УДК 519.6

СРАВНЕНИЕ ДВУХ МЕТОДОВ РАСПАРАЛЛЕЛИВАНИЯ ПРОГОНКИ НА ГИБРИДНЫХ ЭВМ С ГРАФИЧЕСКИМИ УСКОРИТЕЛЯМИ

А. А. Федоров, А. Н. Быков
(ФГУП "РФЯЦ-ВНИИЭФ", г. Саров Нижегородской области)

Дается описание двух методов (параллельно-конвейерного и метода Яненко) решения системы линейных алгебраических уравнений с трехдиагональной матрицей на параллельных ЭВМ с графическими ускорителями. Описываются особенности их реализации как на параллельных ЭВМ без ускорителей, так и на гибридных ЭВМ. Оценивается арифметическая сложность метода Яненко и приводятся результаты численных экспериментов по исследованию масштабируемости методов.

Ключевые слова: система линейных алгебраических уравнений с трехдиагональной матрицей, метод прогонки, параллельно-конвейерный метод, метод Яненко, параллельные ЭВМ с графическими ускорителями, методика РАМЗЕС-КП.

Введение

Для численного решения трехмерных нестационарных задач газовой динамики с учетом теплопроводности применяется эйлерово-лагранжева методика РАМЗЕС-КП [1]. Конечно-разностная аппроксимация дифференциальных уравнений, используемая в данной методике, сводит задачу нахождения численного решения к необходимости решения систем линейных алгебраических уравнений (СЛАУ) специального (трехдиагонального) вида. Для нахождения решения таких СЛАУ используется метод прогонки.

Метод прогонки является прямым методом и всегда привлекал своей экономичностью и простотой реализации в последовательном режиме. Появление параллельных ЭВМ с графическими ускорителями (GPU) сделало актуальной задачу реализации на них методов решения СЛАУ с трехдиагональной матрицей с помощью прогонки. Особенности архитектуры таких ЭВМ заставили пересмотреть подходы к распараллеливанию прогонки.

Ранее для многопроцессорных ЭВМ с распределенной памятью была разработана версия параллельно-конвейерного метода (ПКМ) [2], одной из особенностей которой является автоматический выбор числа порций *линий точек*[1] для такта конвейера.

В ПКМ предполагается распараллеливание не одной, а множества прогонок: на процессоре выполняется работа для части (порции) линий точек, после чего эта часть передается для дальнейшей обработки на другой процессор и начинается работа со следующей порцией. В GPU число параллельно исполняющихся нитей, каждая из которых может обрабатывать одну линию точек, очень велико и для многих задач превышает число линий точек. Поэтому авторы обратили внимание на другой подход к распараллеливанию прогонки, основанный на идее редукции.

В англоязычной литературе известно множество статей на данную тему. Среди них можно выделить следующие. В [3, 4] предложены решатели, использующие сочетание параллельной циклической редукции (ПЦР) [5] и алгоритма Томаса [6] с механизмами принятия решений для переключения с первого алгоритма на второй, чтобы сбалансировать параллелизм и вычислительную

[1]Под линией точек будем понимать строки трехдиагональных матриц, обрабатываемые на текущем процессоре.

сложность. Фирма NVIDIA выпустила трехдиагональный решатель, используя сочетание алгоритмов циклической редукции [5] и ПЦР в библиотеке CUSPARSE [7]. В [8—10] предложены трехдиагональные решатели для GPU с использованием циклической редукции, а в [11] — решатель на основе ПЦР. В [12] описана реализация алгоритма Томаса, а в [13] введено сочетание алгоритмов ПЦР и Томаса. В [14, 15] предложен гибридный метод, включающий в себя алгоритм Томаса, циклическую редукцию, ПЦР и метод рекурсивного удвоения [16]. Вычислительно устойчивый масштабируемый решатель на основе алгоритма SPIKE [17, 18] предложен в [19].

В русскоязычной литературе известен метод, предложенный Н. Н. Яненко [20]. Метод Яненко позволяет редуцировать исходную систему с большим числом неизвестных к системе с числом неизвестных, равным числу процессоров. Редуцированная система для точек на границе областей, рассчитываемых процессорами (гранично-процессорных точек), решается методом прогонки. Для того чтобы распараллелить нахождение решений этой системы, исследуются метод встречной прогонки (ВП) и метод ПЦР.

Параллельно-конвейерный метод

Как было сказано, в методике РАМЗЕС-КП первоначально для решения СЛАУ с трехдиагональной матрицей на параллельных ЭВМ с распределенной памятью был реализован ПКМ. Основная идея метода состоит в следующем. После обработки находящихся на процессоре точек линии данные передаются на следующий (на прямом ходе прогонки) или предыдущий (на обратном ходе прогонки) процессор, чтобы работа могла продолжиться уже на нем, в то время как на текущем процессоре начинается обработка следующей линии точек. Таким образом, имеет место конвейерная обработка информации. Если исключить время разгона и торможения конвейера, то все остальное время в счете задействованы все процессоры.

Поясним вышесказанное с помощью рис. 1. На рисунке конвейер состоит из шести процессоров, строки на рисунке (сверху вниз) соответствуют последовательным моментам времени, стрелки вправо — "протягиванию" вычислений прямой прогонки в одном направлении, а стрелки влево — в другом. Светло-серым цветом отмечены процессоры, "протянувшие" прогонку в одном направлении, темно-серым — в обоих. Граничные условия заданы на правой и левой границах и в центре.

