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

процессе сортировки; термин «малый» - количество элементов при котором не требуется использование видов (типов) памяти, существенно замедляющих информационный обмен в процессе выполнения сортировки.

Отношение порядка (наличие *упорядоченности*) на множестве A предполагает указание для *некоторых* элементов $a_i, a_j \in A$; $i,j \in (1,2,3,...,n)$; $i \neq j$, одного из следующих соотношений: $a_i \prec a_j$, $a_i \succ a_j$. Символы «$\prec$» и «$\succ$» есть отношения следования: «предшествует» и «следует за». В частности они могут интерпретироваться как арифметические соотношения между числами – бинарные отношения «меньше» (<) и «больше» (>). Множество A является *упорядоченным*, если отношение порядка определено для *любых* $a_i, a_j \in A$. Множество A является *частично упорядоченным*, если отношение порядка определено для *некоторых* пар $a_i, a_j \in A$.

Будем различать множество A, как совокупность элементов, и *последовательность* элементов множества, как выборку элементов множества с учётом порядка размещения. Для последовательности, включающей все элементы A, допустимо упорядочение *по возрастанию* $a_1 \le a_2 \le a_3 \le ... \le a_i \le ... \le a_n$, либо *по убыванию* $a_1 \ge a_2 \ge a_3 \ge ... \ge a_i \ge ... \ge a_n$. При этом для любой пары элементов справедливо $a_i \le a_j$ ($a_i \ge a_j$), где $i,j \in (1,2,3,...,n)$, $i < j$. Если хотя бы для одной пары условие $a_i \le a_j$ ($a_i \ge a_j$) не выполняется, последовательность является *не упорядоченной по возрастанию (убыванию)*. Процесс упорядочения – переход от одной расстановки к другой иллюстрируется графом рис. 1 для множества из четырёх элементов.

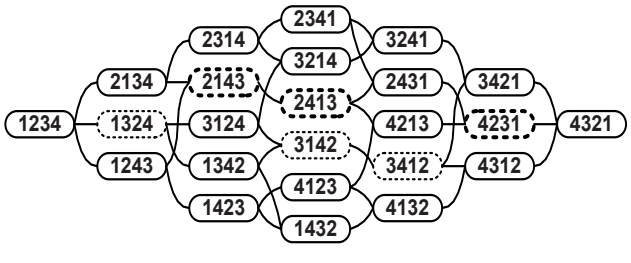

Рис. 1. Структуры частично упорядоченных множеств перестановок четырёхэлементной последовательности

Вводится *мера разупорядоченности*, как число перестановок соседних элементов, которое необходимо произвести чтобы прийти к упорядоченной последовательности элементов. Может быть показано, что множества, упорядоченные по возрастанию и убыванию максимально взаимно разупорядочены. Для данного множества число перестановок в парах соседствующих элементов является максимальным при переупорядочении множества из упорядоченности по возрастанию в упорядоченность по убыванию.

При рассмотрении цепочек перестановок необходимо исключить циклы, ограничившись только продуктивными перестановками. Графы рис. 1 а, б позволяют проследить различные варианты таких цепочек, поскольку охватывают все возможные варианты связей по перестановкам в парах соседствующих элементов. Так, для 4-элементной последовательности (рис. 1 б) одна из цепочек преобразования $(1,2,3,4) \to (4,3,2,1)$ имеет вид:

$$(\overleftrightarrow{1,2},3,4) \to (2,1,\overleftrightarrow{3,4}) \to (2,\overleftrightarrow{1,4},3) \to$$
$$\to (\overleftrightarrow{2,4},1,3) \to (4,2,\overleftrightarrow{1,3}) \to (4,\overleftrightarrow{2,3},1) \to (4,3,2,1)$$

Две соседние последовательности, разделённые знаком «$\to$», различаются перестановкой местами одной (обозначенной «$\leftrightarrow$») пары двух соседних элементов (цифр). Процедура сортировки – последовательная, поэтому соседствующие пары переставляются по одной. В результате, $(1,2,3,4) \to (4,3,2,1)$ реализуется за 6 шагов.

В отличие от последовательной процедуры сортировки, ЛП процедура поддерживает обмен местами сразу нескольких (всех имеющихся) пар соседствующих элементов последовательности. Вследствие этого число переходов «$\to$» может быть существенно сокращено.

3. Локальная параллельность

Принцип ЛП обработки информации, может быть проиллюстрирован вычислительным примером. Имеется n выражений $c_i = a_i + b_i$; $i = 1,2,...,n$. Значения a_i и b_i заданы, требуется найти c_i. В последовательном варианте задача разрешима за 4n шагов: загрузить a_i, загрузить b_i, произвести суммирование, выгрузить результат c_i. Если a_i, b_i и c_i положительные и имеют ограниченную разрядность, исходные данные могут быть конкатенированы: $A = a_1 \oplus a_2 \oplus ... \oplus a_i \oplus ... \oplus a_n$; $B = b_1 \oplus b_2 \oplus ... \oplus b_i \oplus ... \oplus b_n$.

