УДК 004.051

Ахмед-Заки Д.Ж.[1*]., Лебедев Д.В.[1**]., Перепёлкин В.А.[2***]

[1] Казахский национальный университет имени аль-Фараби, Республика Казахстан, г. Алматы
[2] Институт вычислительной математики и математической геофизики СО РАН, Россия, г. Новосибирск
E-mail: *darhan.ahmed-zaki@kaznu.kz, **dan-lebedev@mail.ru, ***perepelkin@ssd.sscc.ru

Сравнение эффективности параллельных реализаций метода прогонки: параллельно-конвейерный метод, параллельная прогонка

Исследуется задача решения серий систем трехдиагональных уравнений методом прогонки для применения в трехмерных задачах, таких как задача теплопроводности, для больших размеров сеток (10003 и более). При росте производительности суперкомпьютеров необходимо использование высокоэффективных параллельных программ. Рассматривается метод прогонки из-за того что он является прямым методом, экономичным и простым в реализации в случае последовательной программы, но сложно распараллеливаемым из-за информационных зависимостей между операциями алгоритма. Данный метод также применяется при решении двумерных и трехмерных задач, что приводит к возникновению серии прогонок. И эффективный алгоритм распараллеливания прогонки позволит решать такие задачи на суперкомпьютерах с хорошей производительностью. В работе кратко представлены два алгоритма распараллеливания метода прогонки, на примере решения одномерного эллиптического уравнения с различными размерами порций с использованием стандарта MPI, а также произведено сравнение их эффективности при использовании серии прогонок. Приводятся результаты численных экспериментов по исследованию оптимального размера порций, делаются вывод о применимости исследуемых алгоритмов для больших трёхмерных задач на суперкомпьютерах, содержащих десятки тысяч вычислительных узлов и более.
Ключевые слова: MPI, параллельная программа, метод Яненко, параллельно-конвейерный метод.

Akhmed-Zaki D.Zh., Lebedev D.V., Perepelkin V.A.

Efficiency comparison of parallel implementations Thomas algorithm: pipelined Thomas algorithm, parallel Thomas algorithm

Study the problem of solutions a series of tridiagonal systems of equations Thomas algorithm for use in three-dimensional problems such as heat conduction problem for large mesh sizes (1000^3 and more). At Supercomputing performance growth requires the use of highly efficient parallel programs. The Thomas algorithm examine of because of that that he is a direct method, economical and easy to implement in the case of a sequential program, but it's hard to parallelize because of information dependencies between the operations of the algorithm. This method is also used for solving the three-dimensional and two-dimensional problems, which gives rise to a series of Thomas algorithm. And an efficient algorithm parallelization Thomas algorithm will allow to solve such problems on supercomputers with a good performance. The paper summarizes the two algorithm parallelization Thomas algorithm, the example of the one-dimensional solutions of an elliptic equation with different sizes of servings, using MPI standard, as well as a comparison of their efficiency by using a series Thomas algorithm. The results of numerical experiments to study the optimum size of servings, to draw conclusions about the applicability of the studied algorithms for large three-dimensional problems on supercomputers containing tens of thousands of compute nodes and more.
Key words: MPI, Parallel program, Janenko method, Pipelined Thomas algorithm.

Ахмед-Заки Д.Ж., Лебедев Д.В., Перепёлкин В.А.

Қуалау әдісінің параллельді жұмысының тиімділгін салыстару: параллельді-конвейерлік әдіс, параллельді қуалау

Үлкен өлшемді торлар үшін (1000^3 және одан да көп) жылу өткізгіштік теңдеуі сияқты есептерге үшдиагоналді теңдеулер жүйесін шешетін қуалау әдісін пайдаланып үш өлшемді есептерді шешу қарастырылады. Суперкомпьютерлердің өнімділігі арттқан сайын, жоғарғы өнімді параллельді бағдарламалар пайдаланылуы тиіс. Қуалау әдісін тізбектей жүзеге асыру тікелей, тиімді және қарапайым әдіс болғандықтан пайдаланылады, бірақ, алгоритмнің операциялары арасында мәліметтердің тәуелділігінен параллельдеу қиын. Ұсынылып отырған әдіс екі өлшемді және үш өлшемді есептерге де пайдаланылады және мұндай жағдайда бірнеше қуалау сериясы пайда болады. Қуалау әдісін тиімді параллельдеу алгоритмі есептерді суперкомпьютерлерде жақсы өнімділікпен шығаруға мүмкіндік береді. Мақалада бір өлшемді эллиптикалық теңдеу мысалында MPI стандарттарын пайдаланып қуалаудың екі параллельдеу әдісі көрсетілген, сонымен қатар бірнеше қуалау кезіндегі тиімділіктерді салыстыру жүзеге асырылған. Сандық эксперименттердің нәтижелері көрсетілген, зерттеліп отырған алгоритмнің үш өлшемді, оң мыңдағын есептеу торларында үлкен есептерді суперкомпьютерлерде пайдаланылуы жайлы қорытынды жасалған.

Түйін сөздер: MPI, паралелльді бағдарлама, Яненко әдісі, параллельді-конвейерлік әдіс.

