Література

1. Jarke, M. Query Optimization in Database Systems [Текст] / M. Jarke, J. Koch // Computing Surveys. – 1984. – Vol. 16, № 2 – P. 111-152.
2. Білоусова, Л. І. Потенціал комп'ютерного тестування [Текст] / Л. І. Білоусова // Вісник ТІМО. – 2008. – №10. – С. 40-44.
3. Фісун, М. Т. Розробка синтаксичного аналізатора мови програмування PL/I для реінженірінгу блок-схем алгоритмів [Текст] / М.Т. Фісун, О.В. Гнездьонова // Проблеми програмування. – 2006. – №2-3. – С. 617-625.
4. Моисеенко, С. SQL. Задачи и решения. Интерактивный учебник [Електронній ресурс] – Режим доступу: http://www.sql-tutorial.ru/ru/content.html.
5. Интерфейс Ltd. Internet&Software Company – Режим доступу: http://www.interface.ru/home.asp.

Застосування принципів локально-паралельної обробки до клітинних автоматів дозволяє суттєво підвищити розміри моделей. Запропоновано і продемонстровано на тестових прикладах локально-паралельний стековий алгоритм бінарного клітинного автомату

Ключові слова: клітинні автомати, локальна паралельність, стек

Применение принципов локально-параллельной обработки к клеточным автоматам позволяет существенно увеличить размеры моделей. Предложен и продемонстрирован на тестовых примерах локально-параллельный стековый алгоритм бинарного клеточного автомата

Ключевые слова: клеточные автоматы, локальная параллельность, стек

Application of principle of local-parallel processing to cellular automaton allows greatly enlarge the sizes of models. Local-parallel stacking algorithm of the binary cellular automaton is offered and demonstrated on test examples

Keywords: cellular automatons, local parallelism, stack

УДК 519.87

СТЕКОВЫЙ АЛГОРИТМ ДЛЯ ЛОКАЛЬНО-ПАРАЛЛЕЛЬНОГО БИНАРНОГО КЛЕТОЧНОГО АВТОМАТА

Бенаддия Абдельлатиф
Аспирант*

О.Ф.Михаль
Доктор технических наук, доцент, профессор*
*Кафедра Электронно-вычислительных машин
Харьковский национальный университет радиоэлектроники
пр. Ленина, 14, г. Харьков, 61166
Контактный тел. (057) 40-93-54
E-mail: fuzzy16@pisem.net

1. Введение

Моделирование крупномасштабных систем с распределённой обработкой информации актуально в связи с изучением динамики процессов, происходящих в человеческом окружении, на различных уровнях от глобальных климатических явлений до живой природы и социальных структур (экология, климатология, демография, социология). Общность подхода состоит в том, что система рассматривается как состоящая из большого числа дискретных элементов (ячеек), взаимодействующих друг с другом в соответствии с фиксированным наборами правил. Перспективным аппаратом компьютерного моделирования подобных систем являются *клеточные автоматы* (КА). Распространённость моделирования поведения систем на КА обусловлена простотой описания конфигурации и набора рабочих правил. Дополнительный выигрыш по эффективности работы КА может быть обеспечен *локально-параллельными* (ЛП) алгоритмами [1], которые разработаны [2, 3] для целочисленных неотрицательных значений операндов, что применимо к описанию ячеек КА. В частности, если ячейка принимает одно из значений (0, 1), это соответствует подвиду КА – *бинарным клеточным автоматам* (БКА) [4]. Для полноты оперирования данными, необходимо дополнить аппарат описания [2, 3] ЛП алгоритмами, реализующими

смену состояний ячеек КА для конкретных видов (вариантов) КА.

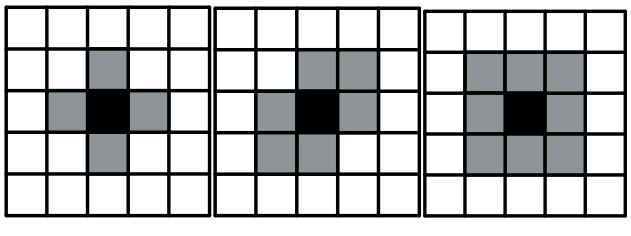
Цель настоящей работы – разработка и исследование ЛП варианта одной из процедур работы КА – стекового алгоритма определения степени соседства ячеек БКА.

