Исследование масштабируемости алгоритма Чиммино для линейных неравенств*

И.М. Соколинская, Л.Б. Соколинский

Южно-Уральский государственный университет

Работа посвящена исследованию масштабируемости алгоритма Чиммино для решения систем неравенств. Данный алгоритм является типичным представителем класса итерационных проекционных алгоритмов для решения систем линейных уравнений и неравенств. Для аналитического анализа масштабируемости используется модель параллельных вычислений BSF (Bulk Synchronous Farm). Дается представление алгоритма Чиммино в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*. Выводятся аналитические оценки верхней границы масштабируемости алгоритма для многопроцессорных вычислительных систем с распределенной памятью. Приводятся данные о реализация алгоритма Чиммино над списками на языке C++ с использованием программного шаблона BSF и библиотеки параллельного программирования MPI. Демонстрируются результаты масштабных вычислительных экспериментов, выполненных на кластерной вычислительной системе. На основе экспериментальных результатов дается анализ адекватности оценок, полученных аналитическим путем с помощью стоимостных метрик модели BSF.

Ключевые слова: алгоритм Чиммино, система линейных неравенств, итерационный алгоритм, проекционный алгоритм, релаксация, модель параллельных вычислений BSF, оценка масштабируемости, эффективность распараллеливания, кластерные вычислительные системы.

Введение

Задача решения системы линейных неравенств возникает во многих областях вычислительной математики. В качестве примеров можно привести линейное программирование [1,2], восстановление изображений по проекциям [3], обработку изображений в магнитно-резонансной томографии [4], радиационную терапию с модулируемой интенсивностью [5]. В настоящее время известно достаточно много методов решения систем линейных неравенств, среди которых можно выделить класс «самоисправляющихся» итерационных методов, допускающих эффективное распараллеливание. Пионерскими работами здесь являются публикации [6,7], в которых предложен метод релаксаций Агмона-Моцкина-Шоенберга для решения систем линейных неравенств. Метод релаксаций можно отнести к алгоритмам проекционного типа, использующим операцию ортогональной проекции на гиперплоскость в Евклидовом пространстве. Одним из первых итерационных алгоритмов проекционного типа был алгоритм Чиммино (Cimmino) [8], предназначенный для решения систем линейных уравнений и неравенств. Алгоритм Чиммино оказал большое влияние на развитие вычислительной математики [9]. Обобщениям и расширениям алгоритма Чиммино было посвящено значительное количество работ (см., например, [3,10–13]).

Системы линейных неравенств, возникающие при решении практических задач, во многих случаях могут включать в себя сотни миллионов переменных, при этом количество неравенств составляет величину того же порядка [2]. В этом случае актуальным становится вопрос разработки масштабируемых параллельных алгоритмов для решения сверхбольших систем линейных неравенств на многопроцессорных системах с распределенной памятью. При создании параллельных алгоритмов для больших многопроцессорных систем важно уже на ранней стадии разработки алгоритма (до написания программы) получить аналитические оценки его масшта-

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00352 а, Правительства РФ в соответствии с Постановлением №211 от 16.03.2013 г. (соглашение № 02.A03.21.0011) и Министерства образования и науки РФ (государственное задание 2.7905.2017/8.9).

бируемости. Для этой цели используются различные модели параллельных вычислений [14]. В настоящее время существует большое количество различных параллельных вычислительных моделей. Наиболее известными среди них являются модели PRAM [15], BSP [16] и LogP [17]. Указанные модели подверглись обобщениям и уточнениям, породив целые семейства, насчитывающие десятки параллельных вычислительных моделей (см., например, [18–20]). Задача разработки новых параллельных вычислительных моделей не утратила актуальности и в настоящее время. Это объясняется тем, что невозможно создать модель параллельных вычислений – «хорошую во всех отношениях». Поэтому необходимо ограничиваться определенными многопроцессорными архитектурами и определенными классами алгоритмов. В работе [21] была предложена модель параллельных вычислений BSF (Bulk Synchronous Farm), ориентированная на кластерные вычислительные системы и алгоритмы итерационного типа. Указанная модель позволяет с высокой точностью оценить верхнюю границу масштабируемости параллельного итерационного алгоритма до написания программы. Пример использования модели BSF приводится в работе [22].

Целью настоящей статьи является исследование масштабируемости алгоритма Чиммино для решения больших систем линейных неравенств на многопроцессорных системах с распределенной памятью путем использования модели параллельных вычислений BSF. Статья имеет следующую структуру. В разделе 1 делается постановка задачи и дается описание алгоритма Чиммино для решения системы линейных неравенств. В разделе 2 строится представление алгоритма Чиммино в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*, определяемых формализмом Бёрда-Миртенса. Раздел 3 посвящен аналитическому исследованию масштабируемости алгоритма Чиммино над списками с помощью метрик модели BSF; приводятся итоговые формулы для оценки ускорения и эффективности распараллеливания; вычисляется верхняя граница масштабируемости алгоритма в зависимости от размера задачи. В разделе 4 дается информация о реализации алгоритма Чиммино над списками, выполненной на языке C++ с использованием программного каркаса BSF и библиотеки параллельного программирования MPI, и приводится сравнение результатов, полученных аналитическим и экспериментальным путем. В заключении суммируются полученные результаты и намечаются направления дальнейших исследований.

