

АЛГОРИТМЫ ПАРАЛЛЕЛЬНОЙ СОРТИРОВКИ ДАННЫХ. МЕТОДЫ ЧЕТНО-НЕЧЕТНОЙ БЛОЧНОЙ СОРТИРОВКИ. ПАРАЛЛЕЛЬНЫЙ МЕТОД СОРТИРОВКИ ШЕЛЛА

1. Общие понятия сортировки данных

Задана исходная последовательность данных (массив) вида $S = (a_1, a_2, \dots, a_n)$ либо $S = (a_i | i = \overline{1, n})$. На основе последовательности S должна быть сформирована последовательность S' вида: $S' = (a'_1, a'_2, \dots, a'_n)$, где $a'_i \leq a'_{i+1}$ ($i = \overline{1, n-1}$).

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 $O(n^2)$

Для более эффективных алгоритмов (сортировка слиянием, сортировка Шелла, быстрая сортировка) трудоемкость определяется величиной $O(n \log_2 n)$.

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.

2. 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 a_{l,n_l} (n_l -ый элемент в l -ом блоке данных) на процессоре P_l не больше значения $a_{l+1,1}$ (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 P обозначено количество процессорных элементов, через n – количество данных в последовательности. При $i < j$ имеем: $a_i \rightarrow P_i$, $a_j \rightarrow P_j$ (данные a_i соответствуют процессору P_i , данные a_j соответствуют процессору P_j). Параллельная реализация операции «Сравнить и переставить» предполагает:

1) обмен имеющимися на процессорах P_i и P_j значениями a_i и a_j ; в результате на каждом процессоре рассматриваются одинаковые пары значений (a_i, a_j) (т.к. $(a_i, a_j) \rightarrow P_i$, $(a_i, a_j) \rightarrow P_j$);

2) сравнение на каждом процессоре P_i и P_j пар (a_i, a_j) т.о., чтобы при $i < j$ на P_i сохранялся минимальный элемент в паре, на P_j – максимальный элемент в паре (a_i, a_j) . Т.о. на основе пары (a_i, a_j) формируется новый элемент a'_i , закрепленный за процессором P_i , следующим образом:

$$a'_i = \min(a_i, a_j)$$

На основе пары (a_i, a_j) определяется новый элемент a'_j , закрепленный за процессором P_j , следующим образом:

$$a'_j = \max(a_i, a_j)$$

Итоговая запись результата выполнения операций «Сравнить и переставить»:

$$a'_i \rightarrow P_i, \text{ где } a'_i = \min(a_i, a_j)$$

$$a'_j \rightarrow P_j, \text{ где } a'_j = \max(a_i, a_j)$$

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

При $p < n$ должно быть определено P блоков данных, каждый из блоков имеет размер n/p . Тогда при $p < n$ каждому процессору ставится в соответствие не единственное значение a_i 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. $a_{l,n_l} < a_{l+1,1}$),

где элементы a_{l,n_l} и $a_{l+1,1}$ – последний и первый элементы в упорядоченных блоках l и $(l+1)$ процессорных элементах.

Т.о. на основе исходных составов блоков, формируемых на базе исходной последовательности $S = (a_1, a_2, \dots, a_n)$, 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 значений, соответствующих процессорному элементу P_l , вид блока b_l следующий: $b_l = (a_{l1}, a_{l2}, \dots, a_{ln_l})$.

Тогда в результате обмена блоками b_l и b_h между ПЭ P_l и P_h формируется фрагмент (блок) данных длиной $2n/p$ (в результате слияния блоков b_l и b_h формируется блок $b_l \cup b_h$, где « \cup » — операция конкатенации (соединения) блоков).

$(b_l \cup b_h)'_2 \rightarrow P_h$ Результирующий блок $b_l \cup b_h$ должен быть упорядочен в соответствии с введенным правилом. После чего формируется левая половина блока $(b_l \cup b_h)'$ в виде $(b_l \cup b_h)'_1$ и правая половина блока в виде $(b_l \cup b_h)'_2$. Левая половина $(b_l \cup b_h)'_1$ блока закрепляется за ПЭ P_l , правая за P_h . Таким образом $(b_l \cup b_h)'_1 \rightarrow P_l$,

Вид последовательности действий при реализации операции «сравнить и разделить»