Введение

В настоящее время все больше растет производительность суперкомпьютеров и становится актуальным использование высокоэффективных параллельных программ. Для решения разностных эллиптических уравнений, возникающих при аппроксимации эллиптических уравнений второго порядка, часто используют метод прогонки[1], который является прямым методом, экономичным и простым в реализации в случае последовательной программы, но сложно распараллеливается из-за информационных зависимостей между операциями алгоритма. Данный метод также применяется при решении двумерных и трехмерных задач, что приводит к возникновению серии прогонок. И эффективный алгоритм распараллеливания прогонки позволит решать такие задачи на суперкомпьютерах с хорошей производительностью. Существует множество статей, описывающих варианты распараллеливания прогонки. Среди них можно выделить следующие: в [2] и [3] предложены решатели, использующие сочетание параллельной циклической редукции (PCR) [4] и правой прогонки [4] с механизмами принятия решений для переключения с PCR на метод правой прогонки, чтобы сбалансировать параллелизм и вычислительную сложность. NVIDIA выпустила трехдиагональный решатель, используя сочетание циклической редукции (CR) [2] и алгоритма PCR в библиотеке CUSPARSE [5]. В [6–8] предложены трехдиагональные решатели для графических ускорителей, использующих CR, а в [9] – решатель на основе PCR. В [10] введено сочетание алгоритмов PCR и правой прогонки. В [11] предложен гибридный метод, включающий в себя метод правой прогонки, CR, PCR и метод рекурсивного удвоения (RD) [12]. В [13] предложен вычислительно устойчивый масштабируемый решатель на основе алгоритма SPIKE [14, 15]. Также известен метод, предложенный Н.Н.Яненко [16]. Этот метод позволяет редуцировать исходную систему с большим числом неизвестных к системе с числом неизвестных, равным числу процессоров. Редуцированная система, состоящая из гранично процессорных точек, решается методом прогонки. Чтобы распараллелить решения этой системы, можно применить такие методы, как метод встречной прогонки и метод параллельно циклической редукции. Также известен параллельно-конвейерный метод

[17, 18], предназначенный для решения множества трехдиагональных СЛАУ, возникающих при решении двумерных и трехмерных задач. Были реализованы метод Яненко и параллельно-конвейерный метод с точки зрения их применимости для решения трехмерного уравнения теплопроводности. Дано описание этих методов, а также произведено сравнение их эффективности при использовании серии прогонок

1 Постановка задачи

Для сравнения эффективности рассматриваемых методов будем решать одномерную модельную задачу. В качестве такой задачи возьмем уравнение Пуассона в единичном отрезке с начальными условиями типа Дирихле.

$$\frac{\partial^2 u}{\partial x^2} = -f(x,y), 0 < x < 1 \tag{1}$$

$$u(0) = 0, u(1) = 1 \tag{2}$$

Правая часть f выбиралась из условия, что бы точным решением задачи (1)-(2) была функция $u = x^3$. Будем решать описанное уравнение методом прогонки [1].

2 Алгоритм решения

2.1 Параллельно-конвейерный метод

Основная идея параллельно-конвейерного метода [18] в том, что процессор, выполнив прямой или обратный ход прогонки по части точек, передает вычисленные значения на границе отрезка (при прямом ходе - правую границу при обратном - левую) соседнему процессору, чтобы тот мог продолжить прогонку, а сам в это время начинает обрабатывать следующую порцию точек. Если исключить время разгона и торможения конвейера, то все остальное время в счете задействованы все процессоры. Поясним вышесказанное с помощью рисунка 1. На рисунке конвейер состоит из 3 процессоров, строки на рисунке (сверху вниз) соответствуют количеству тактов, стрелки вправо – протягиванию прямого хода прогонки в одном направлении, а влево – в другом. Светло-серым цветом отмечены процессоры, протянувшие прогонку в одном направлении, темным – в обоих.

2.1 Метод Яненко

В отличие от параллельно-конвейерного метода метод Яненко [16] можно применить не только для решения множества систем трехточечных уравнений, но и для решения одной системы. Рассмотрим систему линейных уравнений с трех диагональной матрицей следующего вида:

$$a_i y_{i-1} + b_i y_i + c_i y_{i+1} = d_i, b_i \neq 0, i = 1, 2, \ldots, N-1, b_0 y_0 + c_0 y_1 = d_0, a_N y_{N-1} + b_N y_N = d_N \tag{3}$$

Для простоты пусть на каждом процессоре будет одинаковое количество точек $m = K/M$, где K – число неизвестных (в нашем случае $K = N+1$), M – число процессоров, индексация будет глобальная. Таким образом, на отдельном процессоре с номером j будет находиться лишь часть уравнений системы (3) с номерами от $(j-1) \cdot m + 1$ до

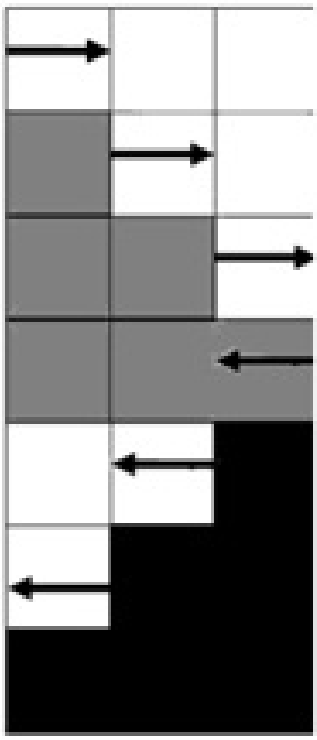

Рисунок 1 –Схема параллельного конвейерного алгоритма

$j \cdot m$, где j – номер процессора. Обозначим $y_{j \cdot m}$ через z_j, $j = 0, .., M$ и будем искать решения системы (3) в виде.