Следует отметить, что число порций, определяющее число тактов конвейера, подбирается автоматически во время решения задачи на ЭВМ исходя из времени счета на предыдущем временном шаге задачи. В случае, когда число порций становится равно единице, ПКМ вырождается в классический метод прогонки, когда в каждый момент времени работа выполняется только на одном процессоре. Подробно этот метод описан в [2].

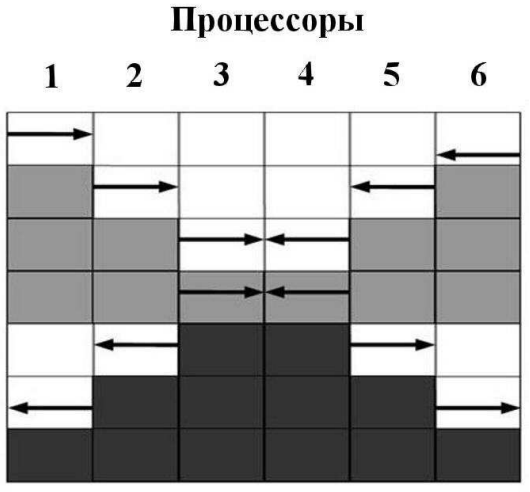

Рис. 1. Схема параллельно-конвейерного алгоритма

Особенности реализации ПКМ на гибридных ЭВМ

При реализации данного метода на GPU выяснилось, что число порций линий точек, необходимых для достижения наименьшего времени счета, равно единице. Это связано с тем, что архитектура GPU подразумевает наличие большого числа одновременно выполняющихся потоков.

Например, пусть имеется двумерная задача, содержащая $1\,000 \times 1\,000$ точек. Тогда при выполнении прогонки вдоль одного направления необходимо рассчитать 1 000 прогонок. Многие GPU поддерживают до 2 048 активных нитей на мультипроцессоре и число мультипроцессоров около 15, благодаря чему можно одновременно запустить на выполнение более 30 000 нитей. Этого количества

достаточно, чтобы каждая прогонка рассчитывалась в отдельном потоке. Следовательно, разбиение на порции не будет иметь смысла. В случае же трехмерной задачи на этапе прогонки с числом точек $100 \times 100 \times 100$ необходимо одновременно рассчитывать $100 \times 100 = 10\,000$ прогонок. Этого количества также недостаточно, чтобы полностью загрузить GPU.

Таким образом, на параллельных ЭВМ с графическими ускорителями ПКМ вырождается в классический метод прогонки. Осознание данного факта заставило обратить внимание на другие методы решения СЛАУ с трехдиагональными матрицами в параллельном режиме, в частности метод, предложенный Н. Н. Яненко [20].

Метод Яненко

Будем решать систему линейных уравнений с трехдиагональной матрицей

$$\begin{aligned} &a_i y_{i-1} + b_i y_i + c_i y_{i+1} = d_i, \quad b_i \neq 0, \quad i = 1, 2, \ldots, N-1; \\ &b_0 y_0 + c_0 y_1 = d_0; \qquad a_N y_{N-1} + b_N y_N = d_N. \end{aligned} \tag{1}$$

Для простоты пусть на каждом процессоре будет одинаковое количество точек $m = K/M$, где K — число неизвестных (в рассматриваемом случае $K = N + 1$), M — число процессоров, и будет использоваться глобальная индексация. Таким образом, на отдельном процессоре с номером j будет находиться лишь часть уравнений системы (1) с номерами от $(j-1)m + 1$ до jm. Обозначим y_{jm} через z_j $(j = 0, \ldots, M)$, и будем искать решения системы (1) в виде

$$y_{(j-1)m+i} = z_{j-1} u_i + z_j v_i + w_i, \qquad i = 0, \ldots, m, \qquad j = 1, \ldots, M, \tag{2}$$

где u_i, v_i, w_i есть решения следующих систем уравнений:

$$\begin{aligned} &a_i u_{i-1} + b_i u_i + c_i u_{i+1} = 0, && u_{(j-1)m} = 1, && u_{jm} = 0; \\ &a_i v_{i-1} + b_i v_i + c_i v_{i+1} = 0, && v_{(j-1)m} = 0, && v_{jm} = 1; \\ &a_i w_{i-1} + b_i w_i + c_i w_{i+1} = d_i, && w_{(j-1)m} = 0, && w_{jm} = 0; \\ &i = (j-1)\,m + 1, \ldots, jm - 1; && j = 1, \ldots, M. \end{aligned} \tag{3}$$

Решения трех систем из (3) можно найти методом прогонки, причем независимо на каждом процессоре. Будем называть их предрешениями, а данный этап решения задачи — этапом нахождения предрешений.

В уравнения из системы (1) с номерами jm: $a_{jm} y_{jm-1} + b_{jm} y_{jm} + c_{jm} y_{jm+1} = d_{jm}$, $j = 0, \ldots, M$, подставляем вместо y комбинации (2). Таким образом, получаем систему трехточечных уравнений для нахождения z_j, имеющую вид

$$\begin{aligned} &A_j z_{j-1} + B_j z_j + C_j z_{j+1} = D_j, \qquad j = 1, \ldots, M-1, \\ &B_0 z_0 + C_0 z_1 = D_0, \qquad A_M z_{M-1} + B_M z_M = D_M \end{aligned} \tag{4}$$

с коэффициентами

$$\begin{aligned} &B_0 = b_0 + c_0 u_1; \qquad C_0 = c_0 v_1; \qquad D_0 = d_0 - c_0 w_1; \\ &A_j = a_{jm} u_{jm-1}, \qquad B_j = a_{jm} v_{jm-1} + b_{jm} + c_{jm} u_{jm+1}, \\ &C_j = c_{jm} v_{jm+1}, \qquad D_j = d_{jm} - a_{jm} w_{jm-1} - c_{jm} w_{jm+1}, \qquad j = 1, \ldots, M-1; \\ &A_M = a_{Mm} u_{Mm-1}; \qquad B_M = b_{Mm} + a_{Mm} v_{Mm-1}; \qquad D_M = d_{Mm} - a_{Mm} w_{Mm-1}. \end{aligned}$$

Этап решения системы (4) назовем этапом нахождения решений в гранично-процессорных точках. Число уравнений в системе (4) равно количеству процессоров, что существенно меньше количества точек задачи.

