УДК 681.3

М.К. Буза, Цзяхуэй Лю

РЕАЛИЗАЦИЯ ЛОГИСТИЧЕСКОГО ОТОБРАЖЕНИЯ НА СИСТЕМЕ ПАРАЛЛЕЛЬНОГО ДЕЙСТВИЯ

Рассматриваются вычисление логистического отображения на параллельных системах и создание нерегулярного временного ряда с использованием логистического уравнения. Предлагается метод вычисления динамического хаоса на параллельных системах. По сравнению с последовательным разработанный метод позволяет ускорить вычисления. Анализируется зависимость позиций точности, количества итераций, времени вычисления и времени передачи данных.

Введение

В последние десятилетия в связи с бурным развитием компьютерных сетей кластерные архитектуры стали эффективным средством решения фундаментальных научных и инженерных задач, характеризующихся большим объемом вычислений [1, 2]. Эти задачи трудно, а порой и практически невозможно решить, используя только традиционные методы. Поэтому все чаще исследователи обращаются к возможностям применения альтернативных методов решения, используемых в других областях науки и техники. Такими областями, в частности, являются теория динамического хаоса и технология параллельных вычислений.

Исследования хаотических систем время от времени появлялись в литературе по прикладным вопросам [3–5]. Наиболее известная модель, построенная на основе хаотических систем, была предложена Э. Лоренцем в 1963 г. Он предложил модель конвенции в атмосфере, создав приближения очень сложных уравнений, описывающих это явление, значительно более простыми уравнениями с тремя неизвестными. Э. Лоренц обнаружил, что решения колеблются нерегулярным, почти случайным образом. Он также установил, что если незначительно изменять начальные значения переменных, то отклонения от исходного решения будут усиливаться, пока новое решение не окажется совершенно непохожим на исходное. Проведенный анализ показал, что малейшее изменение начальных условий радикально меняет характер движения. Поэтому движение оказывается динамически неустойчивым.

Наиболее значимым приложением теории нелинейных систем с хаотическим поведением является прогнозирование динамики порождаемых ими временных рядов. Как известно, большинство систем (природных, таких, например, как атмосфера, или искусственных, таких, например, как биржа) в силу их сложности не могут быть смоделированы с достаточной точностью. В связи с широким применением логистического отображения встает задача построения метода его параллельного вычисления, проведения исследования по различным показателям с указанием на возможные сферы приложения. Ниже предложено одно из решений данной задачи.

Параллельные технологии характеризуются тем, что, объединяя множество современных процессоров в единую вычислительную систему (систему параллельного действия, кластер), позволяют значительно увеличить вычислительные возможности. Это вселяет надежду на повышение точности моделирования поведения сложных систем.

Можно достаточно просто создать параллельную систему на базе компьютеров, соединенных при помощи коммуникационного оборудования и, таким образом, образующих один вычислительный ресурс. Системы, построенные на этом принципе, называются кластерами и относятся к классу параллельных систем с распределенной памятью [6, 7]. Узким местом кластеров является то, что чаще всего для взаимодействия отдельных узлов привлекается наиболее распространенное и дешевое коммуникационное оборудование Fast Ethernet, которое использует общую среду передачи данных, но обладает не очень большой пропускной способностью (по сравнению со скоростью обработки данных современными процессорами). Поэтому на подобных системах решаются в основном задачи с небольшим числом обменов по сравнению с объемом вычислений. Неоспоримыми преимуществами таких систем являются относительная дешевизна и простота

создания. Для их программирования применяются системы передачи сообщений, в которых отдельные компьютеры взаимодействуют посредством передачи и приема данных.

1. Логистическое отображение

В 1976 г. Роберт Мэй (Robert May) открыл интересное уравнение, которое использовалось для моделирования народонаселения. Логистическое отображение (также известное как квадратичное отображение, или отображение Фейгенбаума) является полиномиальным отображением, хрестоматийно упоминаемым в качестве типичного примера возникновения сложного, хаотического поведения из очень простых нелинейных уравнений. Логистическое отображение является дискретным аналогом указанного уравнения, отражая тот факт, что прирост популяции происходит в дискретные моменты времени.

