# алгоритмы ПАРАЛЛЕЛЬНОЙ сортировки данных.

# методы четно-нечетной блочной сортировки. параллельный Метод сортировки Шелла

## Общие понятия сортировки данных

Задана исходная последовательность данных (массив) вида либо . На основе последовательности S должна быть сформирована последовательность S’ вида: , где .

Базовая операция при реализации методов последовательной сортировки – операция «Сравнить и переставить» (compare-exchange). Операция предполагает сравнение одной пары значений из сортируемой последовательности и перестановку этих значений в том случае, если их порядок не соответствует условиям сортировки. Методы (алгоритмы) реализации сортировки различаются способами выбора пар значений для сравнения.

Реализация базовой операции **«Сравнить и переставить»** (при *i*<*j*)

if (a[i]>a[j]) {

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

Вычислительная трудоемкость процедуры упорядочивания является достаточно высокой. Так, для ряда известных простых методов (пузырьковая сортировка, сортировка включением и др.) количество необходимых операций определяется квадратичной зависимостью от числа упорядочиваемых данных *О ( n*2 )

Для более эффективных алгоритмов (сортировка слиянием, сортировка Шелла, быстрая сортировка) трудоемкость определяется величиной О*(n* log2 *n).*

Ускорение сортировки может быть обеспечено при использовании нескольких (*p>1*) *вычислительных элементов* (процессоров). Исходный упорядочиваемый набор в этом случае разделяется на блоки, которые могут обрабатываться вычислительными элементами параллельно.

## Принципы распараллеливания сортировки

В основу реализации подхода к распараллеливанию процесса сортировки данных положены следующие принципы:

1. исходный набор данных (значений) разделяется между устройствами, т. е. набор данных разделяется на блоки, каждый из которых закрепляется за конкретным вычислительным устройством (номер блока соответствует номеру процессорного элемента);
2. в ходе сортировки данные пересылаются между устройствами и сравниваются между собой (выполняется сравнение данных, входящих в разные блоки);
3. результирующий набор данных также разделен между устройствами, при этом значения, расположенные на процессоре с меньшими номерами, не превышает значения, расположенные на процессорах с большими номерами. Т.е. если блоки данных (идентификаторы блоков данных) являются закрепленными за соответствующими процессорными элементами, тогда в процессе сортировки изменяется состав этих блоков.

Обозначим через *l* идентификатор блока данных, соответствующий *l*-му процессорному элементу P*l* , n*l* –количество элементов в *l*-ом блоке данных. Тогда значение (n*l*–ый элемент в *l*-ом блоке данных) на процессоре P*l* не больше значения *al*+1,1 (первый элемент) на процессоре P*l*+1.

Внутри *l*-ого блока данные упорядочиваются по рассматриваемому признаку.

### *Реализация операции «Сравнить и переставить» для (P = n*)

Здесь через *Р* обозначено количество процессорных элементов, через n–количество данных в последовательности. При *i*<*j* имеем: *ai*–>*Pi*, *aj*–>*Pj*(данные *ai* соответствуют процессору *Pi*, данные *aj* соответствуют процессору *Pj*). Параллельная реализация операции «Сравнить и переставить» предполагает:

1. обмен имеющимися на процессорах *Pi* и *Pj* значениями *ai* и *aj*; в результате на каждом процессоре рассматриваются одинаковые пары значений (*ai*, *aj*) (т.к. (*ai*, *aj*)–>*Pi*, (*ai*, *aj*)–>*Pj*);
2. сравнение на каждом процессоре *Pi* и *Pj* пар (*ai*, *aj*) т.о., чтобы при *i*<*j* на *Pi* сохранялся минимальный элемент в паре, на *Pj*– максимальный элемент в паре (*ai*, *aj*). Т.о. на основе пары (*ai*, *aj*) формируется новый элемент *a′i*, закрепленный за процессором Pi, следующим образом:

На основе пары (*ai, aj* ) определяется новый элемент *a′j* , закрепленный за процессором *Pj*, следующим образом:

Итоговая запись результата выполнения операций «Сравнить и переставить»:

, где

, где

### 

### Распространение базовой операции «Сравнить и переставить» для случая p<n. Операция «Сравнить и разделить»

При *p*<*n* должно быть определено *Р* блоков данных, каждый из блоков имеет размер *n*/*p*. Тогда при *p<n* каждому процессору ставится в соответствие не единственное значение *ai* и совокупность значений (блок) из сортируемого набора данных.

