УДК 004.051

МОДЕЛЬ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ BSF-MR*

Н.А. Ежова (ezhovana@susu.ru),
Л.Б. Соколинский (leonid.sokolinsky@susu.ru)
Южно-Уральский государственный университет

В статье дается описание модели параллельных вычислений BSF-MR (Bulk Synchronous Farm with Map and Reduce), ориентированной на итерационные алгоритмы с высокой вычислительной сложностью, разрабатываемые для многопроцессорных систем с распределенной памятью экзафлопсного уровня производительности.

Ключевые слова: итерационный алгоритм, модель параллельных вычислений, BSF-MR, оценка масштабируемости, ускорение, верификация, гравитационная задача, кластерные вычислительные системы.

BSF-MR PARALLEL COMPUTATION MODEL*

Nadezhda A. Ezhova (ezhovana@susu.ru),
Leonid B. Sokolinsky (leonid.sokolinsky@susu.ru)
South Ural State University

The paper describes the parallel computation model BSF-MR (Bulk Synchronous Farm with Map and Reduce), oriented to compute-intensive iterative algorithms and designed for multiprocessor systems with distributed exaflops memory.

Keywords: iterative algorithm, parallel computation model, BSF MR, scalability estimation, speedup, verification, gravitational problem, cluster computing systems.

1. Введение

При создании параллельных алгоритмов для больших многопроцессорных систем важно уже на ранней стадии разработки алгоритма оценить его масштабируемость. Для этой цели используются модели параллельных вычислений [1]. Среди большого количества существующих моделей наиболее известными являются модели PRAM [2], BSP [3] и LogP [4]. Указанные модели подверглись обобщениям и уточнениям, породив целые семейства, насчитывающие десятки параллельных вычислительных моделей (см., например, [5–7]). Задача разработки новых моделей актуальна 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 работах [8,9] была

* 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 науки и высшего образования РФ (государственное задание 2.7905.2017/8.9).

предложена модель параллельных вычислений BSF (Bulk Synchronous Farm), ориентированная на кластерные вычислительные системы и алгоритмы итерационного типа с высокой вычислительной сложностью. Модель BSF является развитием модели BSP и основана на методе программирования SPMD [10,11] и парадигме «мастер-рабочие» [12]. Модель BSF позволяет с высокой точностью оценить верхнюю границу масштабируемости параллельного итерационного алгоритма до написания программы. Примеры использования модели BSF для исследования параллельных алгоритмов можно найти в работах [13,14].

Целью настоящей статьи является верификация расширения BSF-MR модели BSF, предложенного в работе [14]. Расширение BSF-MR предполагает представление итерационного алгоритма в виде операций над списками с использованием функций высшего порядка Map и Reduce. Верификация модели BSF-MR осуществляется с помощью гравитационной задачи, моделирующей движение тела малой массы среди n неподвижных тел большой массы. Указанная задача является упрощением задачи n тел [15,16]. Верификация осуществляется следующим образом: в соответствии с требованиями модели BSF-MR строится алгоритм решения гравитационной задачи Gravitation-MR; с помощью стоимостных метрик BSF-MR получаются оценки для ускорения и верхней границы масштабируемости алгоритма Gravitation-MR; выполняется его реализация на языке C++ с использованием MPI; проводятся эксперименты на кластере; сравниваются графики ускорения, полученные аналитически и экспериментально.

Статья организована следующим образом. В разделе 2 дается описание модели BSF-MR, являющейся расширением модели параллельных вычислений BSF. В разделе 3 приводятся стоимостные метрики модели BSF-MR и выводятся аналитические оценки для ускорения и верхней границы масштабируемости параллельного алгоритма. В разделе 4 дается формальное описание алгоритма решения задачи n тел. В разделе 5 с помощью стоимостных метрик модели BSF-MR выводятся аналитические оценки для ускорения и верхней границы масштабируемости алгоритма Gravitation-MR. В разделе 6 дается информация о реализации алгоритма Gravitation-MR, выполненного на языке C++ с использованием программного каркаса BSF-MR и библиотеки MPI. Производится сравнение результатов, полученные аналитически и экспериментально. В заключении суммируются результаты и намечаются направления дальнейших исследований.