$$y_{(j-1)\cdot m+i} = z_{j-1}u_i + z_j v_i + w_i, i = 1, \ldots, m-1, j = 1, \ldots, M \tag{4}$$

где u, v, w - решения следующих систем уравнений:

$$\begin{cases} a_i u_{i-1} + b_i u_i + c_i u_{i+1} = 0 \\ u_{(j-1)\cdot m} = 1, u_{j\cdot m} = 0 \end{cases} \\ \begin{cases} a_i v_{i-1} + b_i v_i + c_i v_{i+1} = 0 \\ u_{(j-1)\cdot m} = 0, u_{j\cdot m} = 1 \end{cases} \\ \begin{cases} a_i w_{i-} + b_i w_i + c_i w_{i+1} = d_i \\ w_{(j-1)\cdot m} = 0, w_{j\cdot m} = 0 \end{cases} \tag{5}$$

Решения этих трех систем можно найти методом прогонки, причем независимо на каждом процессоре. Будем называть их предрешениями, а этот этап решения задачи – этапом нахождения предрешений. В уравнения с номерами $j*m$ из системы (3) : $a_{j\cdot m}y_{j\cdot m-1}+ b_{j\cdot m}y_{j\cdot m} + c_{j\cdot m}y_{j\cdot m+1} = d_{j\cdot m}, j = 0, \ldots, M$, используя формулу (4) получаем систему трехточечных уравнений относительно z_j, имеющую следующий вид:

$$A_j z_{j-1} + B_j z_j + C_j z_{j+1} = D_j j = 1, \ldots, M-1 \\ B_0 z_0 + C_0 z_1 = D_0, A_M z_{M-1} + B_M z_M = D_M, B_0 = b_0 + c_0 u_1, C_0 = c_0 v_1, D_0 = d_0 + c_0 w_1 \\ A_j = a_{j\cdot m}u_{j\cdot m-1}, B_j = b_{j\cdot m} + a_{j\cdot m}v_{j\cdot m-1} + c_{j\cdot m}u_{j\cdot m+1}, C_j = c_{j\cdot m}v_{j\cdot m+1} \\ D_j = d_{j\cdot m} - a_{j\cdot m}w_{j\cdot m-1} - c_{j\cdot m}w_{j\cdot m+1}, j = 1, \ldots, N-1 \\ A_M = a_{M\cdot m}u_{M\cdot m-1}, B_M = b_{M\cdot m} + a_{M\cdot m}v_{M\cdot m-1}, D_M = d_{M\cdot m} - a_{M\cdot m}w_{M\cdot m-1} \tag{6}$$

Назовем этот этап – этапом нахождения гранично-процессорных решений. Размерность этой системы уравнений равна количеству процессоров, что существенно меньше количества точек задачи. На последнем этапе восстанавливаем окончательное решение по формуле (4). Итак, метод Яненко содержит три этапа:

1. нахождение предрешений,
2. нахождения гранично-процессорных решений,
3. восстановление решения.

Первый и третий этапы выполняются независимо на каждом устройстве (универсальном процессоре, ускорителе или сопроцессоре). А второй этап требует коммуникаций между MPI процессами.

3 Применимость алгоритмов

Рассмотрим применимость рассматриваемых параллельных алгоритмов для решения трехмерных задач, где можно использовать метод стабилизирующей поправки[19], что приводит к возникновению серии прогонок, где каждая из прогонок решает задачу эквивалентную задаче (1) с точки зрения параллельной реализации. Серия прогонок получается при декомпозиции исходной трехмерной области по каждому измерению, как показано на рисунке 2. Для оценки эффективности рассматриваемых параллельных алгоритмов, достаточно рассматривать только один шаг метода стабилизирующей поправки и часть трехмерной области, состоящей из всех точек по оси x и части точек по остальным осям, на рисунке 2 такая область выделена темным цветом. Такой выделенный фрагмент может считаться независимо и коммуникации будут только внутри этой группы процессоров. При выборе этих условий и возникает серия прогонок, в которой количество прогонок определяется как произведение количества точек по измерению y на количество точек по измерению z. Для конвейерно-параллельного алгоритма количество тактов и количество порций определим как произведение, равное произведению количества точек по оси y и количества точек по оси z. Соответственно, мы должны будем так подбирать соотношение количества порций и количества тактов, что бы их произведение всегда было равно размеру выделенной зеленным трехмерной области. Для метода Яненко под количеством порций будем понимать сколько раз, решаются уравнения (4)-(6), а под количеством тактов – сколько раз будет выполняться этот процесс.

