УДК 519.87

Мохамад Али, О.Ф. Михаль
ХНУРЕ, г. Харьков, Украина, fuzzy16@pisem.net

ЛОКАЛЬНО-ПАРАЛЛЕЛЬНАЯ СОРТИРОВКА ОГРАНИЧЕННЫХ МАЛЫХ НАБОРОВ ДАННЫХ

Рассмотрены принципы локально-параллельного представления информации применительно к задаче сортировки данных. Процедура сортировки проанализирована на комбинаторном уровне на примере 3-, 4- и 8-элементных числовых последовательностей. Описан принцип работы алгоритма локально-параллельной сортировки. Моделированием на языке Python показано, что локально-параллельный алгоритм максимально эффективен применительно к малым выборкам, суммарным размером в пределах разрядности процессора.

ЛОКАЛЬНАЯ ПАРАЛЛЕЛЬНОСТЬ, СОРТИРОВКА ДАННЫХ

Введение

Сортировка данных – классическая задача теории алгоритмов, сочетающая в себе прозрачность общей концепции и простоту постановки с многовариантностью решений в зависимости от дополнительных условий, налагаемых в связи с конкретными особенностями обрабатываемой информации. Алгоритмы сортировки подробно изучены для *вычислительных систем* (ВС) последовательного действия. Параллельные ВС до недавнего времени были исключительно многопроцессорными, дорогими и громоздкими, поэтому они разрабатывались преимущественно для специальных применений, а задачи сортировки рассматривалась в них главным образом в контексте реализации общих принципов распараллеливания алгоритмов применительно к большим массивам данных. Ситуация изменилась в последние десятилетия с появлением многоядерных процессоров и широким распространением (массовым использованием) сетевых вычислительных структур [1]: приобрела актуальность обработка *ограниченных малых наборов данных* (ОМНД). Задача сортировки применительно к ОМНД [2, 3] эффективно решается с использованием алгоритмов, реализующих принципы *локальной-параллельной* (ЛП) обработки информации [4, 5]. Область применения ОМНД – задачи принятия решений при детерминированном наборе вариантов. К задачам этого типа относится, в частности, значительная часть тематики *искусственного интеллекта* (ИИ). Так, одна из парадигм ИИ – исследование и воспроизведение структур и функций человеческого интеллекта. Человек же в каждый конкретный момент своей практической деятельности эффективно оперирует *ограниченным малым* числом понятий [6]. Эффективность (скорость принятия решений) зачастую определяется правильной последовательностью рассмотрения вариантов, а реализация эффективности сводится к сортировке ОМНД. ЛП алгоритмы обеспечивают повышение скорости обработки без задействования дополнительных аппаратных ресурсов. Поэтому они целесообразны как структурный элемент при реализации обработки ОМНД. Сказанным определяется актуальность и практическая ценность данного направления.

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.

1. Гносеологический аспект

Рассматриваемый предмет имеет глубокие корни в природе человеческого интеллекта. Эффективная человеческая деятельность характеризуется, в частности:

– ограниченным (не слишком большим) количеством одновременно выполняемых (отслеживаемых) заданий (процессов, объектов, дел, работ);

– правильным определением приоритетности (важности, последовательности) выполнения заданий.

Деятельность с избыточным числом выполняемых заданий, или с плохо организованной приоритетностью, как правило, малоэффективна. Данные аспекты базируются на ограниченности ресурсов реализации человеческого интеллекта. Факт, известный в психологии [6]: средний человек способен эффективно взаимодействовать (удерживать в памяти, отслеживать изменения, мысленно оперировать) одновременно не более, чем с 5–7 объектами (предметами, понятиями, блоками информации). Данная закономерность обусловлена ограниченным числом информационных каналов взаимодействия с объектами, а также ограниченными ресурсами параллельной обработки информации (мыслительной деятельности) по планированию выполнения действий с надлежащей приоритетностью.