Математическая формулировка отображения имеет следующий вид:

$$x_{n+1} = a * x_n * (1 - x_n), \quad (1)$$

где параметр x_n принимает значения от 0 до 1 и отражает численность популяции в n-м году, а x_0 обозначает начальную численность (в год номер 0). Параметр a – положительный параметр, характеризующий скорость размножения (роста) популяции, причем a принадлежит открытому интервалу (0, 4).

При значениях a, превышающих a_1 (a_1=3,569 94...), могут возникнуть хаотические итерации, т. е. поведение модели на больших временах не укладывается в рамки простого периодического движения. В интервале $a_1<a<4$ также присутствуют определенные узкие интервалы, для которых существуют периодические орбиты [3]. Такие интервалы на рис. 1 обозначены черным цветом.

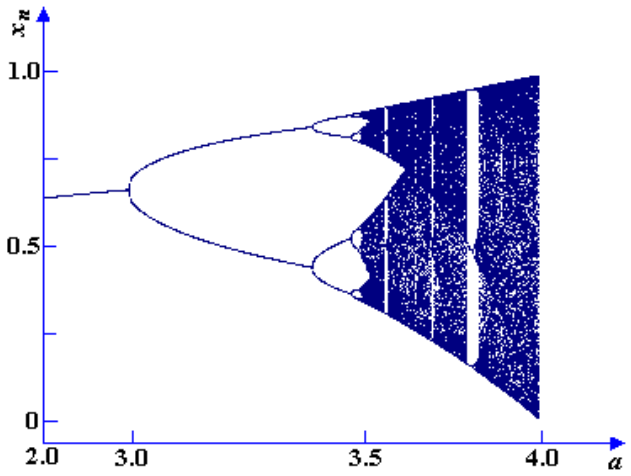


Рис. 1. Бифуркационная диаграмма логистического отображения

2. Параллельный алгоритм реализации логистического отображения

Чтобы достичь требуемой точности для хранения полученных результатов вычисления, будем использовать массивы. Это позволит нейтрализовать влияние ограниченной точности компьютера.

Алгоритм разделен на две части (рис. 2): $x = x * (1-x)$ и $x = a * x$.

Шаг 1. $x = x * (1-x)$.

Начальное значение (x_0) параметра x и значение параметра a заданы в текстовом файле. На первом этапе параллельного алгоритма главный процессор открывает данный файл и получает значения этих параметров. Известно, что в языке программирования Си полученные результаты вычисления с типом данных «double», «float», «int», «long» и т. д. состоят из ограни-

ченного количества байтов. Чтобы обойти это ограничение, используем одномерные массивы для хранения полученных результатов вычисления.

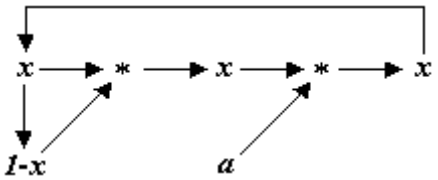


Рис. 2. Схема алгоритма

Начальное значение параметра x будем хранить в одномерном массиве X. Так как при выполнении новой итерации количество позиций точности параметра x изменяется, одномерный массив X является динамическим.

Значение параметра a находится в одномерном массиве A, который является статическим, так как значение параметра a остается неизменным.

Начальный объем одномерных массивов X и A выберем равным 256 байт, т. е. заданное максимальное количество позиций точности начального значения параметра x и значения параметра a не могут задаваться числами, которые требуют для своей записи более 256 байт памяти.

Главный процессор рассылает всем обрабатывающим процессорам значение параметра a и начальное значение параметра x. В каждом обрабатывающем процессоре вычисляется значение параметра y ($y = 1–x$), которое размещается в одномерном массиве Y. Одномерный массив Y является динамическим, начальный его объем – 256 байт.

Положим, что a=3,987 654 321, x_0=0,012 345 678 9.

Множество X имеет вид

$$X = \{ \alpha_1, \alpha_2, \dots , \alpha_N \},$$

где α_1 – первый элемент множества X; N – количество позиций точности параметра x.

