Использование усечённого варианта алгоритма SPIKE из библиотеки Intel Adaptive SPIKE-Based Solver для решения упругопластической задачи*

А.В. Толмачев, А.В. Коновалов, А.С. Партин

Институт Машиноведения УрО РАН

Исследованы возможности усечённого алгоритма SPIKE из библиотеки Intel Adaptive Spike-Based Solver для распараллеливания решения системы линейных уравнений внутри упругопластической задачи. Исследование проводилось на примере задачи сжатия цилиндра. Расчёты проводились на кластере um64 Института математики и механики УрО РАН. Решён ряд проблем, возникающих при использовании этой библиотеки из языка C++.

1. Постановка упругопластической задачи

Упругопластическая задача с большими пластическими деформациями физически и геометрически существенно нелинейная и требует большого количества времени для ее решения на персональном компьютере. 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 данные задачи на кластерных системах.

Решение упругопластических задач методом конечных элементов осуществляется шагами по нагрузке, и на каждом таком шаге 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 (СЛАУ)

$$AX = F \qquad (1)$$

относительно искомого вектора X обобщённой скорости в узлах конечно-элементной сетки;

2. решение СЛАУ (1);

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

Матрица A имеет ленточный вид. На каждом шаге нагрузки этап 1 выполняется один раз, а этапы 2 и 3 – от десяти до пятнадцати раз для удовлетворения итерационно с приемлемой точностью условию пластичности. При этом матрица жёсткости не меняется, а изменяется только правая часть системы уравнений.

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

Целью работы является исследование возможностей параллельного алгоритма SPIKE [5, 6] для решения СЛАУ в упругопластических задачах на кластерной системе.

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

*Работа выполнена в рамках программы Президиума РАН "Интеллектуальные информационные технологии, математическое моделирование, системный анализ и автоматизация".

2. Описание алгоритма SPIKE

Использовали решатель систем линейных уравнений Intel Adaptive Spike-Based Solver (библиотеку программ), доступный по адресу http://software.intel.com/en-us/articles/intel-adaptive-spike-based-solver/. Лежащий в его основе алгоритм SPIKE для решения СЛАУ с ленточной матрицей состоит из трёх этапов: разбиения системы, разложения матрицы системы и решения системы с разложенной матрицы. На первом этапе систему уравнений разделяют на участки, удобные для дальнейшей работы, на втором этапе производится разложение матрицы системы на более удобные для решения СЛАУ матрицы, а на последнем этапе происходит непосредственно решение СЛАУ с использованием полученных матриц.

2.1. Разбиение системы

Рассмотрим СЛАУ вида (1) с ленточной матрицей A размерности $n \times n$ с узкой лентой и матрицей-столбцом F. Разобьём матрицы A и F на p участков горизонтальными линиями и распределим i-й участок разбиения на i-й процессор. Рис. 1 показывает распределение матриц A и F для $p = 4$. Выделим на i-м участке разбиения три блока: диагональный квадратный блок $A_i \left(i = 1,\ldots, p\right)$, который имеет размерность n_i; над-диагональный квадратный блок $B_i \left(i = 1,\ldots, p-1\right)$ и под-диагональный квадратный блок $C_i \left(i = 2,\ldots, p\right)$. Поскольку лента матрицы A узкая, то размерность блоков B_i и C_i, равная m, будет много меньше n_i.