Данная особенность человеческого интеллекта имеет отношение к компьютерной технике и информационным технологиям в следующем аспекте. Практика развития компьютерных си-

стем (hardware и software, включая ИИ) такова, что явно или неявно — мерилом прогресса в этой области является степень соответствия человеческому интеллекту. Данное положение соответствует, в частности, известному тесту Тюринга [7]. Исходя из этого, при разработке ВС целесообразно изучать «прототип» (hardware – человеческий мозг и software – человеческую психику) и следовать тенденциям, предначертанным в этой «уже разработанной» «эталонной» и «априорно совершенной» системе, являющейся «технической базой», на которой реализован человеческий интеллект. Известно, что «прототип» «аппаратно реализован» на естественных (живых) *нейронных сетях* (НС). Основной особенностью обработки информации на НС и ключевым преимуществом НС является *параллельность*. Параллельность реализуема в настоящее время преимущественно на многопроцессорных и многоядерных вычислительных структурах, а также в виде распределённых (локально- и глобально-сетевых) ВС. Аппаратная база искусственных НС развита в настоящее время в существенно меньшей степени. Исходя из этого, наряду с развитием аппаратных искусственных НС, целесообразны исследования и разработки в области алгоритмов, поддерживающих параллельную обработку информации на имеющихся наиболее развитых на настоящий момент типах аппаратного обеспечения. Перспективны ЛП алгоритмы, которыми распараллеливание обработки информации может быть реализовано на ВС последовательного действия (работающих в рамках концепции Фон-Неймана), то есть на компьютерах общего назначения. Таким образом, с использованием принципа ЛП параллельные вычисления претерпевают некоторое качественное изменение. Соответственно, требуют пересмотра в рамках концепции ЛП также и классические процедуры сортировки данных [8].

2. Основные определения

Будем рассматривать набор данных A как множество, состоящее из элементов:

$$A: (a_1, a_2, a_3,..., a_{i-1}, a_i, a_{i+1},..., a_{n-2}, a_{n-1}, a_n). \quad (1)$$

Каждое из данных, соответствующее $a_i, \in A$, $i \in (1, 2,...., n)$, – объект произвольной природы: число, фрагмент текста, запись в БД, картинка, фрейм и др. При этом элементы множества A могут быть самими объектами набора данных, либо указателями на них, позволяющими взаимнооднозначно соотносить $a_i, \in A$ и соответствующий объект. В первом случае операции, производимые над объектами множества A, связаны с перемещением самих объектов (данных) в памяти, во втором – объекты непосредственно не перемещаются, а операции над ними сводятся к работе с указателями. С точки зрения рассматриваемых далее алгоритмов ЛП сортировки данное различие не существенно.

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. Разумеется, интерпретация термина «малый» может быть уточнена с учётом конкретной архитектуры ВС.

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» (>). При этом интерпретацией арифметического соотношения $a_i = a_j$ может быть «произвольное следование»: справедливо любое из отношений следования $a_i \prec a_j$, или $a_i \succ a_j$.

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 только для *некоторых* из пар $a_i, a_j \in A$. Элементы, для которых отношения порядка не определены, являются *несоизмеримыми*. Множество является *неупорядоченным*, если ни для одной пары элементов не определено отношение порядка.

Будем различать *множество* как совокупность элементов (1) и *последовательность* элементов множества как выборку некоторых элементов множества с учётом порядка их размещения. Для *упорядоченной* последовательности, включающей все элементы множества A (1), в арифметической интерпретации с учётом замечания о соотношении $a_i = a_j$ справедливо одно из выражений:

$$a_1 \le a_2 \le a_3 \le ... \le a_{i-1} \le a_i \le a_{i+1} \le ... \le a_{n-2} \le a_{n-1} \le a_n; \quad (2)$$

$$a_1 \ge a_2 \ge a_3 \ge ... \ge a_{i-1} \ge a_i \ge a_{i+1} \ge ... \ge a_{n-2} \ge a_{n-1} \ge a_n. \quad (3)$$

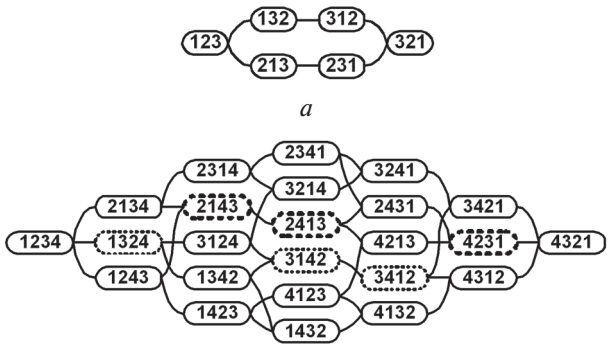
Последовательность является упорядоченной *по возрастанию* (2) (соответственно, *по убыванию* (3)), если для любой пары её элементов справедливо $a_i \le a_j$ (соответственно, $a_i \ge a_j$), где $i, j \in (1, 2, 3,..., n)$, $i < j$. Если хотя бы для одной пары элементов условие $a_i \le a_j$ ($a_i \ge a_j$) не выполняется, последовательность *не является упорядоченной* по возрастанию (убыванию), то есть является *неупорядоченной по возрастанию* (*неупорядоченной по убыванию*). Таким образом, последовательность, упорядоченная по возрастанию, является неупорядоченной по убыванию; последовательность, упорядоченная по