4 Тестирование

Были реализованы три следующие программы: одна последовательная и две параллельные, реализующие конвейерный метод и метод Яненко. Параллельные программы реализованы с использованием стандарта MPI. Тестирование проводилось для различного количества точек по пространственной координате и различного количества серий. Все программы запускались на кластере ССКЦ СО РАН. Для чистоты тестирования параллельные программы запускались по одному процессу на узел. Целью тестирования было сравнение эффективности параллельных алгоритмов и определения оптимального соотношения количества порций и количества тактов из соображений дальнейшего использования одного из параллельных алгоритмов для решения трехмерных задач фильтрации [20]. Из этих же соображений были выбраны следующие параметры. Количество

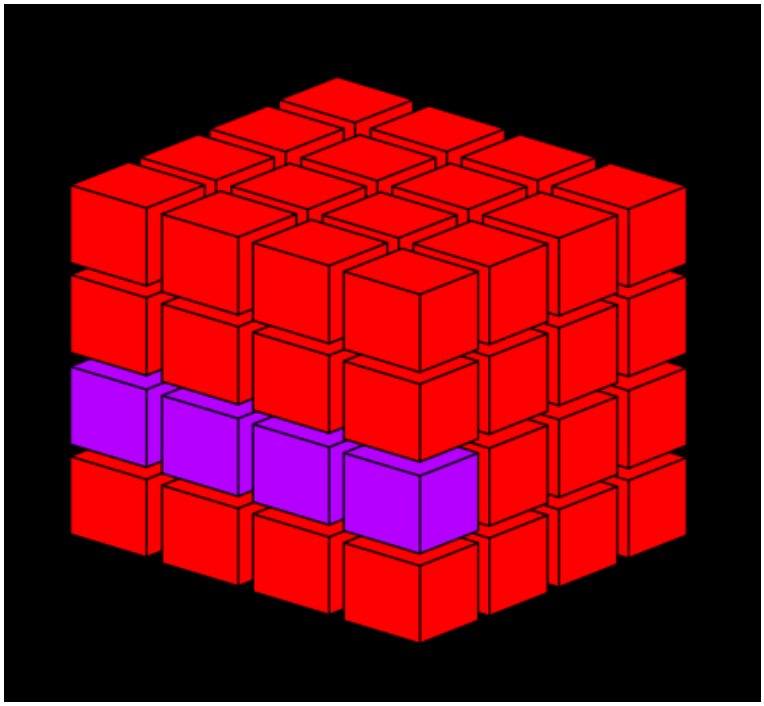

Рисунок 2 –Декомпозиция трехмерной области.

точек по пространственной координате бралось равным 1000, 2000 или 3000. Количество порций и количество тактов конвейера изменялось так, чтобы их произведение было равным 10000. Количество процессов бралось равным 10, 20 или 30. Эти параметры соответствуют трехмерной области размеров $1000 \times 1000 \times 1000$, $2000 \times 2000 \times 2000$ или $3000 \times 3000 \times 3000$ соответственно, и количеству процессов 1000, 8000 или 27000 (10^3, 20^3 и 30^3 соответственно). Результаты тестирования для количества процессов, равного 10 приведены на рисунке 3. Из рисунка 3 видно, что время расчета методом Яненко для различного количества порций всегда меньше чем время выполнения последовательной программы в отличии от конвейерного метода, который выполняется медленнее для размера порций равной всей подобласти. Такое поведение объясняется тем, что при таких параметрах конвейерный метод представляет собой, по сути, последовательную прогонку, но при этом добавляется время на коммуникации между процессами. Но, за исключением последних трех наборов, конвейерный метод всегда выполняется быстрее, чем метод Яненко. А при размере числа порций равным количеству тактов и равному 100 эффективность конвейерного метода равна 0.98, что является очень хорошим результатом, в отличии от метода Яненко, где наибольшая эффективность достигается при тех же параметрах и равна 0.62. На следующем графике (Рисунок 4) показано время расчета для количества процессов, равного 20. Из рисунка 4 видно, что общее поведение сохраняется и при граничном значении, когда количество тактов больше или равно 20 конвейерный метод проигрывает методу Яненко. Для всех остальных случаев конвейерный метод лучше. Но при увеличении количества процессов, что соответствует увеличению длины конвейера, накладные расходы увеличиваются, и максимальная эффективность для конвейерного метода равна 0.71, но при этом и метод Яненко показывает меньшую эффективность (0.42). На рисунке 5 показано время выполнения