Множество Y имеет вид

$$Y = \{ \beta_1, \beta_2, \dots , \beta_N \},$$

где β_1 – первый элемент множества Y; N – количество позиций точности параметра y.

Хранение значения x_0 в памяти будет осуществлено в виде одномерного массива

$$X = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}.$$

В первом элементе одномерного массива X ($X(\alpha_1)$) находится значение первой цифры после запятой, т. е. цифра 0 ($X[1]=0$), во втором элементе массива X ($X(\alpha_2)$) – значение второй цифры после запятой, т. е. цифра 1 ($X[2]=1$), и т. д.

Хранение значения a в памяти будет осуществлено в виде одномерного массива

$$A = \{3, 9, 8, 7, 6, 5, 4, 3, 2, 1\}.$$

В первом элементе одномерного массива A находится значение целой части параметра a, т. е. цифра 3 ($A[1]=3$), во втором элементе массива A – значение первой цифры после запятой, т. е. цифра 9 ($A[2]=9$), и т. д.

Одномерный массив Y хранит значение параметра y ($y =1–x_0$). На первом шаге вычислений оно равно 0,987 654 321 1.

Хранение значения y в памяти будет осуществлено в виде одномерного массива

$$Y = \{ 9, 8, 7, 6, 5, 4, 3, 2, 1, 1 \}.$$

Затем вычисляется $X(\alpha_1, \alpha_2, \ldots, \alpha_N) * Y(\beta_1, \beta_2, \ldots, \beta_N)$. В итоге в главном процессоре получаем произведение $x*(1–x)$.

Полученные результаты вычисления записываются в матрицу M размера $n\times 2n$:

$$M = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1n} & m_{1(n+1)} & \cdots & m_{1(2n)} \\ m_{21} & m_{22} & \cdots & m_{2n} & m_{2(n+1)} & \cdots & m_{2(2n)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} & m_{n(n+1)} & \cdots & m_{n(2n)} \end{pmatrix}_{n\times 2n}, \quad (2)$$

где n – количество позиций точности параметра x в текущей итерации.

Полученный результат произведения $x*(1–x)$ в текущей итерации имеет следующий вид:

$$X = (\ \alpha_1\ \alpha_2 \ldots \alpha_n\ \alpha_{n+1} \ldots \alpha_{2n}); \quad (3)$$

$$\alpha_i = (\ \eta_i + d_i\ )\ (Mod\ 10)\ (1 \leqslant i \leqslant 2n\ ); \quad (4)$$

$$\eta_i = \sum_{j=1}^{n} m_{ji}\ ; \quad (5)$$

$$d_{i\text{-}1} = \text{int}\ [(\eta_i + d_i)\ /\ 10]\ (1 \leqslant i \leqslant 2n\ ), \quad (6)$$

где функция int – получение целой части результатов вычисления, начальное значение переменной d_{2n} равно нулю.

Рассмотрим три стратегии загрузки процессоров при вычислении произведения $x*(1–x)$. Пусть p – количество обрабатывающих процессоров в системе:

1) $n < p$ – некоторые обрабатывающие процессоры не будут использованы для вычисления, i-я строка матрицы M вычисляется обрабатывающим процессором i;

2) $n = p$ – каждый обрабатывающий процессор вычисляет одну строку матрицы M, i-я строка матрицы M вычисляется обрабатывающим процессором i;

3) $n > p$ – некоторые обрабатывающие процессоры должны вычислять две или более строк матрицы M. В зависимости от количества строк матрицы M номер обрабатывающего процессора P_{No} для строки i определяется следующим образом:

$$P_{No} = (i–1)\ (\ Mod\ p)+1\ (\ i = 1, 2, \ldots; P_{No} = 1, 2, \ldots, p), \quad (7)$$

где i – i-я строка матрицы M.