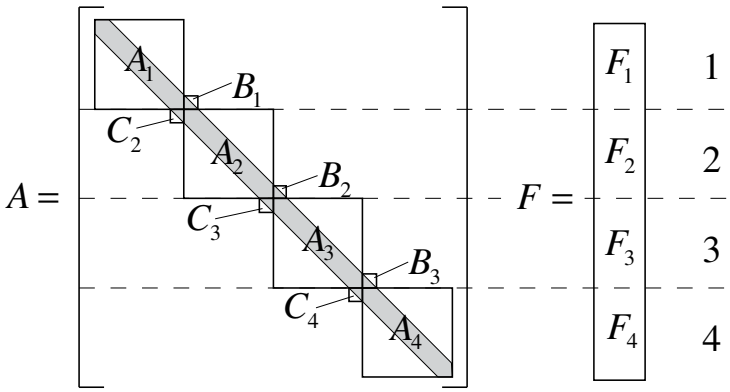

Рис. 1. Распределение матрицы системы A и правой части F на 4 процессора

2.2. Разложение матрицы системы

На данном этапе вычисляется разложение матрицы системы A в виде

$$A = DS\,. \qquad (2)$$

Здесь D – матрица, составленная только из диагональных блоков A_i,

$$D = diag\left(A_1,...,A_p\right),$$

а матрица S, показанная на рис. 2, вычисляется путём умножения i-го участка разбиения матрицы A на матрицу A_i^{-1} слева. Матрица S имеет на диагонали единичные матрицы I_{n_i} размерности n_i и матрицы $V_i \left(i = 1,\ldots, p-1\right)$ и $W_i \left(i = 2,\ldots, p\right)$ на месте внедиагональных блоков. Матрицы V_i и W_i называют шипами (spikes) из-за того, что они имеют размерность $n_i \times m$, то есть являются узкими и высокими.

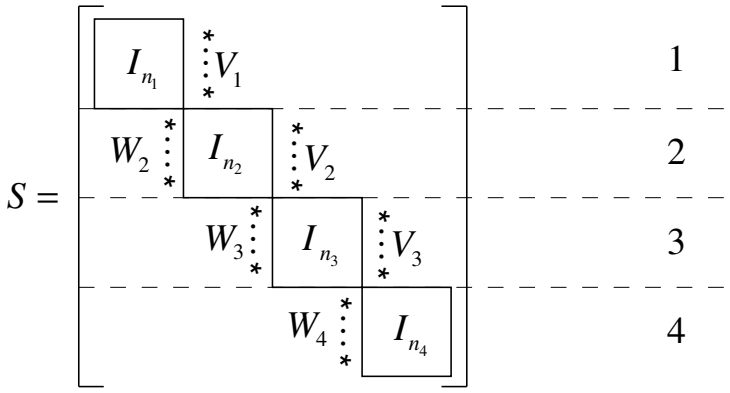

Рис. 2. Матрица S, распределённая на 4 процессора

Шипы V_i и W_i находятся из решения матричного уравнения

$$A_i \begin{bmatrix} V_i & W_i \end{bmatrix} = \begin{bmatrix} 0 & C_i \\ \vdots & 0 \\ 0 & \vdots \\ B_i & 0 \end{bmatrix}. \qquad (3)$$

2.3. Решение системы с разложенной матрицей

После разложения матрицы A в виде (2) решение системы (1) сводится к последовательному решению двух систем:

$$DG = F \qquad (4)$$

$$SX = G. \qquad (5)$$

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

Представим шипы V_i и W_i следующим образом:

$$V_i = \begin{bmatrix} V_i^t \\ V_i^r \\ V_i^b \end{bmatrix} \text{ и } W_i = \begin{bmatrix} W_i^t \\ W_i^r \\ W_i^b \end{bmatrix}$$

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 как

$$X_i = \begin{bmatrix} X_i^t \\ X_i^r \\ X_i^b \end{bmatrix} \text{ и } G_i = \begin{bmatrix} G_i^t \\ G_i^r \\ G_i^b \end{bmatrix}.$$

Из системы (5) формируется сокращённая система

$$\overline{S}\overline{X} = \overline{G}, \qquad (6)$$

состоящая из m рядов матричного уравнения (5), находящихся выше и ниже линий разделения матриц этой системы (это возможно т.к. $m << n$). Она имеет размерность $2m(p-1)$. Матрица сокращённой 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 размерности m.

