УДК 519.6

ИСПОЛЬЗОВАНИЕ УСЕЧЕННОГО ВАРИАНТА АЛГОРИТМА SPIKE БИБЛИОТЕКИ INTEL ADAPTIVE SPIKE-BASED SOLVER ДЛЯ РЕШЕНИЯ УПРУГОПЛАСТИЧЕСКИХ ЗАДАЧ

А. В. Толмачев[1], **А. В. Коновалов**[1], **А. С. Партин**[1]

Исследованы возможности усеченного алгоритма SPIKE библиотеки Intel Adaptive SPIKE-Based Solver для распараллеливания решения систем линейных уравнений, возникающих при дискретизации упругопластических задач. Исследование проводилось на примере задачи сжатия цилиндра. Расчеты выполнены на кластере um64 Института математики и механики УрО РАН. Работа выполнена в рамках программы Президиума РАН “Интеллектуальные информационные технологии, математическое моделирование, системный анализ и автоматизация”. Статья рекомендована к публикации Программным комитетом Международной научной конференции “Параллельные вычислительные технологии” (ПАВТ–2011; http://agora.guru.ru/pavt2011).

Ключевые слова: алгоритм SPIKE, системы линейных алгебраических уравнений, упругопластические задачи.

1. Постановка упругопластической задачи. Упругопластическая задача с большими пластическими деформациями физически и геометрически существенно нелинейна и требует больших затрат компьютерного времени. На решение двумерной задачи затрачивается несколько часов, для трехмерной задачи это время увеличивается до нескольких суток. Существенно сократить время вычислений можно с помощью техники параллельных вычислений, в частности решая такие задачи на кластерных системах.

Решение упругопластических задач методом конечных элементов осуществляется шагами по нагрузке. На каждом таком шаге процесс решения состоит из трех основных этапов [1]:

1) расчет локальных матриц жесткости для конечных элементов и формирование матрицы A (глобальной матрицы жесткости) и вектора F правой части системы линейных алгебраических уравнений

$$AX = F, \tag{1}$$

где X — искомый вектор обобщенной скорости в узлах конечно-элементной сетки;

2) решение линейной системы (1);

3) вычисление напряженно-деформированного состояния в конечных элементах в конце шага нагрузки.

Матрица A имеет ленточный вид. На каждом шаге нагрузки этап 1 осуществляется один раз, а этапы 2 и 3 — от десяти до пятнадцати раз для выполнения условия пластичности с требуемой точностью. В процессе решения матрица жесткости не меняется, а изменяется только правая часть системы уравнений.

Если этапы 1 и 3 легко распараллеливаются, то распараллеливание процесса решения линейной системы является сложной задачей. Решение этой задачи итерационными методами рассмотрено в работе [2]. Исследование эффективности распараллеливания трехдиагонального алгоритма LU-разложения из библиотеки ScaLAPACK [3] при решении получающейся линейной системы приведено в [4].

Цель настоящей статьи состоит в исследовании возможностей параллельного алгоритма SPIKE [5, 6] для решения линейных систем в упругопластических задачах на кластерных системах.

Все численные эксперименты проводились на кластерной системе um64 Института математики и механики УрО РАН на примере решения методом конечных элементов задачи сжатия плоскими плитами цилиндра из упругопластического изотропного и изотропно-упрочняемого материала, постановка которой приведена в работе [2].

2. Описание алгоритма SPIKE. Решатель систем линейных уравнений Intel Adaptive SPIKE-Based Solver доступен по адресу http://software.intel.com/en-us/articles/intel-adaptive-spike-based-solver/. Лежащий в его основе алгоритм SPIKE для решения системы с ленточной матрицей состоит из трех

[1] Институт машиноведения Уральского отделения РАН, ул. Комсомольская, 34, 620049, г. Екатеринбург; А. В. Толмачев, инженер, e-mail: tolmachev.arseny@gmail.com; А. В. Коновалов, зав. лаб., e-mail: avk@imach.uran.ru; А. С. Партин, ст. науч. сотр., e-mail: lmd@imach.uran.ru

этапов: разбиение системы, разложение матрицы системы и решение системы с разложенной матрицей. На первом этапе систему уравнений разделяют на участки, удобные для дальнейшей работы, на втором этапе производится разложение матрицы системы на более удобные для решения матрицы, а на последнем этапе происходит непосредственно решение системы с использованием полученных матриц.