Обрабатывающие процессоры посылают результаты каждого цикла главному процессору. Данные размещаются в соответствующей строке матрицы M, которая находится в главном процессоре для вычисления результата текущей итерации и представляет собой двухмерные массивы размера $n\times 2n$:

$$M = \begin{pmatrix} 0 & (\alpha_1\beta_1) & (\alpha_2\beta_1) & \cdots & (\alpha_n\beta_1) & 0 & \cdots & 0 \\ 0 & 0 & (\alpha_1\beta_2) & (\alpha_2\beta_2) & \cdots & (\alpha_n\beta_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & (\alpha_1\beta_n) & (\alpha_2\beta_n) & \cdots & \alpha_n\beta_n \end{pmatrix}_{n\times 2n}. \quad (8)$$

Затем главный процессор передает новые значения x и изменившееся значение массива X каждому обрабатывающему процессору для выполнения следующего шага.

Шаг 2. $x = a * x$.

На втором шаге параллельного алгоритма в каждом обрабатывающем процессоре параметр a представляется как одномерный массив. Заметим, что в этот раз только размер матрицы M изменяется, а процесс параллельного вычисления аналогичен предыдущему шагу. Результа-

ты итерации получает главный процессор, который передает результаты новых значений и изменившуюся величину одномерного массива X каждому обрабатывающему процессору для выполнения следующей итерации.

Таким образом, на p процессорах реализуется алгоритм решения логистического отображения.

3. Определение количества требуемых итераций

Естественно, возникает задача определения количества выполняемых итераций для достижения требуемой точности, которая исследуется ниже.

В принципе, хаотические последовательности непериодические [8], но в компьютере полученные результаты представляются при ограниченном количестве байтов, т. е. количество позиций точности ограничено, поэтому некоторые значения периодически повторяются. Это позволяет выделить периодические последовательности значений. Пусть $a = 3{,}9$, $x_0 = 0{,}1$. Заметим, что когда количество позиций точности равняется 1, через 4 итерации некоторые значения возникают снова (рис. 3, *а*). Когда количество позиций точности равняется 2, в начальной части полученные последовательности не являются периодическими, затем итерация продолжается и последовательности становятся периодическими (рис. 3, *б*). Когда количество позиций точности равняется 4 или 5, период полученных последовательностей становится длиннее (рис. 3, *в*, *г*). Таким образом, когда количество позиций точности увеличивается, период полученных хаотических последовательностей удлиняется.

Генераторы случайных и псевдослучайных последовательностей являются неотъемлемыми элементами современных систем криптографической защиты информации. Важнейшим требованием, предъявляемым к генераторам псевдослучайных последовательностей, является равномерно распределенная случайная последовательность, т. е. случайная последовательность, элементы которой независимы в совокупности и имеют равномерное распределение вероятностей. При возрастании количества позиций точности период полученных последовательностей становится более длинным.

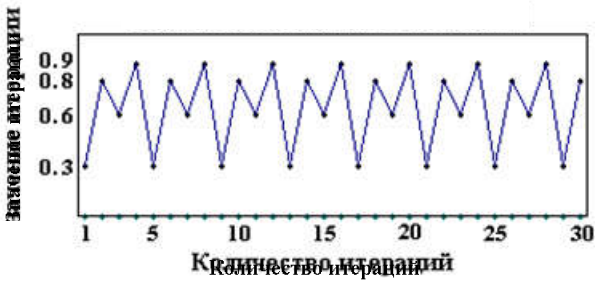
Так как хаотические псевдослучайные последовательности обладают почти равномерным распределением, то, выполнив соответствующее количество итераций, всегда можно получить хаотическую последовательность с заданным пользователем количеством позиций точности. Распределение цифр в максимальных для данного множества итераций хаотических последовательностях имеет лучшее равномерное распределение вероятностей, чем при небольшом заданном количестве позиций точности в полученных хаотических последовательностях. При $x_0 = 0{,}1$ и $a = 3{,}9$ полученная зависимость максимального количества позиций точности от количества итераций приведена в таблице.