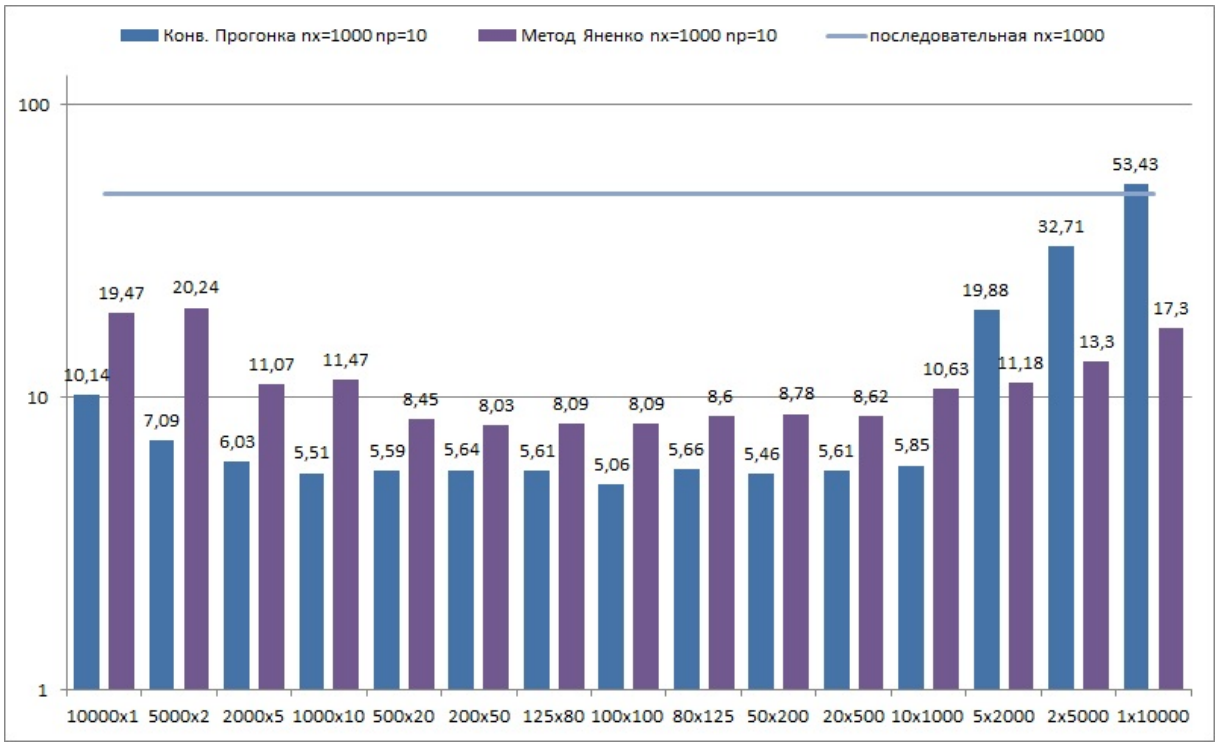

Рисунок 3 –Время расчета (в сек.) для количества процессов равного 10.

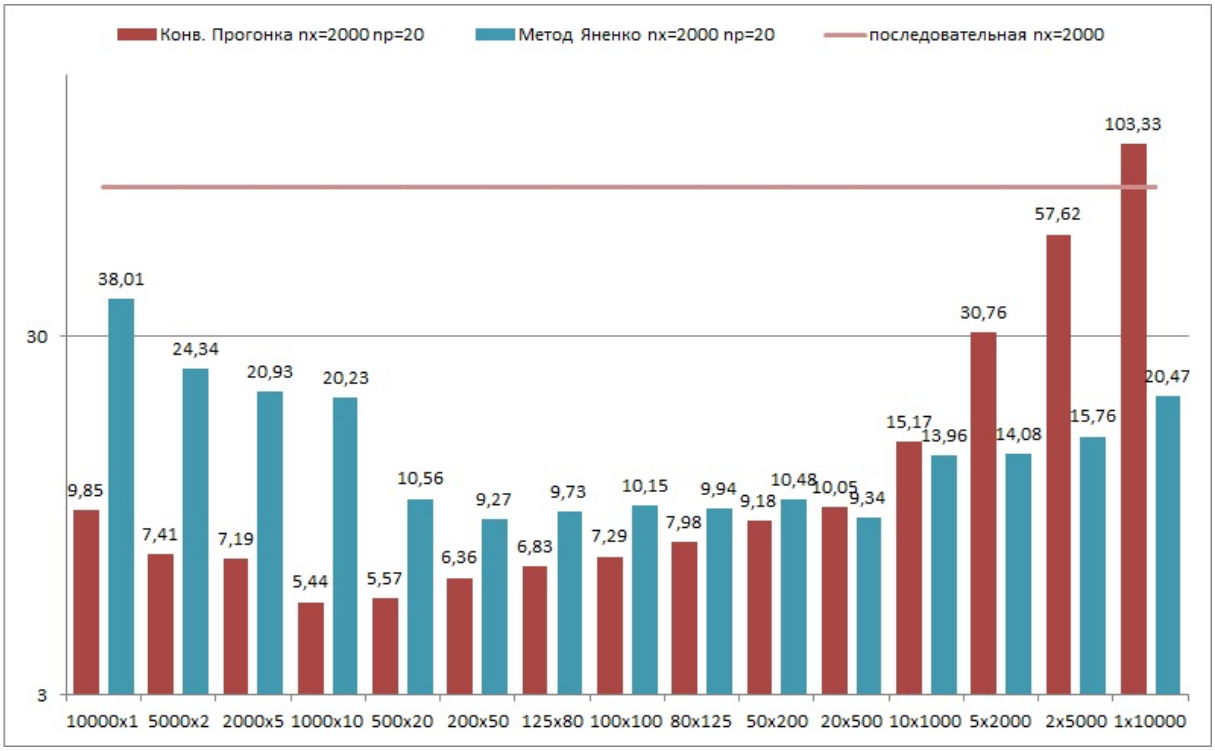

Рисунок 4 –Время расчета (в сек.) для количества процессов равного 20.