1. Алгоритм Чиммино для неравенств

Пусть в евклидовом пространстве $\mathbb{R}^n$ задана совместная система линейных неравенств

$$l_i(x) = \langle a_i, x \rangle - b_i \le 0 \qquad (i = 1, \ldots, m). \tag{1}$$

Необходимо найти одно любое решение системы (1). Для решения этой задачи удобно использовать геометрический язык. В соответствии с этим $x = (x_1, \ldots, x_n)$ рассматривается как точка в n-мерном евклидовом пространстве $\mathbb{R}^n$, а каждое неравенство $l_i(x) \le 0$ – как полупространство P_i. Таким образом, множество решений системы (1) можно представить как выпуклый n-мерный многогранник $M = \prod_{i=1}^{m} P_i$. Каждое уравнение $l_i(x) = 0$ определяет гиперплоскость H_i:

$$H_i = \left\{ x \in \mathbb{R}^n \mid \langle a_i, x \rangle = b_i \right\}. \tag{2}$$

Ортогональная проекция $\pi_{H_i}(x)$ точки x на гиперплоскость H_i вычисляется по следующей формуле:

$$\pi_{H_i}(x) = x + \frac{b_i - \langle a_i, x \rangle}{\|a_i\|^2} a_i, \tag{3}$$

где $\|\cdot\|$ – евклидова норма. Определим *ортогональное отражение* точки x относительно гиперплоскости H_i следующим образом

$$\rho_{H_i}(x) = \pi_{H_i}(x) - x = \frac{b_i - \langle a_i, x \rangle}{\|a_i\|^2} a_i. \quad (4)$$

Алгоритм Чиммино для неравенств с равными весовыми коэффициентами состоит из следующих шагов:

Шаг 1. $k := 0$; $x_0 := \mathbf{0}$.

Шаг 2. $x_{k+1} := x_k + \frac{\lambda}{m} \sum_{i=1}^{m} \rho_{H_i}(x_k)$.

Шаг 3. Если $\|x_{k+1} - x_k\|^2 < \varepsilon$, перейти на шаг 5.

Шаг 4. $k := k + 1$; перейти на шаг 2.

Шаг 5. Стоп.

В качестве начального приближения x_0 в алгоритме Чиммино может быть взят произвольный вектор в Ў^n. На шаге 1 в качестве начального приближения x_0 берется нулевой вектор. На шаге 2 организуется итерационный процесс:

$$x_{k+1} = x_k + \frac{\lambda}{m} \sum_{i=1}^{m} \rho_{H_i}(x_k). \quad (5)$$

Каждое новое приближение x_{k+1} получается путем добавления к текущему приближению x_k центра масс ортогональных отражений точки x_k относительно гиперплоскостей H_1, K, H_m, умноженного на *коэффициент релаксации* λ. Известно [10], что при $0 < \lambda < 2$ итерационный процесс (5) сходится к точке, принадлежащей многограннику M.

2. Представление алгоритма Чиммино в виде операций над списками

Для получения аналитических оценок алгоритма в соответствии с метриками модели BSF он должен быть представлен в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*, определяемых формализмом Бёрда-Миртенса (Bird-Meertens formalism) [23]. Для заданных функции $F: \text{A} \to \text{B}$ и списка $[a_1, \text{K}, a_m]$ функция высшего порядка *Map* формирует новый список той же длины путем применения функции F ко всем элементам списка $[a_1, \text{K}, a_m]$:

$$Map(F, [a_1, \text{K}, a_m]) = [F(a_1), \text{K}, F(a_m)]. \quad (6)$$

Для заданных бинарной ассоциативной операции $\oplus: \text{B} \times \text{B} \to \text{B}$ и списка $[b_1, \text{K}, b_m]$ функция высшего порядка *Reduce* редуцирует список $[b_1, \text{K}, b_m]$ к одному элементу путем многократного применения операции $\oplus$ к элементам списка:

$$Reduce(\oplus, [b_1, \text{K}, b_m]) = b_1 \oplus \text{K} \oplus b_m. \quad (7)$$