Зависимость максимального количества позиций точности от количества итераций

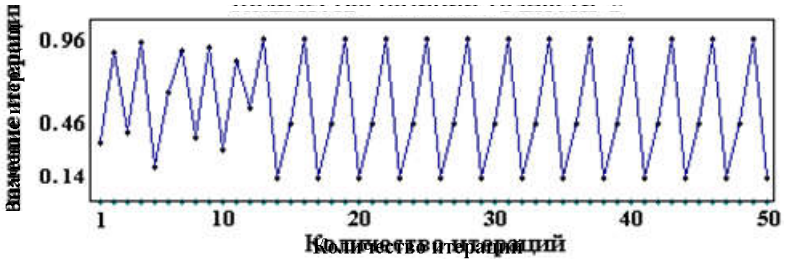
Количество итераций	Максимальное количество позиций точности	Количество итераций	Максимальное количество позиций точности
1	3	11	4 095
2	7	12	8 191
3	15	13	16 383
4	31	14	32 767
5	63	15	65 535
6	127	16	131 071
7	255	17	262 143
8	511	18	524 287
9	1 023	19	1 048 575
10	2 047	20	2 097 151

Когда начальное количество позиций точности параметров x и a равняется 1, через одну итерацию максимальное количество позиций точности полученных хаотических последова-

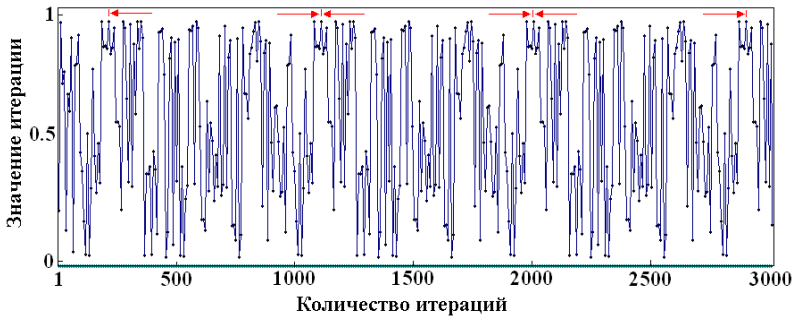
тельностей будет равным 3, через 5 итераций – 63, через 20 итераций – 2 097 151. На практике количество позиций точности полученных хаотических последовательностей задается по требованию пользователя в зависимости от поставленной цели в каждом конкретном случае.



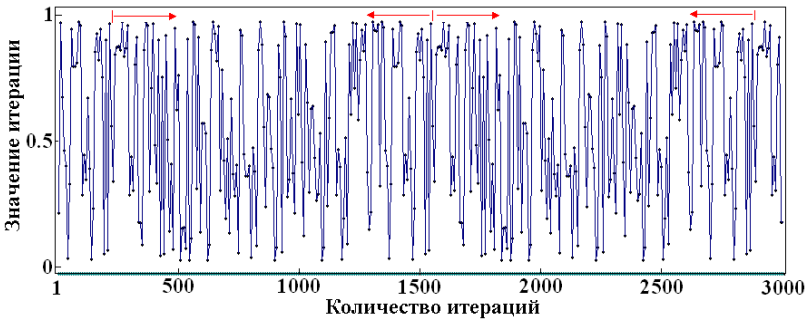
а)



б)



в)



г)

Рис. 3. Зависимость количества позиций точности от периода полученных хаотических последовательностей: *а)* количество позиций точности – 1; *б)* количество позиций точности – 2; *в)* количество позиций точности – 4; *г)* количество позиций точности – 5

Положим, что $a = 3{,}987\,65$, $x_0 = 0{,}123\,45$. Начальное количество позиций точности параметров x и a равняется 5. Если количество позиций точности ограничить значениями 50, 1000 и 10 000, то при 50 итерациях получим три псевдохаотических последовательности $S1$, $S2$ и $S3$ (рис. 4).