Реализация параллельной (распределенной) сортировки должна предусматривать:

1. упорядочивание элементов (значений) внутри блоков;
2. упорядочивание элементов (значений) между блоками (т.е. ), где элементы и*al*+1,1 – последний и первый элементы в упорядоченных блоках *l* и (*l*+1) процессорных элементах.

Т.о. на основе исходных составов блоков, формируемых на базе исходный последовательности , определяются модифицированные составы блоков в соответствии с введенными принципами сортировки. Для формирования модифицированных составов блоков используется операция «сравнить и разделить».

Обозначим через b*l* блок значение, соответствующих процессорному элементу P*l*, вид блока b*l* следующий: .

Тогда в результате обмена блоками *bl* и *bh* между ПЭ *Pl* и *Ph*формируется фрагмент (блок) данных длиной 2*n*/*p* (в результате слияния блоков *bl*и *b*h формируется блок , где «∪» — операция конкатенации (соединения) блоков).

Результирующий блок должен быть упорядочен в соответствии с введенным правилом. После чего формируется левая половина блока ′ в виде и правая половина блока в виде . Левая половина блока закрепляется за ПЭ *Pl*, правая за *Ph*. Таким образом ,

### Вид последовательности действий при реализации операции «сравнить и разделить»

В результате реализации процедуры «сравнить и разделить» блоки на процессорах *Pl* и *Рh* совпадают по размеру с блоками *bl* и *bh*(исходными блоками). Все значения, расположенные на процессоре *Pl*, не превышают значений на процессоре *Рh*

Формализация введенного утверждения:

.

Операция «сравнить и разделить» является базовой подзадачей при организации параллельной сортировки.

Схема реализации процедуры «сравнить и разделить» представлена на Рис. 1.1.

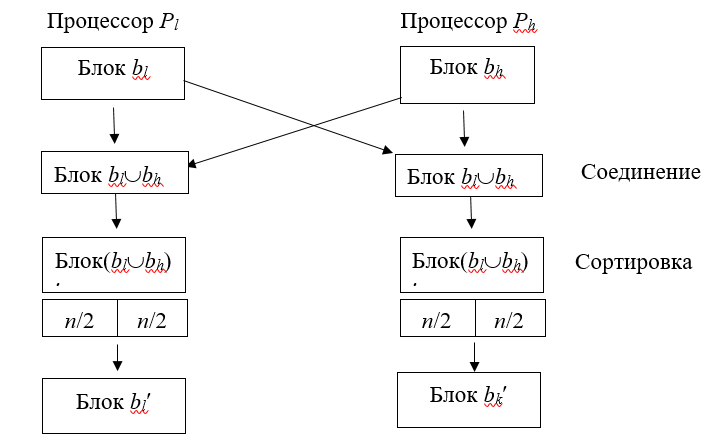


Рисунок 1.1. – Схема реализации процедуры «сравнить и разделить»

Особенности использования операции «сравнить и разделить» при реализации алгоритмов сортировки:

1. составы блоков данных, относящиеся к процессорным элементам, изменяются в ходе выполнения сортировки;
2. размер блоков данных может быть постоянным и одинаковым, либо может различаться в ходе реализации сортировки.

### Реализация чет-нечетной перестановки при P=*n*

Для параллельной реализации метода пузырьковой сортировки используется его модификация, называемая чет-нечетной перестановкой.

Этапы реализации метода чет-нечетной перестановки:

1. Разбиение массива (последовательности) *S* на пары вида   
   (*a*0, *a*1), (*a*2, *a*3), (*a*4, *a*5), …, (*a*n-2, *a*n-1) – четная сортировка.