2. Модель параллельных вычислений BSF-MR

Идея модели параллельных вычислений BSF (Bulk Synchronous Farm) впервые была предложена в работе [9]. В данном разделе мы опишем усовершенствованный вариант этой модели, получивший название BSF-MR (BSF with Map and Reduce). Абстрактный компьютер согласно модели BSF-MR представляет собой множество однородных процессорных узлов с приватной памятью, соединенных сетью, позволяющей передавать данные

от одного процессорного узла другому. Среди процессорных узлов выделяется один узел-мастер. Остальные K узлов называются узлами-рабочими. Модель BSF-MR предполагает представление итерационного алгоритма в виде операций над списками с использованием функций высшего порядка Map и Reduce, определяемых формализмом Бёрда-Миртенса (Bird–Meertens formalism) [17]. Для заданной функции $F:\mathfrak{A}\to\mathfrak{B}$ и списка $[a_1,\ldots,a_\ell]$ функция высшего порядка Map формирует новый список той же длины путем применения функции F ко всем элементам списка:

$$\mathrm{Map}(F,[a_1,\ldots,a_\ell])=[F(a_1),\ldots,F(a_\ell)]. \qquad (1)$$

Для бинарной ассоциативной операции $\oplus:\mathfrak{B}\times\mathfrak{B}\to\mathfrak{B}$ и списка $[b_1,\ldots,b_\ell]$ функция высшего порядка Reduce сводит список $[b_1,\ldots,b_\ell]$ к одному элементу путем многократного применения операции $\oplus$ к элементам списка:

$$\mathrm{Reduce}(\oplus,[b_1,\ldots,b_\ell])=b_1\oplus\ldots\oplus b_\ell. \qquad (2)$$

Таблица 1. Алгоритм 1: шаблон итерационного алгоритма в BSF-MR.

1	**input** A, x_0
2	$i = 0$
3	$B = \mathrm{Map}(F_{x_i}, A)$
4	$s = \mathrm{Reduce}(\oplus, B)$
5	$x_{i+1} = \mathrm{Compute}(x_i, s)$
6	$i = i + 1$
7	**if** $\mathrm{StopCond}(x_i, x_{i-1})$ **goto** 9
8	**goto** 3
9	**output** x_i
10	**stop**

Общий шаблон итерационного алгоритма в модели BSF-MR имеет структуру, приведенную в таблице 1. Переменная i обозначает номер итерации; x_0 – начальное приближение, x_i – i-ое приближение (в качестве приближения может фигурировать число, вектор или другая структура данных); A – список элементов множества $\mathfrak{A}$, исходные данные задачи; $F_x:\mathfrak{A}\to\mathfrak{B}$ – параметризованная функция, отображающая множество $\mathfrak{A}$ в некоторое множество $\mathfrak{B}$; B – список элементов множества $\mathfrak{B}$, получающийся путем применения функции F_x к каждому элементу списка A; $\oplus$ – бинарная ассоциативная операция над множеством $\mathfrak{B}$. На шаге 1 вводятся исходные данные задачи (список A) и начальное приближение x_0. На шаге 2 номер итерации устанавливается в значение 0. На шаге 3 вычисляется список B как результат выполнения функции высшего порядка $\mathrm{Map}(F_{x_i}, A)$. На шаге 4 получается промежуточная переменная $s\in\mathfrak{B}$ в результате выполнения функции высшего порядка $\mathrm{Reduce}(\oplus, B)$. На шаге 5 выполняется пользовательская функция Compute, вычисляющая следую-

щее приближение x_{i+1}. На шаге 6 номер итерации i увеличивается на единицу. На шаге 7 выполняется пользовательская функция StopCond с аргументами x_i и x_{i-1}, проверяющая условие завершения итерационного алгоритма. Если эта функция возвращает значение «истина», то происходит переход на шаг 9. На шаге 8 осуществляется переход на шаг 3 для выполнения очередной итерации. На шаге 9 выводится последнее полученное приближение. Шаг 10 останавливает работу алгоритма.

Таблица 2. Алгоритм 2: шаблон параллельной реализации итерационного алгоритма в модели BSF-MR.