В контексте алгоритма Чиммино определим список L_{Map} следующим образом:

$$L_{Map} = [i_1, \text{K}, i_m], \quad (8)$$

где $i_k \in \{1, \text{K}, m\}$ и $i_k \neq i_l$ при $k \neq l$ $(k, l = 1, \text{K}, m)$. Другими словами, L_{Map} – список номеров неравенств системы (1), взятых в некотором порядке. Для произвольного $x \in \text{Ў}^n$ определим функцию $F_x: \{1, \text{K}, m\} \to \text{Ў}^n$:

$$F_x(i) = \rho_{H_i}(x) \quad (9)$$

для всех $i \in \{1, \text{K}, m\}$. Иначе говоря, функция $F_x(i)$ вычисляет ортогональное отражение точки x относительно гиперплоскости H_i. Для произвольного $x \in \text{Ў}^n$ определим список $L_{Reduce}^{(x)} \subset \text{Ў}^n$ следующим образом:

$$L_{Reduce}^{(x)} = [F_x(i_1), \text{K}, F_x(i_m)]. \quad (10)$$

Список $L_{Reduce}^{(x)}$ включает в себя ортогональные отражения точки x на гиперплоскости H_1, K, H_m, взятые в порядке, определяемом списком L_{Map}. Таким образом, список $L_{Reduce}^{(x)}$ получается из списка L_{Map} путем применения к нему функции высшего порядка Map, использующей в качестве параметра функцию F_x:

$$L_{Reduce}^{(x)} = Map(F_x, L_{Map}). \quad (11)$$

Определим бинарную ассоциативную операцию $\oplus : \breve{\mathrm{Y}}^n \times \breve{\mathrm{Y}}^n \to \breve{\mathrm{Y}}^n$ следующим образом:

$$x \oplus y = x + y \quad (12)$$

для любых $x, y \in \breve{\mathrm{Y}}^n$. В данном случае операция $\oplus$ выполняет обычное сложение векторов. Тогда сумма ортогональных отражений точки x может быть получена путем применения к списку $L_{Reduce}^{(x)}$ функции высшего порядка *Reduce*, использующей в качестве параметра операцию сложения векторов $\oplus$:

$$\sum_{i=1}^{m} \rho_{H_i}(x) = Reduce(\oplus, L_{Reduce}^{(x)}). \quad (13)$$

Теперь мы можем записать *алгоритм Чиммино в виде операций над списками*:

Шаг 1. $k := 0$; $x_0 := \mathbf{0}$; $L_{Map} := [1, \text{K}, m]$.

Шаг 2. $L_{Reduce}^{(x_k)} := Map(F_{x_k}, L_{Map})$.

Шаг 3. $s := Reduce(\oplus, L_{Reduce}^{(x_k)})$.

Шаг 4. $x_{k+1} := x_k + \frac{\lambda}{m} \cdot s$

Шаг 5. Если $\left\| x_{k+1} - x_k \right\|^2 < \varepsilon$, перейти на шаг 7.

Шаг 6. $k := k + 1$; перейти на шаг 2.

Шаг 7. Стоп.

Модель BSF предполагает, что алгоритм выполняется вычислительной системой, состоящей из одного узла-мастера и K узлов-рабочих ($K > 0$). При этом шаг 1 выполняется и мастером, и рабочими в ходе инициализации итерационного процесса; шаг 2 (*Map*) выполняется только на узлах-рабочих; шаг 3 (*Reduce*) выполняется на узлах-рабочих и частично на узле-мастере; шаги 4-6 выполняются только на узле-мастере. В модели BSF предполагается, что все арифметические операции (сложение и умножение), а также операции сравнения над числами с плавающей точкой занимают одинаковое время τ_{op}.

3. Аналитическое исследование масштабируемости

Введем следующие обозначения для анализа масштабируемости алгоритма Чиммино:

c_s – количество вещественных чисел, передаваемых от мастера одному рабочему;

c_{Map} – количество арифметических операций, выполняемых на шаге *Map* (шаг 2 алгоритма);

c_{Reduce_W} – количество арифметических операций, выполняемых одним рабочим на шаге *Reduce* (шаг 3 алгоритма);

c_r – количество вещественных чисел, передаваемых от одного рабочего мастеру;

c_{Reduce_M} – количество арифметических операций, выполняемых мастером на шаге *Reduce* (шаг 3 алгоритма);

c_a – количество арифметических операций, выполняемых мастером на шагах 4 и 5 алгоритма (агрегация).

Вычислим указанные значения. В начале каждой итерации мастер передает каждому рабочему очередное приближение x_k, являющееся вектором размерности n. Следовательно:

$$c_s = n. \tag{14}$$

Подсчитаем количество арифметических операций, выполняемых на шаге *Map*. Для каждого элемента списка L_{Map} вычисляется один вектор по формуле (4). Заметим, что величины $\|a_i\|^2$ ($i = 1,\mathrm{K},m$) не зависят от x_k, и поэтому могут быть вычислены заранее на этапе инициализации. С учетом этого, количество операций при вычислении одного ортогонального отражения точки x_k составляет $n^2 + 2$. Умножив это число на количество неравенств, получаем

