УДК 519.688

ВЫЧИСЛИТЕЛЬНАЯ ПРОИЗВОДИТЕЛЬНОСТЬ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ПРОГОНКИ НА КЛАСТЕРНЫХ СУПЕРКОМПЬЮТЕРАХ С РАСПРЕДЕЛЕННОЙ ПАМЯТЬЮ

В. Э. Витковский[1], **М. П. Федорук**[1]

Рассматривается алгоритм параллельной прогонки для моделирования нелинейного уравнения Шредингера с помощью неявной схемы Кранка–Николсон с переменным шагом по пространственной и временно́й переменной для анализа производительности на кластерных суперкомпьютерах с распределенной памятью. В вычислительных экспериментах и на основе теоретической модели (закон Амдала) показано, что исследуемый алгоритм эффективно распараллеливается и достигает максимальной вычислительной эффективности и ускорения с показателями 0.7 и 30 соответственно по сравнению с последовательным алгоритмом. Обсуждаются особенности влияния размера сетки (в диапазоне $10^4 - 10^6$ ячеек) и сетевых задержек межпроцессорных обменов (число используемых процессоров варьировалось в диапазоне 6 – 128) на производительность вычислений.

Ключевые слова: математическое моделирование, параллельные алгоритмы, высокопроизводительные вычисления, уравнение Шредингера.

1. Введение. Численные методы решения дифференциальных уравнений часто приводят к системам линейных алгебраических уравнений с ленточными (в простейшем случае — с трехдиагональными) матрицами. Подобные системы возникают при решении параболических или эллиптических уравнений в одномерном случае либо в многомерном при покоординатном расщеплении решения. Для решения систем с такими матрицами разработан специальный алгоритм, называемый методом прогонки, который является не чем иным, как методом исключения Гаусса в реализации, учитывающей структуру матрицы [1] решаемой линейной системы. Большинство существующих в настоящее время способов распараллеливания алгоритма прогонки (например, recursive doubling, cyclic reduction, partition method и их модификации, рассмотренные в [2 – 8]) эффективны для векторных компьютеров, но не позволяют достигать высокой производительности вычислений для систем с архитектурой DMP (distributed-memory multiprocessors) и SMP (symmetric-memory multiprocessors) вследствие высокой интенсивности межпроцессорных обменов для этих алгоритмов.

В настоящей статье изучается производительность параллельного алгоритма прогонки, предложенного в работе [9], на высокопроизводительных вычислительных системах с архитектурой DMP на примере решения нелинейного уравнения Шредингера с кубической нелинейностью в цилиндрических координатах, моделирующего самофокусировку лазерного импульса в прозрачных диэлектриках с начальным условием типа чирпованного гауссового (G) ($m = 0$) и кольцевого распределения (R1) ($m = 1$):

$$i\,\frac{\partial\Psi}{\partial z} + \Delta_r\Psi + |\Psi|^2\Psi = 0, \quad \Delta_r = \frac{1}{r}\,\frac{\partial}{\partial r}\left(r\,\frac{\partial\Psi}{\partial r}\right), \quad \Psi(0,r) = A_0 r^m \exp\left(-\frac{(1+C^2)r^2}{2a_0^2}\right), \quad m = 0, 1.$$

Здесь $\Psi(z,r)$ — медленная огибающая электрического поля, a_0 — поперечный радиус пучка и C — параметр чирпа.

Численные расчеты производились по разностной схеме Кранка–Николсон при $\sigma = 0.5$, образующей

[1] Институт вычислительных технологий СО РАН, пр. Лаврентьева, 6, 630090, Новосибирск; e-mail: wsiewolod@gmail.com, mife@ict.nsc.ru

трехдиагональную матрицу для сетки с переменным шагом по r и z:

$$i\,\frac{\widehat{\Psi}_n-\Psi_n}{\Delta z}+\sigma L\widehat{\Psi}_n+(1-\sigma)L\Psi_n+|\Psi_n|^2\big(\sigma\widehat{\Psi}_n+(1-\sigma)\Psi_n\big)=0,$$

$$L\Psi_n=\frac{1}{r_n}\left[\frac{\left(r_n+\dfrac{h_{n+1}}{2}\right)\left(\dfrac{\Psi_{n+1}-\Psi_n}{h_{n+1}}\right)-\left(r_n-\dfrac{h_n}{2}\right)\left(\dfrac{\Psi_n-\Psi_{n-1}}{h_n}\right)}{\dfrac{h_n}{2}+\dfrac{h_{n+1}}{2}}\right],$$

$$h_{n+1}=h_n+\frac{h_n}{M},\quad M=25000,\quad h_1=10^{-6},\quad \Delta z=\frac{\Delta z_0\left|\Psi(0,0)\right|^2}{\left|\Psi(z,0)\right|^2}.$$

