$$

где коэффициент K_{32} вычислен по расчетам для матрицы 8 млн. строк и равен $K_{32} = 9.27$. Пиковые значения такой оценочной экстраполяции приведены в табл. 1. С учетом той особенности, что при увеличении размеров тестовой матрицы, наблюдается улучшение показателей масштабируемости на фиксированном количестве узлов, можно утверждать, что такой подход позволяет получить оценку «снизу», то есть реальные ускорения должны быть как минимум не меньше указанных в таблице.

Сравнительные результаты ускорения времени решения тестовой системы уравнений для матрицы размером 8 млн. строк на вычислительных системах «Чебышев» и «Ломоносов» приведены на рис. 5. Результаты для «Ломоносова» демонстрируют существенно лучшие показатели на малом количестве узлов, ввиду, в первую очередь, более эффективной работы ядер внутри вычислительного узла (ускорение в пределах 1 узла, в среднем, составляет 3.2 раза в сравнении с 2.4 для «Чебышева»), однако пиковые значения ускорения для этих расчетов оказываются примерно одинаковыми. Незначительное преимущество «Чебышева» может быть вызвано различиями в заданных параметрах предобуславливателя и использованием различных реализаций библиотеки MPI. Вместе с тем, если принять во внимание тот факт, что время решения системы уравнений на одном вычислительном ядре системы «Ломоносов» оказывается меньшим в 1.7 раза, то наблюдаемое насыщение кривых на меньшем количестве узлов представляется закономерным: время проведения вычислений с локальными фрагментами данных, сосредоточенных в каждом вычислительном процессе для «Ломоносова» оказывается существенно меньшим. С этой точки зрения, для сравнения вычислительных систем более показателен третий из приведенных на рис. 5 графиков, отражающий ускорение времени решения задачи относительно времени решения на одном ядре «Чебышева», который демонстрирует более чем 300-кратное ускорение.

Табл. 1. Пиковые значения масштабируемости для различных тестовых матриц (hypre 2.6.0b, «Чебышев»)

Размер матрицы, млн. строк	Оптимальное количество узлов	Пиковое ускорение
8	192	195
27	192	336
42,9	192	340
64	192	375

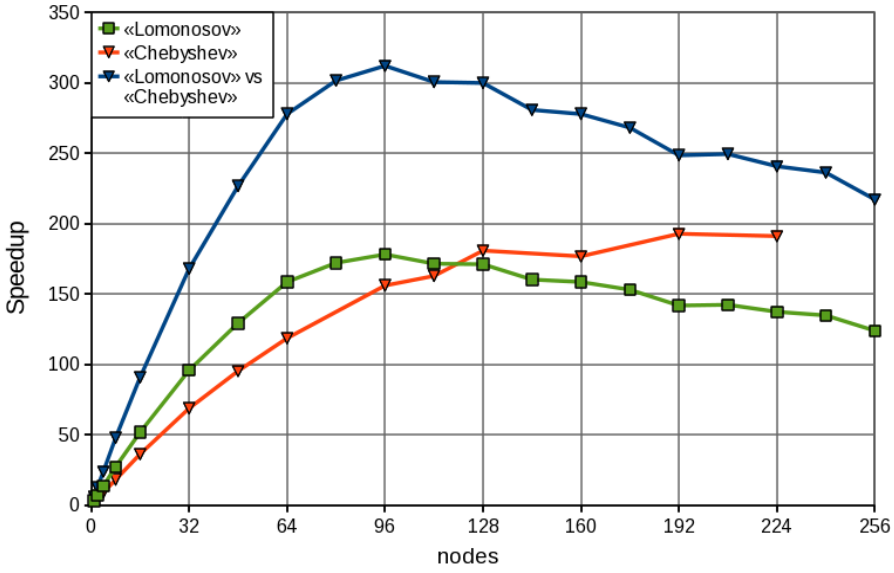

Рис. 5. Ускорение времени решения системы уравнений в зависимости от количества узлов для сетки 2003 на различных аппаратных платформах (тестовая матрица 8 млн. строк; hypre 2.6.0b).

Полученные на вычислительной системе «Ломоносов» результаты исследования масштабируемости решателя в зависимости от размера тестовой матрицы приведены на рис. 6. Как и для «Чебышева», из тех же

соображений, ускорение времени решения системы уравнений было определено относительно времени решения на 32 узлах (256 вычислительных ядер). Приведенный график демонстрирует близкую к линейной масштабируемость на больших матрицах вплоть до 512 узлов (4096 вычислительных ядер) и положительный эффект от увеличения задействованных ресурсов вплоть до 768 узлов (6144 вычислительных ядра). Аналогичным образом получены экстраполированные данные об абсолютных показателях масштабируемости метода и реализованного решателя, которые приведены в табл. 2, где соответствующий коэффициент K_{256} для матрицы 8 млн. строк составил $K_{256} = 95.93$. Результаты пикового ускорения показывают практически линейный рост относительно размера матрицы системы.

Времена решения различных тестовых систем уравнений и соответствующее количество итераций до сходимости решения приведены в табл. 3. Реализованный решатель демонстрирует слабую зависимость скорости сходимости итерационного процесса как от размера матрицы, так и от количества вычислительных ядер, что обеспечивает хорошие результаты масштабируемости не только в пересчете на одну итерацию метода, но и непосредственно при решении различных систем уравнений.