$$c_{Map} = m(n^2 + 2). \tag{15}$$

При выполнении шага *Reduce* список L_{Reduce}, состоящий из m векторов, делится поровну между всеми рабочими. Везде далее мы предполагаем, что $K \le m$. Для простоты мы будем считать, что m всегда кратно количеству рабочих K. Тогда на каждого рабочего приходится список длиной m/K. Для сложения m/K векторов размерности n требуется $((m/K)-1)n$ арифметических операций. Следовательно:

$$c_{Reduce\_W} = ((m/K)-1)n. \tag{16}$$

Вектор, полученный каждым рабочим на шаге *Reduce*, пересылается мастеру. Значит:

$$c_r = n. \tag{17}$$

Мастер должен выполнить шаг *Reduce* для полученных от рабочих векторов, то есть он должен сложить K векторов размерности n. Таким образом:

$$c_{Reduce\_M} = (K-1)n. \tag{18}$$

Выполнение шага 4 требует $2n$ операций (константу λ/m мы предполагаем вычисленной заранее). Выполнение шага 5 требует $3n-1$ арифметических операций и одну операцию сравнения. Отсюда получаем следующую формулу:

$$c_a = 5n. \tag{19}$$

Положим τ_{op} – время, затрачиваемое рабочим на выполнение одной арифметической операции, τ_{tr} – время, затрачиваемое на пересылку по сети одного вещественного числа без учета латентности. Тогда мы получаем следующие значения для стоимостных параметров модели BSF [21] в случае алгоритма Чиммино:

$$t_s = \tau_{tr} n; \tag{20}$$

$$t_w = \tau_{op} \cdot \left(m(n^2 + 2) + (m-K)n\right); \tag{21}$$

$$t_r = \tau_{tr} n; \tag{22}$$

$$t_p = \tau_{op} \cdot \left((K-1)n + 5n\right). \tag{23}$$

Уравнение (20), полученное на основе (14), дает оценку времени t_s, затрачиваемого мастером на передачу сообщения одному рабочему без учета латентности. Формула (21) получена с использованием формул (15) и (16). В соответствии со стоимостной метрикой модели BSF t_w обозначает суммарные временные затраты рабочих на вычисления над локальными данными. Поэтому выражение, стоящее в правой части уравнения (16), при подстановке в формулу (21) умножено на количество рабочих K. Уравнение (22), полученное на основе (17), дает оценку времени t_r, затрачиваемого мастером на получение сообщения от одного рабочего без учета латентности. Формула (23), полученная на основе (18) и (19), вычисляет время t_p, затрачиваемое мастером на выполнение своей части шага *Reduce*, вычисление очередного приближения и проверку условия завершения.

В соответствии с этим время решения задачи системой в составе одного мастера и одного рабочего ($K=1$) может быть вычислено следующим образом:

$$T_1 = 2(L+\tau_{tr}n)+\tau_{op}\cdot\left(m(n^2+2)+(m-1)n+5n\right). \quad (24)$$

Здесь L обозначает латентность, которая определяется как время передачи сообщения длиной в один байт от мастера рабочему. Время решения задачи системой в составе одного мастера и K рабочих может быть вычислено по следующей формуле:

$$T_K = 2(L+\tau_{tr}n)K+\tau_{op}\cdot\left(\frac{m(n^2+n+2)}{K}+Kn+3n\right) \quad (25)$$

На основании формул (24) и (25) мы можем записать формулу для ускорения a как функции от K :

$$a(K)=\frac{T_1}{T_K}=\frac{2(L+\tau_{tr}n)+\tau_{op}\cdot\left(m(n^2+n+2)+4n\right)}{2(L+\tau_{tr}n)K+\tau_{op}\cdot\left(\frac{m(n^2+n+2)}{K}+Kn+3n\right)}. \quad (26)$$

Для определения верхней границы масштабируемости алгоритма Чиммино в соответствии с методикой, описанной в [21], вычислим производную $a'(K)$ и решим уравнение

$$a'(K)=0. \quad (27)$$

С помощью несложных алгебраических преобразований из формулы (26) получаем следующую формулу для производной ускорения:

$$a'(K)=\frac{\left(2(L+\tau_{tr}n)+\tau_{op}\cdot\left(m(n^2+n+2)+4n\right)\right)\cdot\left(\tau_{op}\cdot\left(\frac{m(n^2+n+2)}{K^2}-n\right)-2(L+\tau_{tr}n)\right)}{\left(2(L+\tau_{tr}n)K+\tau_{op}\cdot\left(\frac{m(n^2+n+2)}{K}+Kn+3n\right)\right)^2}. \quad (28)$$

Решим уравнение

$$\frac{\left(2(L+\tau_{tr}n)+\tau_{op}\cdot\left(m(n^2+n+2)+4n\right)\right)\cdot\left(\tau_{op}\cdot\left(\frac{m(n^2+n+2)}{K^2}-n\right)-2(L+\tau_{tr}n)\right)}{\left(2(L+\tau_{tr}n)K+\tau_{op}\cdot\left(\frac{m(n^2+n+2)}{K}+Kn+3n\right)\right)^2}=0. \quad (29)$$