Для каждой пары элементов выполняется операция «сравнить и переставить». В каждой паре слева помещается наименьший элемент, справа – наибольший.

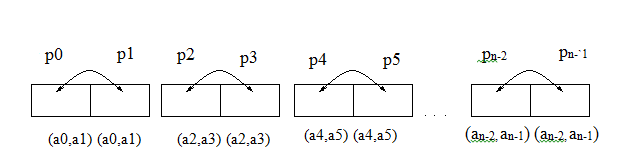
1. Массив значений (последовательности) *S* разбивается на пары вида (a1, *a*2), (*a*3, *a*4), …, (*a*n-3, *a*n-2)– нечетная сортировка.

Для каждой пары элементов выполняется операция «сравнить и переставить». Таким образом при четной сортировке пары начинаются с четных индексов, при нечетной сортировке с нечетных индексов.

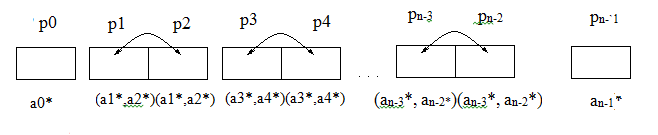
Рассмотренный алгоритм непосредственно может быть использован при реализации параллельной сортировки в случае, если число ПЭ равно количеству n элементов в последовательности *S*.

Порядок реализации обмена при формировании пар значений:

1. Первый этап (четные пары)



1. Второй этап (нечетные пары процессов)



Примерреализации чет-нечетной перестановки элементов при пузырьковой сортировке

Исходная последовательность: 18752

1 этап: 18 *75* 2 –> 18 *57* 2

2 этап: 1 *85* *72* –> 1 *58* *27*

1 этап: 15 *82* 7 –> 15 *28* 7

2 этап: 1 *52* *87* –> 1 *25* *78*

Выводпо параллельной реализации пузырьковой сортировки в случае P= n:

1. Процессоры с номерами, соответствующими элементам в парах, обмениваются друг с другом значениями и формируют их пары.{\displaystyle \forall }
2. Каждый ПЭ, сформировавший на данном этапе пару элементов, реализует операцию «сравнить и переставить» параллельно с другими ПЭ.
3. Если номер ПЭ соответствует меньшему номеру элемента в паре, то он сохраняет минимальный элемент, если номер процессорного элемента соответствует большему номеру, то он сохраняет максимальный элемент.

### Реализация четно-нечетной блочной сортировки при P<n (упрощенная интерпретация)

Последовательность из n элементов разделяется на части одинакового размера n/p. Каждая из частей назначается соответствующему устройству. Элементы, входящие в блок предварительно, сортируются (блок закреплен b*l* за устройством P*l*). После начальной инициализации и сортировки блоков алгоритм пузырьковой сортировки предполагает реализацию следующих этапов (предполагает реализацию обмена между вычислительными устройствами следующих образом):

1. Обмен внутри четных пар процессоров с номерами (0, 1), (2, 3), (4,5),...;
2. Обмен внутри нечетных пар процессоров с номерами (1, 2), (3, 4), (5, 6);

После обмена каждое устройство, реализовывавшее обмен, выполняет операцию «сравнить и разделить» параллельно с другими устройствами.

Пример реализации обмена при пузырьковой сортировке, при условии *P*<*n*, представлен на Рис.1.2.