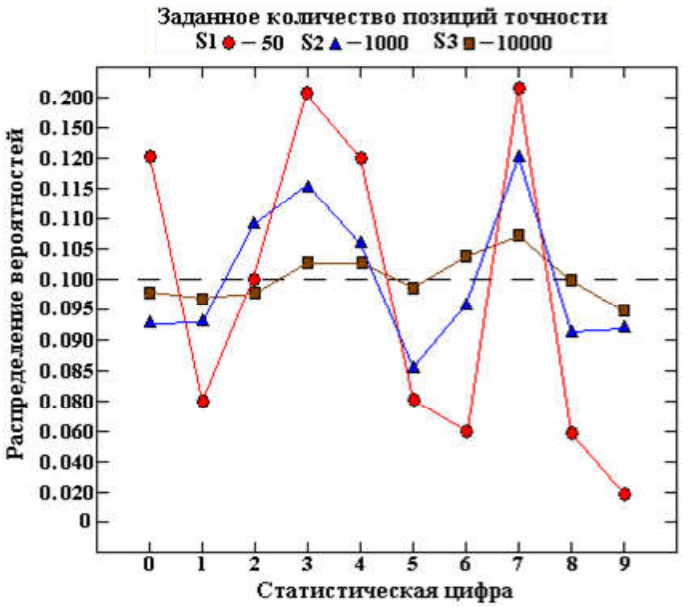


Рис. 4. Зависимость величины позиции точности от статистического распределения вероятностей

Видно, что последовательности $S1$, $S2$ и $S3$ различны по заданному количеству позиций точности полученных хаотических последовательностей, хотя начальные значения параметров x и a одинаковы.

Через 50 итераций получены хаотические псевдослучайные последовательности $S1$, $S2$ и $S3$. Заметим, что чем больше заданное количество позиций точности, тем равномернее становится распределенная случайная последовательность.

4. Оценка скорости вычислений

Подсчитаем время, затрачиваемое на вычисления. Сначала рассмотрим время последовательных вычислений. Каждый шаг итерации требует времени

$$T_{\text{onestep-}s} = T_{\text{initial-}s} + T_{\text{calculation-}s}\,, \tag{9}$$

где $T_{\text{initial-}s}$ – время инициализации: получения начальных значений параметров x и a, вычисления значений одномерного массива Y, инициализации матрицы M для вычисления произведений $x*(1–x)$ и произведения $a*x$. Параметр $T_{\text{calculation-}s}$ обозначает время последовательных вычислений на одном компьютере:

$$T_{\text{calculation-}s} = T_{x\cdot(1-x)\text{-}s} + T_{a\cdot x\text{-}s}\,, \tag{10}$$

где $T_{x\cdot(1-x)\text{-}s}$ – время вычисления произведения $x*(1–x)$; $T_{a\cdot x\text{-}s}$ – время вычисления произведения $a*x$ (x является новым значением, полученным по результатам предыдущего шага).

Величина $T_{x\cdot(1-x)\text{-}s}$ находится по формуле

$$T_{x\cdot(1-x)\text{-}s} = L_n \cdot L_n \cdot t_{\text{mult}} + 2 \cdot L_n \cdot L_n \cdot t_{\text{add}}\,, \tag{11}$$

где L_n – количество позиций точности параметра x; t_{mult} – время операции умножения двух целых чисел на одном компьютере; t_{add} – время операции сложения двух целых чисел на одном компьютере.

Для вычисления $T_{a\cdot x\text{-}s}$ верна формула

$$T_{a\cdot x\text{-}s} = 2 \cdot L_n \cdot L_a \cdot t_{\text{mult}} + (2 \cdot L_n + L_a) \cdot L_a \cdot t_{\text{add}} , \quad (12)$$

где L_a – количество позиций точности параметра a.

При достижении требуемой точности после выполнения N итераций получим следующее время вычислений на одном компьютере:

$$T_{\text{total-}s} = N \cdot T_{\text{average\_onestep-}s} . \quad (13)$$

Этот алгоритм при параллельных вычислениях требует времени

$$T_{\text{total-}m} = N \cdot T_{\text{average\_onestep-}m} , \quad (14)$$

где N – количество итераций; $T_{\text{onestep-}m}$ – время каждого шага итерации при параллельных вычислениях. Последняя компонента вычисляется из соотношения

$$T_{\text{onestep-}m} = T_{\text{initial-}m} + T_{\text{calculation-}m} + T_{\text{transmit-}m} , \quad (15)$$

где $T_{\text{initial-}m}$ – время инициализации параллельных вычислений: каждый обрабатывающий процессор получает одномерные массивы X, Y и A для выполнения параллельных вычислений, при этом синхронизация должна выполняться после каждого этапа; $T_{\text{transmit-}m}$ – время передачи сообщения через коммуникационную систему; $T_{\text{calculation-}m}$ – время параллельных вычислений, состоящее из времени вычисления в главном процессоре и среднего времени вычисления в обрабатывающих процессорах.