Разделив обе части уравнения (29) на положительную величину

$$\left(2(L+\tau_{tr}n)+\tau_{op}\cdot\left(m(n^2+n+2)+4n\right)\right)$$

и умножив на положительную величину

$$\left(2(L+\tau_{tr}n)K+\tau_{op}\cdot\left(\frac{m(n^2+n+2)}{K}+Kn+3n\right)\right)^2,$$

переходим к уравнению

$$\tau_{op}\cdot\left(\frac{m(n^2+n+2)}{K^2}-n\right)-2(L+\tau_{tr}n)=0,$$

откуда получаем

$$K=\sqrt{\frac{m(n^2+n+2)\tau_{op}}{2(L+\tau_{tr}n)+\tau_{op}n}}.$$

Таким образом, уравнение (27) при $K > 0$ имеет только один корень $K_0 = \sqrt{m(n^2+n+2)\tau_{op} \big/ \left(2(L+\tau_{tr}n)+\tau_{op}n\right)}$. Нас интересует случай, когда $K_0 > 1$. Это имеет место при выполнении условия $m(n^2+n+2)\tau_{op} > 2(L+\tau_{tr}n)+\tau_{op}n$. В соответствии с (28) имеем

$$a'(1) = \frac{\tau_{op}\cdot\left(m(n^2+n+2)-n\right)-2(L+\tau_{tr}n)}{2(L+\tau_{tr}n)+\tau_{op}\cdot\left(m(n^2+n+2)+4n\right)} > 0. \quad (30)$$

Следовательно, в точке K_0 достигается максимум ускорения. Таким образом, верхняя граница масштабируемости K_{max} алгоритма Чиммино в соответствии с моделью BSF определяется следующей формулой:

$$K_{max} = \sqrt{\frac{m(n^2+n+2)\tau_{op}}{2(L+\tau_{tr}n)+\tau_{op}n}}. \quad (31)$$

Упростим формулу (31). При $n, m \to \infty$ имеем

$$m(n^2+n+2)\tau_{op} = O(mn^2) \quad (32)$$

и

$$2(L+\tau_{tr}n)+\tau_{op}n = O(n). \quad (33)$$

Подставив правые части уравнений (32) и (33) в (31), получим

$$K_{max} = \sqrt{\frac{O(mn^2)}{O(n)}},$$

что равносильно

$$K_{max} = \sqrt{O(mn)}.$$

Если предположить, что $m > n$, то есть количество неравенств больше количества переменных, то в этом случае имеем $K_{max} = \sqrt{O(n^2)}$, что равносильно $K_{max} = O(n)$.

$$K_{max} = O(n). \quad (34)$$

Таким образом, при $m > n$ верхняя граница масштабируемости алгоритма Чиммино над списками растет пропорционально размерности задачи n.

В заключение этого раздела напишем формулу для оценки параллельной эффективности e как функции от K. С учетом формулы (26) имеем

$$e(K) = \frac{a(K)}{K} = \frac{2(L+\tau_{tr}n)+\tau_{op}\cdot\left(m(n^2+n+2)+4n\right)}{2(L+\tau_{tr}n)K^2+\tau_{op}\cdot\left(m(n^2+n+2)+K^2n+3Kn\right)}. \quad (35)$$

4. Вычислительные эксперименты

Для верификации результатов, полученных аналитическим путем, мы выполнили реализацию алгоритма Чиммино на языке C++ с использованием программного каркаса BSF и библиотеки параллельного программирования MPI. Данная реализация свободно доступна в сети Интернет по адресу https://github.com/leonid-sokolinsky/BSF-Cimmino. Систему неравенств мы заимствовали из модельной масштабируемой задачи линейного программирования *Model-n* размерности n, приведенной в работе [24]. Количество неравенств в этой системе равно $m = 2n+2$. Мы исследовали ускорение и эффективность распараллеливания алгоритма Чиммино на суперкомпьютере «Торнадо ЮУрГУ» [25]. Тестирование проводилось для размерностей 1500, 5000, 10000 и 16000. Одновременно мы построили для этих же размерностей графики ускорения и эффективности с использованием формул (26) и (35). Для этого экспериментальным путем были определены величины в секундах: $L = 1.5\cdot10^{-5}$, $\tau_{op} = 2.9\cdot10^{-8}$ и

$\tau_{tr} = 1.9 \cdot 10^{-7}$. Результаты приведены на рис. 1-8. Во всех случаях аналитические оценки оказались очень близки к экспериментальным. Кроме этого, проведенные эксперименты показывают, что верхняя граница масштабируемости программы растет пропорционально размерности задачи, что было предсказано аналитически с помощью формулы (34).