На последнем этапе восстанавливаем окончательное решение по формуле (2).

Итак, метод Яненко содержит три этапа:

1) нахождение предрешений;
2) нахождение решений в гранично-процессорных точках;
3) восстановление решения.

Первый и третий этапы выполняются независимо на каждом устройстве (универсальном процессоре или ускорителе). Второй этап требует коммуникаций между MPI-процессами. Таким образом, эффективность распараллеливания метода Яненко напрямую зависит от эффективности распараллеливания второго этапа.

Возникает вопрос, каким методом решать систему для гранично-процессорных точек. В качестве возможных вариантов выбраны метод ВП [21] и метод ПЦР [5]. Рассмотрим эти методы подробнее.

Формулы для ВП выглядят следующим образом:

$$
\begin{aligned}
&\alpha_{1,0} = -\frac{c_0}{b_0}; \qquad \beta_{1,0} = \frac{d_0}{b_0};\\
&\alpha_{1,i} = -\frac{c_i}{b_i + a_i\alpha_{1,i-1}}, \qquad \beta_{1,i} = \frac{d_i - a_i\beta_{1,i-1}}{b_i + a_i\alpha_{1,i-1}}, \qquad i = 1, \ldots, k;\\
&\alpha_{2,M} = -\frac{a_M}{b_M}; \qquad \beta_{2,M} = \frac{d_M}{b_M};\\
&\alpha_{2,i} = -\frac{a_i}{b_i + c_i\alpha_{2,i+1}}, \qquad \beta_{2,i} = \frac{d_i - c_i\beta_{2,i+1}}{b_i + c_i\alpha_{2,i+1}}, \qquad i = M-1, \ldots, k+1;\\
&y_k = \frac{\beta_{1,k} + \alpha_{1,k}\beta_{2,k+1}}{1 - \alpha_{1,k}\alpha_{2,k+1}};\\
&y_i = \alpha_{1,i}y_{i+1} + \beta_{1,i}, \quad i = 0, \ldots, k-1; \qquad y_i = \alpha_{2,i}y_{i-1} + \beta_{2,i}, \quad i = k+1, \ldots, M.
\end{aligned}
\tag{5}
$$

Особенность этого метода заключается в том, что во время его использования в параллельном режиме загружены одновременно два процессора (за исключением этапа разрешения ВП, когда работа выполняется только на одном процессоре) в отличие от классического метода прогонки [21], когда в каждый момент времени может быть загружен только один процессор. Это является очевидным преимуществом для повышения эффективности распараллеливания метода Яненко.

Теперь рассмотрим метод ПЦР. Формулы из [5] преобразования коэффициентов матрицы (4) на первом шаге редукции (с учетом введенных выше обозначений) принимают следующий вид:

$$
\begin{aligned}
&B_0^1 = -\frac{C_0}{B_1}A_1 + B_0; \qquad C_0^1 = -\frac{C_0}{B_1}C_1; \qquad D_0^1 = -\frac{C_0}{B_1}D_1 + D_0;\\
&C_i^1 = -\frac{C_i}{B_{i+1}}C_{i+1}, \qquad A_i^1 = -\frac{A_i}{B_{i-1}}A_{i-1}, \qquad D_i^1 = -\frac{A_i}{B_{i-1}}D_{i-1} - \frac{C_i}{B_{i+1}}D_{i+1} + D_i,\\
&B_i^1 = -\frac{A_i}{B_{i-1}}C_{i-1} - \frac{C_i}{B_{i+1}}A_{i+1} + B_i, \qquad i = 1, ..., M-1;\\
&A_M^1 = -\frac{A_M}{B_{M-1}}A_{M-1}; \qquad B_M^1 = -\frac{A_M}{B_{M-1}}C_{M-1} + B_M; \qquad D_M^1 = -\frac{A_M}{B_{M-1}}D_{M-1} + D_M.
\end{aligned}
\tag{6}
$$

После такого преобразования имеются две системы трехточечных уравнений, которые можно решать независимо на разных процессорах: одна из них — для четных неизвестных, другая — для нечетных.

На следующем шаге данного метода в результате применения уравнений (6) к двум полученным системам будем иметь уже четыре системы трехточечных уравнений, которые можно решать независимо. Этот процесс продолжается до тех пор, пока в каждой из полученных систем не останется по одному уравнению с одним неизвестным, из которых и находится решение системы.

Особенность метода ПЦР заключается в том, что во время работы в параллельном режиме одновременно загружены почти все процессоры (за исключением тех, на которых уже найдено решение системы уравнений).

Особенности реализации метода Яненко на гибридных ЭВМ

Необходимо учитывать разницу в архитектурах традиционных процессоров (CPU) и графических ускорителей (GPU). Программу необходимо разбивать на части, оптимальные для решения на CPU или GPU. Следует отметить, что при решении системы (1) предполагается, что коэффициенты системы уже находятся на GPU.