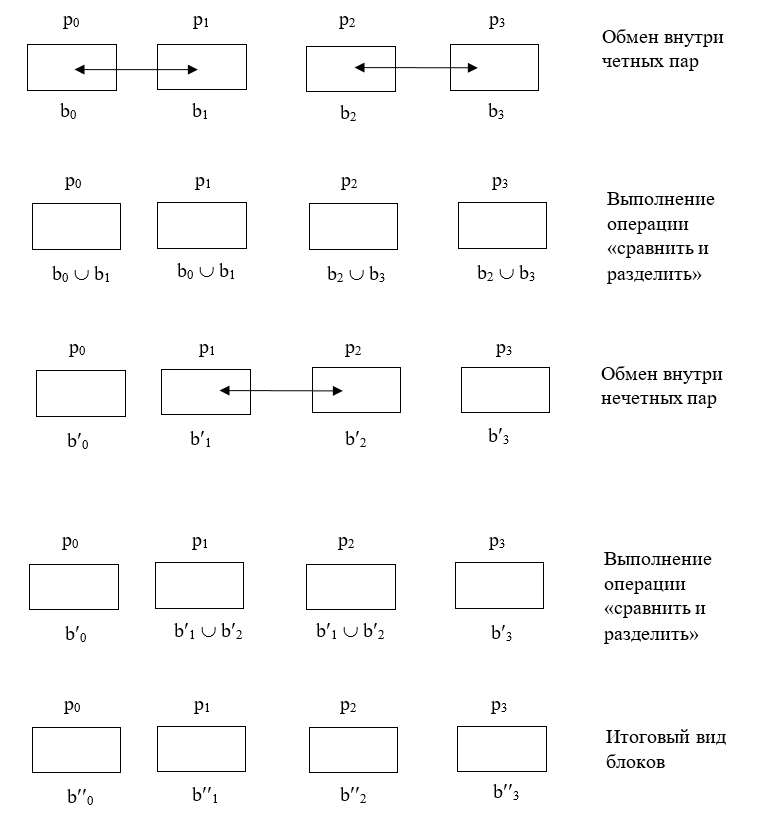


Рисунок 1.2 – Схема реализации четно-нечетной перестановки при *P*<*n*

Пример реализации обмена при пузырьковой сортировке, (*n* = 6, *p* = 3)

Исходная последовательность *S* имеет вид:

*S* = (2, 8, 4, 5, 6, 7)

|  |  |  |  |
| --- | --- | --- | --- |
| 0 этап | 2 8  p0 | 4 5  p1 | 6 7  p2 |
| 1-ый этап | p0  2 8 4 5 | p1  2 8 4 5 | p2  6 7 |
| Операция «сравнить и разделить» | p0  2 4 5 8  2 4 | p1  2 4 5 8  5 8 | p2  6 7  6 7 |
| 2-ой этап | p0  2 4 | p1  5 8 7 6 | p2  5 8 6 7 |
| Операция «сравнить и разделить» | p0  2 4  2 4 | p1  5 6 7 8  5 6 | p2  5 6 7 8  8 |

1.4.3 Формализация метода чет-нечетной перестановки для случая P<n (блочный аналог четно-нечетной перестановки)

Алгоритм «сортировки слиянием» двух упорядоченных массивов.

Заданы исходные массивы, элементы которых должны быть упорядочены. Массивы являются отсортированными.

Вид исходных массивов:

Вид результирующего массива:

,

где , в итоге элементы такие, что ;

Начальными условиями для реализации алгоритма является инициализация индексов массивов A, B, C значением 0 (j=0)

Три возможных варианта реализации вычислительного процесса в соответствии с алгоритмом:

* + 1. если , тогда

1.

2.

ii. если , тогда

1.

2.

т.е. сначала в С записывается один массив, потом другой.

1. Сравнение и , если:
2. , тогда ;

; ; ( индекс не изменяется);

1. , тогда

; ; ( индекс ja не изменяется);

1. сравнение и продолжается до тех пор, пока

либо ;

Если , тогда шаг 4;

Если , тогда шаг 5;

1. если ,тогда

, ; ;

Если , тогда шаг 4;

1. если ,тогда

, ; ;

если , тогда шаг 5.

Т.е. как только исчерпан один из входных массивов (A или B), но не исчерпан другой, тогда оставшаяся часть не законченного массива переписывается в массив С.