2.1. Разбиение системы. Рассмотрим систему (1) с ленточной матрицей A размерности $n \times n$ с узкой лентой и матрицей-столбцом F. Разобьем матрицы A и F на p участков горизонтальными линиями и распределим i-й участок разбиения на i-й процессор.

Рисунок 1 показывает распределение матриц A и F для $p = 4$. Выделим на i-м участке разбиения три блока: диагональный квадратный блок A_i $(i = 1, \ldots, p)$, который имеет порядок n_i; наддиагональный квадратный блок B_i $(i = 1, \ldots, p-1)$ и поддиагональный квадратный блок C_i $(i = 2, \ldots, p)$. Поскольку лента матрицы A узкая, то порядки блоков B_i и C_i, равные m, будут много меньше n_i.

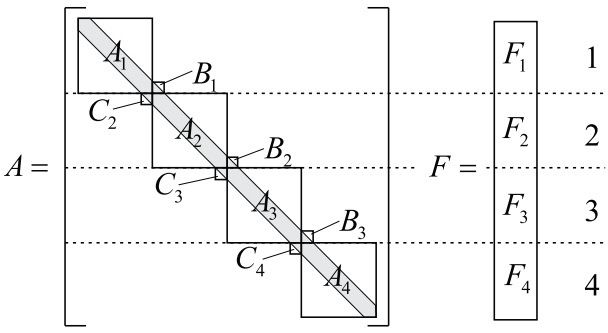

Рис. 1. Распределение матрицы системы A и правой части F на 4 процессора

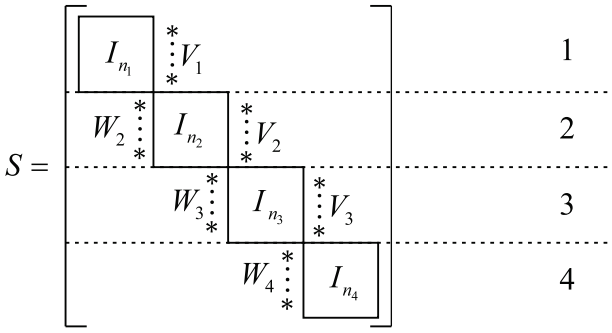

Рис. 2. Матрица S, распределенная на 4 процессора

2.2. Разложение матрицы системы. На этом этапе вычисляется разложение матрицы системы A в виде

$$A = DS, \tag{2}$$

где D — матрица, составленная только из диагональных блоков A_i, $D = \operatorname{diag}(A_1, \ldots, A_p)$, а матрица S, показанная на рис. 2, вычисляется путем умножения i-го участка разбиения матрицы A на матрицу A_i^{-1} слева. Матрица S имеет на диагонали единичные матрицы I_{n_i} порядка n_i, а ее внедиагональные блоки состоят из матриц V_i $(i = 1, \ldots, p-1)$ и W_i $(i = 2, \ldots, p)$. Матрицы V_i и W_i называют шипами (spikes) из-за того, что они имеют размерность $n_i \times m$, т.е. являются узкими и высокими.

Шипы V_i и W_i находятся из решения матричного уравнения

$$A_i[V_i\, W_i] = \begin{bmatrix} 0 & C_i \\ \vdots & 0 \\ 0 & \vdots \\ B_i & 0 \end{bmatrix}. \tag{3}$$

2.3. Решение системы с разложенной матрицей. После разложения матрицы A в виде (2) решение системы (1) сводится к последовательному решению двух систем

$$DG = F, \tag{4}$$

$$SX = G. \tag{5}$$

Решение системы (4) может быть выполнено полностью параллельно, поскольку ее матрица состоит лишь из диагональных блоков.