В данной реализации метода Яненко на GPU осуществляются нахождение предрешений по формулам (3) и восстановление решений по формулам (2). Программа для нахождения предрешений выполняется в двух асинхронных потоках [22], что достигается реализацией метода ВП. За счет использования асинхронных потоков GPU этап нахождения предрешений может выполняться почти в 2 раза быстрее.

Оценим объем копирования данных между CPU и GPU для двумерной задачи с числом точек $m_1 \times \times m_2$ и для трехмерной задачи с числом точек $n_1 \times n_2 \times n_3$ с декомпозицией по второму направлению. Объем пересылаемых данных будет рассчитываться следующим образом:

1) для нахождения коэффициентов системы (4) понадобится 6 предрешений (u_{jm-1}, v_{jm-1}, w_{jm-1}, $u_{(j-1)m+1}$, $v_{(j-1)m+1}$, $w_{(j-1)m+1}$). Соответственно для двумерного случая будем иметь $6m_2$, для трехмерного — $6n_2n_3$ пересылаемых значений;
2) для восстановления решения по формулам (2) необходимо присутствие на GPU решений в гранично-процессорных точках (z_{j-1}, z_j). Следовательно, для двумерного случая пересылается $2m_2$, для трехмерного — $2n_2n_3$ значений.

В итоге общее число пересылаемых значений для двумерной задачи будет составлять $6m_2 + 2m_2 = = 8m_2$, а для трехмерной — $6n_2n_3 + 2n_2n_3 = 8n_2n_3$.

Таким образом, используя графические ускорители, удается редуцировать систему (1) к системе (4), при этом уменьшив число уравнений до числа MPI-процессов, что, как правило, существенно меньше числа уравнений исходной системы.

Вычисление арифметической сложности методов

Для того чтобы оценить, насколько хорошо метод Яненко с различными алгоритмами прогонки на втором этапе будет использоваться в параллельном режиме, вычислим количество операций, которые нужно при этом сделать для нахождения решений системы (1).

В методе Яненко число арифметических операций складывается как сумма операций на трех этапах, описанных ранее. Соответственно искомое число есть $S = s_1 + s_{2,1} + s_{2,2} + s_3$. Здесь s_1 — число операций на этапе нахождения предрешений; $s_{2,1}$ — число операций в расчете коэффициентов матрицы системы для нахождения решений в гранично-процессорных точках; $s_{2,2}$ — число операций в методе прогонки на втором этапе; s_3 — число операций при восстановлении решения исходной системы.

Вычислим число арифметических операций, которые нужно выполнить в методе Яненко, при условии использования классического метода прогонки на втором этапе. Напомним, что число арифметических операций в классическом методе прогонки составляет $8K - 7$, где K — число неизвестных в системе (1) [23].

Сначала заметим, что если решать системы (3) методом прогонки, не учитывая особенностей этих систем, то число арифметических операций на этапе нахождения предрешений составит $s'_1 = = 3\,(8K - 7) = 24K - 21$.

Покажем, как можно упростить нахождение предрешений. Обозначим прогоночные коэффициенты первой системы (относительно u) через $\alpha_{1,i}$, $\beta_{1,i}$, второй системы (относительно v) — через $\alpha_{2,i}$, $\beta_{2,i}$, третьей системы (относительно w) — через $\alpha_{3,i}$, $\beta_{3,i}$.

Все коэффициенты $\alpha_{1,i}$, $\alpha_{2,i}$, $\alpha_{3,i}$ равны между собой, поскольку $u_{(j-1)m}$ не зависит от $u_{(j-1)m+1}$, $v_{(j-1)m}$ не зависит от $v_{(j-1)m+1}$, а $w_{(j-1)m}$ не зависит от $w_{(j-1)m+1}$. Тогда $\alpha_{1,i} = \alpha_{2,i} = \alpha_{3,i} = \alpha_i$.

Покажем методом индукции, что $\beta_{2,i} = 0$. Сначала вычислим коэффициент на левой границе области решения задачи:

$$\beta_{2,(j-1)m+1} = \frac{d_{(j-1)m+1} - a_i\beta_{2,(j-1)m}}{b_{(j-1)m+1} + a_i\alpha_{(j-1)m}} = \frac{0 - a_i v_{(j-1)m}}{b_{(j-1)m+1} + a_i\alpha_{(j-1)m}} = \frac{-a_i \cdot 0}{b_{(j-1)m+1} + a_i\alpha_{(j-1)m}} = 0.$$

Пусть теперь $\beta_{2,i} = 0$ для фиксированного i, тогда найдем значение $\beta_{2,i+1}$:

$$\beta_{2,i+1} = \frac{d_{i+1} - a_{i+1}\beta_{2,i}}{b_{i+1} + a_{i+1}\alpha_i} = \frac{0 - a_i \cdot 0}{b_{i+1} + a_{i+1}\alpha_i} = 0.$$

Таким образом, $\beta_{2,i} = 0$ для всех $i = (j-1)\,m + 1, \ldots, jm - 2$, $j = 1, \ldots, M$.