убыванию, является неупорядоченной по возрастанию. То есть упорядоченности по возрастанию и убыванию взаимно-обратно являются неупорядоченностями. Данные и подобные представления (упорядочение, разупорядочение или переупорядочение) продуктивны применительно к задачам сортировки ограниченных малых наборов элементов. Множество может быть изначально разупорядоченным. Процесс упорядочения – переход от одной последовательности (расстановки) элементов к другой. Может быть введена *мера разупорядоченности* множества как число перестановок соседних элементов, которое необходимо произвести чтобы прийти к упорядоченной последовательности элементов.

3. Упорядочение элементов

С учётом введенного понятия меры разупорядоченности, может быть показано, что множества, упорядоченные по возрастанию и убыванию, взаимно максимально разупорядочены по отношению друг к другу. В частности, для данного множества число перестановок в парах соседствующих элементов является максимальным при переупорядочении множества из упорядоченности по возрастанию в упорядоченность по убыванию, или наоборот: (2) → (3) или (3) → (2)

Графы, представленные на рис. 1 а, б иллюстрируют сказанное для множеств из трёх и четырёх *разных* элементов. Термин «разный» понимается в значении $a_i, \neq a_j$; $a_i, a_j \in A$; $i, j \in (1, 2, 3,..., n)$; $i \neq j$.



а

б

Рис. 1. Структуры частично упорядоченных множеств перестановок трёхэлементной (*а*) и четырёхэлементной (*б*) последовательностей.

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) → (4, 3, 2, 1) имеет следующий вид:

$$
\begin{array}{ccccccc}
 & & \lceil\rceil & & \lceil\rceil & & \\
 & & (1,2,3,4) & \rightarrow & (2,1,3,4) & \rightarrow & \\
 & & \lceil\rceil & & \lceil\rceil & & \\
\rightarrow & & (2,1,4,3) & \rightarrow & (2,4,1,3) & \rightarrow & \\
 & & \lceil\rceil & & \lceil\rceil & & \\
\rightarrow & & (4,2,1,3) & \rightarrow & (4,2,3,1) & \rightarrow & \\
\rightarrow & & (4,3,2,1) & & & &
\end{array}
\qquad (4)
$$

В (4) любые две соседние последовательности, разделённые знаком « → », различаются перестановкой местами одной (обозначенной) пары двух соседних элементов (цифр). Другая соседствующая пара, если она имеется и может быть переставлена, тем не менее остаётся незадействованной, поскольку процедура сортировки – последовательная. В результате, (1, 2, 3, 4) → (4, 3, 2, 1) реализуется за 6 шагов. Эта ситуация имеет место не только для (4), но и для любых вариантов цепочек перестановки элементов, что может быть прослежено непосредственно по графу (рис. 1 б), на котором имеется 7 вертикальных столбцов расположения 4-элементных последовательностей, различающихся перестановкой одной пары элементов.

В отличие от последовательной процедуры сортировки, ЛП процедура поддерживает обмен местами сразу нескольких (всех имеющихся) пар соседствующих элементов последовательности. Вследствие этого, как будет показано далее, число переходов « → » в процедуре сортировки может быть существенно сокращено.

4. Локальная параллельность

ЛП обработка информации на процессорах общего назначения допускает широкий круг приложений. Принцип ЛП обработки информации может быть проиллюстрирован на следующем вычислительным примером. Имеется n выражений $c_i = a_i + b_i$; $i = 1, 2,..., n$. Значения a_i и b_i заданы. Требуется найти значения c_i. Если используется ВС с одним процессором, задача разрешима за $4n$ шагов: загрузить a_i, загрузить b_i, произвести суммирование, выгрузить результат c_i. Если a_i и b_i положительные, имеют ограниченное число разрядов и если в результате сложении разрядность не нарушается (числа a_i, b_i и c_i имеют одинаковую разрядность), исходные данные могут быть представлены в виде:

$$A = a_1 \oplus a_2 \oplus ... \oplus a_i \oplus ... \oplus a_n;$$
$$B = b_1 \oplus b_2 \oplus ... \oplus b_i \oplus ... \oplus b_n, \qquad (5)$$