В результате реализации процедуры «сравнить и разделить» блоки на процессорах P_l и P_h совпадают по размеру с блоками b_l и b_h (исходными блоками). Все значения, расположенные на процессоре P_l , не превышают значений на процессоре P_h

Формализация введенного утверждения:

$$[b_l \cup b_h]' = b'_l \cup b'_h; \forall a_i \in \forall b'_l, \forall a_j \in \forall b'_h: a_i \leq a_j.$$

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

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

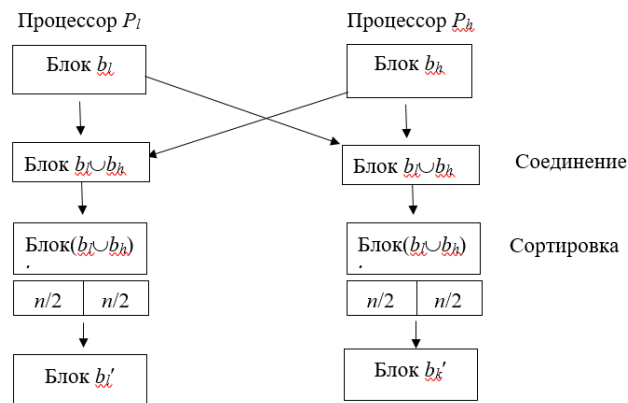


Рисунок 1.1. – Схема реализации процедуры «сравнить и разделить»

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 $(a_0, a_1), (a_2, a_3), (a_4, a_5), \dots, (a_{n-2}, a_{n-1})$ – четная сортировка.

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

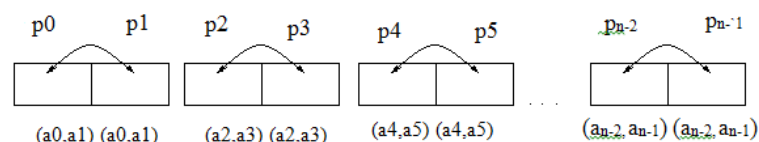
- 2) Массив значений (последовательности) S разбивается на пары вида $(a_1, a_2), (a_3, a_4), \dots, (a_{n-3}, a_{n-2})$ – нечетная сортировка.

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.

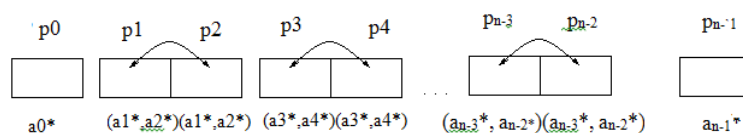
Рассмотренный алгоритм непосредственно может быть использован при реализации параллельной сортировки в случае, если число ПЭ равно количеству n элементов в последовательности S .

Порядок реализации обмена при формировании пар значений:

1. Первый этап (четные пары)



2. Второй этап (нечетные пары процессов)



Пример реализации чет-нечетной перестановки элементов при пузырьковой сортировке

Исходная последовательность: 18752

1 этап: 18 75 2 \rightarrow 18 57 2

2 этап: 1 85 72 \rightarrow 1 58 27

1 этап: 15 82 7 \rightarrow 15 28 7

2 этап: 1 52 87 \rightarrow 1 25 78

Вывод по 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.

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_i за устройством P_i). 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, реализовавшее 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.