	Мастер		j-ый рабочий (j=1,…,K)
1	**input** A, x_0	1	**input** A, x_0
2	$i = 0$	2	
3	SendToAllWorkers(x_i)	3	RecvFromMaster(x_i)
4		4	$B^{[j]} = \text{Map}(F_{x_i}, A^{[j]})$
5		5	$s^{(j)} = \text{Reduce}(\oplus, B^{[j]})$
6	**for** $j = 1$ **to** K **do**	6	
7	RecvFromWorker$(j, s^{(j)})$	7	SendToMaster$(s^{(j)})$
8	**end for**	8	
9	$s = \text{Reduce}(\oplus, [s^{(1)}, \ldots, s^{(K)}])$	9	
10	$x_{i+1} = \text{Compute}(x_i, s)$	10	
11	$i = i + 1$	11	
12	**if** StopCond(x_i, x_{i-1}) **goto** 14	12	
13	**goto** 3	13	**goto** 3
14	**output** x_i	14	
15	**stop**	15	

Общий шаблон параллельной реализации (табл. 2) итерационного алгоритма модели BSF-MR включает в себя $K+1$ потоков управления. Поток управления с номером 0 выполняется на мастере. Потоки $1,\ldots,K$ выполняются на рабочих, имеют идентичный код, однако обрабатывают различные части $A^{[1]},\ldots,A^{[K]}$ списка A, формируя различные части $B^{[1]},\ldots,B^{[K]}$ списка B. На шаге 3 мастер посылает, а рабочие получают текущее приближение x_i. После этого j-тый рабочий выполняет следующие действия в параллельном режиме: на шаге 4 вычисляет подсписок $B^{[j]}$, применяя функцию F_{x_i} к каждому элементу подсписка $A^{[j]}$; на шаге 5 считает частичную «сумму» $s^{(j)}$, применяя операцию $\oplus$ к элементам подсписка $B^{[j]}$; на шаге 7 посылает вычисленное значение $s^{(j)}$ мастеру. Мастер в цикле 6-8 получает значения $s^{(j)}$ от всех рабочих и выполняет оставшиеся шаги алгоритма.

3. Стоимостная метрика модели BSF-MR

Модель BSF-MR включает в себя следующие основные стоимостные параметры в рамках одной итерации:

K – количество узлов-рабочих;

L – латентность (время пересылки сообщения длиной в 1 байт).

t_s – время передачи задания от мастера рабочему (без учета L);

t_{Map} – время выполнения рабочим функции Map для всего списка A;

t_p – время, затрачиваемое мастером на обработку полученных результатов и проверку условия завершения;

t_r – время передачи мастеру результата от рабочего (без учета L);

t_a – время, необходимое для выполнения одной операции;

ℓ – длина списка исходных данных A (совпадает с длиной списка B);

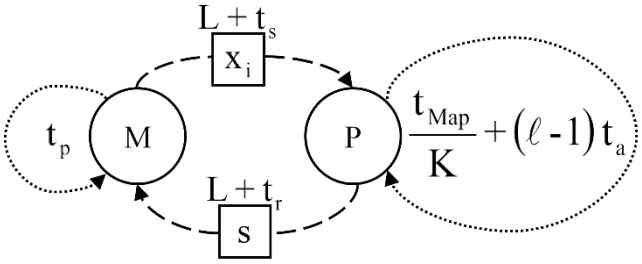


Рис. 1. Схема работы алгоритма 2 (табл. 2) для мастера (M) и рабочего (P).

Обозначим через T_1 время выполнения одной итерации алгоритма 2 (табл. 2) системой из одного мастера и одного рабочего (см. рис. 1). Используя введенные стоимостные параметры можно получить следующую оценку T_1 (при длине списка $\ell \to \infty$):

$$T_1 = 2L + t_s + t_r + t_p + t_{Map} + \ell t_a. \qquad (3)$$

Обозначим через T_K время выполнения одной итерации алгоритма 2 (табл. 2) мастером и K рабочими (рис. 2). При $\ell \to \infty$ получаем оценку T_K:

$$T_K = K(2L + t_s + t_r + t_a) + \\ + \frac{t_{Map} + \ell t_a}{K} - t_a + t_p. \qquad (4)$$

Ускорение в модели BSF-MR вычисляется по формуле:

$$a(K) = \frac{T_1}{T_K} = \\ = (2L + t_s + t_r + t_p + t_{Map} + \ell t_a) \times \\ \times \frac{\left(-2L - t_s - t_r - t_a + \frac{t_{Map} + \ell t_a}{K^2}\right)}{\left(K(2L + t_s + t_r + t_a) + \frac{t_{Map} + \ell t_a}{K} - t_a + t_p\right)^2}. \qquad (5)$$