Представим шипы V_i и W_i следующим образом: $V_i = \left[V_i^t, V_i^r, V_i^b\right]^{\mathrm{T}}$ и $W_i = \left[W_i^t, W_i^r, W_i^b\right]^{\mathrm{T}}$, где V_i^t, V_i^r, V_i^b и W_i^t, W_i^r, W_i^b — соответственно верхние m, средние $n_i - 2m$ и нижние m рядов шипов V_i и W_i. Аналогично локальные части матрицы неизвестных и матрицы правой части X_i и G_i разбиваются в виде $X_i = \left[X_i^t, X_i^r, X_i^b\right]^{\mathrm{T}}$ и $G_i = \left[G_i^t, G_i^r, G_i^b\right]^{\mathrm{T}}$. Из системы (5) формируется сокращенная система

$$\widehat{S}\widehat{X} = \widehat{G}, \tag{6}$$

состоящая из m рядов матричного уравнения (5), находящихся выше и ниже линий разделения матриц этой системы (это возможно, поскольку $m \ll n$). Порядок этой системы равен $2m(p-1)$. Матрица сокращенной системы при распределении исходной системы уравнений на 4 процессора показана на рис. 3. В систему (6) входят только части матриц V_i, W_i, X_i и G_i с верхними индексами b и t. Блоки I_m являются единичными матрицами порядка m.

Матрица системы (6) является блочно-трехдиагональной с $(p-1)$ диагональными блоками, i-й из которых имеет вид $\begin{bmatrix} I_m & V_i^b \\ W_{i+1}^t & I_m \end{bmatrix}$. Левый и правый i-й внедиагональные блоки соответственно выглядят следующим образом: $\begin{bmatrix} W_i^b & 0 \\ 0 & 0 \end{bmatrix}$ и $\begin{bmatrix} 0 & 0 \\ 0 & V_{i+1}^t \end{bmatrix}$. Блок неизвестных и правая часть, соответствующие i-му диагональному блоку, имеют вид $\left[X_i^b, X_{i+1}^t\right]^{\mathrm{T}}$ и $\left[G_i^b, G_{i+1}^t\right]^{\mathrm{T}}$.

$$\hat{S} = \begin{array}{|c|c|c|c|c|c|}
\hline
I_m & V_1^b & & & & \\ \hline
W_2^t & I_m & & V_2^t & & \\ \hline
W_2^b & & I_m & V_2^b & & \\ \hline
 & & W_3^t & I_m & & V_3^t \\ \hline
 & & W_3^b & & I_m & V_3^b \\ \hline
 & & & & W_4^t & I_m \\ \hline
\end{array}$$

Рис. 3. Вид матрицы сокращенной системы для случая распределения матрицы A на 4 процессора

После вычисления решения системы (6) общее решение системы (1) вычисляется по формулам

$$X_1^r = G_1^r - V_1^r X_2^t, \quad X_i^r = G_i^r - V_i^r X_{i+1}^t - W_i^r X_{i-1}^b, \quad i = 2, \ldots, p-1, \quad X_p^r = G_p^r - W_p^r X_{p-1}^t.$$

2.4. Усеченный вариант алгоритма SPIKE. Если матрица системы A имеет диагональное преобладание, т.е. выполняется условие $|a_{ii}| > \sum\limits_{i \neq j} |a_{ij}|$, то шипы W_i и V_i практически полностью состоят из нулей [5]. Это позволяет при решении системы (6) отбросить ее внедиагональные блоки. Полученная система называется усеченной. Тогда достаточно вычислять только те части шипов, которые входят в диагональные блоки системы (6).

2.5. Распараллеливание процесса решения. Этап разложения матрицы выполняется полностью параллельно, поскольку для вычислений достаточно данных, находящихся на локальном процессоре. При этом происходит вычисление LU-разложения [7] блоков A_i, после этого вычисляются шипы как решение матричного уравнения (3). Решение системы (4) также выполняется полностью параллельно, так как матрица D состоит только из диагональных блоков.

Решение системы (6) в стандартном варианте алгоритма может быть выполнено различными способами, например, итерационными методами, применением алгоритма SPIKE рекурсивно и др. Если матрица A имеет диагональное преобладание, то вместо системы (6) решается ее усеченный вариант без внедиагональных блоков. Тогда в процессе решения требуется передать лишь блок W_{k+1}^t с $k+1$-го на k-й процессор и получить обратно блок решений, что увеличивает степень параллельности данного алгоритма. Измененный таким образом вариант алгоритма называется усеченным алгоритмом SPIKE.