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

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

Матрица системы (6) является блочно-трёхдиагональной 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 как

$$\begin{bmatrix} W_i^b & 0 \\ 0 & 0 \end{bmatrix} \text{ и } \begin{bmatrix} 0 & 0 \\ 0 & V_{i+1}^t \end{bmatrix}.$$

Блок неизвестных и правая часть, соответствующие i-му диагональному блоку имеют вид

$$\begin{bmatrix} X_i^b \\ X_{i+1}^t \end{bmatrix} \text{ и } \begin{bmatrix} G_i^b \\ G_{i+1}^t \end{bmatrix}.$$

После нахождения решения X системы (6), общее решение системы (1) вычисляется по формулам

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

2.4. Усечённый вариант алгоритма SPIKE

В случае если матрица системы A имеет диагональное доминирование, то есть выполняется условие

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|,$$

то шипы W_i и V_i практически полностью состоят из нулей [5]. Это позволяет при решении системы (6) отбросить её внедиагональные блоки. Полученная система называется усечённой. При этом достаточно вычислять только те части шипов, которые входят в диагональные блоки системы (6).

2.5. Параллельность 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) так же 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. В случае если матрица A имеет диагональное доминирование, то вместо системы (6) решается её усечённый вариант без внедиагональных блоков. При этом в процессе решения требуется передать лишь блок W_{k+1}^t с $k+1$ на k -й процессор и получить обратно блок решений. Это увеличивает степень параллельности данного алгоритма. Изменённый таким образом вариант алгоритма называется усечённым алгоритмом SPIKE.

В нашем случае матрица системы не имеет диагонального доминирования, степень её диагонального доминирования δ, которая вычисляется как

$$\delta = \frac{|a_{ii}|}{\sum_{i \neq j} |a_{ij}|}$$

имеет порядок $0,4$. Поэтому сочли возможным использовать усечённый вариант алгоритма с контролем точности решения. Вычислительные эксперименты показали, что в нашем случае значение $\|F - AX\|$ имело порядок 10^{-12}. Здесь $\|\square\|$ обозначена ∞-норма, то есть максимальное по модулю значение компоненты вектора.

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 приведёт к тому, что компилятор языка C++ будет игнорировать особенности разрешения имён, специфические для языка C++, такие как пространства имён, функции-члены, перегрузку функций и шаблоны.

В заголовочных файлах библиотеки Intel Adaptive Spike Solver данная директива отсутствует, поэтому при компоновке возникают ошибки невозможности разрешения имён. Это можно исправить либо добавлением указанных директив в заголовочные файлы, либо включать заголовочный файл, как показано на рис. 4.

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

Рис. 4. Способ включения заголовочных файлов библиотеки Intel Adaptive SPIKE-Based Solver из языка C++

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, причём используется нумерация элементов с ведущим столбцом. В языках C и C++, в свою очередь, обычно используется нумерация элементов матриц с ведущей строкой. 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 по-

требления памяти, поскольку сформированную матрицу жёсткости 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. При этом из матрицы, имеющей нумерацию с ведущими строками, получим матрицу, имеющую нумерацию с ведущими столбцами.

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 частей M_i размерности $n_i \left(i = 1 \ldots n\right)$, $n_i > 2\beta$. Каждую матрицу M_i можно разбить на три подматрицы: $H_i \left(i = 1 \ldots n\right)$, которые можно транспонировать независимо друг от друга, и матрицы $J_i \left(i = 2 \ldots n\right)$ и $K_i \left(i = 1 \ldots n-1\right)$, данные в которых надо поменять местами для транспонирования матрицы M. Пример разбиения матрицы на три участка показан на рис. 5.

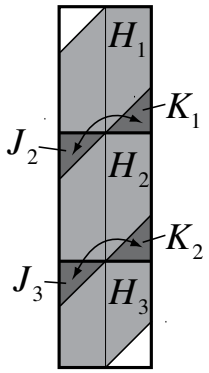

Рис. 5. Разбиение матрицы M для транспонирования при $p = 3$

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 распределённый алгоритм транспонирования ленточной матрицы будет выглядеть следующим образом:

1. Скопировать содержимое матриц J_i и K_i в буферы для передачи данных.
2. Запустить асинхронную передачу матрицы J_i на процессор с номером $i-1$, матрицы K_i - на процессор $i+1$; а также запустить асинхронный приём матриц K_{i-1} и J_{i+1}.
3. Транспонировать матрицу H_i.
4. Завершить асинхронную передачу данных или дождаться её завершения.
5. 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

Решали задачу конечно-элементного моделирования сжатия упруго-пластического цилиндра при использовании регулярной конечно-элементной сетки с одинаковым количеством разбиений d по обеим осям.