где символом $\oplus$ обозначена *конкатенация*. Если разрядность результата превышает разрядность слагаемых для переноса избыточного разряда при суммировании в формате представления данных требуется предусмотреть разделительные нули. В обоих случаях результат $C = A + B$ может быть интерпретирован как конкатенация аналогично (5), в которой сегменты – числа c_i – соответствуют

по формату a_i и b_i. При этом *весь* набор значений c_i может быть получен за 4 шага. Здесь n-кратный выигрыш достигнут без учёта «окаймляющих» операций конкатенации-деконкатенации, предназначенных для формирования A и B и извлечения результатов c_i из C. Если в реальном случае вычислительный блок достаточно сложный и включает последовательное применение нескольких разнообразных 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.

Рост разрядности процессоров является долговременной устойчивой тенденцией развития процессорной техники. Известный закон Мура об удвоении числа транзисторов на кристалле на практике означает удвоение разрядности регистров процессора. Разрядность регистров, как известно, это объём непосредственно адресуемой оперативной памяти, т.е. допустимая размерность задач, разрешимых в данном ВС за приемлемое время. Поэтому разрядность вероятно, будет наращиваться и в дальнейшем, поскольку при этом стимулируется появление новых вычислительных задач, которые в свою очередь стимулируют рост требований к техническим характеристикам вновь разрабатываемых ВС. Данная «обратная связь» исправно работает на протяжении всей истории развития ВС и нет причин предполагать отход от данной тенденции. Соответственно, есть основания прогнозировать рост эффективности (производительности) применения ЛП алгоритмов за счёт увеличения числа конкатенируемых сегментов.

5. ЛП сортировка

Проиллюстрируем работу алгоритма ЛП сортировки ОМНД следующим образом. Пусть имеется множество элементов, на котором определено отношение порядка:

$$A: \{a_1, a_2, a_3, a_4,..., a_n\};\quad a_1 > a_2 > a_3 > a_4 >...> a_n. \quad (6)$$

Пусть из элементов множества A составлена *разупорядоченная* последовательность:

$$B: (b_n, b_{(n-1)},..., b_2, b_1),\ b_i = a_j,\ i,j \in \{1, 2,..., n\},$$
$$\exists\, (i, j): i \neq j. \quad (7)$$

В частности, в максимально разупорядоченном случае (7) может быть обратно упорядоченным набором элементов из (6): B: (a_n a_{n-1}, a_{n-2}, a_{n-3}, a_{n-4},..., a_3, a_2, a_1).

Требуется упорядочить последовательность (7) в порядке убывания величины элементов от младшего (правого) к старшему (левому), т.е. привести к упорядочению согласно (6).

Производим конкатенацию (5) элементов b_i согласно их расположению в последовательности (7). Далее в вербальном описании ЛП процедура сортировки включает следующие 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 сравниваемой 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.

Процедура иллюстрируется рисунком 2. Разделение сегментов на чётные и нечётные организовано посредством пары регистров $R1$ и $R2$. Заштрихованы фрагменты регистров, в которых расположены информационные сегменты. Разделение (прореживание) конкатенации вида (4) на четный ($R2$) и нечётный ($R1$) регистры программно реализуется применением процедуры побитового «И» с использованием пары битовых полумасок вида:

$$E1:\{...(11...1)(00...0)(11...1)(00...0)(11...1)\};$$
$$E2:\{...(00...0)(11...1)(00...0)(11...1)(00...0)\}. \quad (8)$$

Сегменты, соответствующие конкатенированным данным в (5), выделены в (8) круглыми скобками.

