ЛАБОРАТОРНАЯ РАБОТА №7
ИССЛЕДОВАНИЕ АЛГОРИТМОВ ПАРАЛЛЕЛЬНОЙ БЫСТРОЙ СОРТИРОВКИ ДАННЫХ, ИСПОЛЬЗУЕМЫХ ПРИ ПРОЕКТИРОВАНИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОГРАММНЫХ СИСТЕМ

Цель работы: реализовать и исследовать эффективность алгоритмов параллельной быстрой сортировки с использованием функций библиотеки MPI в сравнении с последовательными версиями тех же алгоритмов.

1 Теоретическое введение

1.1 Алгоритм последовательной быстрой сортировки

Алгоритм быстрой сортировки, предложенный Хоаром (*Hoare C.A.R.*), относится к числу эффективных методов упорядочивания данных и широко используется в практических приложениях.

Схема быстрой сортировки представлена на Рис.1.1

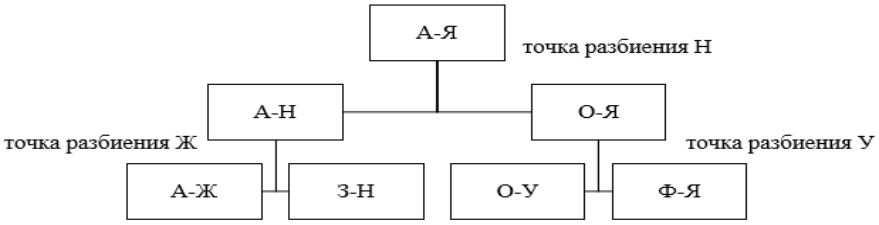

Рисунок 1.1 – Схема реализации последовательной сортировки

Метод основывается на последовательном разделении сортируемого набора на блоки меньшего размера таким образом, чтобы между значениями разных блоков обеспечивалось отношение упорядоченности (для пары блоков все значения одного из них не превышают значения другого).

Особенности реализации алгоритма:

1. В левую часть массива помещаются все элементы, значения которого не больше опорного элемента (меньше или равны), в правую часть массива – те элемента которые не меньше опорного.
2. Выбор опорного элемента должен быть выполнен таким образом, чтобы массив был разделен на примерно равные части.

Один из вариантов реализации алгоритма.

1. В рассмотрение вводятся 2 указателя для обозначения начального и конечного элемента последовательности элементов A (начальный – left, конечный – right), вводится опорный элемент mid.
2. Индекс опорного элемента вычисляется как (left+right)/2. Значение этого элемента присваивается переменной mid.
3. Указатель left смещается с шагом 1 вправо к концу массива до тех пор, пока не выполнится условие фиксации A[left]>mid. Указатель right смещается с шагом -1 влево к началу массива до тех пор, пока не выполнится условие фиксации A[right]<mid.
4. Найденные таким образом элементы меняются местами.
5. Шаги 3 и 4 повторяются пока left<right.
6. Если полученные подмассивы до и после опорного элемента имеют более чем 1 элемент, выполняется быстрая сортировка каждого из них тем же методом.

Пример реализации алгоритма:

Исходный массив A[1..8] имеет вид:

```
6 7 2 5 9 1 3 8
```

1. left – индекс первого элемента, right – индекс последнего элемента. left=1, right=8.
2. Определение mid: (left+right)/2=4.5. Ближайшее значение индекса равно 4. Значение опорного элемента mid=A[4] => mid=5.
3. Элемент A[left]=6 сравнивается с mid=5. Условие фиксации left уже выполнено (A[1]>mid), указатель left не изменяется.
4. Элемент A[right]=8 сравнивается с mid=5. Так как условие A[right]<mid не выполняется, указатель изменяется на -1 и проверка повторяется. Условие фиксации выполняется при right=7.
5. Имеем left=1, right=7. Элемент A[1] и A[7] меняются местами.

Вид массива после реализации обмена:

```
3 7 2 5 9 1 6 8
```

После реализации обмена указатели изменяют свое значение на 1 (left=2, right=6).

6. Выполняется проверка и изменение указателей до условий их фиксации A[left]>mid, A[right]<mid.