В нашем случае матрица системы не имеет диагонального преобладания, поскольку "степень" ее диагонального преобладания δ, которая вычисляется как $\delta = \dfrac{|a_{ii}|}{\sum\limits_{i \neq j} |a_{ij}|}$, имеет порядок 0.4 . Поэтому мы сочли возможным использовать усеченный вариант алгоритма с контролем точности решения. Вычислительные эксперименты показали, что в нашем случае значение $\|F - AX\|_\infty$ имело порядок 10^{-12}.

3. Особенности использования библиотеки из языка C++. Библиотека Intel Adaptive SPIKE Solver (версия от 23.02.2010 г.) рассчитана для использования из языков FORTRAN 90/95 и C, однако при использовании этой библиотеки из языка C++ возник ряд проблем.

3.1. Ошибки компоновки при использовании библиотеки Intel Adaptive SPIKE-Based Solver из языка C++. При использовании из языка C++ скомпилированных библиотек, написанных на языке C, требуется, чтобы экспортируемые из данной библиотеки функции были помечены с использованием директивы extern "C". Это приведет к тому, что компилятор языка C++ будет игнорировать особенности разрешения имен, специфические для языка C++, такие как пространства имен, функции-члены, перегрузку функций и шаблоны. В заголовочных файлах библиотеки Intel Adaptive SPIKE Solver данная директива отсутствует, поэтому при компоновке возникают ошибки невозможности разрешения имен. Это можно исправить либо добавлением указанных директив в заголовочные файлы, либо включать заголовочный файл. Ниже показан способ включения заголовочных файлов библиотеки Intel Adaptive

SPIKE-Based Solver из языка C++:

```
extern "C" {
#include <spike.h>
}
```

3.2. Параллельное транспонирование ленточной матрицы. Библиотека Intel Adaptive SPIKE-Based Solver написана на языке FORTRAN, в котором исторически матрицы представляются в виде двумерного массива по столбцам. В языках C и C++, в свою очередь, предполагается, что матрицы представлены по строкам. При вызове функций библиотеки Intel Adaptive SPIKE-Based Solver в качестве данных требуется указывать описатели динамически выделяемых массивов языка FORTRAN. В поставке библиотеки присутствуют функции выделения, удаления и доступа к элементам таких массивов, однако это приводит к увеличению потребления памяти, поскольку сформированную матрицу жесткости требуется переносить из массива, в который она формируется, в массив, понятный среде исполнения FORTRAN.

Для экономии памяти можно транспонировать полученную матрицу и заполнить структуры данных, описывающие эту матрицу для среды выполнения языка FORTRAN.

Операция параллельного транспонирования ленточной матрицы M размерности $n \times n$ с лентой полуширины β, хранимой в сжатой ленточной форме, происходит следующим образом. Допустим, что матрица M разбита на p матриц M_i порядка n_i, $i = 1, \ldots, n$, $n_i > 2\beta$. Каждую матрицу M_i можно разбить на подматрицы H_i $(i = 1, \ldots, n)$, которые можно транспонировать независимо друг от друга, и на подматрицы J_i $(i = 2, \ldots, n)$ и K_i $(i = 1, \ldots, n-1)$, данные в которых надо поменять местами для транспонирования матрицы M. Пример разбиения матрицы на три матрицы показан на рис. 4. Для улучшения параллельных свойств алгоритма требуется производить вычисления таким образом, чтобы иметь возможность совмещать вычисления и передачу данных. Таким образом, для i-го процессора параллельный распределенный алгоритм транспонирования ленточной матрицы будет выглядеть следующим образом.

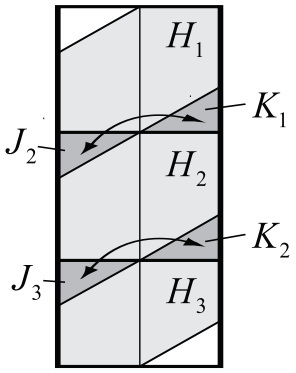

Рис. 4. Разбиение матрицы M для транспонирования при $p = 3$

Скопировать содержимое матриц J_i и K_i в буферы для передачи данных. Запустить асинхронную передачу матрицы J_i на процессор с номером $i-1$ и матрицы K_i — на процессор $i+1$, а также запустить асинхронный прием матриц K_{i-1} и J_{i+1}. Транспонировать матрицу H_i. Завершить асинхронную передачу данных или дождаться ее завершения. Скопировать полученное содержимое матрицы J_{i-1} на место матрицы K_i и содержимое матрицы K_{i+1} на место матрицы J_i.