Здесь $\left|\Psi(z,0)\right|^2$ — интенсивность в центре пучка, Δz_0 — начальный шаг по эволюционной переменной z. Такой способ построения сетки (очевидно не единственный) позволяет учесть для данной задачи возможность образования особенности в центре пучка в фокусе.

2. Алгоритм параллельной прогонки. В работе используется параллельный алгоритм, предложенный в [9] и исследованный в работах [10 – 14]. Алгоритм заключается в следующем: вектор решения X, состоящий из N компонент, разбивается на size групп компонент одинаковой длины M_P; тогда вектор-решение уравнения

$$\begin{aligned}&a_ix_{i-1}-b_ix_i+c_ix_{i+1}=d_i,\quad x_i\in X,\\&x_0=\alpha_0x_1+\beta_0,\quad x_N=\alpha_Nx_{N+1}+\beta_N,\end{aligned}\tag{1}$$

имеет вид $X=\big(x_0^1,\ldots,x_{M_1-1}^1;\ldots;x_0^P,\ldots,x_{M_P-1}^P;\ldots;x_0^{\text{size}},\ldots,x_{M_{\text{size}}-1}^{\text{size}}\big)=\big(X^1,\ldots,X^P,\ldots,X^{\text{size}}\big)$, где size — число процессоров и P — номер процессора.

Пусть $W_P=x_0^{P+1}$ — значение решения на границе группы с номером P. Представим компоненты вектора решения X внутри этой группы в виде

$$X^P=R^P+P^P\,W_{P-1}+Q^P\,W_P.\tag{2}$$

Чтобы получить решение X, следует найти все P^P, Q^P, R^P и W_P, поэтому для каждого P методом прогонки решаются следующие уравнения:

$$\begin{aligned}&a_iP_{j-1}^P-b_iP_j^P+c_iP_{j+1}^P=0,\quad P_0^P=1,\quad P_M^P=0,\\&a_iQ_{j-1}^P-b_iQ_j^P+c_iQ_{j+1}^P=0,\quad Q_0^P=0,\quad Q_M^P=1,\\&a_iR_{j-1}^P-b_iR_j^P+c_iR_{j+1}^P=d_i,\quad R_0^P=0,\quad R_M^P=0.\end{aligned}$$

Каждому P соответствует свой индекс $j\in[0,M_P-1]$, т.е. $i=(P-1)M_P+j$. Затем, подставляя (2) в (1), получаем уравнения на обменные граничные условия:

$$\begin{aligned}&A_kW_{k-1}-B_kW_K+C_kW_{k+1}=D_k,\quad W_0=\xi_0W_1+\eta_0,\quad W_{\text{size}-1}=\xi'_{\text{size}-1}W_{\text{size}}+\eta'_{\text{size}-1},\\&A_k=a_iP_{P-1}^{M-1},\quad B_k=b_i-a_iQ_{P-1}^{M-1}-c_iP_P^1,\quad C_k=c_iQ_P^1,\quad D_k=d_i-a_iR_{P-1}^{M-1}-c_iR_P^1.\end{aligned}$$

Граничные условия слева имеют вид $\xi_0=\dfrac{\alpha_0Q_1^1}{1-\alpha_0P_1^1}$ и $\eta_0=\dfrac{\alpha_0R_1^1+\beta_0}{1-\alpha_0P_1^1}$. В нашем случае граничные условия W_{size} справа полагаются равными Ψ_{LSE}, где Ψ_{LSE} — решение линейного уравнения Шредингера на правой границе. Если граничное условие справа дано в виде соотношения $W_{\text{size}-1}=\xi'_{\text{size}-1}W_{\text{size}}+\eta'_{\text{size}-1}$, то значение W_{size} получаем, решая систему уравнений

$$\begin{aligned}&W_{\text{size}-1}=\xi'_{\text{size}-1}W_{\text{size}}+\eta'_{\text{size}-1},\\&W_{\text{size}-1}=\xi_{\text{size}-1}W_{\text{size}}+\eta_{\text{size}-1},\end{aligned}$$

где $\xi_{\text{size}-1}$ и $\eta_{\text{size}-1}$ получены из прямого хода прогонки для начальных значений ξ_0 и η_0. При расчете трех матриц в прямом и обратном ходе выполняемое число операций в этом алгоритме в два раза меньше трехкратного выполнения расчетов для одной матрицы за счет повторяемости элементов вычислений. Однако число операций для переопределения элементов матрицы может быть значительно больше числа операций для прогонки.