Так как условия снова выполняются, то элементы A[2] и A[6] меняются местами, указатели изменяют значения.

Вид массива:

```
3 1 2 5 9 7 6 8
```

Новые значения указателей: left=3, right=5.

7. Снова выполняется проверка условий A[left]>mid и A[right]<mid. Так как A[3]<mid, left увеличивается на 1.

– Так как A[5]>mid (то есть условие не выполняется), то указатель right уменьшается на 1. Тогда left=right и первый этап упорядочивания завершается. Вид массива не изменен.

8. Так как в каждом из подмассивов больше чем 1 элемент, каждый из подмассивов нуждается в дополнительном упорядочивании. Массив разделяется на два с границами A[1..left] и A[right+1..8]. Каждый из них упорядочивается тем же методом. Пример полной сортировки приведен ниже:

6 7 2 **5** 9 1 **3** 8
3 7 2 **5** 9 **1** 6 8
3 1 **2** **5** **9** 7 6 8

3 **1** 2 5 **9** 7 **6** 8
1 3 2 5 6 7 9 8

1 **3** **2** 5 6 **7** **9** **8**
1 **2** 3 5 6 7 8 **9**

Варианты реализации алгоритма

Эффективность алгоритма зависит от выбора границ подмассивов.

Возможны различные методы определения опорного элемента:

- Среднее арифметическое всего массива
- Среднее арифметическое крайних элементов массива
- Центральный элемент массива.

Таким образом, опорный элемент в общем случае может быть и одним из крайних элементов.

После упорядочивания элементов в левой и правой частях новые рассматриваемые массивы должны включать опорный элемент (опорный элемент включается либо в левую, либо в правую части).

1.2 Алгоритм параллельной быстрой сортировки

Параллельная реализация быстрой сортировки выполняется при условии, что топология связи узлов (ПЭ) представлена в виде *N*-мерного гиперкуба. Тогда $p=2N$ – количество ПЭ в гиперкубе.

Если n – количество элементов в массиве тогда количество элементов в блоках данных, закрепляемых за ПЭ, определяется как n/p. Нумерация блоков соответствует нумерации ПЭ.

Среди p процессорных элементов один ПЭ является ведущим, он определяет первоначальное значение опорного элемента.

Способ реализации 1-й стадии алгоритма

1. На ведущем ПЭ выполняется выбор опорного элемента (способ формирования опорного элемента – среднее арифметическое элементов блока, размещенных на ведущем ПЭ).
2. Широковещательная рассылка ведущим ПЭ опорного элемента между остальными (p-1) ПЭ.
3. На каждом ПЭ выполняется разделение имеющегося блока данных на две части (2 подмассива) с использованием опорного элемента - выполняется быстрая сортировка на каждом ПЭ. В результате будут сформированы подмассивы элементов, не больших, чем опорный (левый подмассив), и не меньших, чем опорный (правый подмассив).
4. Процессы, для которых битовое представление номеров отличается в N-ой позиции реализуют обмен данными.

В результате обмена на ПЭ, номер которых в позиции *N* в битовом представлении содержит 0, окажутся значения меньше опорного элемента, на ПЭ номер которых в позиции *N* в битовом представлении содержит 1, окажутся значения данных (элементов массива) больше опорного элемента.

Пример реализации 1-й стадии процесса быстрой сортировки (N=2, n=16, p=4, n/p=4):

1. Распределение исходной последовательности по процессам имеет вид:

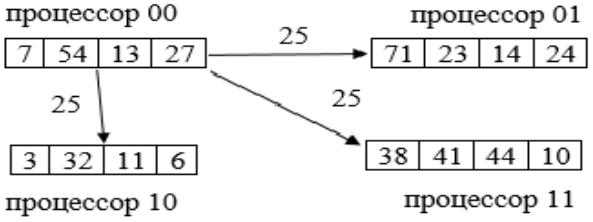

Рисунок 1.2 — Распределение данных и опорного элемента по ПЭ.

–

2. На процессоре 00 в качестве опорного элемента выбирается значение, соответствующее среднему арифметическому значению данных, хранящихся на этом ПЭ (среднее значение = 25).