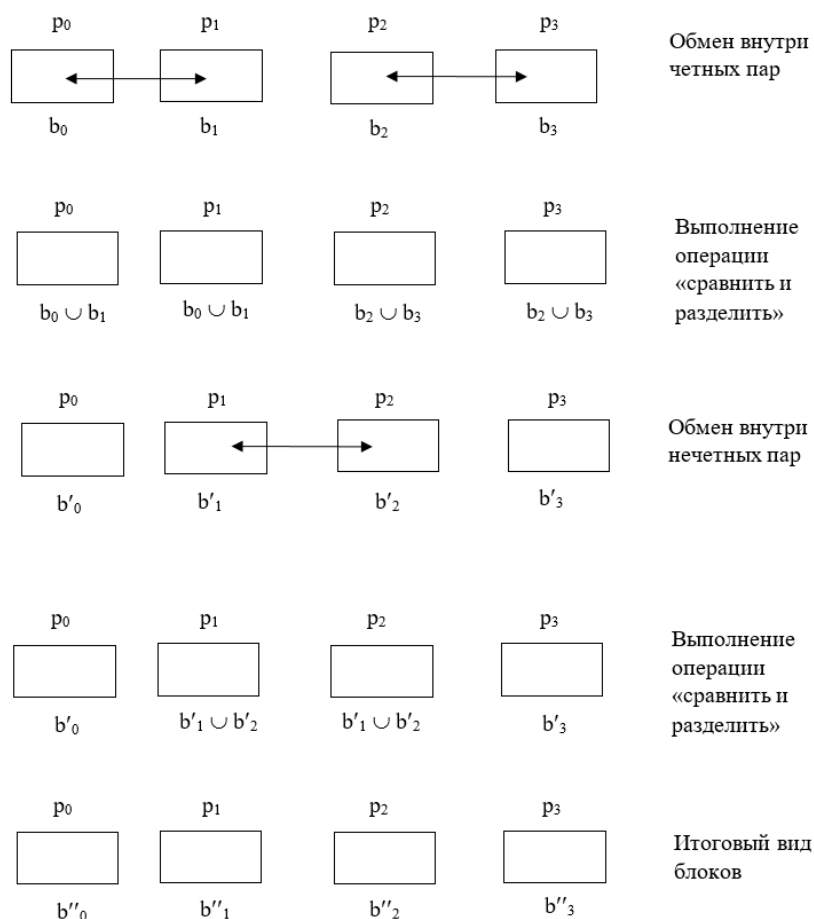


Рисунок 1.2 – Схема реализации четно-нечетной перестановки при $P < n$

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	4 5	6 7
	p ₀	p ₁	p ₂
1-ый этап	2 8 4 5	2 8 4 5	6 7
	p ₀	p ₁	p ₂
Операция «сравнить и разделить»	2 4 5 8	2 4 5 8	6 7
	2 4	5 8	6 7
2-ой этап	2 4	5 8 7 6	5 8 6 7
	p ₀	p ₁	p ₂
Операция «сравнить и разделить»	и 2 4	5 6 7 8	5 6 7 8
	2 4	5 6	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:

$$A = \{a_{j_a} \mid j_a = \overline{0, n_A - 1}\}, a_{j_a+1} \geq a_{j_a}$$
$$B = \{b_{j_b} \mid j_b = \overline{0, n_B - 1}\}, b_{j_b+1} \geq b_{j_b}$$

Вид результирующего массива:

$$C = \{c_j \mid j = \overline{0, n_C - 1}\},$$

где $n_C = n_A + n_B$, в итоге элементы такие, что $c_{j+1} \geq c_j$;

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:

i. если $a_{n_A-1} \leq b_0$, тогда

1. $c_j = a_{j_a}$, где $j_a = \overline{0, n_A - 1}; j = \overline{0, n_A - 1} \setminus$

2. $c_j = b_{j_b}$, где $j_b = \overline{0, n_B - 1}; j = \overline{n_A, n_A + n_B - 1}$

ii. если $b_{n_B-1} \leq a_0$, тогда

1. $c_j = b_{j_b}$, где $j_b = \overline{0, n_B - 1}; j = \overline{0, n_B - 1}$

2. $c_j = a_{j_a}$, где $j_a = \overline{0, n_A - 1}; j = \overline{n_B, n_B + n_A - 1}$

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

iii. Сравнение a_{j_a} и b_{j_b} , если:

1. $a_{j_a} < b_{j_b}$, тогда $c_j = a_{j_a}$;

$$j = j + 1; j_a = j_a + 1; (\text{индекс } j_b \text{ не изменяется});$$

2. $b_{j_b} \leq a_{j_a}$, тогда $c_j = b_{j_b}$;

$$j = j + 1; j_b = j_b + 1; (\text{индекс } j_a \text{ не изменяется});$$

3. сравнение a_{j_a} и b_{j_b} продолжается до тех пор, пока

$$j_a \leq n_A - 1, \text{ либо } j_b \leq n_B - 1;$$

Если $j_a > n_A - 1$, тогда шаг 4;

Если $j_b > n_B - 1$, тогда шаг 5;

4. если $j_a > n_A - 1$, тогда

$$c_j = b_{j_b}, i = i + 1; j_b = j_b + 1;$$

Если $j_b \leq n_B - 1$, тогда шаг 4;

5. если $j_b > n_B - 1$, тогда

$c_j = a_{j_a}, j = j + 1; j_a = j_a + 1;$
 если $j_a \leq n_A - 1$, тогда шаг 5.