Очевидно, что при $K \geq 1$ формула (5) дает только положительные значения ускорения. Для нахождения точек экстремума кривой ускорения (5) найдем производную ускорения по K:

$$a'(K) = \tag{6}$$
$$= (2L + t_s + t_r + t_p + t_{Map} + \ell t_a) \times$$
$$\times \frac{\left(-2L - t_s - t_r - t_a + \frac{t_{Map} + \ell t_a}{K^2}\right)}{\left(K(2L + t_s + t_r + t_a) + \frac{t_{Map} + \ell t_a}{K} - t_a + t_p\right)^2}$$

Уравнение

$$(2L + t_s + t_r + t_p + t_{Map} + \ell t_a) \times$$
$$\times \frac{\left(-2L - t_s - t_r - t_a + \frac{t_{Map} + \ell t_a}{K^2}\right)}{\left(K(2L + t_s + t_r + t_a) + \frac{t_{Map} + \ell t_a}{K} - t_a + t_p\right)^2} =$$
$$= 0. \tag{7}$$

имеет только один корень на интервале

$$K_0 = \sqrt{\frac{t_{Map} + \ell t_a}{2L + t_s + t_r + t_a}}. \tag{8}$$

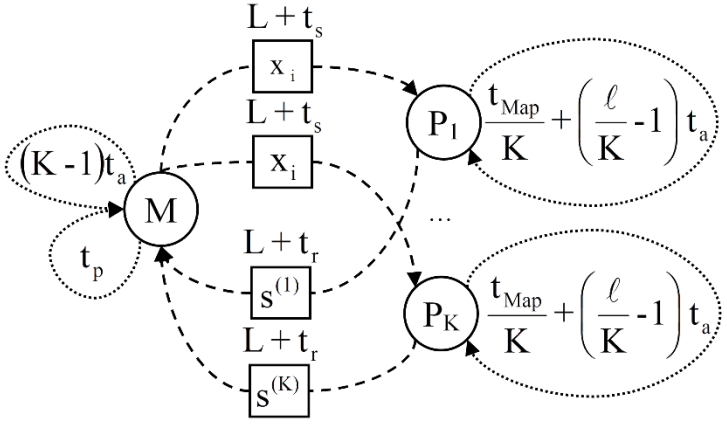


Рис. 2. Схема работы алгоритма 2 (табл. 2) для конфигурации из одного мастера (M) и K рабочих (P_1 … P_K).

На интервале $[1, K_0)$ производная $a'(K)$, вычисляемая по формуле (6), принимает только положительные значения, а на интервале $(K_0, +\infty)$ – только отрицательные. Следовательно, ускорение $a(K)$ достигает максимума в точке K_0. Таким образом, верхняя граница масштабируемости алгоритма 2 (табл. 2) может быть оценена по следующей формуле:

$$K_{MAX} = \sqrt{\frac{t_{Map} + \ell t_a}{2L + t_s + t_r + t_a}}. \qquad (9)$$

4. Алгоритм Gravitation-MR

Выполним верификацию модели BSF-MR с помощью гравитационной задачи, моделирующей движение тела малой массы среди n неподвижных тел большой массы. Указанная задача является упрощенным вариантом известной задачи n тел [15,16].

Пусть в $\Re^3$ задано конечное множество точек $\mathfrak{Y}$, представляющих собой неподвижные тела большой массы. Обозначим эти тела следующим образом: $\mathfrak{Y} = \{Y_1,...,Y_n\} \subset \Re^3$, а их массы – $\{m_1,...,m_n\}$, где $n = |\mathfrak{Y}|$. Пусть также задано некоторое движущееся тело x малой массы m_x. Предполагается, что на x не действуют никакие силы, кроме гравитационных. Для тела x задано его начальное положение $X^{(t_0)} \in \Re^3$ и вектор скорости $V^{(t_0)} \in \Re^3$ в начальный момент времени t_0. Необходимо рассчитать траекторию движения тела x среди тел $\mathfrak{Y}$. Для этого будем последовательно считать новые положения тела x через равные промежутки времени Δt:

$$X^{(t_0)}, X^{(t_0+\Delta t)}, X^{(t_0+2\Delta t)}, X^{(t_0+3\Delta t)}, \ldots \qquad (10)$$