3. В результате быстрой сортировки с учетом опорного элемента, равного 25, получены левые и правые подпоследовательности (подмассивы) в следующем виде:

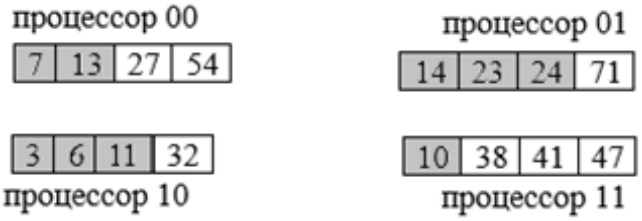

Рисунок 1.3 - Результат разделения на подмассивы относительно опорного элемента

Реализация обмена между процессорами 00-10 и 01-11.

От процессора 00 к процессору 10 передается правая часть, от процессора 10 к процессору 00 - левая часть.

От процессора 01 к процессору 11 передается правая часть, от процессора 11 к процессору 01 передается левая часть.

Полученный вид последовательности представлен на Рис 1.4.

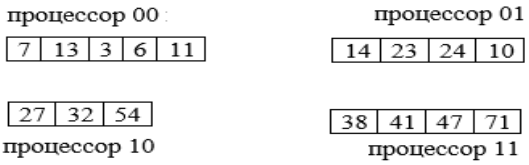

Рисунок 1.4 — Результат обмена подмассивами

Способ реализации 2-й стадии алгоритма

1. Рассматриваются гиперкубы, размерностью на 1 меньше, чем на предыдущей стадии.
   В данном случае размерность гиперкубов равна 1 и процессоры, входящие в гиперкубы, следующие:
   а) гиперкуб 1: процессоры 00 и 01;
   б) гиперкуб 2: процессоры 10 и 11.
2. На процессоре с меньшим номером определяется значение опорного элемента, который рассматривается между ПЭ, входящими в гиперкуб.

После выполнения широковещательной рассылки опорного элемента на каждый ПЭ, входящий в гиперкуб меньшей размерности, реализуется алгоритм быстрой сортировки (с учетом опорного элемента).

На каждом ПЭ гиперкубов:

В результате сформированы подпоследовательности элементов, не больших, чем опорный, и больших опорного.

ПЭ, входящие в гиперкубы меньшей размерности, реализуют обмен подпоследовательностями больших и меньших, чем опорный, элементов. В результате на каждом ПЭ будут сформированы новые последовательности.

Пример реализации 2-й стадии алгоритма параллельной быстрой сортировки.

1. На процессорах 00 и 10 определение опорных элементов (соответственно, значение 10 и 37), их широковещательная рассылка между ПЭ, входящими в гиперкуб (Рис. 1.7).
2. На ПЭ реализация алгоритма быстрой сортировки с учетом опорных элементов, формирование «левых» и «правых» подпоследовательностей (подмассивов) (Рис. 1.8).
3. Реализация обмена подпоследовательностями (Рис. 1.9).

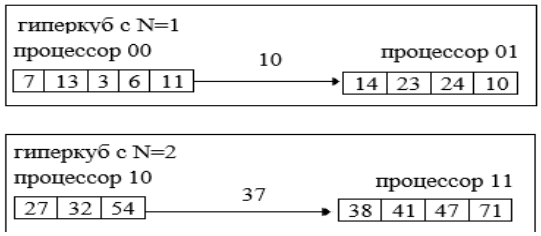

Рисунок 1.5 — Широковещательная рассылка опрного элемента

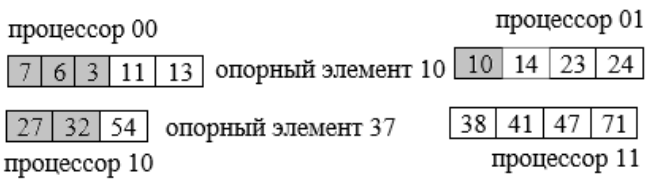

Рисунок 1.6 — Результат разделения на подмассивы относительно опорного элемента

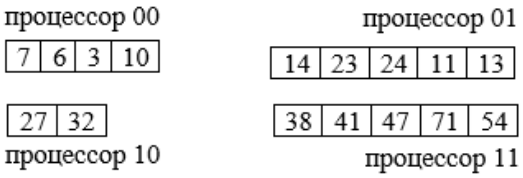

Рисунок 1.7 — Результат обмена подмассивами

Способ реализации 3-й стадии алгоритма

Рассматривается гиперкуб размерности на 1 меньше, чем размерность гиперкуба на предыдущей стадии (размерность гиперкуба $N = 0$).