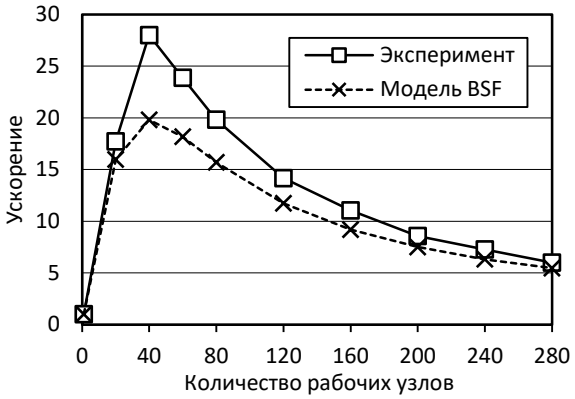

Рис. 1. Ускорение при n = 1500, m = 3002.

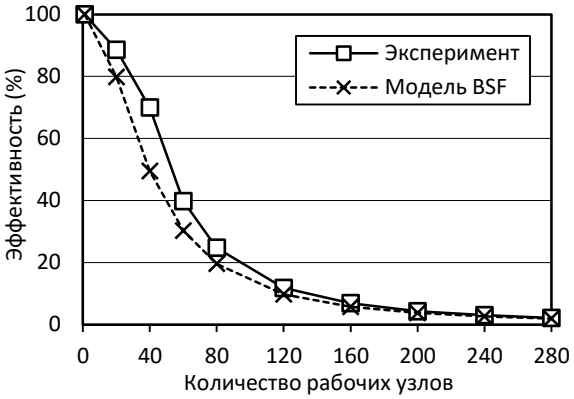

Рис. 2. Эффективность при n = 1500, m = 3002.

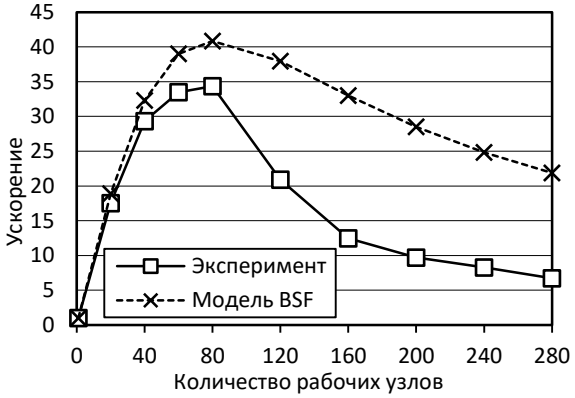

Рис. 3. Ускорение при n = 5000, m = 10002.

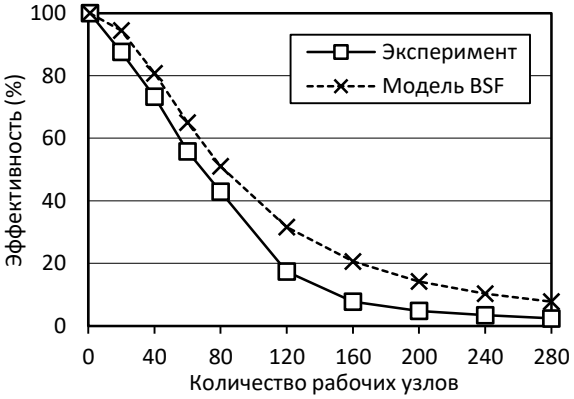

Рис. 4. Эффективность при n = 5000, m = 10002.

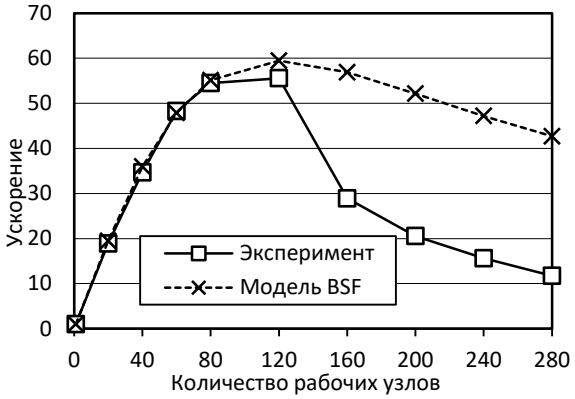

Рис. 5. Ускорение при n = 10000, m = 20002.

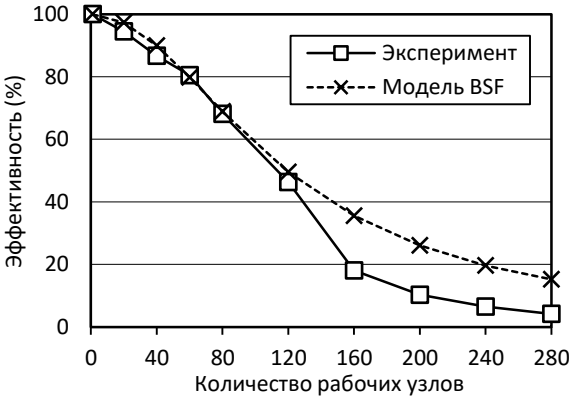

Рис. 6. Эффективность при n = 10000, m = 20002.