### Формализация метода чет-нечетной перестановки для случая *p*<*n* (блочный аналог четно-нечетной перестановке)

Исходный массив , где n-количество элементов в массиве. Если Р-число процессоров, тогда n′ - количество элементов в каждом блоке (всего р блоков). Таким образом всего определено *Р* блоков, сформированных из элементов массива A.

каждый длинной *n′*

;

, либо ;

В общем виде:

По аналогии с методом сортировки слиянием определены условия упорядочивания блоков:

и  имеем:

1. , тогда  (где  отношение упорядочивания массивов).

и имеем:

2.  ,тогда .

### Алгоритм блочной чет-нечетной перестановки

1. Исходный шаг-определение индекса шага алгоритма (*S*=0). Итерации алгоритма соответствуют значениям 0, 1, 2, …, *р*-2, *р*-1.

2. На четных итерациях в парах блоков проверяется выполнение условия

, где *j* ∈ {0, 1, 2, …},

2*j* – индекс четного блока, 2*j*+1 – индекс нечетного блока.

Если введенное условие не выполняется, тогда реализуется упорядочивание элементов в паре блоков с использованием алгоритма «сортировки слиянием».

3. На нечетных итерациях в парах блоков проверяется выполнение условия , где j ∈ {1, 2, …}

2*j*-1 – индекс нечетного блока, 2*j* – индекс четного блока

Если условие не выполняется, тогда упорядочивание элементов в блоке реализуется алгоритмом «сортировки слиянием».

В целом алгоритм осуществляет операции над блоком, называемые «сравнить и разделить».

Распараллеливание блочного варианта «чет-нечетной перестановки» основано на том, что на каждой итерации упорядочивание элементов в любых различных парах блоков можно выполнить независимо (т.е. одновременно).

Распределенный вариант алгоритма предполагает пересылку соседних блоков (блоков, входящих в одну пару) между соседними узлами.

**СОРТИРОВКА ШЕЛЛА**

**Последовательная реализация сортировки Шелла**

Особенность реализации перестановки – обмен при выполнении условия сортировки выполняется между элементами массива, расположенными друг от друга на большом расстоянии.

При этом на 1-м этапе рассматриваются группы по 2 элемента, на 2-м этапе рассматриваются группы по 4 элемента, на 3-м этапе - по 8 элементов и на заключительном этапе рассматривается весь массив.

Таким образом на 1-м шаге происходит упорядочивание элементов в парах следующего вида: (), где .

На 2-м этапе упорядочиваются элементы в группах из 4-х элементов вида: (), где .

На 3-м этапе упорядочиваются элементы в группах по 8 элементов вида: (), где

На заключительном этапе упорядочиваются элементы во всем массиве .

Таким образом на каждом следующем этапе расстояния между элементами в группе уменьшается в 2 раза, а число элементов в группе увеличивается в 2 раза. На последнем этапе сортируется весь массив как одна группа.

При определении элементов, входящих в соответствующие группы, внутри этих групп выполняется сортировка элементов.

Таким образом на каждом этапе выполняется сортировка элементов внутри выделенных групп.

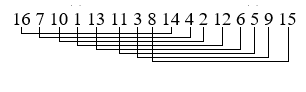
Пример реализации последовательной сортировки Шелла

Исходный массив имеет вид:

10 1 13 11 3 8 14 4 2 12 6 5 9 15

1 этап (1; +1), (2; +2),…,()

8 групп по 2 элемента



2 этап 4 группы по 4 элемента, группы элементов с индексами:



3 этап 2 группы по 8 элементов, группы элементов с индексами:



4 этап 1 группа по 16 элементов:



1 2 3 4 5 6 7 9 8 10 11 13 12 14 15 16

Необходим повторный проход по массиву.

**Параллельная реализация сортировки Шелла**

Реализация параллельной сортировки предполагает наличие параллельно действующих узлов. Здесь должно быть определено различие между взаимодействием блоков массива и взаимодействием ПЭ.

Каждому из блоков данных ставится в соответствие процесс. Нумерация процессов реализуется в двоичной системе, соответственно 00, 01, 10, 11 (номер процесса соответствует номеру блока данных).