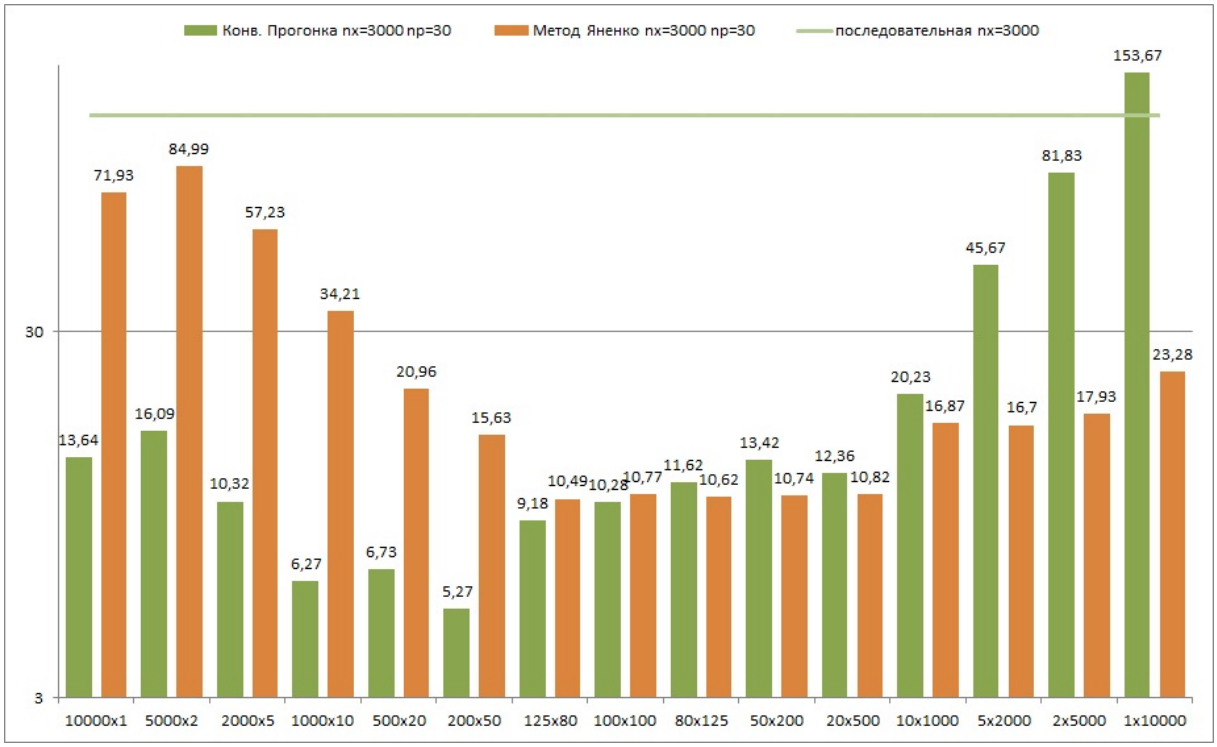

Рисунок 5 – Время расчета (в сек.) для количества процессов равного 20.

программы для количества процессов, равного 30. Из рисунка 5 видно, что общее поведение сохраняется и при граничном значении когда количество тактов больше или равно 80 конвейерный метод проигрывает методу Яненко. Для всех остальных случаев конвейерный метод лучше. Однако при увеличении количества процессов или что то же самое длины конвейера накладные расходы увеличиваются и максимальная эффективность для конвейерного метода равна 0.74, но и метод Яненко показывает еще меньшую эффективность (0.37). Следовательно, для решения трехмерной задачи следует использовать параллельно-конвейерный метод. Также, для более точной оценки эффективности для параллельного конвейерного метода было проведено дополнительное тестирование. Суть его заключается в том, что для количества точек по оси x равного 1000, 2000 и 3000 количество процессов бралось равным 10, 20 и 30 для каждого размера. Результаты тестирования приведены на рисунке 6. На этом рисунке показаны значения эффективности для различных параметров. Видно, что при увеличении размера задачи эффективность снижается, но все равно остается высокой. К тому же, при nx=2000 наибольшая эффективность все равно остается при расчете на 10 процессах. Однако при nx=3000 наилучшая эффективность достигается уже при количестве процессов, равном 20 или 30. Отсюда можно сделать вывод, что при увеличении размера задачи пропорциональное увеличение процессов не дает лучшей эффективности, а оптимальное ускорение может достигаться и на меньшем количестве процессов. Можно сделать вывод, что если не рассматривать граничные значения, параллельный конвейерный метод обладает лучшей эффективностью и для решения трехмерных задач большого размера рекомендуется использовать именно метод конвейерной прогонки. Можно модернизи-