Вычислительные эксперименты проводили на кластере um64 Института математики и механики УрО РАН. Кластер состоит из 32 двухпроцессорных двухядерных 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 года, 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 жёсткости.

Таблица. Параметры сеток, использовавшихся в вычислительных экспериментах

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

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 приняли t_2, поскольку сформированная матрица жёсткости не помещалась в память одного процессора используемой кластерной системы.

4.1. Производительность усечённого алгоритма SPIKE при решении упругопластической задачи

На рис. 6, 7, 8 предоставлена зависимость значения ускорений s усечённого алгоритма SPIKE от количества процессоров p и количества разбиений сетки d соответственно для этапа разложения матрицы системы, этапа решения СЛАУ с уже разложенной матрицей и полного решения СЛАУ на шаге нагрузки упругопластической задачи. При полном решении СЛАУ происходит одно разложение и 15 решений с разложенной матрицей системы.

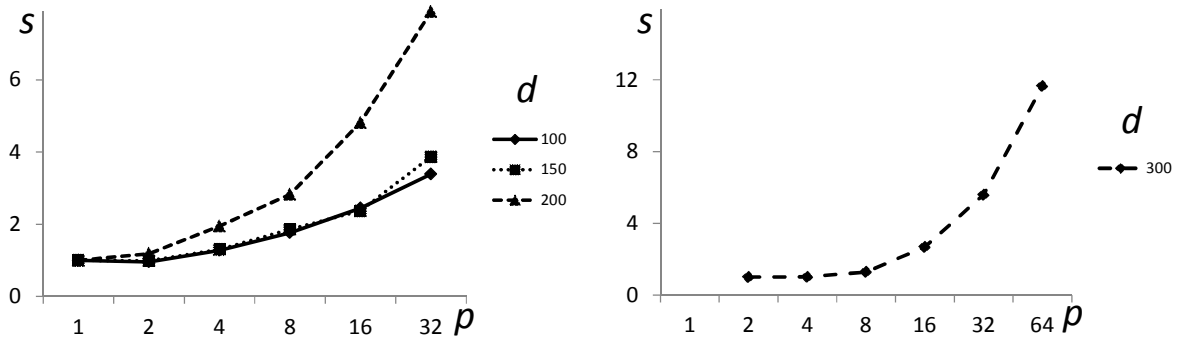

Рис 6. Зависимость ускорений на этапе разложения матрица системы алгоритма SPIKE от количества процессоров p и количества разбиений сетки d

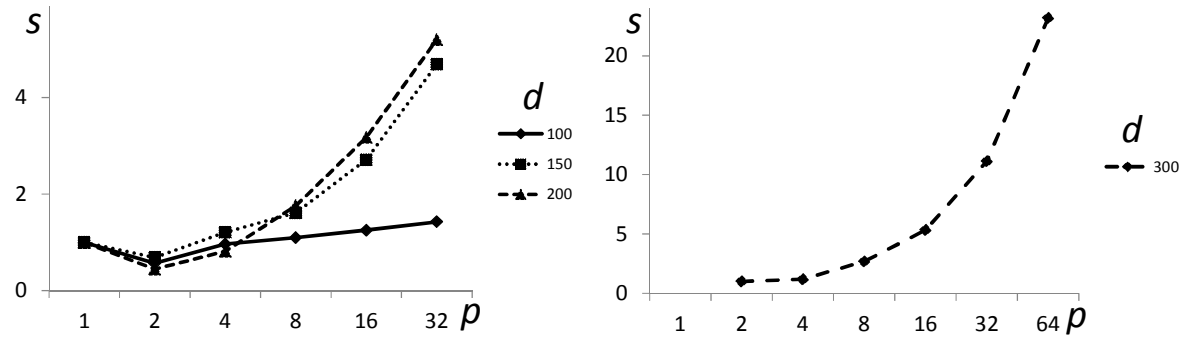

Рис. 7. Зависимость ускорений на этапе решения СЛАУ с разложенной матрицей алгоритма SPIKE от количества используемых процессоров p и количества разбиений d