Опорный элемент на каждом ПЭ определяется как среднее значение из элементов, хранящихся на этом ПЭ (Рис. 1.10). Реализация быстрой сортировки выполняется до тех пор, пока элементы в последовательностях не будут упорядочены.

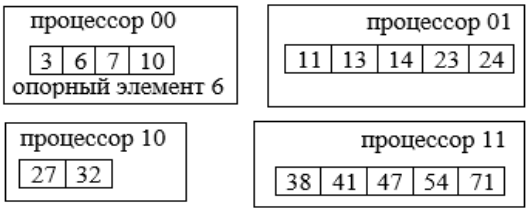

Рисунок 1.8 — Гиперкубы с размерностью N=0

1.3 Модифицированный метод параллельной быстрой сортировки

Первая стадия базового методом параллельной быстрой сортировки и модифицированного метода являются одинаковыми. Методы отличаются последующими стадиями сортировки. Рассмотрим на примере реализацию модифицированного метода быстрой сортировки. (N=2, p=4, n/p=4 при n=16). Исходная последовательность имеет вид:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	54	13	27	71	23	14	24	3	32	11	6	38	41	44	10
Процессор 1				Процессор 2				Процессор 3				Процессор 4			

В качестве опорного элемента выбираем последний элемент последовательности равный 10.

На каждом ПЭ выполняется (локально) алгоритм быстрой сортировки. В результате на каждом ПЭ сформированы «левые» и «правые» последовательности элементов.

Для рассматриваемой последовательности результаты реализации первой итерации алгоритма представлены на Рис 1.11.

Процессор 1			
7	54	13	27
Процессор 3			
3	6	11	32

Процессор 2			
71	23	14	24
Процессор 4			
10	41	44	38

Рисунок 1.9 — Результат первой итерации

Способ реализации 2-й стадии алгоритма

1. Каждый из процессоров определяет сколько элементов оказалась в левой и правой частях. В результате на ведущем ПЭ формируется последовательность значений вида:

а) количество элементов в блоке, меньших или разных заданному опорному элементу.

1	0	2	1

б) Количество элементов в блоке больше заданного.

3	4	2	3

2. Для полученных последовательностей вычисляются префиксные суммы, дополняемые слева 0. Префиксной суммой (префиксом) последовательности чисел $x_1, x_2, \ldots, x_n$ называется другая последовательность $\pi_1, \pi_2, \ldots, \pi_N$, элементы которой определяется следующим образом:

$$\pi_1 = x_1,\ \pi_2 = x_1 + x_2, \ldots, \pi_N = x_1 + x_2 + \cdots + x_N$$

Так как получаемые префиксные суммы для сформированных последовательностей значений дополняются слева нулем, тогда вид префиксных сумм следующий:

0	1	1	3	4
0	3	7	9	12

3. Последнее значение префиксной суммы для левых частей исключается и прибавляется к каждому элементу префиксной суммы для правых частей.

Модифицированные префиксные суммы имеет следующий вид:

а)

0	1	1	3
(1-й ПЭ)	(2-й ПЭ)	(3-й ПЭ)	(4-й ПЭ)

б) 4+

0	3	7	9	12

4	7	11	13	16
(1-й ПЭ)	(2-й ПЭ)	(3-й ПЭ)	(4-й ПЭ)	

Разряды префиксных сумм соответствуют номерам позиций, начиная с которых в результирующей последовательности размещаются соответствующее «левые» и «правые» последовательности для соответствующих ПЭ.

Таким образом префиксная сумма, полученная в пункте а), соответствует номерам позиций левых последовательностей в формируемой результирующей последовательности.

Разряды префиксной суммы, полученной в пункте б), соответствуют номерам позиций правых последовательностей в формируемой результирующей последовательности.