При этом

$$T_{\text{calculation-}m} = T_{\text{calculation-}m\text{-}0} + T_{\text{calculation-}m\text{-other}} , \quad (16)$$

где $T_{\text{calculation-}m\text{-}0}$ – время сбора результатов параллельных вычислений в главном процессоре; $T_{\text{calculation-}m\text{-other}}$ – среднее время вычисления в обрабатывающих процессорах. Указанные составляющие находятся по формулам

$$T_{\text{calculation-}m\text{-}0} = 2 \cdot L_n \cdot L_n \cdot t_{\text{add}} + (2 \cdot L_n + L_a) \cdot L_a \cdot t_{\text{add}} ; \quad (17)$$

$$T_{\text{calculation-}m\text{-other}} = (L_n \cdot t_{\text{mult}}) \cdot (L_n / N_{\text{process}}) + (2 \cdot L_n \cdot t_{\text{mult}}) \cdot (L_a / N_{\text{process}}) , \quad (18)$$

где N_{process} – количество занятых обрабатывающих процессоров.

Время передачи сообщения

$$T_{\text{transmit-}m} = (L_n / N_{\text{process}}) \cdot 2 \cdot L_n \cdot t_{\text{unit}} + (L_a / N_{\text{process}}) \cdot (2 \cdot L_n + L_a) \cdot t_{\text{unit}} , \quad (19)$$

где t_{unit} – время передачи одного слова коммуникационной системой.

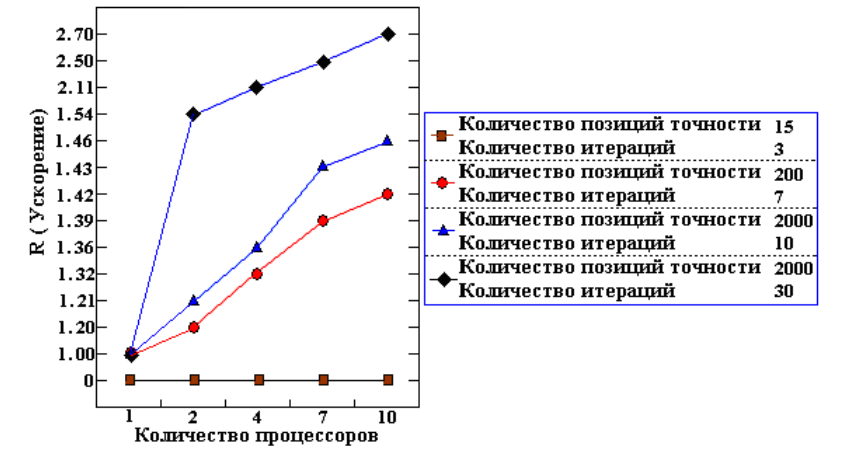
Для получения параметра L_n каждого шага итерации используем выражение