По закону всемирного тяготения сила притяжения F_i тела x к телу Y_i $(i = 1,\ldots,n)$ вычисляется по формуле

$$F_i = G\frac{m_i\, m_x}{\left\|Y_i - X\right\|^3}(Y_i - X), \qquad (11)$$

где $X \in \Re^3$ задает текущие координаты точки x. По второму закону Ньютона ускорение α_i тела x под действием силы F_i вычисляется по формуле

$$\alpha_i = \frac{F_i}{m_x}. \qquad (12)$$

Ускорение под действием всех сил $F_1,\ldots,F_n$ вычисляется по формуле

$$\alpha = \sum_{i=1}^{n} \alpha_i. \qquad (13)$$

В соответствии с этим вектора скорости для последовательности (10) могут быть посчитаны по следующей итерационной формуле

$$V^{(t+\Delta t)} = V^{(t)} + \alpha^{(t+\Delta t)}\,\Delta t, \qquad (14)$$

где

$$\alpha^{(t+\Delta t)} = \sum_{i=1}^{n} G\frac{m_i}{\left\|Y_i - X^{(t)}\right\|^3}\left(Y_i - X^{(t)}\right). \qquad (15)$$

Используя (14) получаем

$$X^{(t+\Delta t)} = X^{(t)} + V^{(t+\Delta t)}\Delta t. \qquad (16)$$

В контексте гравитационного алгоритма определим список A:

$$A = [(Y_1, m_1), \ldots, (Y_n, m_n)], \qquad (17)$$

то есть A – список пар вида (Y_i, m_i), задающих координаты и массу i-того неподвижного тела большой массы. Для произвольной точки $X \in \Re^3$ определим функцию $f_X : \Re^3 \times \Re^3 \to \Re^3$:

$$f_X(Y_i, m_i) = G\frac{m_i}{(Y_i - X)^3}(Y_i - X). \qquad (18)$$

для всех $i \in \{1, \ldots, n\}$. Иначе говоря, функция $f_X(Y_i, m_i)$ с использованием формулы (11) вычисляет значение F_i / m_x, где F_i – гравитационная сила, с которой тело Y_i действует на материальную точку с координатами X и массой m_x. Для $X \in \Re^3$ определим список $B \subset \Re^3$ следующим образом:

$$B^{(X)} = [f_X(Y_1, m_1), \ldots, f_X(Y_n, m_n)], \qquad (19)$$

то есть список B получается из списка A применением к нему функции высшего порядка Map, использующей в качестве параметра функцию f_X: $B = \mathrm{Map}(f_X, A)$. Определим операцию $\oplus : \Re^3 \times \Re^3 \to \Re^3$:

$$Z \oplus Y = Z + Y \qquad (20)$$

для любых $Z, Y \in \Re^3$. В данном случае операция $\oplus$ выполняет сложение векторов в трехмерном пространстве. Тогда ускорение α материальной точки с координатами X под воздействием тел $\mathfrak{Y}$, может быть вычислено путем применения к списку B функции Reduce, использующей в качестве параметра операцию $\oplus$:

$$\alpha = \mathrm{Reduce}(\oplus, B). \qquad (21)$$

На основе шаблона, представленного в алгоритме 1 (табл. 1), запишем гравитационный алгоритм в виде операций над списками (см. таблицу 3).

Таблица 3. Алгоритм 3: гравитационный алгоритм Gravitation-MR.

1	**input** $A, X^{(t_0)}, V^{(t_0)}, \Delta t, t_0, T$
2	$t = t_0$
3	$B = \mathrm{Map}(f_{X^{(t)}}, A)$
4	$\alpha = \mathrm{Reduce}(\oplus, B)$.
5	$V^{t+\Delta t} = V^{(t)} + \alpha \cdot \Delta t;\ X^{(t+\Delta t)} = X^{(t)} + V^{(t+\Delta t)} \cdot \Delta t$
6	$t = t + \Delta t$
7	**if** $t \geq T$ **goto** 10
8	**goto** 3
9	**output** $X^{(t)}$
10	**stop**

На основе алгоритма 2 (табл. 2) получаем алгоритм 4 (табл. 4), представляющий собой параллельную реализацию гравитационного алгоритма.