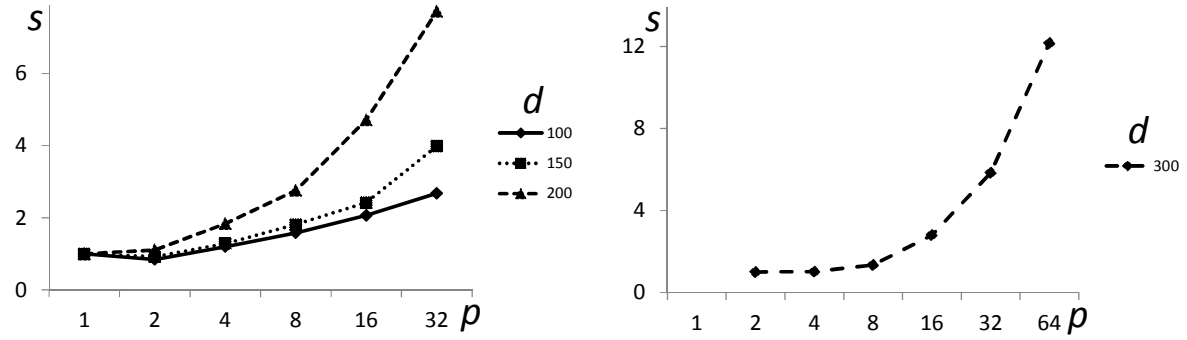

Рис. 8. Зависимость ускорений полного решения СЛАУ внутри шага нагрузки упругопластической задачи для случая одного разложения и 15 решений

Из рис. 6, 7, 8 видно, что рост производительности имеет место как при увеличении количества процессоров, так и при увеличении количества разбиений конечноэлементной сетки. Уменьшение производительности при использовании двух процессоров, наблюдающееся на рис. 7, а так же более пологий наклон графиков на этом рисунке по сравнению с графиками для этапа разложения матрицы для небольших сеток, вызван затратами времени на контроль точности решения.

4.2. Сравнение производительности усечённого алгоритма SPIKE и трёхдиагонального алгоритма LU-разложения из библиотеки ScaLAPACK

На рис. 9 предоставлено сравнение значений ускорений s решения СЛАУ с использованием трёхдиагонального параллельного алгоритма LU-разложения из библиотеки ScaLAPACK и усечённого алгоритма SPIKE, полученные при решении упругопластической задачи сжатия цилиндра с количеством разбиений $d = 200$. Исследование применимости трёхдиагонального алгоритма LU-разложения для решения упругопластической задачи рассмотрено в работе [4].

