СЕКЦИЯ 2. ПАРАЛЛЕЛЬНАЯ И РАСПРЕДЕЛЕННАЯ ОБРАБОТКА ДАННЫХ

ВЫЧИСЛЕНИЕ LOGISTIC ФУНКЦИИ НА ПАРАЛЛЕЛЬНЫХ СИСТЕМАХ

М.К. Буза, Л.Ф. Зимянин, Лю Цзяхуэй
Беларусь, г. Минск

Введение.

В последнее десятилетие в связи с бурным развитием компьютерных сетей, кластерные архитектуры являются эффективным средством решения фундаментальных научных и инженерных задач, характеризующихся большим объёмов вычислений [1]. Но эти задачи трудно, а порой и практически невозможно, решить, используя только традиционные методы. Поэтому взгляды исследователей все чаще обращаются в сторону возможности применения альтернативных методов решения, применяемых в других областях науки и техники. Одной из таких областей является теория динамического хаоса и технология параллельных вычислений.

В 1963 году в динамической системе Лоренцем было обнаружено очень сложное движение, которое воспринималось как хаотическое. Для характеристики таких движений ввели понятие "динамический хаос". Проведенный анализ показал, что малейшее изменение начальных условий радикально меняет характер движения. Тем самым 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. Параллельные технологии характеризуются тем, что объединяя множество современных процессоров в единую вычислительную систему (систему параллельного действия, кластер), позволяют получить мощные вычислительные возможности.

1. Логистическое отображение.

В 1976 году Robert May и Mitchell Feigenbaum открыли интересное и известное уравнение, которое использовалось для моделирования народонаселения. Это логистическое отображение.

$X_{n+1} = a * X_n * (1 - Xn)$

где параметр a принадлежит интервалу (0,4); параметр X_n (n = 0, 1, 2, ...) принадлежит интервалу (0,1) [2].

При значениях a, превышающих a1(a1=3.56994...), могут возникнуть хаотические итерации, т.е. поведение модели на больших временах не укладывается в рамки простого периодического движения. В интервале a1<a<4 также при-

сутствуют определенные узкие интервалы, для которых существуют периодические орбиты, которые показаны на рис. 1. [3]

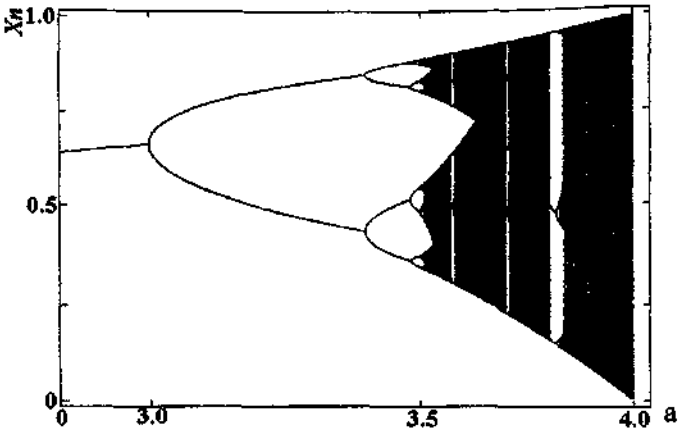

Рис. 1. Бифуркационная диаграмма

2. Параллельный алгоритм решения логистического отображения

Для того чтобы достичь требуемой точности, в параллельном алгоритме для хранения полученных результатов вычисления, используем массивы. Это позволяет нейтрализовать влияние ограниченной точности компьютера.

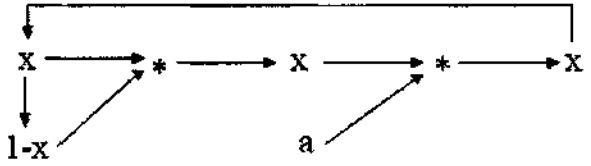

Рис. 2. Схема алгоритма

Алгоритм разделён на две части (рис. 2):

X = X * (1 - X). На первом этапе параллельного алгоритма процессор 0 рассылает начальное значение одномерного массива X всем процессорам. В каждом процессоре получен параметр 1-X, размещенный в одномерном массиве Y, потом по всем номерам процессоров вычислены X (x1, x2, ... , xn, ...) * Y(ymyid , ymyid+n , …), где myid – номер процессора, n – количество процессоров. Каждый процессор посылает результат каждого цикла процессору 0. Данные размещаются в соответствующей строке матрицы, которая находится в процессоре 0 для вычисления суммы и представляет собой двумерные массивы размера N×N, где N – количество позиций параметра X. В процессоре 0 получена сумма X*(1-X). Процессор 0 передает новые значения и изменившуюся величину одномерного массива X каждому процессору для выполнения следующего этапа.