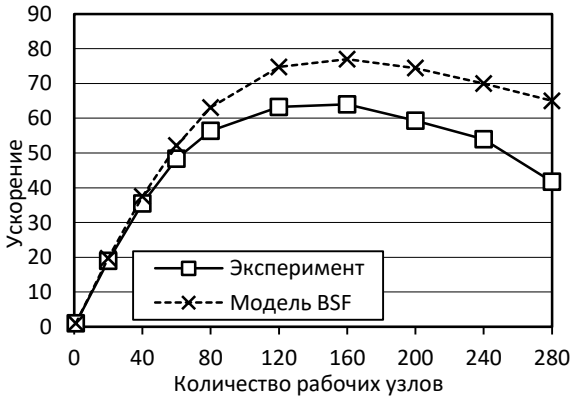

Рис. 7. Ускорение при n = 16000, m = 32002.

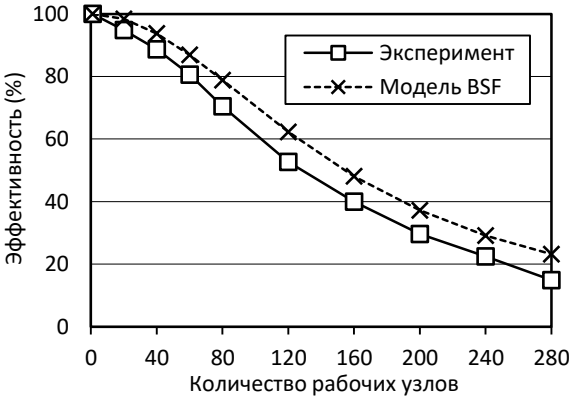

Рис. 8. Эффективность при n = 16000, m = 32002.

Заключение

В статье выполнено аналитическое исследование масштабируемости и эффективности распараллеливания итерационного алгоритма Чиммино для решения больших систем линейных неравенств на многопроцессорных системах с распределенной памятью. Для этого использована модель параллельных вычислений BSF (Bulk Synchronous Farm), основанная на парадигме «мастер-рабочие». Описана реализация алгоритма Чиммино в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*, определяемых формализмом Бёрда-Миртенса. Получена оценка верхней границы масштабируемости алгоритма Чиммино над списками. Данная оценка показывает, что если количество неравенств больше количества переменных, то верхняя граница масштабируемости растет пропорционально размерности задачи. Также получены формулы для оценки ускорения и эффективности распараллеливания алгоритма Чиммино над списками. Выполнена реализация алгоритма Чиммино над списками на языке C++ с использованием программного каркаса BSF и библиотеки параллельного программирования MPI. Данная реализация свободно доступна в сети Интернет по адресу https://github.com/leonid-sokolinsky/BSF-Cimmino. На кластерной вычислительной системе проведены масштабные эксперименты для определения фактических кривых ускорения и параллельной эффективности для задач с количеством переменных 1500, 5000, 10000, 16000 и количеством неравенств 3002, 10002, 20002, 32002 соответственно. Результаты экспериментов показали, что модель BSF с высокой точностью предсказывает верхнюю границу масштабируемости алгоритма Чиммино над списками. В рамках дальнейших исследований мы планируем решить следующие задачи:

1) применить алгоритм Чиммино для реализации фазы *Qwest* алгоритма NSLP [2], предназначенного для решения сверхбольших нестационарных задач линейного программирования;
2) провести вычислительные эксперименты по решению сверхбольших задач линейного программирования на кластерной вычислительной системе в условиях динамического изменения исходных данных.

Литература

1. Cottle R.W., Pang J.-S., Stone R.E. The Linear Complementarity Problem. Society for Industrial and Applied Mathematics, 2009. xxiv + 757 p. DOI: 10.1137/1.9780898719000
2. Соколинская И.М., Соколинский Л.Б. О решении задачи линейного программирования в эпоху больших данных // Параллельные вычислительные технологии — XI международная конференция, ПаВТ'2017, г. Казань, 3–7 апреля 2017 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2017. С. 471-484. URL: http://omega.sp.susu.ru/pavt2017/short/014.pdf.
3. Censor Y. et al. On Diagonally Relaxed Orthogonal Projection Methods // SIAM Journal on Scientific Computing. Society for Industrial and Applied Mathematics, 2008. Vol. 30, No. 1. P. 473–504. DOI: 10.1137/050639399
4. Zhu J., Li X. The Block Diagonally-Relaxed Orthogonal Projection Algorithm for Compressed Sensing Based Tomography // 2011 Symposium on Photonics and Optoelectronics (SOPO). IEEE, 2011. P. 1–4. DOI: 10.1109/SOPO.2011.5780660
5. Censor Y. Mathematical optimization for the inverse problem of intensity-modulated radiation therapy // Intensity-Modulated Radiation Therapy: The State of the Art / ed. Palta J.R., Mackie T.R. Madison, WI: Medical Physics Publishing, 2003. P. 25–49. DOI: 10.1118/1.1628279
6. Agmon S. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 382–392. DOI: 10.4153/CJM-1954-037-2
7. Motzkin T.S., Schoenberg I.J. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 393–404. DOI: 10.4153/CJM-1954-038-x
8. Cimmino G. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari // La Ricerca Scientifica, XVI, Series II, Anno IX, 1. 1938. P. 326–333.
9. Benzi M. Gianfranco Cimmino's Contributions to Numerical Mathematics // Atti del SemiSeminario di Analisi Matematica, Dipartimento di Matematica dell'Universit`a di Bologna (Ciclo di Conferenze in Ricordo di Gianfranco Cimmino, 2004). Bologna, Italy: Tecnoprint, 2005. P. 87–109.