Результат $C = A + B$ так же может быть интерпретирован как конкатенация. При этом весь набор значений c_i может быть получен за 4 шага. n-кратный выигрыш достигнут без учёта «окаймляющих» операций конкатенации и деконкатенации. Если в реальном случае вычислительный блок сложный и включает последовательное применение нескольких операций без промежуточных конкатенаций-деконкатенаций, – выигрыш может быть значительным. Выигрыш растёт с ростом разрядности процессора и с сокращением размеров конкатенируемых сегментов. При этом снижается точность представления данных (система «загрубляется»), что в ряде конкретных приложений бывает допустимо.

Проиллюстрируем работу алгоритма ЛП сортировки. Имеется множество элементов $A: \{a_1, a_2, ..., a_n\}$, на котором определено отношение порядка $a_1 > a_2 > ... > a_n$. Из элементов множества A составлена *разупорядоченная* последовательность $B: \{b_n, b_{n-1}, ..., b_2, b_1\}$; $b_j = a_j$; $i,j \in (1,2,3,...,n)$; $i \neq j$.

Максимально разупорядоченный случай соответствует обратной упорядоченности: $B: \{a_n, a_{n-1}, ..., a_2, a_1\}$. Требуется привести последовательность к правильной исходной упорядоченности: $B \to A$.

Производим конкатенацию элементов b_i согласно их расположению в последовательности. В вербальном описании ЛП процедура сортировки включает 4 шага.

Шаг 1. Скопировать текущее B для последующего сравнения: $B^1 = B$.

Шаг 2. Сравнить попарно нечётные элементы B с чётными, стоящими слева от них. В каждой сравнивае-

мой паре больший из элементов поставить на нечётное место, а меньший – на чётное.

Шаг 3. Сравнить попарно чётные элементы B с нечётными, стоящими слева от них. В каждой сравниваемой паре больший из элементов поставить на чётное место, а меньший – на нечётное.

Шаг 4. Сравнить B и B^1. Если $B^1 = B$, ЛП процедура сортировки закончена. Результат содержится в B. Если $B^1 \neq B$ – перейти к Шагу.1.

Основные черты ЛП сортировки могут быть прослежены на примере.

		(а)		(б)
1)		$\overleftrightarrow{(1,2,3,4)}$		$\overleftrightarrow{(1,2,3,4)}$
2)		$\overleftrightarrow{(2,1,4,3)}$		$\overleftrightarrow{(1,3,2,4)}$
3)		$\overleftrightarrow{(2,4,1,3)}$		$\overleftrightarrow{(3,1,4,2)}$
4)		$\overleftrightarrow{(4,2,3,1)}$		$\overleftrightarrow{(3,4,1,2)}$
		(4,3,2,1)		(4,3,2,1)

В столбце, обозначенном (а), представлены 4 последовательных цикла ЛП сортировки начиная со строки обменов «чётный–нечётный»; в столбце (б) – начиная со строки обменов «нечётный–чётный». При изменении порядка чередования обменов общее число циклов сортировки не меняется и результат – нижняя строка – одинаков. Пример соответствует графу рис. 1. ЛП алгоритм преобразования $(1,2,3,4) \rightarrow (4,3,2,1)$ в варианте, обозначенном (а), соответствует вершинам графа, выделенным жирным пунктиров; в варианте, обозначенном (б), - вершинам, выделенным мелким пунктиров. Сопоставление с графом рис. 1 показывает сокращение числа проходимых вершин, за счёт чего достигается выигрыш в производительности ЛП сортировки по сравнению с традиционной последовательной.

Выводы

Локально-параллельное представление информации обеспечивает повышенную эффективность работы программного обеспечения. Принципы локально-параллельной обработки применимы к задачам сортировки наборов данных. Процедуры сортировки рассмотрены на комбинаторном уровне для 4-элементных последовательностей. Разработано алгоритмическое обеспечение локально-параллельной сортировки. В ходе моделирования, проведенного на языке Python, выявлено, что локально-параллельный алгоритм максимально эффективен применительно к сортировке малых (в пределах разрядности процессора) выборок.

Литература

1. Мохамад Али, Михаль О.Ф. Перспективы реализации локально-параллельных вычислений на многоядерных процессорах // Радиоэлектронные и компьютерные системы. – 2008. - № 6 (33). – с. 234-237.
2. Мохамад Али. Локально-параллельная сортировка данных с ограниченной разрядностью // Материалы международной научно-практической конференции студентов и аспирантов "Информационные технологии в экономике и образовании". – М.: Российский университет кооперации, 2008. – с. 48-52.
3. Мохамед Али, Михаль О.Ф. Локально-параллельная сортировка со встречной чередующейся упорядоченностью данных. Материалы 13-го международного молодёжного форума "Радиоэлектроника и молодёжь в XXI веке" Ч.2. – Харьков: ХНУРЭ, 2009, с. 209.
4. Михаль О.Ф. Локально-параллельные алгоритмы определения степени включения и степени равенства нечетких множеств // "Проблемы бионики", Вып. 55, Харьков, 2001, с. 80-90.
5. 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.