2) X = a * X. На втором этапе параллельного алгоритма в каждом процессоре параметр a представляется как одномерный массив. Заметим, что в этот раз только размер матрицы изменяется, но процесс параллельного вычисления аналогичен предыдущему шагу. Результаты этой итерации получает процессор 0, который передает результаты новых значений и изменившуюся величину одномерного массива X каждому процессору для выполнения следующей итерации.

Для достижения требуемой точности необходимо определелить количество итераций.

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

$T_{onestep\text{-}s} = T_{initial\text{-}s} + T_{calculation\text{-}s}$.

Здесь $T_{initial\text{-}s}$ – время инициализации: получение начальных значений параметра X и начальных значений параметра a, вычисление значения одномерного массива 1-X, инициализация матрицы для вычисления сумм. Параметр Tcalculation-s обозначает время последовательных вычислений на одном компьютере: $T_{calculation\text{-}s} = T_{x\cdot(1\text{-}x)\text{-}s} + T_{a\cdot x\text{-}s}$. Здесь $T_{x\cdot(1\text{-}x)\text{-}s}$ – время умножения x*(1-x), а $T_{a\cdot x\text{-}s}$ – время умножения a*x (x является новым значением, вычисленным по результатам предыдущего шага).

$T_{x\cdot(1\text{-}x)\text{-}s} = K_n \cdot K_n \ tmult + 2\ K_n\ K_n\ tadd$,

где K_n – количество позиций параметра X; tmult – время операции умножения двух целых чисел на одном компьютере; tadd – время операции сложения двух целых чисел на одном компьютере.

$T_{a\cdot x\text{-}s} = 2 \cdot K_n \cdot K_a \cdot tmult + (2 \cdot K_n + K_a) \cdot K_a \cdot tadd$,

где K_a – количество позиций параметра a.

Для достижения требуемой точности после выполнения N итераций, получим время вычислений на одном компьютере $T_{total\text{-}s} = N \cdot T_{onestep\text{-}s}$.

Этот алгоритм при параллельных вычислениях требует времени $T_{total\text{-}m} = N \cdot T_{onestep\text{-}m}$. Здесь N – количество итераций; $T_{onestep\text{-}m}$ – время каждого шага итерации при параллельных вычислениях.

$T_{onestep\text{-}m} = T_{initial\text{-}m} + T_{calculation\text{-}m} + T_{transmit\text{-}m}$.

1) Здесь $T_{initial\text{-}m}$ – время инициализации параллельных вычислений: каждый процессор получает одномерный массив X, одномерный массив 1-X , одномерный массив a для выполнения параллельных вычислений, при этом синхронизация должна выполняться после каждого этапа; $T_{calculation\text{-}m}$ – время вычислений в одном компьютере; $T_{transmit}$ – время передачи сообщения через коммуникационную систему.

2) Здесь $T_{calculation\text{-}m}$ – время параллельных вычислений, состоящее из времени вычисления в процессоре 0 и каждом параллельном процессоре P .

$T_{calculation\text{-}m} = T_{calculation\text{-}m\text{-}0} + T_{calculation\text{-}m\text{-}other}$,

где $T_{calculation\text{-}m\text{-}0}$ – время сбора результатов параллельных вычислений в процессоре 0 ; $T_{calculation\text{-}m\text{-}other}$ – время параллельных вычислений в каждом процессоре P (P=1,2,...).

$T_{calculation\text{-}m\text{-}0} = 2 \cdot K_n \cdot K_n \cdot t_{add} + (2 \cdot K_n + K_a) \cdot K_a \cdot t_{add}$.

$T_{calculation\text{-}m\text{-}other} = (K_n \cdot t_{mult}) \cdot (K_n / (N_{process} - 1)) + (2 \cdot K_n \cdot t_{mult}) \cdot (K_a / (N_{process} - 1))$

3) Время передачи сообщения равно

$T_{transmit\text{-}m} = (K_n / (N_{process} - 1)) \cdot 2 K_n \cdot t_{unit} + (K_a / (N_{process} - 1))(2 \cdot K_n + K_a) \cdot t_{unit}$,

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

4) Для получения параметра K_n каждого шага итерации используем выражение: $K_{n+1} = 2 K_n + K_a - 1$ (n = 0, 1, 2, ...)