С учетом полученных выше утверждений формулы нахождения предрешений будут следующими:

$$\alpha_i = -\frac{c_i}{b_i + a_i\alpha_{i-1}}; \qquad \beta_{1,i} = \frac{d_i - a_i\beta_{1,i-1}}{b_i + a_i\alpha_{i-1}} = \frac{-a_i\beta_{1,i-1}}{b_i + a_i\alpha_{i-1}}; \qquad \beta_{2,i} = 0; \qquad \beta_{3,i} = \frac{d_i - a_i\beta_{3,i-1}}{b_i + a_i\alpha_{i-1}}. \quad (7)$$

При этом предрешения будут иметь вид

$$u_i = \alpha_i u_{i+1} + \beta_{1,i}, \quad v_i = \alpha_i v_{i+1}, \quad w_i = \alpha_i w_{i+1} + \beta_{3,i}, \quad i = (j-1)\,m + 1, \ldots, jm - 1, \quad j = 1, \ldots, M. \quad (8)$$

Для этапа нахождения предрешений требуемое число операций $s_1 = s_{1,1} + s_{1,2}$, где $s_{1,1}$ и $s_{1,2}$ — число операций на прямом и обратном ходе прогонки соответственно. Из (7) и (8) следует, что $s_{1,1} = (2 + 1 + 2 + 0 + 3) \cdot (m - 2) = 8\,(m-2)$, а $s_{1,2} = (2 + 1 + 2) \cdot (m - 2) = 5\,(m-2)$. Таким образом, $s_1 = s_{1,1} + s_{1,2} = 8\,(m-2) + 5\,(m-2) = 13\,(m-2)$, что значительно меньше, чем $s_1' = 24\,(m+1) - 21 = 24m + 3$. То есть, учитывая структуру систем уравнений, удается уменьшить число арифметических операций почти в 2 раза по сравнению с применением обычного метода прогонки к решению трех систем уравнений.

В итоге число операций на всех этапах в методе Яненко будет следующим (из (4) число процессоров равно M):

$$s_1 = 13\,(m-2); \; s_{2,1} = 5 + 10\,(M-1) + 5 = 10M; \; s_{2,2} = 8\,(M+1) - 7 = 8M + 1; \; s_3 = 5\,(m-2);$$

$$S = 13\,(m-2) + 18M + 1 + 5\,(m-2) = 18\,(m-2) + 18M + 1.$$

В случае, когда число процессоров равно 1 ($M = 1$, $m = K$), $S = 18\,(K-2) + 18 + 1 = 18K - 17$ для системы (1). Видно, что при расчете на одном процессоре метод Яненко будет проигрывать классическому методу прогонки, в котором $S = 8K - 7$.

Попробуем оценить арифметическую сложность алгоритмов, которые планируется применить на втором этапе метода Яненко, а именно ВП и ПЦР, предварительно отметив, что число арифметических операций для классического метода прогонки будет составлять $8M + 1$, (поскольку K в этом случае равно $M + 1$).

Из (5) видно, что для метода ВП число арифметических операций равно $2 + 6k + 2 + 6\,(M - k - 1) + 4 + 2k + 2\,(M - k) = 8M + 3$. Это немногим больше, чем в классическом методе прогонки.

Теперь рассмотрим метод ПЦР. Получаемые в дальнейшем оценки будут составлять слагаемое $s_{2,2}$ в формуле арифметической сложности метода Яненко.

Из (6) видно, что на этапе преобразования коэффициентов матрицы для гранично-процессорных точек число операций на первом шаге редукции будет равно $S_{2,1} = 6 + (2 + 1 + 4 + 1 + 4)\,(M-1) + 6 = 6 + 12M - 12 + 6 = 12M$.

На втором шаге преобразований получаются две независимые системы трехточечных уравнений, которые можно решать отдельно друг от друга на разных процессорах (с четными и нечетными номерами соответственно). Формулы преобразований будут теми же, но изменится индексация.

Для системы, в которой ненулевые коэффициенты стоят на четных местах, диапазон изменения индекса следующий: $i = 0, 2, \ldots, 2\,[M/2]$, а для системы, в которой они стоят на нечетных местах, $i = 1, 3, \ldots, 2\,[(M-1)/2] + 1$. В этом случае число арифметических операций будет равно

$$\begin{aligned} S_{2,2} &= 12\left[\frac{M}{2}\right] + 12\left[\frac{M-1}{2}\right] = 12\left(\left(\frac{M}{2} - \left\{\frac{M}{2}\right\}\right) + \left(\frac{M-1}{2} - \left\{\frac{M-1}{2}\right\}\right)\right) = \\ &= 12\left(\left(\frac{M}{2} + \frac{M-1}{2}\right) - \left(\left\{\frac{M}{2}\right\} + \left\{\frac{M-1}{2}\right\}\right)\right) = 12\left(\left(M - \frac{1}{2}\right) - \frac{1}{2}\right) = 12\,(M-1)\,. \end{aligned}$$

Здесь и ниже всюду $[M]$ и $\{M\}$ — соответственно целая и дробная части числа M.

На следующем шаге уже будет четыре системы трехточечных уравнений, которые можно решать независимо. Аналогично предыдущему

$$\begin{aligned} S_{2,3} &= 12\left[\frac{M}{4}\right] + 12\left[\frac{M-1}{4}\right] + 12\left[\frac{M-2}{4}\right] + 12\left[\frac{M-3}{4}\right] = \\ &= 12\left(\frac{M}{4} + \frac{M-1}{4} + \frac{M-2}{4} + \frac{M-3}{4}\right) - 12\left(\left\{\frac{M}{4}\right\} + \left\{\frac{M-1}{2}\right\} + \left\{\frac{M-2}{4}\right\} + \left\{\frac{M-3}{4}\right\}\right) = \\ &= 12\,(M - 6/4) - 12\,(3/4 + 2/4 + 1/4 + 0) = 12\,(M-3)\,. \end{aligned}$$

Этот процесс продолжается до тех пор, пока в каждой из получившихся систем не останется по одному уравнению с одним неизвестным, из которых будет найдено решение системы. Число таких итераций $H = [\log_2(M+1)] + d$, где $d = 0$, если M является степенью двойки, и $d = 1$, если M степенью двойки не является.