4. Результаты вычислительных экспериментов. Решалась задача конечно-элементного моделирования процесса сжатия упругопластического цилиндра при использовании регулярной конечно-элементной сетки с одинаковым количеством разбиений d по обеим осям.

Параметры сеток

d	β	n	β/n
100	205	20402	0.0100
150	305	45602	0.0067
200	405	80802	0.0050
300	605	181202	0.0033

Вычислительные эксперименты проводили на кластере um64 Института математики и механики УрО РАН. Кластер состоит из 32 двухпроцессорных двухъядерных модулей на базе процессоров AMD Operton с тактовой частотой 2.6 ГГц. Для вычислительных экспериментов использовали версию библиотеки Intel Adaptive SPIKE-Based Solver от 23 февраля 2010 г. и библиотеку ScaLAPACK, реализация которой входит в Intel MKL версии 10.0.010. Для межпроцессорной коммуникации использовалась библиотека OpenMPI версии 1.3.3 на сети InfiniBand. Для тестирования были взяты сетки с параметрами, представленными в таблице, β — полуширина матрицы жесткости.

Представленные значения ускорений вычислялись по формуле $s = t_n/t_1$, где t_1 — время выполнения операции на одном процессоре, а t_n — время выполнения операции на n процессорах. Для сетки с $d = 300$ за t_1 мы приняли t_2, поскольку сформированная матрица жесткости не помещалась в память одного процессора используемой кластерной системы.

4.1. Производительность усеченного алгоритма SPIKE при решении упругопластической задачи. На рис. 5–7 представлена зависимость значения ускорений усеченного алгоритма SPIKE от количества процессоров p и количества разбиений сетки d соответственно для этапа разложения матрицы системы, этапа решения линейной системы с уже разложенной матрицей и этапа полного решения системы на шаге нагрузки упругопластической задачи. При полном решении линейной системы выполняются одно разложение и 15 решений с разложенной матрицей системы.

Из рис. 5–7 видно, что рост производительности имеет место как при увеличении количества процессоров, так и при увеличении количества разбиений конечно-элементной сетки. Уменьшение производительности при использовании двух процессоров, наблюдающееся на рис. 6, а также более пологий наклон графиков на этом рисунке по сравнению с графиками для этапа разложения матрицы для небольших сеток, вызваны затратами времени на контроль точности решения.

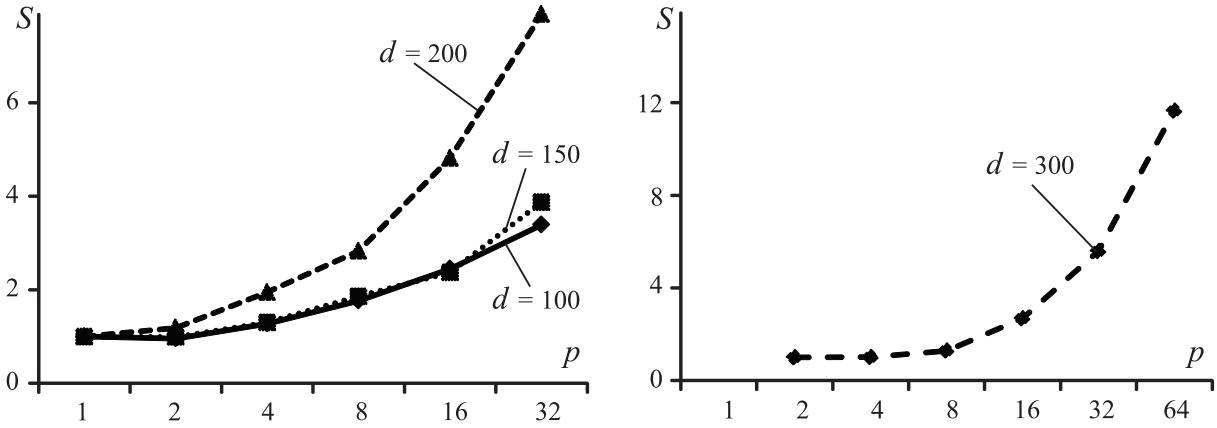

Рис. 5. Зависимость ускорений на этапе разложения матрицы системы алгоритма SPIKE от количества процессоров p и количества разбиений сетки d