Оценочное выражение для ускорения S параллельного алгоритма (закон Амдала) имеет следующий вид [15, 16]:

$$S = \frac{N_{\mathrm{M}}}{\dfrac{aN_{\mathrm{M}}}{N_{\mathrm{CPU}}} + b(N_{\mathrm{CPU}}) + c}. \tag{3}$$

Здесь a — коэффициент, пропорциональный числу операций на один элемент сетки размером N_{M}, N_{CPU} — число используемых процессоров, $b(N_{\mathrm{CPU}})$ — функция, характеризующая задержки сети и зависящая от числа процессоров, c — коэффициент, учитывающий неускоряемую часть программы.

3. Результаты и обсуждения. Алгоритм параллельной прогонки реализован на языке Fortran 77 с использованием технологии MPI. Анализ производительности выполнялся на кластерных суперкомпьютерах на основе процессоров Intel Itanium 2, 1.6 GHz, cache 3 Mb и Intel Xeon64 DP 3,2 GHz, cache 1 Mb и сетевого коммутатора InfiniBand (10 Гбит/сек, Cluster Interconnect, $t_L = 10\mu s$) с использованием компилятора Intel Fortran 10.1 с оптимизационной опцией fast.

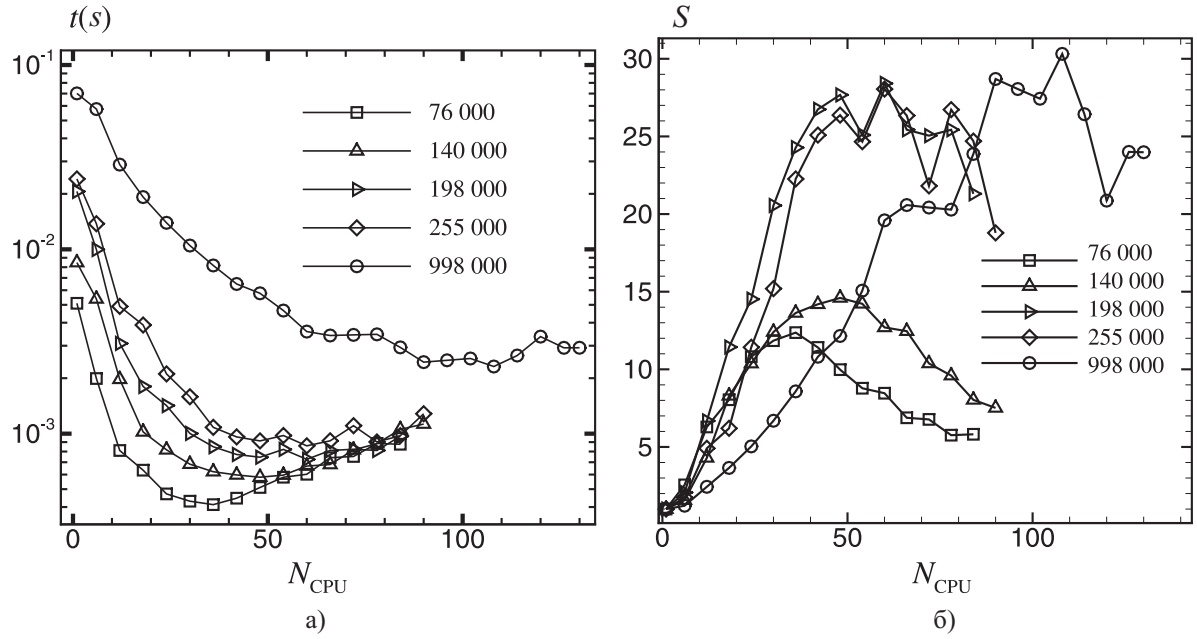

Рис. 1. (а) Время выполнения одного шага t параллельной прогонки на процессорах Itanium для разных сеток и (б) ускорение S параллельной прогонки по сравнению с последовательным алгоритмом для разных сеток на процессорах Itanium

Ускорение параллельного алгоритма вычислялось как отношение среднего времени выполнения одного шага прогонки последовательного алгоритма к среднему времени одного шага параллельной прогонки. Время выполнения одного шага для различного числа используемых процессоров усреднялось за одинаковое время выполнения программы (5 минут) и за одинаковое количество шагов (50 000 шагов). Кривые ускорения для двух типов усреднения отличались на доли процента.