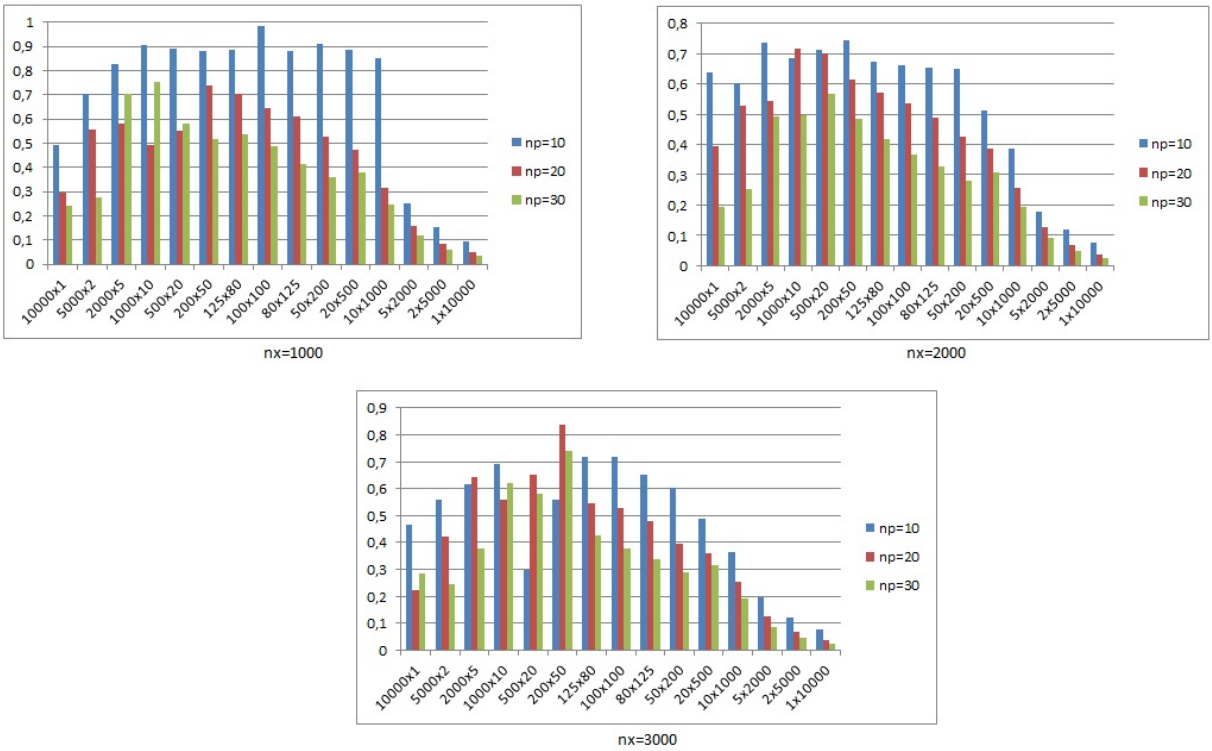

Рисунок 6 – Эффективность параллельно-конвейерного метода.

ровать этот метод, чтобы при расчете он автоматически определял количество порций для наибольшей производительности.

Заключение

Было проведено сравнительное исследование эффективности и масштабируемости двух параллельных алгоритмов прогонки с точки зрения их применения для решения трехмерного уравнения теплопроводности. Была выявлено, что достигается высокая степень эффективности для размеров задач размером до 30003 точек и более, что делает рассмотренные алгоритмы применимыми на суперкомпьютерах, содержащих десятки тысяч вычислительных узлов и более. Работа выполнена при поддержке грантового финансирования научно-технических программ и проектов Комитетом науки МОН РК, грант № 5029/ГФ4

Литература

[1] *Самарский А.А.* Теория разностных систем. - М.: Наука, 1977. - 656с.

[2] *Davidson A., Zhang Y., D. Owens J.* An auto-tuned method for solving large tridiagonal systems on the GPU. URL: http://idav.ucdavis.edu/func/return_pdf?pub_id=1052 (дата обращения: 07.09.2016).

[3] *Kim H.-S., Wu S., Chang L.-W., Hwu W.-M.* A scalable tridiagonal solver for gpus // In Parallel Processing (ICPP), 2011 International Conference on. P. 444 –453.

[4] *Hockney R. W., Jesshope C. R.* Parallel computers: architecture, programming and algorithm. //Hilger. Bristol. 1986. - P. 274-280.

[5] NVIDIA Corporation. CUDA CUSPARSE Library. URL: https://developer.nvidia.com/cusparse (дата обращения: 07.09.2016).

[6] *Sengupta S., Harris M., Zhang Y., Owens J. D.* Scan primitives for GPU computing. URL: http://idav.ucdavis.edu/func/return_pdf?pub_id=915 (дата обращения: 07.09.2016).

[7] *Goddeke D., Strzodka R.* Cyclic reduction tridiagonal solvers on GPUs applied to mixed-precision multigrid // IEEE Transactions on Parallel and Distributed Systems. - 2011. - Vol. 22. - P.22–32.

[8] *Davidson A., Owens J.D.* Register packing for cyclic reduction: A case study. URL: http://idav.ucdavis.edu/func/return_pdf?pub_id=1052 (дата обращения: 07.09.2016).

[9] *Egloff D.* High performance finite difference PDE solvers on GPUs. URL: http://www.researchgate.net/publication/228942149_High_performance_finite_difference_PDE_solvers_on_GPUs (дата обращения: 07.09.2016).

[10] *Sakharnykh N.* Efficient tridiagonal solvers for ADI methods and fluid simulation. // NVIDIA GPU Technology Conference. September 2010.

[11] *Zhang Y., Cohen J., et al.* Fast tridiagonal solvers on the GPU. URL: http://idav.ucdavis.edu/func/return_pdf?pub_id=978 (дата обращения: 07.09.2016).

[12] *Stone H. S.* An efficient parallel algorithm for the solution of a tridiagonal linear system of equations // Journal of the ACM. - 1973. - Vol. 20. - P. 27–38.

[13] *Chang L.-W., Stratton J. A., Kim H.-S., Hwu W.-M.* A scalable, numerically stable, highperformance tridiagonal solver using GPUs. High Performance Computing, Networking, Storage and Analysis (SC) // 2012 International Conference for IEEE Computer Society Press. - 2012. - P. 11.

[14] *Polizzi E., Sameh A.* A parallel hybrid banded system solver: The SPIKE algorithm // Parallel Computing. - 2006. - Vol. 32, No. 2. - P. 177–194.

[15] *15. Polizzi E., Sameh A.* SPIKE: A parallel environment for solving banded linear systems // Computers and Fluids. - 2007. - Vol. 36, No. 1. - P. 113– 120.