Вид сформированной результирующей последовательности представлен на Рис 1.12.

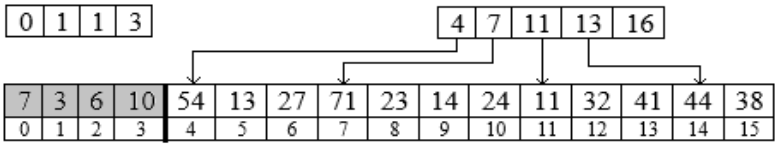

Рисунок 1.10 — Результирующая последовательность (первый этап)

Таким образом элементы левой последовательности с ПЭ3 располагается в позициях, начиная с 1, элементы правой последовательности этого ПЭ размещаются, начиная с 11 позиции.

После формирования результирующей последовательности она распределяется по ПЭ. При этом элементы, входящие в левые части последовательностей на каждом ПЭ, закрепляются за одним ПЭ и сортируются независимо от других элементов.

То есть элементы в позициях с 4 по 15 распределяется по 3 ПЭ и их сортировка выполняется аналогичным образом, независимо от элементов, входящих в левые последовательности на рассматриваемой стадии.

Результат распределения последовательностей по ПЭ представлен на Рис. 2.13.

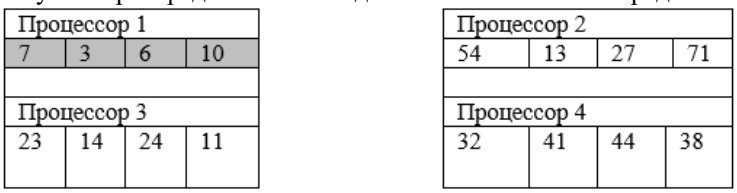

Рисунок 1.11 — Распределение данных по процессорным элементам

Вывод: на последующей итерации сортировка элементов на процессоре 1 и процессоре 2, 3, 4 выполняется независимо.

Реализация следующей итерации алгоритма (опорный элемент 38)

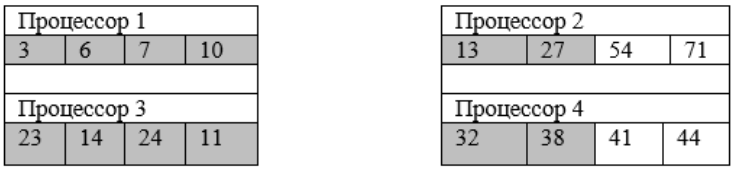

Рисунок 1.12 — Результат второй итерации

Итоговая последовательность после второй итерации алгоритма представлена на Рис. 2.13.

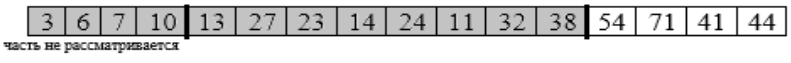

Рисунок 2.13 — Результирующая последовательность (второй этап)

На последующих шагах сортируется последовательность, полученная из левых частей на текущей итерации, и последовательность элементов правых частей (на одном из процессоров).

1.4 Сортировка с использованием регулярного набора образцов

Алгоритм сортировки с использованием регулярного набора образцов (the parallel sorting by regular sampling) является обобщением метода быстрой сортировки. Упорядочивание данных в соответствии с данным вариантом алгоритма быстрой сортировки осуществляется в ходе выполнения следующих четырех этапов:

1) на первом этапе сортировки производится упорядочивание имеющихся на процессорах блоков. Данная операция может быть выполнена каждым процессором независимо друг от друга при помощи обычного алгоритма быстрой сортировки; далее каждый процессор формирует набор из элементов своих блоков с индексами 0, *m*, *2m*, ..., *(p-1)m*, где *m=n/p2* ;

2) на втором этапе выполнения алгоритма все сформированные на процессорах наборы данных собираются на одном из процессоров системы и объединяются в ходе последовательного слияния в одно упорядоченное множество. Далее из полученного множества значений из элементов с индексами

$$p + [p/2] - 1,\ 2p + [p/2] - 1,\ \ldots,\ (p-1)p + [p/2]$$

формируется новый набор ведущих элементов, который передается всем используемым процессорам. В завершение этапа каждый процессор выполняет разделение своего блока на *p* частей с использованием полученного набора ведущих значений;

3) на третьем этапе сортировки каждый процессор осуществляет рассылку выделенных ранее частей своего блока всем остальным процессорам системы; рассылка выполняется в соответствии с порядком нумерации – часть *j*, 0<=*j*<*p*, каждого блока пересылается процессору с номером *j*;

4) на четвертом этапе выполнения алгоритма каждый процессор выполняет слияние *p* полученных частей в один отсортированный блок.