На рис. 1а показано среднее время выполнения одного шага t на процессорах Itanium в зависимости от числа используемых процессоров N_{CPU} для различных по величине сеток N_{M} (76 000, 140 000, 198 000, 255 000 и 998 000 ячеек). Для всех сеток расчеты времени выполнения одного шага выполнены до достижения условий, когда основное расчетное время занимают обменные операции пересылок. Поэтому для достаточно большого N_{CPU} с увеличением N_{CPU} уменьшается время t, а значения t для разных N_{M} совпадают. В этих случаях функция $t(N_{\mathrm{CPU}})$ имеет квадратичную зависимость от N_{CPU}. В обменные операции входят две коллективные операции обмена по 1024 и 256 бит на каждый процессор (обменные граничные условия), не блокирующие отложенные операции обмена по 256 бит на каждый процессор (синхронизация шага по эволюционной переменной z) и не блокирующие операции обмена на трех смежных процессорах по 512 бит на каждый процессор (элементы на краях интервала для схемы Кранка–Николсон).

На рис. 1б представлены кривые ускорения $S(N_{\mathrm{CPU}})$ для различных N_{M} на процессорах Itanium. Здесь обнаруживается соответствие с законом Амдала (3). Максимальное ускорение ограничивается функцией, описывающей задержки сети $b(N_{\mathrm{CPU}})$, т.е. пропускной способностью и латентностью. Проведенные

эксперименты по аппроксимации функции $b(N_{\mathrm{CPU}})$ по данным рис. 1б (кроме сетки с $N_{\mathrm{M}} = 998\,000$) показали наименьший разброс параметров при аппроксимации кривых ускорения для различных сеток по формуле

$$S_{\mathrm{A}} = \frac{N_{\mathrm{M}}}{\dfrac{aN_{\mathrm{M}}}{N_{\mathrm{CPU}}} + bN_{\mathrm{CPU}}^2 + c + dN_{\mathrm{M}}}, \tag{4}$$

где коэффициент d определяет зависимость t от размера сетки N_{M} (в данном случае зависимость линейная). Параметры в (4) принимают следующие значения: $a = 2$, $b = 2$, $c = 1$, $d = -0.02$.