Общее число арифметических операций для метода ПЦР будет равно

$$\begin{aligned} s_{2,2} &= S_{2,1} + S_{2,2} + \ldots + S_{2,H} = 12\,(M + M - 1 + M - 3 + \ldots + M - (2H-3)) = \\ &= 12HM - 12\,(1 + 3 + \ldots + (2H-3)) = 12HM - 12\,(((1 + 2H - 3)/2)\,(H-1)) = \\ &= 12HM - 12\,(H-1)^2 = 12\,([\log_2(M+1)] + d)\,M - 12\,([\log_2(M+1)] + d - 1)^2\,. \end{aligned}$$

Окончательно $s_{2,2} = 12M\,[\log_2(M+1)] - 12\,([\log_2(M+1)] + d - 1)^2 + 12dM$, где $d = 0$, если M является степенью двойки, и $d = 1$, если M степенью двойки не является. Очевидно, что это число значительно больше, чем в методе ВП, но здесь практически всегда все процессоры выполняют работу независимо (за исключением тех случаев, когда на процессоре уже найдено решение системы уравнений).

Сведем полученные результаты для методов распараллеливания прогонки:

1. Классический метод прогонки (метод правой прогонки):
   - число арифметических операций $8M + 1$;
   - число независимо выполняемых MPI-процессов равно 1.
2. Метод ВП:
   - число арифметических операций $8M + 3$;
   - число независимо выполняемых MPI-процессов равно 2 (почти всегда).
3. Метод ПЦР:
   - число арифметических операций $12M\,[\log_2(M+1)] - 12\,([\log_2(M+1)] + d - 1)^2 + 12dM$, где $d = 0$, если M является степенью двойки, и $d = 1$, если M степенью двойки не является;
   - число независимо выполняемых MPI-процессов $M - 1$ (почти всегда).

Из полученных оценок следует, что сказать заранее, какой метод прогонки будет эффективнее выполняться на втором этапе метода Яненко, нельзя. Для метода ВП число операций меньше, но и число независимо выполняемых MPI-процессов мало. Для метода ПЦР число операций существенно больше, но и число независимых MPI-процессов почти максимально.

Далее получим практические результаты, которые можно будет сравнить со сделанными выше теоретическими оценками.

Тестирование методов

В данном разделе приведены результаты численных экспериментов для определения следующих характеристик рассмотренных методов распараллеливания прогонки:

- эффективность распараллеливания (масштабируемость методов) на параллельных системах с распределенной памятью и гибридных ЭВМ с графическими ускорителями;
- ускорение версии программы с использованием GPU относительно программы без использования GPU.

Так как метод Яненко и ПКМ предназначены для решения задач на нескольких вычислительных узлах, исследуем характеристики методов при использовании большого количества вычислительных узлов.

В методике РАМЗЕС-КП метод Яненко реализован только для расчета двумерных задач, поэтому при тестировании будем рассматривать двумерную задачу теплопроводности с числом точек $1\,000 \times 1\,000$, рассчитанную как на универсальных процессорах, так и на графических ускорителях. Будем учитывать только время расчета коэффициентов матрицы и нахождения решения методом прогонки.

Будем масштабировать задачу методом умножения — при увеличении числа MPI-процессов в 2 раза число точек задачи также увеличивается в 2 раза.

Введем следующие обозначения: $S_p = pt_1/t_p$ — ускорение от использования p MPI-процессов; $V_p = (S_p/p) \cdot 100\,\%$ — эффективность распараллеливания, где t_1, t_p — время выполнения программы на одном и на p вычислительных устройствах[2] соответственно.

Сначала сравним времена решения системы (1) с различными методами распараллеливания прогонки на втором этапе метода Яненко на гибридной архитектуре (с использованием GPU). Результаты представлены в таблице. Здесь замерялось время расчетов задачи (в секундах), в которых первый и третий этапы выполнялись на ускорителе одинаково, а второй этап на универсальных узлах — по-разному: методом Яненко с ВП и методом Яненко с ПЦР. В строке таблицы ВП/ПЦР приведено отношение времен расчетов указанными методами.

Из таблицы видно, что при использовании до 8 гибридных узлов время расчета с ВП меньше времени с ПЦР, но на 64 ускорителях метод ПЦР выполняется на четверть быстрее метода ВП. Поэтому всюду в дальнейших оценках на втором этапе метода Яненко будем подразумевать использование ПЦР.

Теперь сравним времена решения двумерной задачи с использованием ПКМ и метода Яненко на двух архитектурах — с применением графических ускорителей и без них.

На рис. 2 приведена зависимость времени выполнения прогонки различными методами распараллеливания от числа узлов. Можно заметить, что метод Яненко "работает" быстрее, чем ПКМ, как на универсальной (более чем в 2 раза на 64 узлах), так и на гибридной архитектуре (более чем в 18 раз на 64 узлах). Такая большая разница во времени на 64 узлах объясняется тем, что, как было показано выше, на гибридной архитектуре ПКМ вырождается в последовательный метод прогонки. Также из рис. 2 видно, что реализация метода Яненко на гибридной архитектуре позволила получить ускорение до 3,67 раза по сравнению с универсальной архитектурой.