Таблица 4. Параллельная реализация гравитационного алгоритма.

	Мастер		j-ый рабочий (j=1,…,K)
1	**input** $A, X^{(t_0)}, V^{(t_0)}, \Delta t, t_0, T$	1	**input** A
2	$t = t_0$	2	
3	$\text{SendToAllWorkers}\left(X^{(t)}\right)$	3	$\text{RecvFromMaster}\left(X^{(t)}\right)$
4		4	$B^{[j]} = \text{Map}\left(f_{X^{(t)}}, A^{[j]}\right)$
5		5	$\alpha^{[j]} = \text{Reduce}\left(\oplus, B^{[j]}\right)$
6	**for** $j = 1$ **to** K **do**	6	
7	$\text{RecvFromWorker}\left(j, \alpha^{[j]}\right)$	7	$\text{SendToMaster}\left(\alpha^{[j]}\right)$
8	**end for**	8	
9	$\alpha = \text{Reduce}\left(\oplus, \left[\alpha^{[1]}, \ldots, \alpha^{[K]}\right]\right)$	9	
10	$V^{(t+\Delta t)} := V^{(t)} + \alpha \cdot \Delta t;$ $X^{(t+\Delta t)} := X^{(t)} + V^{(t+\Delta t)}\,\Delta t$	10	
11	$t = t + \Delta t$	11	
12	**if** $t \geq T$ **goto** 14	12	
13	**goto** 3	13	**goto** 3
14	**output** $X^{(t)}$	14	
15	**stop**	15	

5. Аналитическое исследование алгоритма Gravitation-MR

Для простоты мы будем везде далее предполагать, что количество тел большой массы, то есть

$$n = qK \qquad (22)$$

при некотором $q \in \mathbb{N}$. В рамках одной итерации ведем следующие обозначения для анализа масштабируемости алгоритма Gravitation-MR:

c_s – количество вещественных чисел, передаваемых мастером рабочему;

c_{Map} – количество арифметических операций, выполняемых на шаге Map при обработке всего списка A (шаг 3 алгоритма 3, см. табл. 3);

c_a – количество арифметических операций, необходимое для сложения двух векторов в трехмерном пространстве;

c_r – количество чисел, передаваемых от рабочего мастеру;

c_p – количество арифметических операций, выполняемых мастером на шагах 10 – 12 алгоритма 4 (табл. 4).

Вычислим указанные значения. В начале очередной итерации мастер на шаге передает каждому рабочему текущие координаты $X^{(t)}$ тела x, что в совокупности составляет 3 вещественных числа. Следовательно,

$$c_s = 3. \qquad (23)$$

На шаге Map функция f_X, задаваемая формулой (18), применяется ко всем элементам списка A, имеющего длину n. Предположим, что квадратный

корень вычисляется с помощью первых четырех членов ряда Тейлора, что составляет 11 арифметических операций. Тогда однократное вычисление функции f_X потребует 20 арифметических операций. Следовательно, общее количество операции для обработки всего списка A составит

$$c_{Map} = 20n. \qquad (24)$$

Для сложения двух векторов размерности 3 требуется 3 операции:

$$c_a = 3. \qquad (25)$$

Трехмерный вектор, полученный рабочим на шаге Reduce, пересылается мастеру. Значит:

$$c_r = 3. \qquad (26)$$

Выполнение шага 10 алгоритма 4 (табл. 4) требует 12 арифметических операций, шага 11 – одну арифметическую операцию и шага 12 – одну операцию сравнения. Отсюда получаем следующую формулу:

$$c_p = 14. \qquad (27)$$

Пусть τ_{op} обозначает время, затрачиваемое рабочим на выполнение одной арифметической операции, а τ_{tr} – время, затрачиваемое на пересылку одного числа без учета латентности. Тогда получаем следующие значения для стоимостных параметров алгоритма Gravitation-MR:

$$t_s = c_s\,\tau_{tr} = 3\tau_{tr}; \qquad (28)$$

$$t_{Map} = c_{Map}\,\tau_{op} = 20n\tau_{op}; \qquad (29)$$

$$t_r = c_r\tau_{tr} = 3\tau_{tr}; \qquad (30)$$

$$t_a = c_a\,\tau_{op} = 3\tau_{op}; \qquad (31)$$

$$t_p = c_p\,\tau_{op} = 14\tau_{op}. \qquad (32)$$

Подставляя в формулу (5) значения правых частей из формул (28) – (32) получаем формулу для оценки ускорения алгоритма Gravitation-MR:

$$a_{Gravitation\text{-}MR}(K) = \frac{2L + 6\tau_{tr} + 14\tau_{op} + 20n\tau_{op} + 3\ell\tau_{op}}{K\left(2L + 6\tau_{tr} + 14\tau_{op}\right) + \frac{20n\tau_{op} + 3\ell\tau_{op}}{K} + 11\tau_{op}} \qquad (33)$$