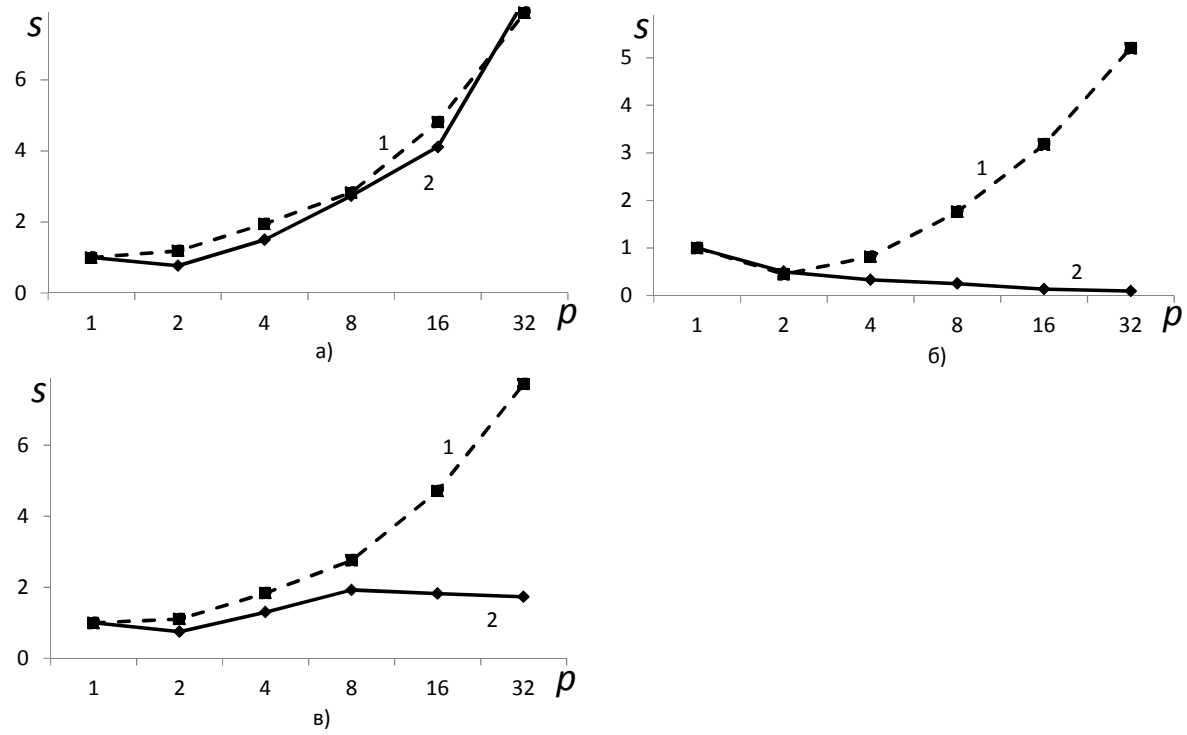

Рис. 9. Ускорение разложения (а); решения СЛАУ (б) и полного решения системы (в) для сетки с количеством разбиений $d = 200$: 1 – SPIKE; 2 – ScaLAPACK

Видно, что усечённый алгоритм SPIKE на сетке с количеством разбиений $d = 200$ имеет приблизительно одинаковую производительность вычисления разложения матрицы системы с трёхдиагональным алгоритмом LU-разложения из библиотеки ScaLAPACK. На этапе решения системы уравнений с использованием разложенной матрицы усечённый алгоритм SPIKE имеет значительно лучшую производительность, чем трёхдиагональный алгоритм. Трёхдиагональный алгоритм LU-разложения имеет коэффициент ускорения $s < 1$ при использовании более одного процессора, однако усечённый алгоритм SPIKE имеет коэффициент ускорения $s > 1$ при использовании более 4 процессоров. При дальнейшем увеличении количества процессоров коэффициент ускорения продолжает увеличиваться.

Литература

1. Поздеев А.А., Трусов П.В., Няшин Ю.И.. Большие упруго-пластические деформации. М: Наука, 1986 - 232с.
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 // Труды международной конференции "Параллельные Вычислительные технологии (ПаВТ’2009)", Нижний Новгород, 30 марта – 3 апреля 2009. – С. 444 – 449. – Челябинск: Изд. ЮУрГУ, 2009.– 839 с. (электронное издание).
3. Blackford L. S., Choi J., Cleary A., D'Azeuedo E. at all. 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 // Труды международной конференции "Параллельные вычислительные технологии (ПаВТ’2010)", Уфа, 29 марта – 2 апреля 2010. – Челябинск: Изд. ЮУрГУ, 2010. – С. 498–506. (электронное издание).

5. Polizzi E, Sameh A. SPIKE: A parallel environment for solving banded linear systems. Computers & Fluids. 2007 Jan 0;36:113--120.
6. Polizzi E, Sameh AH. 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. – 1296 с.