Т.е. как только исчерпан один из входных массивов (А или В), 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 $A = \{a_0, a_1, \dots, a_{n-1}\}$, где n -количество элементов в массиве. Если P -число процессоров, тогда n' - количество элементов в каждом блоке (всего p блоков). Таким образом всего определено P блоков, сформированных из элементов массива А.

$A \rightarrow A_0, A_1, \dots, A_{p-1}$ каждый длиной n'

$A_0 = \{a_0, a_1, \dots, a_{n'-1}\};$

$A_1 = \{a_{n'}, a_{n'+1}, \dots, a_{(n'+n')-1}\}$, либо $A_1 = \{a_n, a_{n'+1}, \dots, a_{2n'-1}\};$

$A_{p-1} = \{a_{(p-1)n'}, \dots, a_{(p-1)n'+(n'-1)}\}$

В общем виде: $A_k = \{a_{kn'}, a_{(k+1)n'}; \dots; a_{(k+1)n'-1}\}$

По аналогии с методом сортировки слиянием определены условия упорядочивания блоков:

$\forall a_{n' \cdot k+l} \in A_k$ и $\forall a_{n' \cdot j+l} \in A_j$ имеем:

1. $a_{n' \cdot k+l} \geq a_{n' \cdot j+s}$, тогда $A_k \geq A_j$ (где \geq отношение упорядочивания массивов).

$\forall a_{n' \cdot k+l} \in A_k$ и $\forall a_{n' \cdot j+l} \in A_j$ имеем:

2. $a_{n' \cdot k+l} < a_{n' \cdot j+s}$, тогда $A_k < A_j$.

Алгоритм блочной чет-нечетной перестановки

1. Исходный шаг-определение индекса шага алгоритма ($S=0$). Итерации алгоритма соответствуют значениям $0, 1, 2, \dots, p-2, p-1$.

2. На четных итерациях в парах блоков $(A_0, A_1), (A_2, A_3), (A_4, A_5), \dots$ проверяется выполнение условия

$A_{2j} \leq A_{2j+1}$, где $j \in \{0, 1, 2, \dots\}$,

$2j$ – индекс четного блока, $2j+1$ – 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 $(A_1, A_2), (A_3, A_4), (A_5, A_6), \dots$ проверяется выполнение условия $A_{2j-1} \leq A_{2j}$, где $j \in \{1, 2, \dots\}$

$2j-1$ – индекс нечетного блока, $2j$ – 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 $n/2$ парах следующего вида: $(a_i; a_{n/2+i})$, где $i = \overline{1, n/2}$.

На 2-м этапе упорядочиваются элементы в $n/4$ группах из 4-х элементов вида: $(a_i; a_{n/4+i}; a_{n/2+i}; a_{3n/4+i})$, где $i = \overline{1, n/4}$.

На 3-м этапе упорядочиваются элементы в $n/8$ группах по 8 элементов вида: $(a_i; a_{n/8+i}; a_{2n/8+i}; a_{3n/8+i}; a_{4n/8+i}; a_{5n/8+i}; a_{6n/8+i}; a_{7n/8+i})$, где $i = \overline{1, n/8}$.

На заключительном этапе упорядочиваются элементы во всем массиве (a_1, a_2, \dots, a_n) .

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

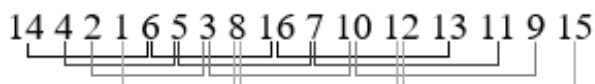
1 этап $(1; \frac{n}{2} + 1), (2; \frac{n}{2} + 2), \dots, (\frac{n}{2}; n)$

8 групп по 2 элемента



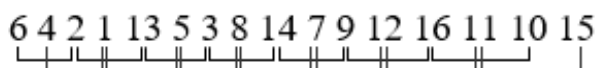
2 этап 4 группы по 4 элемента, группы элементов с индексами:

$(1; \frac{n}{4} + 1; \frac{n}{2} + 1; \frac{3n}{4} + 1), \dots, (\frac{n}{4}; \frac{2n}{4}; \frac{3n}{4}; n)$

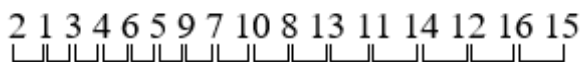


3 этап 2 группы по 8 элементов, группы элементов с индексами:

$(1; \frac{n}{8} + 1; \frac{n}{4} + 1; \frac{3n}{8} + 1; \frac{n}{4} + 1; \frac{5n}{8} + 1; \frac{6n}{8} + 1; \frac{7n}{8} + 1),$
 $\dots, (2; \frac{n}{8} + 2; \frac{2n}{8} + 2; \frac{3n}{8} + 2; \frac{4n}{8} + 2; \frac{5n}{8} + 2; \frac{6n}{8} + 2; \frac{7n}{8} + 2)$



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 ($i=\overline{0, N}$ 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).