2. Клеточные автоматы

В основе КА [4] лежат следующие представления. В регулярной решётке каждая ячейка может находится в одном из конечного множества состояний. Для ячейки определено её окружение: множество *соседних* ячеек. Задаются начальные состояние ячеек, и набор правил перехода из одного состояния в другое. На каждой итерации определяется новое состояние каждой ячейки. При этом весь КА последовательно сменяет состояния (поколения).

Работа КА необратима, как и процессы в живой природе. Текущее j-е состояние КА однозначно определяетследующее $(j+1)$-есостояние,непредыдущее $(j-1)$-е состояние не может быть однозначно восстановлено. Таким образом, КА «автоматически» воспроизводят важную особенность реального мира – однонаправленность времени.

В БКА для записи состояния ячейки используется один бит. Допустимы различные варианты соседства ячеек. Рис. 1 иллюстрирует некоторые из возможных топологий БКА: ситуации с 4, 6 и 8 ячейками-соседями. Центральная ячейка условно выделена чёрным цветом, её соседи – серым.



а) б) в)

Рис. 1. Бинарные КА с вариантами соседства: 4 (а), 6 (б) и 8 клеток (в)

Известен вариант БКА - «игра Жизнь» [4], структурно соответствующий рис. 1 в. Ячейка КА может находиться в одном из двух состояний: 0 или 1. При степени соседства 3 или 4 единичное состояние сохраняется или 0 переходит в 1. Иначе – 0 сохраняется или 1 переходит в 0. В компьютерной реализации, даже по таким простым правилам, «игра Жизнь» демонстрирует поведение «сообщества», аналогичное наблюдаемому у микроорганизмов.

Содержательность моделей растёт с их размерами. Применительно к воспроизведению поведения, характерного для живой природы, целесообразны модели объёмом 10^4 ячеек и более. В связи с этим при компьютерной реализации возникает проблема экономного расходования ресурсов *вычислительной системы* (ВС) – оперативной памяти, дискового пространства, времени работы программы и др. Информация о состоянии i-й ячейки должна храниться отдельно от информации о состоянии других ячеек. То есть, для каждой ячейки требуется отдельный элемент хранения. Стандартные форматы хранения данных – расточительны в отношении использования ресурсов ВС. Существенный выигрыш обеспечивает ЛП представлении информации [2, 3].

3. Принцип локальной параллельности

Пусть имеется два n-компонентных вектора:

$$A:\{a_1,a_2,a_3,...,a_i,...,a_n\};\ B:\{b_1,b_2,b_3,...,b_i,...,b_n\};$$

$$i\in(1,2,3,...,n);\ a_i\le a_{max};\ b_i\le b_{max};\ a_{max}=b_{max}.$$

Требуется найти покомпонентную сумму векторов: $C:\{c_1,c_2,c_3,...,c_i,...,c_n\}$; $c_i=a_i+b_i$. Согласно последовательной схеме, для этого должен быть выполнен следующий набор операций.

Шаг 1. Начальная установка: $i=1$.

Шаг 2. Извлечение значения операнда a_i из регистра хранения.

Шаг 3. Извлечение значения операнда b_i из регистра хранения.

Шаг 4. Суммирование: $c_i=a_i+b_i$.

Шаг 5. Помещение результата c_i в регистр хранения.

Шаг 6. $i=i+1$. Если $i>n$, завершение вычисления; иначе переход к Шагу 2.

Согласно ЛП схеме, для нахождения покомпонентной суммы необходимо:

Шаг 1. Конкатенация:

$$A:\{a_1,a_2,a_3,...,a_n\}\to a_\#=(a_1\oplus a_2\oplus a_3\oplus...\oplus a_n);$$

$$B:\{b_1,b_2,b_3,...,b_n\}\to b_\#=(b_1\oplus b_2\oplus b_3\oplus...\oplus b_n).$$

Шаг 2. Извлечение операнда (конкатенанта) $a_\#$ из регистра хранения.

Шаг 3. Извлечение операнда $b_\#$ из регистра хранения.

Шаг 4. Суммирование конкатенантов: $c_\#=a_\#+b_\#$.

Шаг 5. Помещение результата $c_\#$ в регистр хранения.