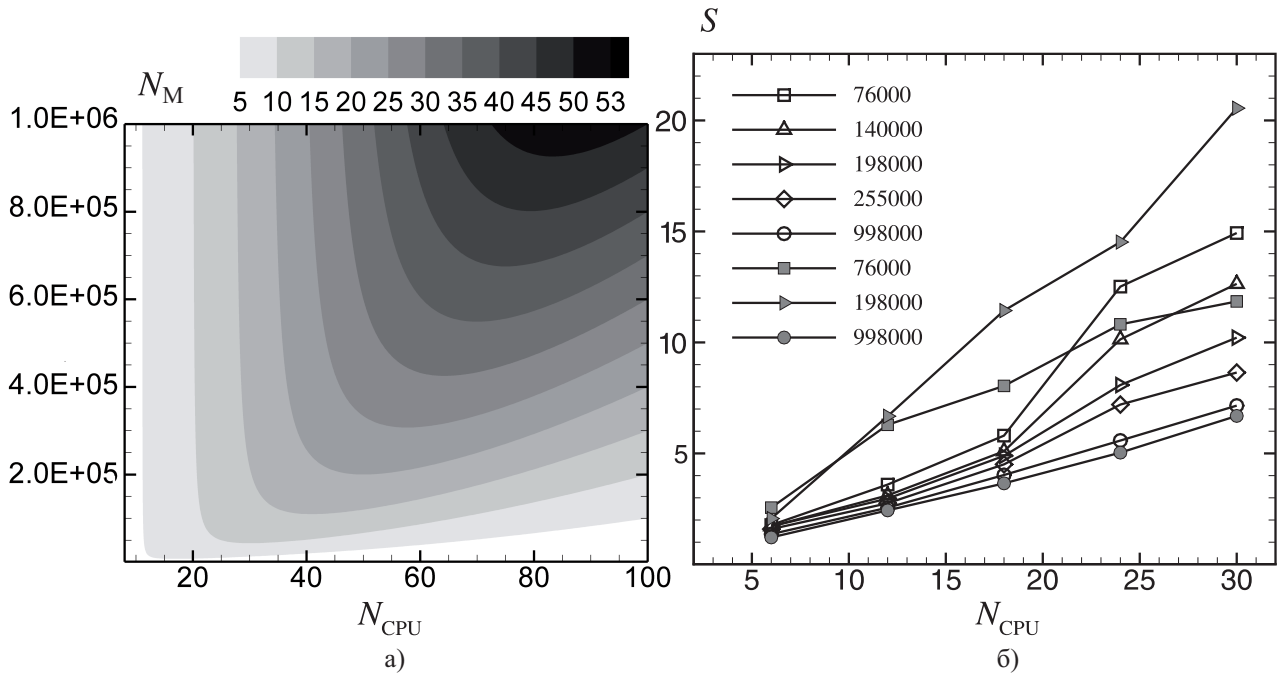

Рис. 2. (а) Изолинии функции ускорения S_{A} в координатах N_{CPU} и N_{M} и (б) кривые ускорения $S(N_{\mathrm{CPU}})$ для процессоров Itanium (заполненные символы) и Xeon (пустые символы) для разных сеток N_{M}

На рис. 2а представлены изолинии функции ускорения S_{A} в координатах N_{CPU} и N_{M}. Здесь можно обнаружить, что максимальное ускорение увеличивается с ростом размера сетки. Если зависимость $b(N_{\mathrm{CPU}})$ пропорциональна $bN_{\mathrm{CPU}}^{\sigma}$, то получаем следующие зависимости для максимального ускорения и точки максимума соответственно:

$$S^{\max} \propto \frac{N_{\mathrm{M}}^{1/(\sigma+1)}}{2a^{\sigma/(\sigma+1)}\, b^{1/(\sigma+1)}}, \quad N_{\mathrm{CPU}}^{\max} \propto \sqrt[\sigma+1]{\frac{aN_{\mathrm{M}}}{b}}. \tag{5}$$

На рис. 2а видно, что скорость роста ускорения S незначительно зависит от N_{M}. Однако на рис. 1б наклон кривой ускорения для сетки 998 000 в 2.5 раза меньше наклона $S(N_{\mathrm{CPU}})$ для остальных сеток. Предполагается, что уменьшение наклона $S(N_{\mathrm{CPU}})$ для сетки с $N_{\mathrm{M}} = 998\,000$ связано с ограниченным размером и многоуровневой структурой кэша [17]. Скачки ускорения могут быть связаны с перепадами в пропускной способности сети вследствие того, что пропускная способность является основным фактором, ограничивающим ускорение при большом числе процессоров. При увеличении числа процессоров N_{CPU} уменьшается M_P и тем бóльшая доля вычислений происходит в кэше первого уровня; вследствие этого постепенно увеличивается наклон S для сетки с $N_{\mathrm{M}} = 998\,000$.

На рис. 2б приведены кривые ускорения $S(N_{\mathrm{CPU}})$ для процессоров Itanium и Xeon для сеток с различными значениями N_{M}. Здесь для графиков ускорения на процессорах Xeon также обнаруживается постепенное уменьшение S с увеличением N_{M}. Для сеток с $N_{\mathrm{M}} = 76\,000$ и $N_{\mathrm{M}} = 140\,000$ на процессорах Itanium достигаются меньшие ускорения, чем на процессорах Xeon. Предполагается, что это связано с меньшим временем выполнения одного шага на процессоре Itanium, вследствие чего увеличивается влияние сетевых задержек.

В табл. 1 представлены данные в секундах о времени выполнения одного шага последовательного алгоритма для разных сеток на процессорах Itanium и Xeon. Максимальная эффективность параллельного алгоритма $E_{\max} = \dfrac{S_{\max}}{N_{\mathrm{CPU}}}$ и максимальное ускорение $S_{\max}$ для разных сеток приведены в табл. 2 (в скобках указано соответствующее число процессоров N_{CPU}). Значения $E_{\max}$ и $S_{\max}$ для процессоров Xeon получены при измерении производительности алгоритма вплоть до 30 процессоров, для процессоров Itanium — до 108 процессоров. При анализе полученных данных наблюдается уменьшение эффективности E и ускорения S параллельного алгоритма по мере роста размера расчетной сетки (за исключением вышеоговоренных случаев при малых временах t). Особенно этот эффект заметен при анализе производительности на процессорах Xeon.