$$L_{n+1} = 2 \cdot L_n + L_a - 1 \ (n = 0, 1, 2, \ldots) . \quad (20)$$

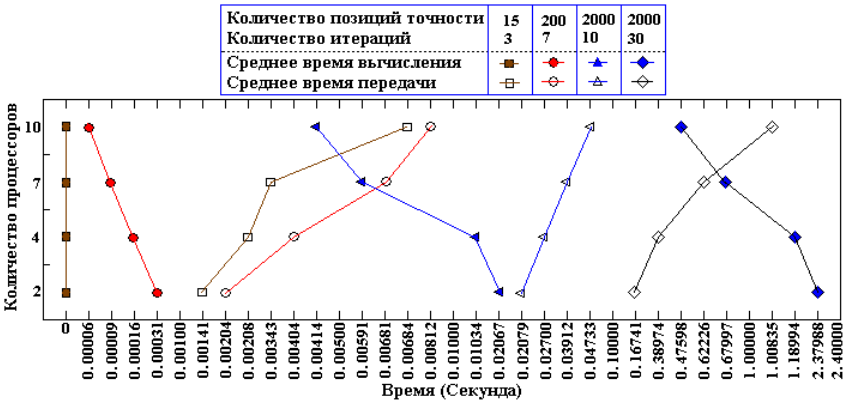
Ускорение вычисляется по формуле

$$R = T_{\text{total-}s} / T_{\text{total-}m} . \quad (21)$$

Тестирование проведено для кластера типа Beowulf, построенного на базе сети Fast Ethernet с числом процессоров до 10. Расчет произведен для t_{unit} = 1 мкс (рис. 5).



а)



б)

Рис. 5. Пример вычисления на кластере типа Beowulf с сетью Fast Ethernet:
а) зависимость ускорения вычислений от количества процессоров;
б) среднее время вычисления и передачи данных

Заключение

Проведенные в работе исследования показывают, что в случае параллельных вычислений получается большее количество позиций точности, чем при последовательных вычислениях за одинаковое время. Если количество итераций и позиций точности уменьшается (см. рис. 5), время вычисления приближается к нулю, а главным становится время передачи. При большом количестве позиций точности или большом количестве итераций главным становится время вычисления.

Когда количество позиций точности и количество итераций существенно увеличивается, каждый процессор приближается к среднему времени вычисления и среднему времени передачи, что позволяет достичь лучшей производительности системы. При увеличении количества позиций точности период полученных последовательностей становится более длинным, т. е. распределение их цифр начинает обладать хорошим случайным свойством.

Полученные результаты можно использовать для уменьшения времени вычислений, требующих большой точности, для ускорения процесса шифрования текстовой, графической и речевой информации, а также при требовании большой точности восстановления переданной информации.

Список литературы

1. Буза, М.К. Архитектура компьютеров / М.К. Буза. – Минск : Новое знание, 2007. – 559 с.

2. Буза, М.К. Модели распределенной разделяемой памяти / М.К. Буза, Л.Ф. Зимянин // Сб. тр. Междунар. науч. конф. «Сетевые компьютерные технологии» (NCT' 2005). – Минск : Изд. центр БГУ, 2005. – С. 15–20.

3. Garnett, P.W. Chaos theory tamed / P.W. Garnett. – Washington : Joseph Henry Press, 1997. – 405 p.

4. Kossakowski, A. How can we observe and describe chaos / A. Kossakowski, M. Ohya, Y. Togawa // Open Systems and Information Dynamics. – 2003. – Vol. 10, № 3. – P. 221–233.

5. Jiang, B.T. An encryption method based on chaos and Fibonacci pseudo-random sequences / B.T. Jiang, J.H. Liu, B. Xu // Journal of Northeastern University (Natural Science). – 2005. – Vol. 26, № 9. – P. 864–866.

6. Шпаковский, Г.И. Программирование для многопроцессорных систем в стандарте MPI / Г.И. Шпаковский, Н.В. Серикова. – Минск : Изд-во БГУ, 2002. – 323 с.

7. Барский, А.Б. Параллельные информационные технологии : учебное пособие / А.Б. Барский. – М. : БИНОМ. Лаборатория знаний, 2007. – 503 с.

8. Ditto, W. Principles and applications of chaotic systems / W. Ditto, T. Munakata // Communications of the ACM. – 1995. – Vol. 38, № 11. – P. 96–102.

Поступила 28.01.09

Белорусский государственный университет,
Минск, пр. Независимости, 4
e-mail: bouza@bsu.by

M.K. Bouza, Jiahui Liu

IMPLEMENTATION OF A LOGISTIC MAPPING IN A PARALLEL SYSTEM

Computing the logistic mapping using parallel systems and generating an irregular time series by the Logistic equation are considered. A method of dynamic chaos computing based on the parallel systems is introduced. In comparison with the sequential processing mode, the method makes possible certain acceleration of computations. The mutual relationships of interrelation of the precision position, the iteration number, the time of computing and transmitting data have been analyzed.