[16] *16. Яненко Н.Н., Коновалов А.Н., Бугров А.Н., Шустов Г.В.* Об организации параллельных вычислений и “распараллеливании” прогонки // Численные методы механики сплошной среды. - 1978. - Т. 9, №7. С. 139-146.

[17] *Povitsky A.* Parallelization of the pipelined Thomas algorithm. ICASE Report No. 98-48. Hampton. NASA Langley Research Center. 1998. 22 p.

[18] *Сапронов И.С., Быков А.Н.* Параллельно-конвейерный алгоритм // Атом. - 2009. - № 44. - С. 24-25.

[19] *19. Яненко Н. Н.* Метод дробных шагов решения многомерных задач математической физики. - Новосибирск: Наука, 1967 - 197 с.

[20] *Akhmed-Zaki D., Lebedev D., Perepelkin V.* Implementation of a Three-Phase Fluid Flow ("Oil-Water-Gas") Numerical Model in the LuNA Fragmented Programming System // Proc. of the 13th Conference on Parallel Computing Technologies, LNCS 9251. - 2015. - P. 489-497.

References

[1] *Samarskiy A.A.* The theory of difference systems. - M.: Nauka, 1977. -656p.

[2] *Davidson A., Zhang Y., D. Owens J.* An auto-tuned method for solving large tridiagonal systems on the GPU. URL: http://idav.ucdavis.edu/func/return_pdf?pub_id=1052 (дата обращения: 07.09.2016).

[3] *3. Kim H.-S., Wu S., Chang L.-W., Hwu W.-M.* A scalable tridiagonal solver for gpus. // In Parallel Processing (ICPP), 2011 International Conference on. P. 444 –453.

[4] *Hockney R. W., Jesshope C. R.* Parallel computers: architecture, programming and algorithm. // Hilger. Bristol. - 1986. - P. 274-280.

[5] NVIDIA Corporation. CUDA CUSPARSE Library. URL: https://developer.nvidia.com/cusparse (дата обращения: 07.09.2016).

[6] *Sengupta S., Harris M., Zhang Y., Owens J. D.* Scan primitives for GPU computing. URL: http://idav.ucdavis.edu/func/return_pdf?pub_id=915 (дата обращения: 07.09.2016).

[7] *Goddeke D., Strzodka R.* Cyclic reduction tridiagonal solvers on GPUs applied to mixed-precision multigrid // IEEE Transactions on Parallel and Distributed Systems. - 2011. - Vol. 22. - P. 22–32.

[8] *Davidson A., Owens J.D.* Register packing for cyclic reduction: A case study. URL: http://idav.ucdavis.edu/func/return_pdf?pub_id=1052 (дата обращения: 07.09.2016).

[9] *Egloff D.* High performance finite difference PDE solvers on GPUs. URL: http://www.researchgate.net/publication/228942149_High_performance_finite_difference_PDE_solvers_on_GPUs (дата обращения: 07.09.2016).

[10] *Sakharnykh N.* Efficient tridiagonal solvers for ADI methods and fluid simulation. // NVIDIA GPU Technology Conference. - 2010. - P.17-21

[11] *Zhang Y., Cohen J., et al.* Fast tridiagonal solvers on the GPU. URL: http://idav.ucdavis.edu/func/return_pdf?pub_id=978 (дата обращения: 07.09.2016).

[12] *Stone H. S.* An efficient parallel algorithm for the solution of a tridiagonal linear system of equations // Journal of the ACM. - 1973. - Vol. 20. - P. 27–38.

[13] *Chang L.-W., Stratton J. A., Kim H.-S., Hwu W.-M.* A scalable, numerically stable, highperformance tridiagonal solver using GPUs. High Performance Computing, Networking, Storage and Analysis (SC). // 2012 International Conference for. IEEE Computer Society Press. 2012. - P. 11.

[14] *Polizzi E., Sameh A.* A parallel hybrid banded system solver: The SPIKE algorithm // Parallel Computing. - 2006. - Vol. 32, No. 2. - P. 177–194.

[15] *15. Polizzi E., Sameh A.* SPIKE: A parallel environment for solving banded linear systems // Computers and Fluids. - 2007. - Vol. 36, No. 1. - P. 113– 120.

[16] *15. Yanenko N.N., Konovalov A.N., Bugrov A.N., Shustov G.V* On organization of parallel computations and parallelization of the tridiagonal matrix algorithm // Numerical Methods of Continuum Mechanics. - 1978. - Vol. 9. №7. - P. 139-146.

[17] *Povitsky A.* Parallelization of the pipelined Thomas algorithm. // ICASE Report No. 98-48. Hampton. NASA Langley Research Center. - 1998. - P. 22.

[18] *Sapronov I.S., Bykov A.N.* Pipelined Thomas algorithm // Atom. - 2009. - № 44. - P. 24-25.

[19] *19. Janenko N. N.* The method of fractional steps for solving multidimensional problems of mathematical physics. - Novosibirsk: Nauka, 1967. - 197p.

[20] *Akhmed-Zaki D., Lebedev D., Perepelkin V.* Implementation of a Three-Phase Fluid Flow ("Oil-Water-Gas") Numerical Model in the LuNA Fragmented Programming System //Proc. of the 13th Conference on Parallel Computing Technologies, LNCS 9251. -2015. - P. 489-497.