Таблица 1

Среднее время выполнения в секундах одного шага последовательного алгоритма прогонки для разных сеток на процессорах Itanium и Xeon

$t(s)$	76000	140000	198000	255000	998000
Itanium	0.00509	0.00847	0.02062	0.02409	0.07014
Xeon	0.00728	0.01347	0.01905	0.02451	0.09602

Таблица 2

Максимальное ускорение параллельного алгоритма прогонки $S_{\max}$ и максимальная эффективность $E_{\max}$ и соответствующее число процессоров, полученные на разных сетках на процессорах Itanium и Xeon

$E_{\max}(N_{\mathrm{CPU}})$ $S_{\max}(N_{\mathrm{CPU}})$	76000	140000	198000	255000	998000
Itanium	0.52(12) 12.36(36)	0.46(18) 14.6(48)	0.7(30) 28.4(60)	0.62(36) 28(60)	0.33(60) 30.4(108)
Xeon	0.52(24) 15(30)	0.42(24) 12.65(30)	0.34(30) 10.2(30)	0.3(24) 8.6(30)	0.24(30) 7.2(30)

Алгоритм был применен в численных исследованиях свойств локального коллапса решений нелинейного уравнения Шредингера. В процессе расчетов были определены, например, критические мощности локального коллапса для гауссового и кольцевого распределений $P_{\mathrm{Cr,G}} = 1.900$ и $P_{\mathrm{Cr,R1}} = 2.132$ [18]. Из-за возникновения численной особенности при коллапсе для увеличения точности расчетов необходимо значительное измельчение сетки по эволюционной переменной z и поперечному радиусу r. Рабочая сетка по переменной r состояла из 255 000 ячеек. Расчеты выполнялись на процессорах Itanium с использованием 36 процессоров. Ускорение алгоритма при этом составляло $S = 22.3$. Число используемых процессоров выбиралось исходя из условий наибольшего наклона кривой $S(N_{\mathrm{CPU}})$ для данной расчетной сетки. В этих условиях происходит более рациональный расход вычислительных ресурсов, поскольку большее ускорение для данной расчетной сетки, например $S = 26.4$, достигается при $N_{\mathrm{CPU}} = 48$. Таким образом, при ограниченных вычислительных ресурсах оптимально провести большее число расчетов (на 33 % для задачи с $N_{\mathrm{CPU}} = 36$) с меньшим ускорением (на 15 %), чем меньше расчетов с бóльшим ускорением одновременно.

В данной работе расчеты производились одновременно на 108 процессорах, т.е. три задачи по 36 процессоров, поскольку три задачи по 48 процессоров занимали бы 144 процессора, но такое число процессоров в наших вычислениях одновременно не доступно. С помощью этого приема вычисления были проведены в два раза быстрее, что следует из следующих рассуждений. Пусть T — время вычислений одной последовательной задачи, ξ — число расчетов для трех задач по 36 процессоров, η — число расчетов для двух задач по 48 процессоров, τ — время вычислений всех последовательных задач. Тогда получаем следующие равенства: $\dfrac{T}{36}\,3\,\xi = \tau$, $\dfrac{T}{48}\,2\,\eta = \tau$, из которых следует $\eta = 2\,\xi$.

4. Выводы. Анализ производительности параллельного алгоритма прогонки показал, что основным фактором, ограничивающим ускорение, являются не только задержки сети, но и эффекты кэширования