Шаг 6. Деконкатенация:

$$c_\#=(c_1\oplus c_2\oplus c_3\oplus...\oplus c_n)\to C:\{c_1,c_2,c_3,...,c_n\}.$$

Операции конкатенации и деконкатенации выполняются над целыми положительными числами в двоичном представлении. При обработке числа располагаются в центральном процессоре в отдельных регистрах – линейках бинарных ячеек памяти. Отсюда применяемый далее термин – *регистровое представление* (РгП). В ЛП представлении результат конкатенации есть целое положительное число, помещаемое в едином регистре, – РгП, которое интерпретируется как состоящее из сегментов согласно длине конкатенантов. При деконкатенации сегменты извлекаются из РгП. Таким образом, конкатенация есть операция уплотнения представления компонентов векторов (1) в виде сегментов в РгП $a_\#$ и $b_\#$ (2). Содержимые РгП $a_\#$ и $b_\#$ обрабатываются целиком, как числа. Деконкатенация (3) – обратная операция: содержимое i-го

сегмента c_i извлекается и помещается в индивидуальный регистр.

В результате, согласно ЛП схеме, операция производится над РгП, а результат $C:\{c_1,c_2,c_3,...,c_i,...,c_n\}$ выглядит так, как если бы операции производились над парами операндов раздельно.

Вычислительные блоки операций Шаги 2 – 5 в последовательной и ЛП схемах совпадают, но в последовательной схеме блок выполняется n -кратно, в ЛП схеме – однократно. Этим обеспечивается выигрыш в производительности. Конкатенация и деконкатенация связаны с дополнительным расходованием ресурсов ВС.

Однако, если в системе реализуются сложные многошаговые вычисления, причём РгП составляются из исходных данных только в самом начале вычислительного блока, а результаты извлекаются в самом конце, – затраты ресурсов на конкатенацию-деконкатенацию могут быть пренебрежимо малы по сравнению с общими затратами ресурсов ВС. Эффективность ЛП схемы растёт пропорционально числу конкатенантов n [2, 3].

Выигрыш в производительности и точность системы конкурируют между собой. При фиксированной разрядности, производительность ЛП системы тем выше, чем больше число сегментов. Для повышения производительности целесообразно сократить разрядность сегментов до минимума. При этом уменьшается число градаций значений обрабатываемых величин.

При проектировании ЛП систем должны приниматься компромиссные решения по соотношению производительности вычислений и точности результатов.

4. Локально-параллельное представление КА

Рассматриваем 32-разрядный процессор Фон-Неймановского типа, реализующий стандартный набор арифметических и логических операций, включая поразрядную (побитовую) логику и регистровые сдвиги. При ЛП обработке предполагается хранение нескольких элементов информации в виде раздельных не пересекающихся сегментов РгП. В случае ЛП представления КА логично разместить ячейки КА в отдельных сегментах. В обычной записи для хранения матрицы ячеек КА требуется двумерный массив. При размере поля M строк на N столбцов требуется MN индивидуально адресуемых элементов хранения информации, что составляет 32MN бит. В ЛП представлении индивидуально адресуемыми элементами хранения становятся РгП. В случае БКА в РгП может быть до 32 однобитовых сегментов. Строка поля БКА может быть помещена в $[M/32]+1$ РгП (квадратные скобки обозначают целую часть от деления). В результате, для представления всего БКА необходим массив из $([M/32]+1)N \approx MN/32$ РгП. Соответственно, если ячейка КА имеет n состояний, для её представления требуется k бит с соблюдением соотношения $2^k \geq n$, то есть $k \geq \log_2 n$, то есть $k=[\log_2 n]+1$ бит.

Таким образом, в 32-разрядном ВУ для представления строки элементов КА требуется $[Mk/32]+1$, а для представления всего массива – $MNk/32$ индивидуально адресуемых элементов хранения информации.

Алгоритм функционирования КА предполагает рассмотрение окружения каждой клетки. Для БКА с 8-ю соседями рис. 1 в при ЛП вычислениях для j -й строки (j -го РгП a_j) должны суммироваться 8 штук РгП:

$$(a_{j-1} << p)+(a_{j-1})+(a_{j1} >> p)+(a_j << p)+$$

$$+(a_j >> p)+(a_{j+1} << p)+(a_{j+1})+(a_{j+1} >> p)$$

где « $<<p$ » и « $>>p>>$ » означают процедуры регистрового сдвига на p бит (длину сегмента) в сторону старших или младших разрядов, соответственно. Для БКА $p=1$, а для представления результата по каждому сегменту требуется 4 бита, поскольку имеется 9 градаций возможных значений сегментной суммы: (0, 1, 2,..., 7, 8). Чтобы соседние сегменты не взаимодействовали, должно быть изначально зарезервировано достаточно места для размещения результата. Типовым подходом при подобных ЛП вычислениях является прореживание РгП [2, 3] с использованием битовых масок. Массив РгП преобразуется в 4 массива с непересекающимися наборами сегментов с номерами сегментов: (1, 5, 9, 13,...), (2, 6, 10, 14,...), (3, 7, 11, 15,...), (4, 8, 12, 16,...). После прореживания процедура выполняется последовательно для каждого из четырёх наборов с последующим сведением результатов в единый РгП. Эффективность ЛП алгоритма при этом снижается четырёхкратно. Прореживание системой масок и соответствующее кратное снижение эффективности ЛП обработки снимается полностью с применением *стекового* ЛП алгоритма.