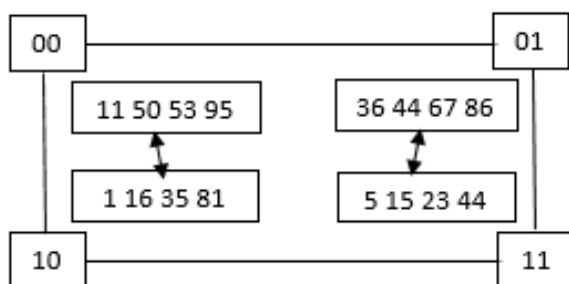
Пример реализации итераций алгоритма параллельной сортировки Шелла

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:

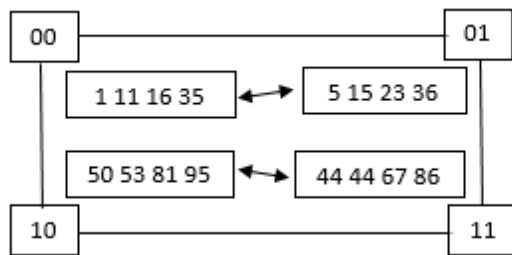
Процессы/блоки



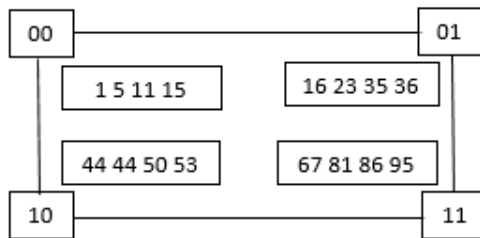
Гиперкуб с $N=1$ для хостов

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 $N=3$, $p=8$.

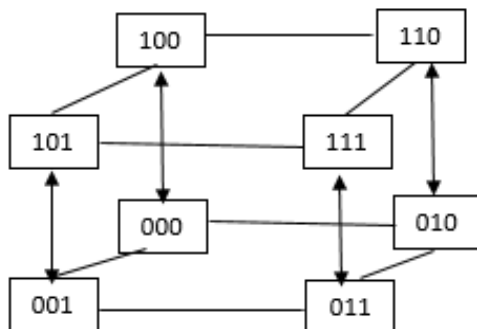
Нумерация блоков (процессов): 000, ..., 111 (0, ..., 7)

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 ($i=0$) $\Rightarrow N_{\text{позиции}} = N - 1 - i = 2$.

Вид обмена между процессами в гиперкубе



Обмен между парами блоков:

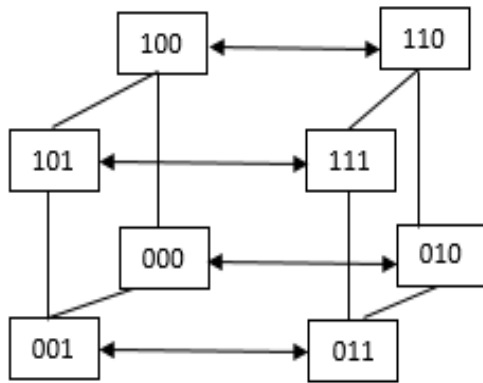
$0(000) \Leftrightarrow 4(100)$

$1(001) \Leftrightarrow 5(101)$

$2(010) \Leftrightarrow 6(110)$

$3(011) \Leftrightarrow 7(111)$

Итерация 1 ($i=1$) $N_{\text{позиции}} = N - 1 - i = 1$.



Обмен между парами блоков:

0(000) \Leftrightarrow 2(010)

1(001) \Leftrightarrow 3(011)

4(100) \Leftrightarrow 6(110)

5(101) \Leftrightarrow 7(111)

Итерация 2 ($i=2$) $N_{\text{позиции}} = N - 1 - i = 0$.

Обмен между парами блоков:

0(000) \Leftrightarrow 1(001)

2(010) \Leftrightarrow 3(011)

4(100) \Leftrightarrow 5(101)

6(110) \Leftrightarrow 7(111)