**Реализация параллельной сортировки Шелла выполняется в 2 этапа:**

**1 этап:** предполагает выполнение операции «сравнить и разделить» для соответствующих пар процессов в кубе.

Правило формирования номеров взаимодействующих процессов (номеров блоков данных, для которых выполняется операция "сравнить и разделить").

Если i-номер итерации (i= всего N+1 итерация), тогда пары образуют те процессы (блоки), у которых различие в битовом представлении их номеров имеются в позиции (разряде) N-i.

При i=0 для первых пар блоков должно быть различие в левом разряде. На первой итерации обмен блоками и реализация операции «сравнить и разделить» должен быть выполнены для номеров 00 и 10, 01 и 11.

При i=1 для вторых пар блоков должно быть различие в правом разряде. Таким образом на второй итерации обмен блоками и реализация операции "сравнить и разделить" выполняется для номеров 00 и 01, 10 и 11.

Таким образом при N=2 должно быть выполнено 2 итерации, на которых реализуется обмен блоками между процессами с соответствующими номерами и выполнение процессами операции «сравнить и разделить».

**2 этап** предполагает реализацию итераций алгоритма четной-нечетной перестановок (т.е. последовательный обмен блоками между процессами, номера которых определяются алгоритмом четной-нечетной перестановки, и реализация операции «сравнить и разделить».

Итерации продолжаются до прекращения изменения сортируемого набора.

В рассматриваемом случае четная перестановка – обмен данными между процессами с номерами (00, 01) и (10, 11) (обмен блоками в указанных парах процессов). Нечетная перестановка – (01, 10).

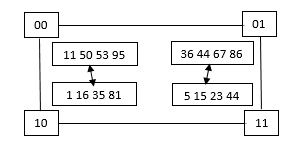
Пример реализации итераций алгоритма параллельной сортировки Шелла

Некоторый массив (с разбиением на блоки) имеет вид:

11 50 53 95 | 36 44 67 86 | 1 16 35 81 | 5 15 23 44

**1-я итерация 1-го этапа алгоритма:**

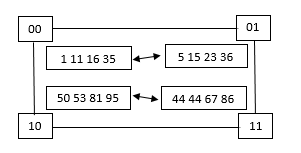
Процессы/блоки



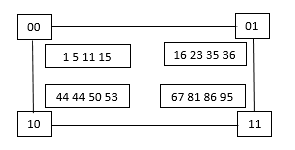
Гиперкуб с N=1 для хостов

Гиперкуб с N=2 для процессов

**2-я итерация 1-го этапа**



Результат второй итерации 1-го этапа



Исходя из результатов 1-го этапа четно-нечетные перестановки не требуется.

Пример организации обмена между процессами на первом этапе при N=3, p=8.

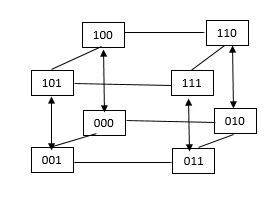
Нумерация блоков (процессов): 000, … ,111 (0, … ,7)

Формат номера блока: 2-я позиция, 1-я позиция, 0-я позиция.

Определение номеров процессов, реализующих обмен блоками данных на каждой итерации 1-го этапа.

Итерация 0 (i=0) =>.

Вид обмена между процессами в гиперкубе



Обмен между парами блоков:

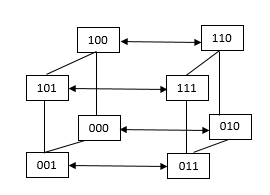
0(000)⬄4(100)

1(001)⬄5(101)

2(010)⬄6(110)

3(011)⬄7(111)

Итерация 1 (i=1) .



Обмен между парами блоков:

0(000)⬄2(010)

1(001)⬄3(011)

4(100)⬄6(110)

5(101)⬄7(111)

Итерация 2 (i=2) .

Обмен между парами блоков:

0(000)⬄1(001)

2(010)⬄3(011)

4(100)⬄5(101)

6(110)⬄7(111)