5) Ускорение вычисляется по формуле $R = T_{total\text{-}s} / T_{total\text{-}m}$

Таблица 1. Зависимость количества итераций от количества позиций точности

Количество итераций	1	2	3	4	5	6	7	8	9	10
Количество позиций	3	7	15	31	63	127	255	511	1023	2047
Количество итераций	11	12	13	14	15	16	17	18	19	20
Количество позиций	4095	8191	16383	32767	65535	131071	262143	524287	1048575	2097151

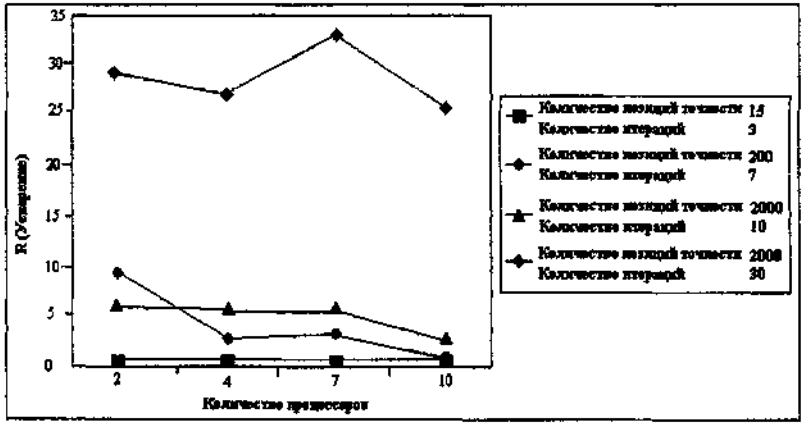

Рис. 3. Ускорение для кластера типа Beowulf с сетью Fast Ethernet

Тестирование приведено для кластера типа Beowulf, построенного на базе сети Fast Ethernet с числом процессоров до 10. Расчет производился для: tunit = 1 мкс.

При X_0 =0.1 и a = 3.9, полученная зависимость количества итераций от количества позиций точности приведена в табл. 1.

Соответствующие графики представлены на рис. 3 и рис. 4.

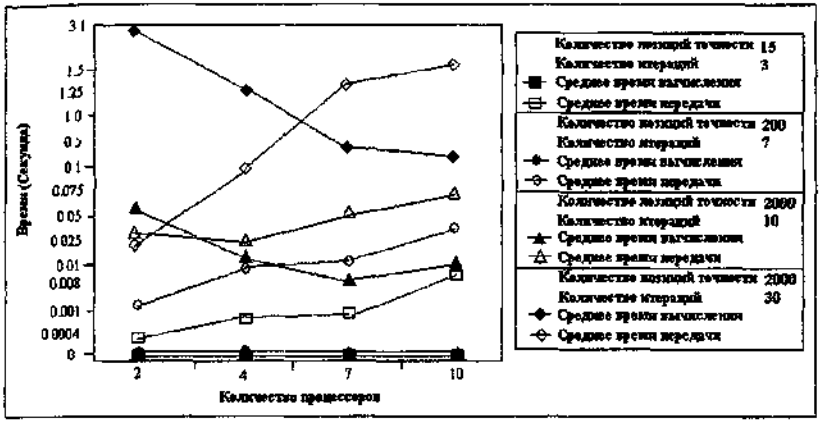

Рис. 4. Среднее время вычисления и среднее время передачи

Выводы

1. В случае параллельных вычислений получаем существенно большее количество позиций точности, чем при последовательных вычислениях на одном процессоре.

2. Чем больше позиций точности, тем больше требуется вычислений, поэтому, например, в данном случае при количестве итераций равным 3 и количестве позиций точности равным 15, время вычисления приближается к нулю. В этом случае время передачи становится главным. При большом количестве позиций точности или большом количестве итераций становится главным время вычисления.

3. Когда количество позиций точности и количество итераций существенно увеличивается, каждый процессор подходит к среднему времени вычисления и среднему времени передачи, поэтому достигается лучшая производительность системы.

4. Полученные результаты можно использовать: а) для уменьшения времени вычислений результатов, требующих большой точности; б) для ускорения процесса шифрования текстовой, графической и речевой информации. Особенно существенное ускорение может быть получено при требовании большой точности восстановления переданной информации.

Литература

1. Буза М.К.,Зимянин Л.Ф. Модели распределенной разделяемой памяти. // II Международная научн. Конф. «Сетевые компьтерные технологии» NCT'2005 Сб Тр. -Мн.: Изд.центр БГУ, 2005. – С.15-20.

2 Garnett P. Williams. Chaos Theory Tamed. Joseph Henry Press, 1997.