Представим данные рис. 2 в виде зависимости эффективности распараллеливания V_p от числа узлов (рис. 3). Видно, что эффективность метода Яненко как на универсальной, так и на гибридной

[2]Для универсальной архитектуры (только универсальные процессоры) термин *вычислительное устройство* означает одно ядро, а для гибридной архитектуры (универсальные процессоры вместе с графическими ускорителями) — один ускоритель.

Сравнение двух методов нахождения решений в гранично-процессорных точках с использованием GPU

Метод	Количество гибридных узлов						
	1	2	4	8	16	32	64
ВП	6,66	14,60	14,98	15,32	16,18	17,90	21,21
ПЦР	6,66	14,60	15,14	15,45	15,87	16,35	16,93
ВП/ПЦР	1,00	1,00	0,99	0,99	1,02	1,09	1,25

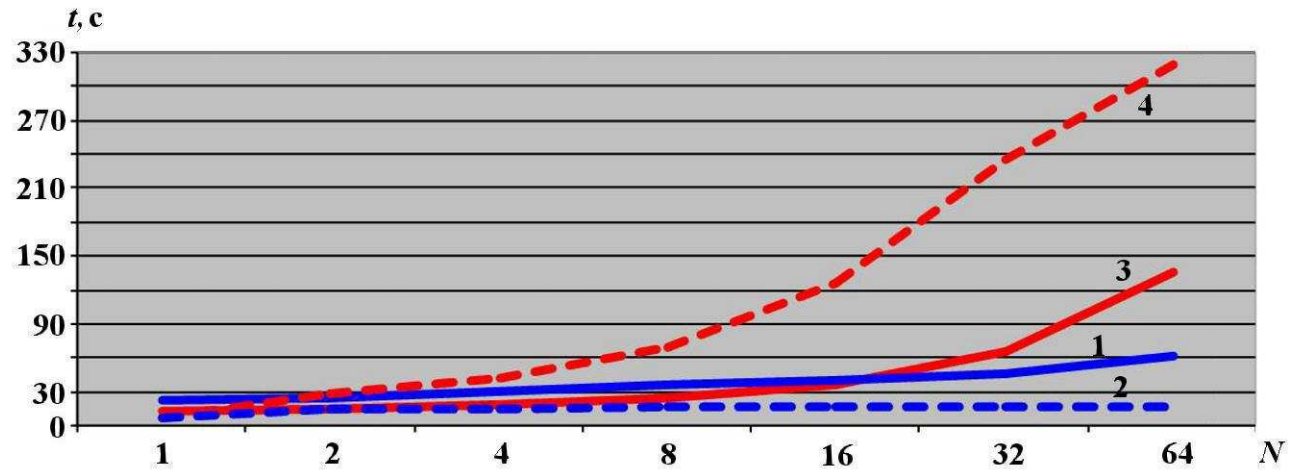

Рис. 2. Время выполнения методов распараллеливания прогонки в зависимости от числа узлов: 1 — метод Яненко на универсальной архитектуре; 2 — метод Яненко на гибридной архитектуре; 3 — ПКМ на универсальной архитектуре; 4 — ПКМ на гибридной архитектуре

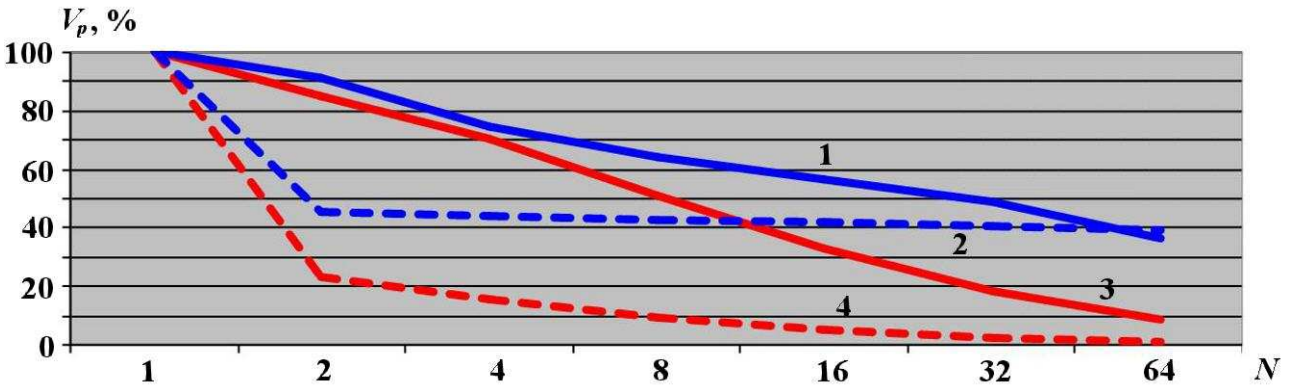

Рис. 3. Эффективность методов распараллеливания прогонки в зависимости от числа узлов: 1 — метод Яненко на универсальной архитектуре; 2 — метод Яненко на гибридной архитектуре; 3 — ПКМ на универсальной архитектуре; 4 — ПКМ на гибридной архитектуре

архитектуре при масштабировании до 64 узлов выше 36 %. Эффективность ПКМ на универсальной архитектуре падает более плавно, чем на гибридной архитектуре, а именно на 8 узлах эффективность составляет более 50 %, а на 64 узлах — около 9 %. На гибридной же архитектуре на 64 узлах эффективность составила около 1 %.

Заключение

Выполнена реализация и сравнение двух методов решения СЛАУ с трехдиагональной матрицей: ПКМ и метода Яненко.