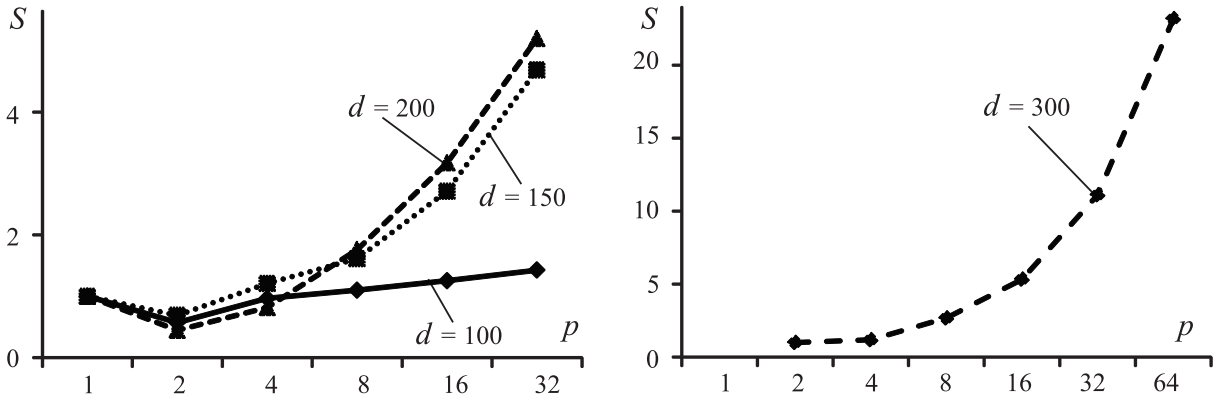

Рис. 6. Зависимость ускорений на этапе решения линейной системы с разложенной матрицей алгоритма SPIKE от количества используемых процессоров p и количества разбиений d

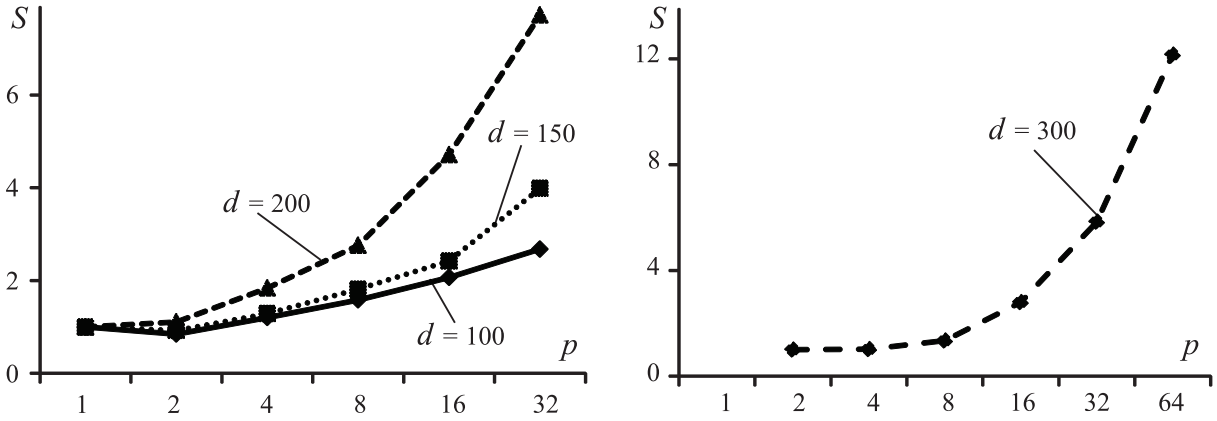

Рис. 7. Зависимость ускорений полного решения линейной системы внутри шага нагрузки упругопластической задачи для случая одного разложения и 15 решений

4.2. Сравнение производительности усеченного алгоритма SPIKE и трехдиагонального алгоритма LU-разложения из библиотеки ScaLAPACK. На рис. 8 предоставлено сравнение