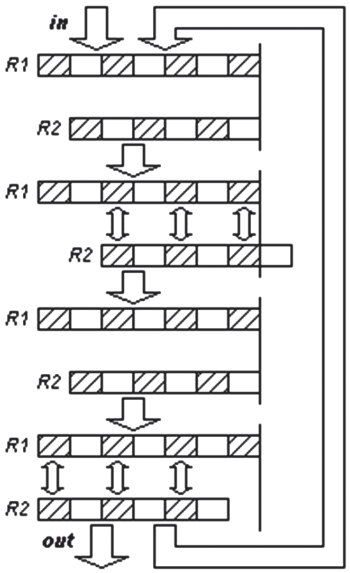


Рис. 2. Работа ЛП алгоритма сортировки

Регистры последовательно совмещаются для сопоставления (Шаги 2 и 3) применением регистровых сдвигов на число бит, равное длине сегмента. Посегментное сопоставление показано двунаправленными прозрачными стрелками. Однонаправленные прозрачные стрелки показывают общее

прохождение ЛП процедуры по Шагам 1 – 4. Загрузка исходных данных и вывод результата обозначены *in* и *out*.

6. Чередование обменов

Основные черты ЛП сортировки ОМНД согласно алгоритму (рис. 2) могут быть прослежены на примере пересортировки набора из 8 элементов:

$$(7, 6, 5, 4, 3, 2, 1, 0) \rightarrow (0, 1, 2, 3, 4, 5, 6, 7). \quad (9)$$

Соответствующая процедура представлен на рис. 3.

Существенным (и тривиальным) ограничением при выполнении обычной (не ЛП) процедуры сортировки является последовательный характер выполнения действий. Как отмечалось, в каждом акте сопоставления рассматривается исключительно одна пара соседствующих элементов (a_i, a_{i+1}). Предшествующие (a_1, a_2), (a_2, a_3),..., (a_{i-1}, a_i) и последующие (a_{i+1}, a_{i+2}), (a_{i+2}, a_{i+3}),..., (a_{n-1}, a_n) пары остаются вне рассмотрения. Таким образом проделывается n-1 однократных операций сопоставления. При этом, когда одна пара сопоставляется, n-2 не задействованы. Введением ЛП обработки эффективность существенно повышается: параллельно проводится ещё до $(n/2)$-1 сопоставлений.

Нумерация шагов процесса дана в левой колонке. Нижняя строка – результат сортировки, полученный после ЛП операций обмена в предпоследней строке.

Строками двунаправленных стрелок над соответствующими нумерованными строками обозначены ЛП операции обмена между сегментами. Каждая строка стрелок есть одна ЛП операция. Как показано, инвертирование последовательности реализуется за n шагов. В примере (рис. 3) в нечётных шагах процесса осуществляется обмен между чётными и нечётными сегментами (чётный сегмент стоит *слева* от нечётного), а в чётных строках – обмен между нечётными и чётными сегментами (чётный сегмент стоит *справа* от нечётного).

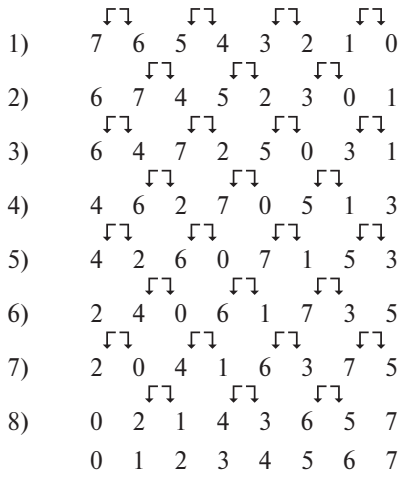


Рис. 3. ЛП сортировка последовательности из 8 элементов

В примере (рис. 3) ЛП сортировка начата со строки обменов «чётный–нечётный». Возможен также другой порядок: начало сортировки со строки «нечётный–чётный». При изменении порядка чередования обменов общее число шагов сортировки не изменяется: полная ЛП пересортировка 4 элементов (1, 2, 3, 4) → (4, 3, 2, 1) реализуется за равное число шагов.

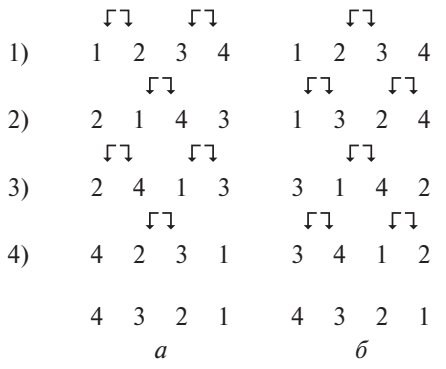


Рис. 4. ЛП сортировка последовательности из 4 элементов в двух вариантах: начиная с «чётный–нечётный» (*а*) и «нечётный–чётный» (*б*)

Результат сортировки также остаётся прежним, поскольку изменение касается только фаз сортировки по чередованию чётностей.