Реализация метода Яненко отличается от описанной в [20] тем, что на этапе нахождения решений в гранично-процессорных точках может использоваться как ВП, так и метод ПЦР. Численные эксперименты показали, что метод ПЦР позволяет быстрее решать СЛАУ, хотя и обладает большей вычислительной сложностью. Такой результат достигается за счет более высокой эффективности распараллеливания на большом числе вычислительных устройств.

По эффективности распараллеливания метод Яненко выигрывает у ПКМ как на универсальной, так и на гибридной архитектуре.

Список литературы

1. *Быков А. Н., Веселов В. А., Воронин Б. Л., Ерофеев А. М.* Методика РАМЗЕС-КП для расчета пространственных движений многокомпонентных теплопроводных сред в эйлеро-лагранжевых координатах // Труды РФЯЦ-ВНИИЭФ. 2008. Вып. 13. С. 50—57.
2. *Сапронов И. С., Быков А. Н.* Параллельно-конвейерный алгоритм // Атом. 2009. № 44. С. 24—25.
3. *Davidson A., Zhang Y., Owens J. D.* An auto-tuned method for solving large tridiagonal systems on the GPU // IPDPS'11. Proc. 2011 IEEE Int. Parallel & Distributed Processing Symposium. Washington, DC: IEEE Press, 2011. P. 956—965.
4. *Kim H.-S., Wu S., Chang L.-W., Hwu W.-M.* A scalable tridiagonal solver for GPUs // ICPP'11. Proc. 2011 Int. Conf. on Parallel Processing. Washington, DC: IEEE Press, 2011. P. 444—453.
5. *Hockney R. W., Jesshope C. R.* Parallel Computers: Architecture, Programming and Algorithm. Bristol: Hilger, 1986. P. 274—280.
6. *Conte S. D., Boor C. W. D.* Elementary Numerical Analysis: An Algorithmic Approach. 3rd ed. NY: McGraw-Hill Higher Education, 1980. P. 153—156.
7. NVIDIA Corporation. NVIDIA CUDA Sparse Matrix Library (cuSPARSE). https://developer.nvidia.com/cusparse.
8. *Sengupta S., Harris M., Zhang Y., Owens J. D.* Scan primitives for GPU computing // Graphics Hardware. Aug. 2007. P. 97—106.
9. *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.
10. *Davidson A., Owens J. D.* Register Packing for Cyclic Reduction: A Case Study. http://idav.ucdavis.edu/func/return_pdf?pub_id=1052.
11. *Egloff D.* High Performance Finite Difference PDE Solvers on GPUs. http://download.quantalea.net/fdm_gpu.pdf.
12. *Hockney R. W.* A fast direct solution of Poisson's equation using Fourier analysis // J. ACM. 1965. Vol. 12. P. 95—113.
13. *Sakharnykh N.* Efficient tridiagonal solvers for ADI methods and fluid simulation // NVIDIA GPU Technology Conference. San Jose. September 2010.
14. *Zhang Y., Cohen J., Owens J. D.* Fast tridiagonal solvers on the GPU // Proc. 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. NY: ACM, 2010. P. 127—136.
15. *Zhang Y., Cohen J., Davidson A. A., Owens J. D.* A hybrid method for solving tridiagonal systems on the GPU // GPU Computing Gems. 2011. Vol. 2. P. 117—133.

16. *Stone H. S.* An efficient parallel algorithm for the solution of a tridiagonal linear system of equations // J. ACM. 1973. Vol. 20. P. 27—38.

17. *Polizzi E., Sameh A.* A parallel hybrid banded system solver: The SPIKE algorithm // Parallel Computing. 2006. Vol. 32, No. 2. P. 177—194.

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

19. *Chang L.-W., Stratton J. A., Kim H.-S., Hwu W.-M.* A scalable, numerically stable, high-performance tridiagonal solver using GPUs // SC'12. Proc. Int. Conf. on High Performance Computing, Networking, Storage and Analysis. Los Alamitos: IEEE Press, 2012. Article No 27.

20. *Яненко Н. Н., Коновалов А. Н., Бугров А. Н., Шустов Г. В.* Об организации параллельных вычислений и "распараллеливании" прогонки // Числ. методы мех. спл. среды. 1978. Т. 9, № 7. С. 139—146.

21. *Самарский А. А.* Введение в численные методы. М.: Наука, 1982. С. 34—38.

22. *Боресков А. В, Харламов А. А., Марковский Н. Д. и др.* Параллельные вычисления на GPU. Архитектура и программная модель CUDA. М.: Изд-во Московского ун-та, 2012. С. 241—256.

23. *Вержбицкий В. М.* Вычислительная линейная алгебра. М.: Директ-Медиа, 2013. С. 296—299.

Статья поступила в редакцию 17.09.15.

COMPARISON OF TWO SWEEP PARALLELING METHODS FOR HYBRID COMPUTERS WITH GRAPHICS PROCESSING UNITS / A. A. Fedorov, A. N. Bykov (FSUE "RFNC-VNIIEF", Sarov, Nizhny Novgorod region).

The paper describes two methods (the parallel pipeline and the Yanenko method) for solving a system of linear algebraic equations with a tridiagonal matrix on GPU-accelerated parallel computes. Specific features of their implementation both on parallel computers having no accelerators and on hybrid computers are discussed. Arithmetic complexity of the Yanenko method is analyzed, and results of numerical scalability experiments are reported.

Keywords: system of linear algebraic equations with tridiagonal matrix, sweep method, parallel pipeline, Yanenko method, GPU-accelerated parallel computers, code RAMZES-KP.