10. Censor Y., Zenios S.A. Parallel Optimization: Theory, Algorithms, and Applications. New York: Oxford University Press, 1997.
11. Censor Y., Elfving T. New methods for linear inequalities // Linear Algebra and its Applications. North-Holland, 1982. Vol. 42. P. 199–211. DOI: 10.1016/0024-3795(82)90149-5
12. Censor Y. Sequential and parallel projection algorithms for feasibility and optimization // Proc. SPIE 4553, Visualization and Optimization Techniques, (25 September 2001) / ed. Censor Y., Ding M. International Society for Optics and Photonics, 2001. P. 1–9. DOI: 10.1117/12.441550
13. Kelley C.T. Iterative Methods for Linear and Nonlinear Equations. Philadelphia: Society for Industrial and Applied Mathematics, 1995. xiii + 156 p. DOI: 10.1137/1.9781611970944
14. Bilardi G., Pietracaprina A. Models of Computation, Theoretical // Encyclopedia of Parallel Computing. Boston, MA: Springer US, 2011. P. 1150–1158. DOI: 10.1007/978-0-387-09766-4_218
15. JaJa J.F. PRAM (Parallel Random Access Machines) // Encyclopedia of Parallel Computing. Boston, MA: Springer US, 2011. P. 1608–1615. DOI: 10.1007/978-0-387-09766-4_23
16. Valiant L.G. A bridging model for parallel computation // Communications of the ACM. 1990. Vol. 33, No. 8. P. 103–111. DOI: 10.1145/79173.79181
17. Culler D. et al. LogP: towards a realistic model of parallel computation // Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming - PPOPP '93. New York, New York, USA: ACM Press, 1993. P. 1–12. DOI: 10.1145/155332.155333
18. Forsell M., Leppänen V. An extended PRAM-NUMA model of computation for TCF programming // Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012. Washington, DC, USA: IEEE Computer Society, 2012. P. 786–793. DOI: 10.1109/IPDPSW.2012.97
19. Gerbessiotis A. V. Extending the BSP model for multi-core and out-of-core computing: MBSP // Parallel Computing. Elsevier B.V., 2015. Vol. 41. P. 90–102. DOI: 10.1016/j.parco.2014.12.002
20. Lu F., Song J., Pang Y. HLognGP: A parallel computation model for GPU clusters // Concurrency and Computation: Practice and Experience. 2015. Vol. 27, No. 17. P. 4880–4896. DOI: 10.1002/cpe.3475
21. Ежова Н.А., Соколинский Л.Б. Модель параллельных вычислений для многопроцессорных систем с распределенной памятью // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2018. Т. 7, № 2. С. 32-49. DOI: 10.14529/cmse180203.
22. Sokolinskaya I., Sokolinsky L.B. Scalability evaluation of the NSLP algorithm for solving non-stationary linear programming problems on cluster computing systems // Суперкомпьютерные дни в России: Труды международной конференции (25-26 сентября 2017 г., г. Москва). М.: Изд-во МГУ, 2017. С. 319-332.
URL: http://russianscdays.org/files/pdf17/319.pdf.
23. Cole M.I. Parallel programming with list homomorphisms // Parallel Processing Letters. 1995. Vol. 5, No. 2. P. 191–203. DOI: 10.1142/S0129626495000175
24. Соколинская И.М., Соколинский Л.Б. Модифицированный следящий алгоритм для решения нестационарных задач линейного программирования на кластерных вычислительных системах с многоядерными ускорителями // Суперкомпьютерные дни в России: труды международной конференции (26-27 сентября 2016 г., г. Москва). М.: Изд-во МГУ, 2016. С. 294-306. URL: http://2016.russianscdays.org/files/pdf16/294.pdf.
25. Kostenetskiy P.S., Safonov A.Y. SUSU Supercomputer Resources // Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies (PCT 2016). CEUR Workshop Proceedings. Vol. 1576. 2016. P. 561–573.