Пример (рис. 4) соответствует графу на рис. 1 б. При этом ЛП алгоритм преобразования (1, 2, 3, 4) → (4, 3, 2, 1) реализует пару цепочек (10, 11) типа (4), но с сокращённым числа промежуточных состояний:

$$(1, 2, 3, 4) \rightarrow (2, 1, 4, 3) \rightarrow (2, 4, 1, 3) \rightarrow \rightarrow (4, 2, 3, 1) \rightarrow (4, 3, 2, 1) \quad (10)$$

$$(1, 2, 3, 4) \rightarrow (1, 3, 2, 4) \rightarrow (3, 1, 4, 2) \rightarrow \rightarrow (3, 4, 1, 2) \rightarrow (4, 3, 2, 1). \quad (11)$$

Выражение (10) соответствует рис. 4 а – ЛП сортировка начинается с сопоставления «чётный–нечётный»; выражение (11) – рис. 4 б – начало сортировки с «нечётный–чётный». На графе рис. 1 б элементы (10) (кроме исходного и конечного) выделена жирным пунктиров, элементы (11) – мелким пунктиром. Сопоставление рис. 4 а, б с графом рис. 1 б позволяет видеть, каким образом реализуется выигрыш в производительности ЛП сортировки по сравнению с традиционной последовательной.

7. Обсуждение результатов

Граф (рис. 1 а) демонстрирует симметрию различных вариантов сортировки. Различные виды последовательностей из 3 элементов расположены в вершинах 6-угольника. Каждая вершина соединена с двумя другими, потому что связь соответствует обмену местами в паре соседствующих элементов, а у последовательности из трёх элементов имеется ровно 2 варианта пар обмена. Изменение порядка

элементов на обратный реализуется за 3 шага не только для (1, 2, 3) → (3, 2, 1), но и для любых пар последовательностей (вершин графа) с взаимно-инверсным расположением символов. Легко видеть, что на графе соответствующие парные вершины в 6-угольнике расположены диагонально.

Аналогично, граф (рис. 1 б) также демонстрирует симметрию. Каждая из вершин графа соединена с тремя другими, так как у последовательности из 4 элементов имеется три пары соседствующих элементов, что соответствует трём вариантам обмена. Изменение порядка элементов на обратный реализуется за 6 шагов не только для (1, 2, 3, 4) → (4, 3, 2, 1), но и для любых пар последовательностей (вершин графа) со взаимно-инверсным расположением символов. Это может быть непосредственно прослежено на графе. Изображение графа (рис. 1 б) укрупнено, имеет очертания ромба. Вершины графа, расположенные по левой верхней стороне ромба, в последовательности слева направо (т.е. (1, 2, 3, 4), (2, 1, 3, 4), (2, 3, 1, 4), (2, 3, 4, 1)) взаимно-инверсны вершинам графа, расположенным по правой нижней стороне ромба, в последовательности справа налево (т.е. (4, 3, 2, 1), (4, 3, 1, 2), (4, 1, 3, 2), (1, 4, 3, 2)). Аналогично, взаимно-инверсны вершины расположенные по правой верхней и левой нижней сторонам ромба.

Граф (рис. 1 б) – непланарный. При других его начертаниях (начиная, например, с изменения порядка расположения вершин второго слева столбца) другие его вершины будут «выведены» на стороны ромба. При этом в силу симметрии (каждая вершина соединена ровно с тремя другими) сохраняется взаимно-инверсное расположение вершин. Между взаимно-инверсными вершинами непосредственно и наглядно может быть прослежена на графе цепочки длиной ровно 6 шагов.

Описанная ситуация не является неожиданной, поскольку сводится к циклической замене символов или может рассматриваться как перекодировка записи выражений (1, 2, 3) → (3, 2, 1) и (1, 2, 3, 4) → (4, 3, 2, 1) и соответствующая замена символов в графах (рис. 1).

Таким образом, рассмотренное выше ЛП преобразование (1, 2, 3, 4) → (4, 3, 2, 1), проиллюстрированное графом (рис. 1 б) является достаточно общим, справедливым для любых пар взаимно-инверсных вершин.

В алгебраическом смысле рассматриваемые последовательности образуют конечные группы относительно операции перестановки пары элементов. В связи с этим представленные особенности, касающиеся симметрии в преобразованиях последовательностей, являются отражением теоретико-групповых свойств.