Верхняя граница масштабируемости алгоритма Gravitation-MR получается подстановкой значений правых частей из формул (28) – (32) в формулу (9):

$$K_{Gravitation\text{-}MR} = \sqrt{\frac{20n\tau_{op} + 3\ell\tau_{op}}{2L + 6\tau_{tr} + 3\tau_{op}}}. \qquad (34)$$

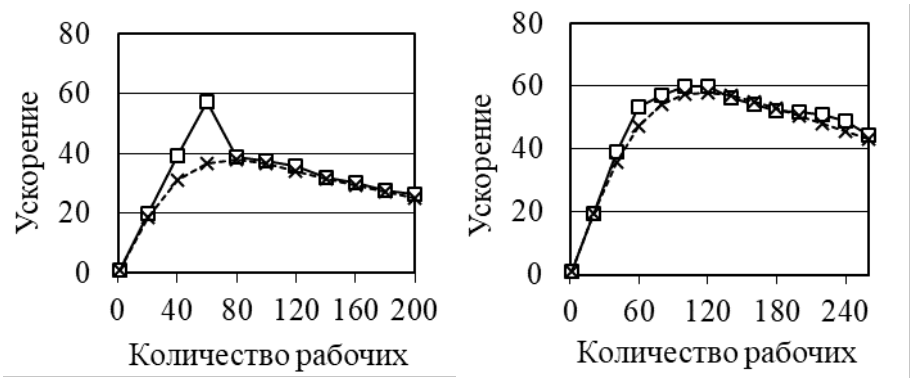
При $n \to \infty$ имеем

$$K_{Gravitation\text{-}MR} = O\left(\sqrt{n}\right). \qquad (35)$$

Таким образом, верхняя граница масштабируемости алгоритма Gravitation-MR над списками растет пропорционально корню квадратному из числа, задающего количество неподвижных тел большой массы.

6. Вычислительные эксперименты

Для верификации результатов, полученных аналитическим путем, была выполнена реализация алгоритма Gravitation-MR 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/nadezhda-ezhova/Gravitation-MR. С помощью этой реализации исследованы ускорение алгоритма Gravitation-MR на суперкомпьютере «Торнадо ЮУрГУ» [18]. Тестирование проводилось для 450, 600, 900 и 1200 тел. Выполнялось 10 итераций для каждого случая. Для верификации модели BSF-MR были также построены графики ускорения алгоритма Gravitation-MR с использованием формулы (33). Для этого экспериментально были определены следующие величины в секундах: $L = 1.5 \cdot 10^{-5}$, $\tau_{op} = 2.9 \cdot 10^{-8}$ и $\tau_{tr} = 1.9 \cdot 10^{-7}$. Во всех случаях аналитические оценки оказались близки к экспериментальным (рис. 3). Кроме того, верхние границы масштабируемости алгоритма Gravitation-MR, полученные аналитически, оказались близки к границам масштабируемости, полученным в результате вычислительных экспериментов. Это подтверждает адекватность модели параллельных вычислений BSF-MR. Отметим, что ранее в работах [14,19] была подтверждена адекватность модели BSF-MR для итерационных алгоритмов Чиммино для СЛН и Якоби для СЛАУ.



а) n = 450. б) n = 600.

в) n = 900. г) n = 1200.

Рис. 3. Эксперименты для разного количества больших тел (□ – модель BSF, × – экспериментальные данные).