По завершении четвертого этапа исходный набор данных становится отсортированным.

Ниже приведен пример сортировки массива данных с помощью алгоритма, описанного выше. Следует отметить, что число процессоров для данного алгоритма может быть произвольным, в данном примере оно равно 3.

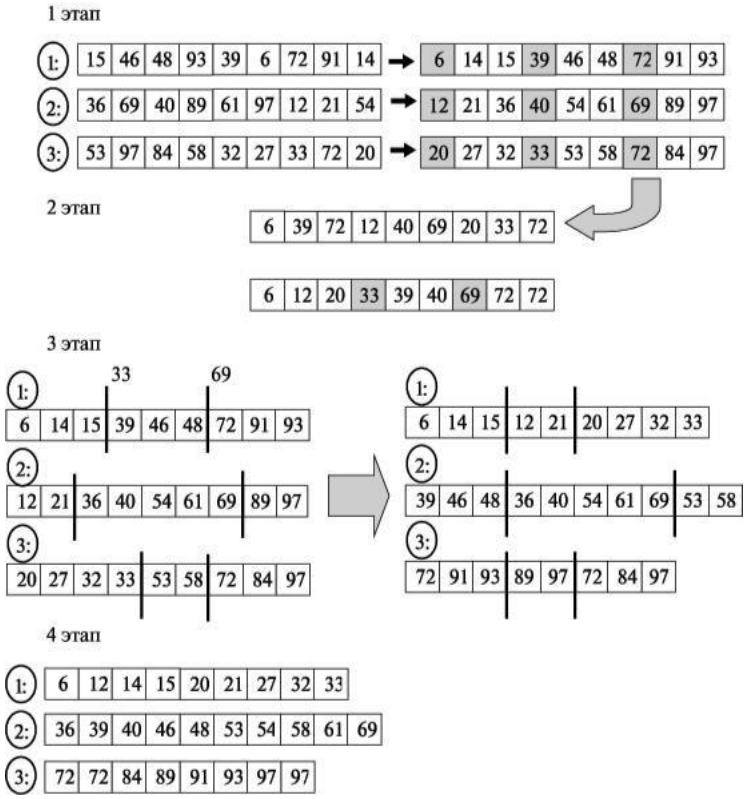

Рисунок 1.4 — Пример работы алгоритма сортировки с использованием регулярного набора образцов

2. Задание на работу.

Выполнить разработку и отладку программы быстрой сортировки данных с использованием вызовов требуемых функций библиотеки MPI для реализации варианта сортировки в соответствии с вариантом, указанным преподавателем. В качестве базового варианта реализовать также сортировку последовательным методом. Получить результаты работы программы в виде протоколов сообщений, комментирующих параллельное выполнение процессов и их взаимодействие в ходе выполнения. Оценить эффективность параллельного процесса сортировки в сравнении с последовательным на том же наборе исходных данных.

Таблица 7.1 — Варианты заданий

№ варианта	Вид сортировки

1	Быстрая сортировка
2	Модифицированная быстрая сортировка
3	Сортировка с использованиемрегулярного набора образцов

3. Контрольные вопросы

3.1. Назовите особенности реализации и использования рассматриваемых моделей взаимодействия распределенных процессов.
3.2. Сформулируйте понятие топологии кластера и остовного дерева, определите форматы рассылаемых сообщений, алгоритмы построения топологии кластера и остовного дерева, алгоритм рассылки.
3.3. Сформулируйте алгоритм реализации механизма «Распределенных семафоров», назначение логических часов и очереди сообщений, определите форматы передаваемых сообщений.
3.4. Сформулируйте алгоритм реализации модели «передачи маркера».