Алгоритмы параллельных вычислений. Сортировка слиянием (Algorithm M)

В. С. Верхотуров

БСБО-05-20
РТУ МИРЭА

6 октября 2022 г.

Содержание

Определение

Реализация сортировки в одном потоке

Вычислительная и временная сложность

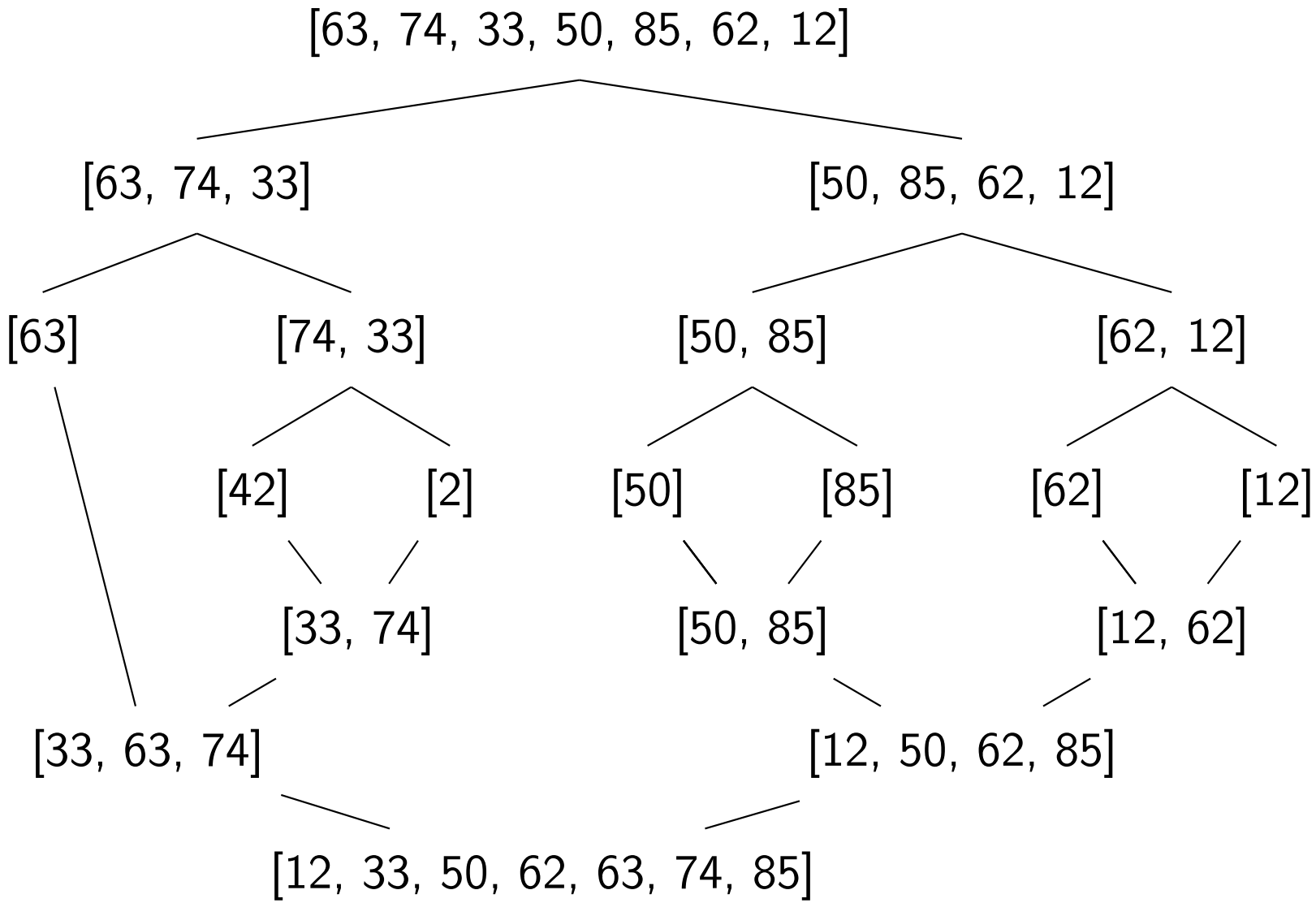
Параллельные вычисления в CPython

Многопроцессная реализация алгоритма слияния

Временная сложность многопроцессного алгоритма

Определение

Сортировка слиянием — это алгоритм «разделяй и властвуй».