7. Заключение

В статье представлена новая модель параллельных вычислений BSF-MR и выполнена ее верификация на основе алгоритма Gravitation-MR для решения гравитационной задачи, моделирующей движение тела малой массы среди n неподвижных тел большой массы. Представлено описание модели BSF-MR, основанное на представлении итерационного численного алгоритма в виде операций над списками с использованием функций высшего порядка Map и Reduce. Построены последовательный и параллельный алгоритмические шаблоны для представления итерационных алгоритмов в контексте модели BSF-MR. Разработана стоимостная метрика модели BSF-MR, позволяющая получать оценки для ускорения и верхней границы масштабируемости алгоритма. Описывается алгоритм Gravitation-MR для решения гравитационной задачи, построенный на основе алгоритмических шаблонов модели BSF-MR. С использованием стоимостной метрики модели BSF-MR получена формула для оценки ускорения алгоритма Gravitation-MR. Также получена оценка верхней границы масштабируемости алгоритма Gravitation-MR, которая показывает, что при количестве рабочих узлов больше $O(\sqrt{n})$ (n – количество неподвижных тел большой массы) кривая ускорения переходит в убывающий тренд. Алгоритм Gravitation-MR был реализован на языке C++ с использованием библиотеки параллельного программирования MPI. Вычислительные эксперименты подтвердили высокую адекватность модели BSF-MR. В рамках дальнейших исследований авторы предполагают разработать инструментальную систему «BSF-Studio» для быстрого создания параллельных программ на основе алгоритмического шаблона модели BSF-MR в рамках фреймворка «мастер-рабочие». Указанная система будет инкапсулировать детали, связанные с параллельным программированием на распределенной памяти.

Литература

1. 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.
2. 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.
3. Valiant L.G. A bridging model for parallel computation // Commun. ACM. 1990. Vol. 33, no. 8. P. 103–111. DOI:10.1145/79173.79181.
4. 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.
5. Forsell M., Leppanen 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.
6. Gerbessiotis A. V. Extending the BSP model for multi-core and out-of-core computing: MBSP // Parallel Comput. Elsevier B.V., 2015. Vol. 41. P. 90–102. DOI:10.1016/j.parco.2014.12.002.
7. Lu F., Song J., Pang Y. HLognGP: A parallel computation model for GPU clusters // Concurr. Comput. Pract. Exp. 2015. Vol. 27, no. 17. P. 4880–4896. DOI:10.1002/cpe.3475.
8. 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.
9. Sokolinsky L.B. Analytical Estimation of the Scalability of Iterative Numerical Algorithms on Distributed Memory Multiprocessors // Lobachevskii J. Math. 2018. Vol. 39, no. 4. P. 571–575. DOI:10.1134/S1995080218040121.
10. Silva L.M., Buyya R. Parallel programming models and paradigms // High Performance Cluster Computing: Architectures and Systems. Vol. 2. 1999. P. 4–27.
11. Darema F. SPMD Computational Model // Encyclopedia of Parallel Computing. Boston, MA: Springer US, 2011. P. 1933–1943. DOI:10.1007/978-0-387-09766-4_26.
12. Sahni S., Vairaktarakis G. The master-slave paradigm in parallel computer and industrial settings // J. Glob. Optim. 1996. Vol. 9, no. 3–4. P. 357–377. DOI:10.1007/BF00121679.
13. 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). Москва: Изд-во МГУ, 2017. P. 319–332.
14. Ежова Н.А., Соколинский Л.Б. Исследование масштабируемости итерационных алгоритмов при суперкомпьютерном моделировании физических процессов // Вычислительные методы и программирование. 2018. Т. 19, № 4. С. 416–430. DOI:10.26089/NumMet.v19r437.
15. Diacu F. The solution of the n-body problem // Math. Intell. 1996. Vol. 18, no. 3. P. 66–70. DOI:10.1007/BF03024313.
16. Marciniak A. Numerical Solutions of the N-Body Problem. Dordrecht, Boston, Lancaster: D. Reidel Publishing Company, 1985. 242 p. DOI:10.1007/978-94-009-5412-0.
17. Cole M.I. Parallel programming with list homomorphisms // Parallel Process. Lett. 1995. Vol. 05, no. 02. P. 191–203. DOI:10.1142/S0129626495000175.

18. 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.
19. Соколинская И.М., Соколинский Л.Б. Исследование масштабируемости модифицированного алгоритма Чиммино для линейных неравенств // Суперкомпьютерные дни в России: Труды международной конференции (24-25 сентября 2018 г., г. Москва). Москва: Изд-во МГУ, 2018. С. 673–983.