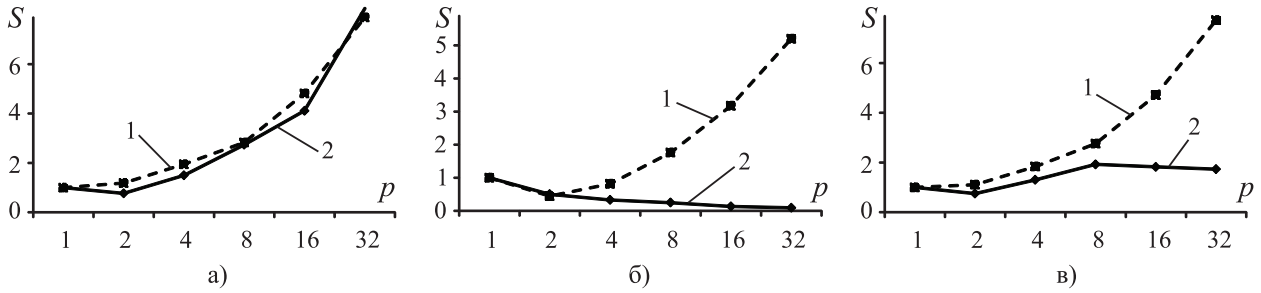

Рис. 8. Ускорение разложения (а), решения линейной системы (б) и полного решения системы (в) для сетки с количеством разбиений $d = 200$: 1) SPIKE; 2) ScaLAPACK

значений ускорений s решения линейной системы с использованием трехдиагонального параллельного алгоритма LU-разложения из библиотеки ScaLAPACK и усеченного алгоритма SPIKE, полученные при решении упругопластической задачи сжатия цилиндра с количеством разбиений $d = 200$. Исследование применимости трехдиагонального алгоритма LU-разложения для решения упругопластической задачи рассмотрено в работе [4].

Из рис. 8 следует, что при разложении матрицы системы усеченный алгоритм SPIKE на сетке с количеством разбиений $d = 200$ имеет приблизительно ту же производительность, что и трехдиагональный алгоритм LU-разложения из библиотеки ScaLAPACK. На этапе решения системы уравнений с использованием разложенной матрицы усеченный алгоритм SPIKE имеет значительно лучшую производительность, чем трехдиагональный алгоритм. Трехдиагональный алгоритм LU-разложения имеет коэффициент ускорения $s < 1$ при использовании более одного процессора, однако усеченный алгоритм SPIKE имеет коэффициент ускорения $s > 1$ при использовании более 4 процессоров. При дальнейшем увеличении количества процессоров коэффициент ускорения продолжает увеличиваться.

СПИСОК ЛИТЕРАТУРЫ

1. *Поздеев А.А., Трусов П.В., Няшин Ю.И.* Большие упруго-пластические деформации. М.: Наука, 1986.
2. *Демешко И.П., Акимова Е.Н., Коновалов А.В.* Применение параллельных алгоритмов для решения системы линейных алгебраических уравнений с ленточной матрицей итерационными методами на кластерной системе // Тр. междунар. конф. “Параллельные вычислительные технологии (ПаВТ-2009)”. Нижний Новгород. 30 марта–3 апреля 2009. Челябинск: Изд-во ЮУрГУ, 2009. 444–449.
3. *Blackford L.S., Choi J., Cleary A., D’Azeuedo E., et al.* ScaLAPACK User’s Guide. 1997. URL: http://www.netlib.org/scalapack/slug.
4. *Коновалов А.В., Толмачев А.В., Партин А.С.* Опыт применения параллельного алгоритма LU-разложения для решения линейных систем уравнений в упругопластических задачах // Тр. междунар. конф. “Параллельные вычислительные технологии (ПаВТ-2010)”. Уфа. 29 марта–2 апреля 2010. Челябинск: Изд-во ЮУрГУ, 2010. 498–506.
5. *Polizzi E., Sameh A.* SPIKE: A parallel environment for solving banded linear systems // Computers & Fluids. 2007. **36**. 113–120.
6. *Polizzi E., Sameh A.* A parallel hybrid banded system solver: the SPIKE algorithm // Parallel Comput. 2006. **32**. 177–194.
7. *Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л., Штайн К.* Алгоритмы: построение и анализ. М.: Вильямс, 2008.

Поступила в редакцию
14.03.2011