данных в процессоре и сама производительность вычислительного модуля. Задержки сети сказываются при малых N_{M}, т.е. на грубых сетках, и при большом числе процессоров N_{CPU} при больших N_{M}, когда интервал расчетной сетки, обрабатываемый на одном процессоре, достаточно мал и бóльшая часть вычислений происходит в кэше первого уровня, так что время вычислений сравнимо со временем сетевых обменов. При больших N_{M} ожидалось получить лучшие результаты вследствие меньшего влияния сетевых задержек и зависимости $S^{\max}(N_{\mathrm{M}})$ из формулы (5), но из-за влияния размера кэша оказалось, что с ростом размера расчетной сетки уменьшается эффективность и ускорение параллельного алгоритма (различие до в 2.5 раз). Максимальные ускорения, полученные для обсуждаемого алгоритма, следующие: для процессоров Itanium в 30.4 раз при $N_{\mathrm{M}} = 998\,000$ и $N_{\mathrm{CPU}} = 108$, а для процессоров Xeon в 15 раз при $N_{\mathrm{M}} = 76\,000$ и $N_{\mathrm{CPU}} = 30$. Наиболее оптимальные условия для ускорения достигаются при вычислениях на расчетной сетке, состоящей из 198 000 ячеек на 30 процессорах Itanium; при этих условиях достигается ускорение $S = 20.6$ с максимальной эффективностью, равной 0.7.

Авторы признательны В. И. Паасонену и Д. Л. Чубарову за многочисленные полезные обсуждения.

СПИСОК ЛИТЕРАТУРЫ

1. *Thomas L.H.* Elliptic problems in linear difference equations over a network. Technical report. New York: Columbia University Press, 1949.
2. *Hockney R.W., Jesshope C.R.* Parallel computers: architecture, programming and algorithms. Bristol: IOP Publishing Ltd., 1981.
3. *Cox C. L.* Implemention of a divide and conquer cyclic reduction algorithm on the FPS T-20 hypercube // Proc. of the Third Conference on Hypercube Concurrent Computers and Applications. January 1989. Pasadena, California, United States. 1532–1538.
4. *Ortega J.M., Voigt R.G.* Solution of partial differential equations on vector and parallel computers // SIAM Rev., June 1985. 149–240.
5. *Sun X.-H., Zhang H., Ni L.M.* Efficient tridiagonal solvers on multicomputers // IEEE Transactions on Computers. 1992. **41**, N 3. 286–296.
6. *Sun X.-H., Moitra S.* A fast parallel tridiagonal algorithm for a class of CFD applications. NASA Technical Paper N 3585. 1996.
7. *Chawla M.M., Passi K., Zalik R.A.* A recursive partitioning algorithm for inverting tridiagonal matrices // Int. J. Computer Math. 1990. **35**. 153–158.
8. *Povitsky A.* Parallel directionally split solver based on reformulation of pipelined Thomas algorithm. ICASE Technical Report N 45. 1998.
9. *Яненко Н.Н., Коновалов А.Н., Бугров А.Н., Шустов Г.В.* Об организации параллельных вычислений и "распараллеливание" прогонки // Числ. методы механики сплошн. среды. 1978. **9**, № 7. 139–146.
10. *Paasonen V.I.* Boundary conditions of high-order accuracy at the poles of curvilinear coordinate systems // Russian Journal of Numerical Analysis and Mathematical Modelling. 1999. **14**, N 4. 369–382.
11. *Паасонен В.И.* Параллельный алгоритм для компактных схем в неоднородных областях // Вычислительные технологии. 2003. **8**, № 3. 98–106.
12. *Паасонен В.И.* Сходимость параллельного алгоритма для компактных схем в неоднородных областях // Вычислительные технологии. 2005. **10**, № 5. 81–89.
13. *Воеводин А.Ф., Шугрин С.М.* Методы решения одномерных эволюционных систем. Новосибирск: Наука, 1993.
14. *Кудряшова Т.А., Поляков С.В.* О некоторых методах решения краевых задач на многопроцессорных вычислительных системах // Труды IV международной конференции по математическому моделированию. 27 июня – 1 июля 2000 г., Москва (под ред. Л. А. Уваровой). **2**. 134–145. М.: СТАНКИН, 2001.
15. *Amdahl G.* Validity of the single processor. Approach to achieving large-scale computing capabilities // Proceedings of the AFIPS Conference. 1967. 483–485.
16. *Рычков А.Д.* Численные методы и параллельные вычисления // Учебное пособие. Новосибирск: СибГУТИ, 2007.
17. *Прокопьева Л.Ю., Чубаров Д.Л.* Производительность параллельной прогонки // Труды IV Российско-германской школы по параллельным вычислениям и высокопроизводительным вычислительным системам. Новосибирск: ИВТ СО РАН, 2007.
18. *Витковский В.Э., Федорук М.П.* Численное исследование свойств решений нелинейного уравнения Шредингера при распространении лазерных импульсов в световодах // Вычислительные технологии. 2008. **13**, № 6.

Поступила в редакцию
07.08.2008