5. Стековый локально-параллельный алгоритм

Пусть имеются 8-битные числа

$A_1 = 188_{10} = 10111100_2$,

$A_2 = 113_{10} = 01110001_2$,

$A_3 = 235_{10} = 11101011_2$,

$A_4 = 25_{10} = 00011001_2$, случайным образом выбранные в интервале от 0 до 255. Нижний индекс обозначает основание системы счисления представления числа. В двоичном представлении для наглядности лидирующие нули сохранены. Интерпретируем двоичные позиции как сегменты в ЛП представлении. Таким образом, имеется четыре 8-сегментных РгП. Размер каждого сегмента – 1 бит. Стеками являются столбцы из сегментов.

После первоначального заполнения единичные сегменты «проваливаются» максимально вниз, а нулевые, соответственно, «всплывают» вверх согласно алгоритму сортировки «методом пузырька», который применяется к РгП посегментно, локально-параллельно ко всем сегментам.

С набором чисел $A_1 – A_4$ это выглядит следующим образом:

	10111100		10111100	}→
	01110001		01110001	
	11101011	}→{	00001001	
	00011001		11111011	
	10111100	}→{	00000000	
→{	00000001		10111101	
	01111001		01111001	}→
	11111011		11111011	
	00000000		00000000	
	10111101	}→{	00111001	
→{	01111001		11111101	}→
	11111011		11111011	
			00000000	
			00111001	
		→{	11111001	
			11111111	

Данное выражение изображает в совокупности последовательную смену 7 состояний набора РгП $A_1 - A_4$. Первое состояние соответствует исходному, последнее –полную отсортированность: все единицы максимально смещены вниз. Фигурные скобки и стрелки переходов показывают, какие из РгП сопоставляются на каждом шаге алгоритма сортировки. Процесс включает два вложенных цикла. Внешний цикл обеспечивает $(n-1)$-кратное (n – число РгП, в примере $n=4$) повторение внутреннего цикла с сокращением его объёма от $(n-1)$ (три первые смены состояний) до 1 (последняя смена состояний). Первый (соответствующий младшим разрядам чисел $A_1 - A_4$) стек не претерпевал изменений: расстановка единиц в нём не требует сортировки. 4-е и 5-е состояния – тождественны: в этом цикле не происходит вертикального перемещения единиц. Показателен 3-й справа разряд: тремя последовательными обменами единица спускается из A_1 в A_4.

Бинарная процедура сопоставления, обозначенная значком «→», включает арифметические операции «–» и «+», а так же поразрядные логические операции «END» и «XOR».

```
Local_Parallel_Stack()
1 M ← A “END” ( B “XOR” E0 )
2 A ← A – M
3 B ← B + M
```

Входными данными являются A и B, выходными – изменённые значения A и B.

В процедуре используются константа $E_0 = 11111...11$ («ЛП единица») и промежуточная переменная M. После завершения ЛП сортировки стеков, имеется исходный материал для формирования результата – нового набора значений соответствующего фрагмента КА.

Так, если интерпретировать пример (1) как работу БКА с четырьмя соседями (рис. 1 а), разность

$$(A_2 - A_1) = (00111001)_2 - (00000000)_2 = (00111001)_2$$

будет соответствовать правилу наличия трёх «живых» ячеек-соседей. Единицами отмечены те позиции (ячейки КА), которые «выжили» в новом поколении КА, поскольку в текущем поколении они были окружены тремя единичными соседями.

6. Обсуждение результатов

Работа БКА с применением предложенного стекового алгоритма смоделирована в варианте с 8 соседями (рис. 1 а) на 32-разрядном процессоре, на языке Python. Размер поля модели 32×32 ячейки. Моделированием подтверждена работоспособность стекового алгоритма. В процессе разработки выявлены следующие особенности алгоритма. Позитивной стороной является универсальность подхода применительно к различным вариантам правил поведения КА: независимость от объёма и конфигурации окружения (соседства), простота выбора порогового уровня при определении «выживающих» ячеек. Имеется существенное (в 3 – 4 раза) сокращение времени обработки по сравнению с альтернативным ЛП вариантом – с прореживанием РгП. Ограничением является представление строки КА в виде одно РгП.