Работа ЛП процедуры сортировки (рис. 2) смоделирована в варианте с 8 3-битовыми сегментами. Заполнение сегментов выполнено наборами чисел (0, 1, 2, 3, ..., 7) со случайной перестановкой от генератора случайных чисел с равномерным законом распределения. Модель реализована на языке Python и демонстрирует работоспособность алгоритма.

8. Перспективы использования

Сортировка данных – классическая задача, исчерпывающе разработанная для ВС последовательного типа. С применением элементов параллельной обработки сортировка приобретает новые очертания. В случае ЛП целесообразна работа с ограниченными наборами данных. При этом скорость выполнения сортировки возрастает пропорционально числу конкатенированных сегментов, вследствие чего она может быть использована в приложениях с быстрым принятием оперативных решений. Применительно к ситуациям типа систем массового обслуживания, наряду с дисциплинами выборки заданий из очередей FIFO и FILO, получает право на существование очередь с заданиями, последовательность (очерёдность, важность, актуальность, срочность) выполнения которых изменяется в процессе нахождения заданий в очереди. Одним из приложений для подобных систем является Интернет: распределённая система хранения информации с децентрализованным управлением и мультиагентной обработкой. Другая важная прикладная область – интерфейс взаимодействия ВС с оператором, в частности, обучающие системы.

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 3-, 4- и 8-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.

Список литературы: 1. *Мохамад, Али* Перспективы реализации локально-параллельных вычислений на многоядерных процессорах [Текст] / *Мохамад Али, О.Ф. Михаль* // Радиоэлектронные и компьютерные системы. – 2008. – № 6 (33). – С. 234–237. **2**. *Мохамад, Али*. Локально-параллельная сортировка данных с ограниченной разрядностью [Текст] / *Мохамад Али*. // Материалы международной научно-практической конференции студентов и аспирантов “Информационные технологии в экономике и образовании”. – М.: Российский университет кооперации, 2008. – С. 48–52. **3.** *Мохаммед, Али*

Локально-параллельная сортировка со встречной чередующейся упорядоченностью данных [Текст] / *Мохамед Али, О.Ф. Михаль* // 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 [Текст] / *О.Ф. Михаль* // Проблемы бионики. – Вып. 55. – 2001. – С. 80–90. **5**. *Михаль, О.Ф.* Принципы организации систем нечеткого регулирования на однородных локально-параллельных алгоритмах [Текст] / *О.Ф. Михаль, О.Г. Руденко* // Управляющие системы и машины. – 2001. – № 3. – С. 3-10. **6**. *Милнер, П.* Физиологическая психология [Текст] / *П. Милнер* М.: Мир, 1973. – 648 с. **7.** *Люггер, Дж.Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем [Текст] / *Дж.Ф. Люгер* 4-е изд.: Пер. с англ. – М.: Изд. дом «Вильямс», 2005. – 864 с. **8**. *Кормен, Т.* Алгоритмы: построение и анализ [Текст] / *Т. Кормен, Ч. Лейзерсон, Р. Ривест* М.: МЦНМО, 2001. – 960 с.

Поступила в редколлегию 28.09.2011

УДК 519.87

Локально-паралельне сортування обмежених малих наборів даних / Мохамад Алі, О.П. Міхаль // Біоніка інтелекту: наук.-техн. журнал. – 2011. – № 3 (77). – С. 119-125.

Розглянуто локально-паралельне подання інформації стосовно задач сортування наборів даних. Процедури сортування проаналізовано на комбінаторному рівні для 3-, 4- и 8-елементних послідовностей. Подано алгоритмічне забеспечення локально-паралельного сортування. В ході моделюрования на мові Python виявлено, що локально-паралельний алгоритм максимально эфективний стосовно до сортування виборок в межах разрядности процесора.

Іл. 4. Бібліогр.: 8 найм.

UDK 519.87

Local-parallel sorting of limited small sets of data / Mohamad Ali, O.Ph. Mikhal // Bionics of Intelligense: Sci. Mag. – 2011. – № 3 (77). –P. 119-125.

Local-parallel presentation of information is considered referring to problem of the sorted data sets. The Procedures of the sorting is analysed on combinatorial level for 3-, 4- and 8-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.

Fig. 4. Ref.: 8 items.