Заключение. В работе представлены результаты исследования масштабируемости предобусловленного переупорядоченного метода BiCGStab для решения систем линейных алгебраических уравнений. Благодаря использованному гибридному ходу (MPI+ShM) реализованный метод с алгебраическим многосеточным предобуславливанием демонстрирует хорошие результаты масштабируемости (более 3 порядков) для выбранных тестовых матриц дискретизации эллиптических уравнений на большом количестве вычислительных узлов: до 200 на вычислительной системе «Чебышев» и до 1024 и более на вычислительной установке «Ломоносов». Реализованный решатель демонстрирует слабую зависимость скорости сходимости итерационного процесса как от размера матрицы, так и от количества вычислительных ядер, что обеспечивает хорошие результаты масштабируемости при решении различных систем уравнений.

Представленная работа частично поддержана грантом РФФИ №09-08-00390-а.

Приведенные в работе результаты тестирования были получены на суперкомпьютерах СКИФ МГУ «Чебышев» и «Ломоносов» суперкомпьютерного комплекса МГУ им. М.В. Ломоносова. Автор благодарит Вл.В. Воеводина и С.А. Жуматия за предоставленную возможность проведения тестов на указанных вычислительных ресурсах и оперативную помощь при решении технических вопросов.

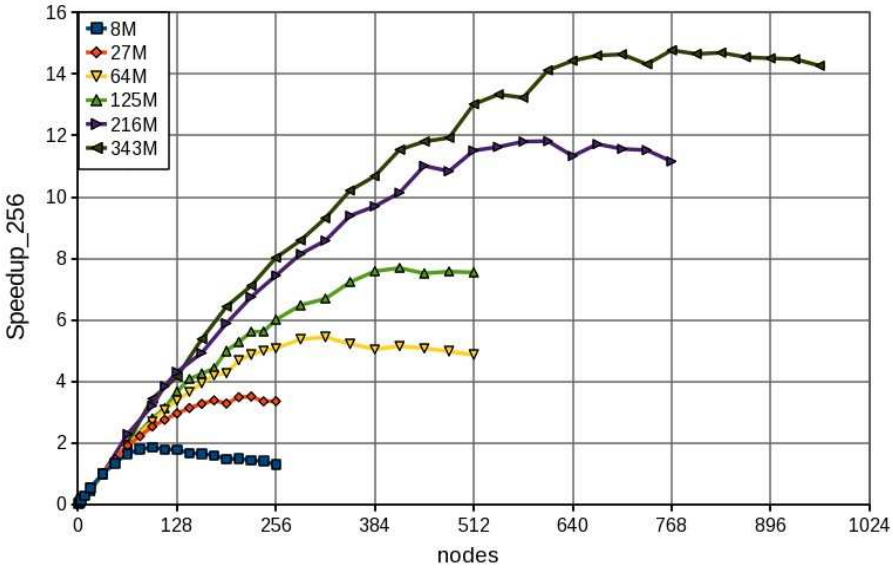

Рис. 6. Ускорение времени решения системы уравнений в зависимости от количества узлов и размера тестовых матриц относительно времени расчета на 32 узлах (256 вычислительных ядер; hypre 2.6.0b, «Ломоносов»).

Табл. 2. Пиковые значения масштабируемости для различных тестовых матриц (hypre 2.6.0b, «Ломоносов»)

Размер матрицы, млн. строк	Оптимальное количество узлов	Пиковое ускорение
8	96	178
27	224	337
64	320	522
125	416	738
216	608	1134

343	768	1417

Табл. 3. Время решения системы линейных алгебраических уравнений для различных тестовых матриц при использовании 128 узлов (1024 вычислительных ядра; hypre 2.6.0b, «Ломоносов»)

Размер матрицы, млн. строк	Количество итераций	Время решения, сек
8	10	0,19
27	11	0,4
64	12	0,88
125	13	1,69
216	14	3,13
343	14	5

ЛИТЕРАТУРА:

1. P.N. Swarztrauber. A direct method for the discrete solution of separable elliptic equations // SIAM Journal on Numerical Analysis, 1974, v. 11, pp. 1136-1150.
2. H. A. van der Vorst. BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems // SIAM Journal on Scientific and Statistical Computing, 1992, v. 13, pp. 631-644.
3. Y. Saad. Iterative methods for sparse linear systems, 2nd edition. SIAM, 2003, 528 p.
4. U. Trottenberg, C.W. Oosterlee, A. Schuller. Multigrid. New York, Academic Press, 2001, 631 p.
5. P. Arbenz, W. Petersen. Introduction to Parallel Computing - A practical guide with examples in C. Oxford Texts in Applied and Engineering Mathematics № 9. Oxford University Press, 2004, 278 p.
6. T. Jacques, L. Nicolas, C.Vollaire. Electromagnetic scattering with the boundary integral method on MIMD systems // High-Performance Computing and Networking, Lecture Notes in Computer Science, 1999, v. 1593, pp. 1025-1031.
7. L. Yang, R. Brent. The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures // Proceedings of Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02), 2002, pp. 324-328.
8. Б.И. Краснопольский. Модифицированный метод BiCGStab для высокопроизводительных вычислений // Конференция молодых ученых Института механики МГУ им. Ломоносова, (Москва, МГУ, октябрь, 2009) (в печати).
9. B. Krasnopolsky. The reordered BiCGStab method for distributed memory computer systems // Procedia Computer Science, 2010, v. 1, pp. 213-218. (ICCS 2010, May 31 - June 02, Amsterdam, the Netherlands).
10. Hypre: a library of high performance preconditioners. https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html
11. R. S. Tuminaro, M. A. Heroux, S. A. Hutchinson, J. N. Shadid. Official Aztec Users Guide, Version 2.1, 1999.
12. A. Yakhot, T. Anor, H. Liu, N. Nikitin. Direct numerical simulation of turbulent flow around a wall-mounted cube: spatio-temporal evolution of large-scale vortices // Journal of Fluid Mechanics, 2006, v. 566, pp. 1-9.