Перспективным развитием стекового алгоритма явится представление строки КА несколькими РгП, а так же распространение на более широкий класс КА, в частности с введением нечёткого описания для учёта случайных влияющих факторов в моделируемой системе.

Выводы

Локально-параллельное представление информации повышает эффективность работы программы клеточного автомата и существенно увеличивает размеры модели. Предложен и продемонстрирован на тестовых примерах локально-параллельный стековый алгоритм бинарного клеточного автомата, инвариантный относительно выбора уровней «выживающих» ячеек.

Литература

1. Бенаддия Абдельлатиф. Перспективы реализации келлюлярных авторматов на локально-параллельных алгоритмах // Материалы международной научно-практической конференции студентов и аспирантов ”Информационные технологии в экономике и образовании”. – М.: Российский университет кооперации, 2008. – С. 45 – 48.

2. Михаль О.Ф. Моделирование распределенных информационно-управляющих систем средствами локально-параллельных алгоритмов обработки нечеткой информации // Проблемы бионики. Всеукраинский межведомственный научно-технический сборник. Харьков: ХНУРЭ. 2001. Вып. 54. С. 28-34.
3. Михаль О.Ф., Руденко О.Г. Принципы организации систем нечеткого регулирования на однородных локально-параллельных алгоритмах // Управляющие системы и машины. 2001. № 3. С. 3-10.
4. Люггер Д.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем М.: Изд. дом «Вильямс», 2005. – 864 с.

Процедуру сортування проаналізовано на комбінаторному рівні для 4-елементних послідовностей. Подано алгоритмічне забезпечення локально-паралельного сортування. В ході моделювання мовою Python з'ясовано, що локально-паралельний алгоритм максимально эфективний щодо сортування виборок в межах розрядності процесора

Ключові слова: локальна паралельність, сортування даних

Процедура сортировки проанализирована на комбинаторном уровне на примере 4-элементных числовых последовательностей. Описана работа алгоритма локально-параллельной сортировки. Моделированием на языке Python показано, что алгоритм эффективен применительно к малым выборкам

Ключевые слова: локальная параллельность, сортировка данных

The Procedures of the sorting is analysed on combinatorial level for 4-element sequences. Algorithmic provision of local-parallel sorting is presented. In the course of modeling in language Python is revealled that local-parallel algorithm is greatly efficient with reference to sorting the samples within processor register size

Key words: local parallelity, data sorting

УДК 519.87

ЛОКАЛЬНО-ПАРАЛЛЕЛЬНАЯ СОРТИРОВКА МАЛЫХ НАБОРОВ ДАННЫХ

Мохамад Али
Аспирант*

О.Ф. Михаль
Доктор технических наук, доцент, профессор*
*Кафедра Электронно-вычислительных машин
Харьковский национальный университет радиоэлектроники
пр. Ленина, 14, г. Харьков, 61166
Контактный тел. (057) 40-93-54
E-mail: fuzzy16@pisem.net

1. Введение

Сортировка данных – классическая задача, сочетающая в себе прозрачность общей концепции и многовариантность решений. Алгоритмы сортировки подробно изучены для *вычислительных систем* (ВС) последовательного действия. Параллельные ВС до недавнего времени были исключительно многопроцессорными и разрабатывались для специальных приложений. Задачи параллельной сортировки рассматривалась в них преимущественно в контексте общих принципов распараллеливания алгоритмов. С распространением многоядерных процессоров и сетевых вычислительных структур [1]: приобрела актуальность сортировка ограниченных малых наборов данных [2, 3]. Эта задача эффективно решается с использованием, *локально-параллельных* (ЛП) алгоритмов обработки информации [4, 5].

В настоящей работе развиваются рассмотренные ранее [1-3] принципы сортировки на ЛП алгоритмах, проводится детальное сравнение с последовательными алгоритмическими процедурами и формулируются направления использования выявленных закономерностей в прикладных задачах.

2. Основные определения

Будем рассматривать набор данных A – множество, состоящее из элементов: $A:\{a_1,a_2,a_3,...,a_i,...,a_n\}$, где $a_i \in A$; $i \in (1,2,...,n)$, – объект произвольной природы: число, фрагмент текста, запись в БД и др. Элементы множества A могут быть самими объектами набора данных, либо указателями на них, позволяющими взаимно-однозначно соотносить $a_i \in A$ и соответствующий объект. Термин «ограниченный» предполагает конечное число n элементов, неизменное в