$$\begin{cases} 33 & 63 & 74 & \\ 12 & 50 & 62 & 85 \end{cases} \tag{1}$$

$$12 \quad \begin{cases} 33 & 63 & 74 \\ 50 & 62 & 85 \end{cases} \tag{2}$$

$$12 \quad 33 \quad \begin{cases} 63 & 74 & \\ 50 & 62 & 85 \end{cases} \tag{3}$$

$$12 \quad 33 \quad 50 \quad \begin{cases} 63 & 74 \\ 62 & 85 \end{cases} \tag{4}$$

$$12 \quad 33 \quad 50 \quad 62 \quad \begin{cases} 63 & 74 \\ 85 & \end{cases} \tag{5}$$

$$\begin{array}{cccccccc} & & 12 & 33 & 50 & 62 & 63 & \begin{cases} 74 \\ 85 \end{cases} \end{array} \tag{6}$$

$$\begin{array}{cccccccc} & 12 & 33 & 50 & 62 & 63 & 74 & \begin{cases} \\ 85 \end{cases} \end{array} \tag{7}$$

$$\begin{array}{cccccccc} 12 & 33 & 50 & 62 & 63 & 74 & 85 & \begin{cases} \\ \\ \end{cases} \end{array} \tag{8}$$

Реализация однопоточного алгоритма на CPython

```
from random import randint
from numbers import Number

def merge(arrays: list[list[Number]]) -> list[Number]:
        assert len(arrays) == 2
        x, y = arrays
        index_x = index_y = 0
        out = []
        while index_x < len(x) and \
                        index_y < len(y):
                if x[index_x] < y[index_y]:
                        out.append(x[index_x])
                        index_x += 1
                else:
                        out.append(y[index_y])
                        index_y += 1
        out += x[index_x:] + y[index_y:]
        return out

def merge_sort(arr : list[Number]) -> list[Number]:
        if len(arr) <= 1:
                return arr
        if len(arr) == 2:
                return arr if arr[0] < arr[1] else [arr[1], arr[0]]
        mid = len(arr) // 2
        return merge(merge_sort(arr[:mid]), merge_sort(arr[mid:]))


def test_merge_sort():
        input_array = [randint(1, 100) for i in range(10)]
        print(merge_sort(input_array))

test_merge_sort()
```

Реализация однопоточного алгоритма на CPython. Часть 1

```
def test_merge_sort():
        input_array = [randint(1, 100)
                       for i in range(10)]
        print(merge_sort(input_array))

test_merge_sort()
```

Реализация однопоточного алгоритма на CPython. Часть 2

```
def merge_sort(arr : list[Number]) -> list[Number]:
        if len(arr) <= 1:
                return arr
        if len(arr) == 2:
                return arr
                        if arr[0] < arr[1]
                        else [arr[1], arr[0]]
        mid = len(arr) // 2
        return merge(merge_sort(arr[:mid]),
                     merge_sort(arr[mid:]))
```

Реализация однопоточного алгоритма на CPython. Часть 3

```
def merge(arrays: list[list[Number]]) \
                                    -> list[Number]:
        assert len(arrays) == 2
        x, y = arrays
        index_x = index_y = 0
        out = []
        while index_x < len(x) and \
                        index_y < len(y):
            if x[index_x] < y[index_y]:
                        out.append(x[index_x])
                        index_x += 1
            else:
                        out.append(y[index_y])
                        index_y += 1
        out += x[index_x:] + y[index_y:]
        return out
```

Вычислительная сложность

Худшая выч. сложность:

$$O(n \log n) \tag{9}$$

Лучшая выч. сложность:

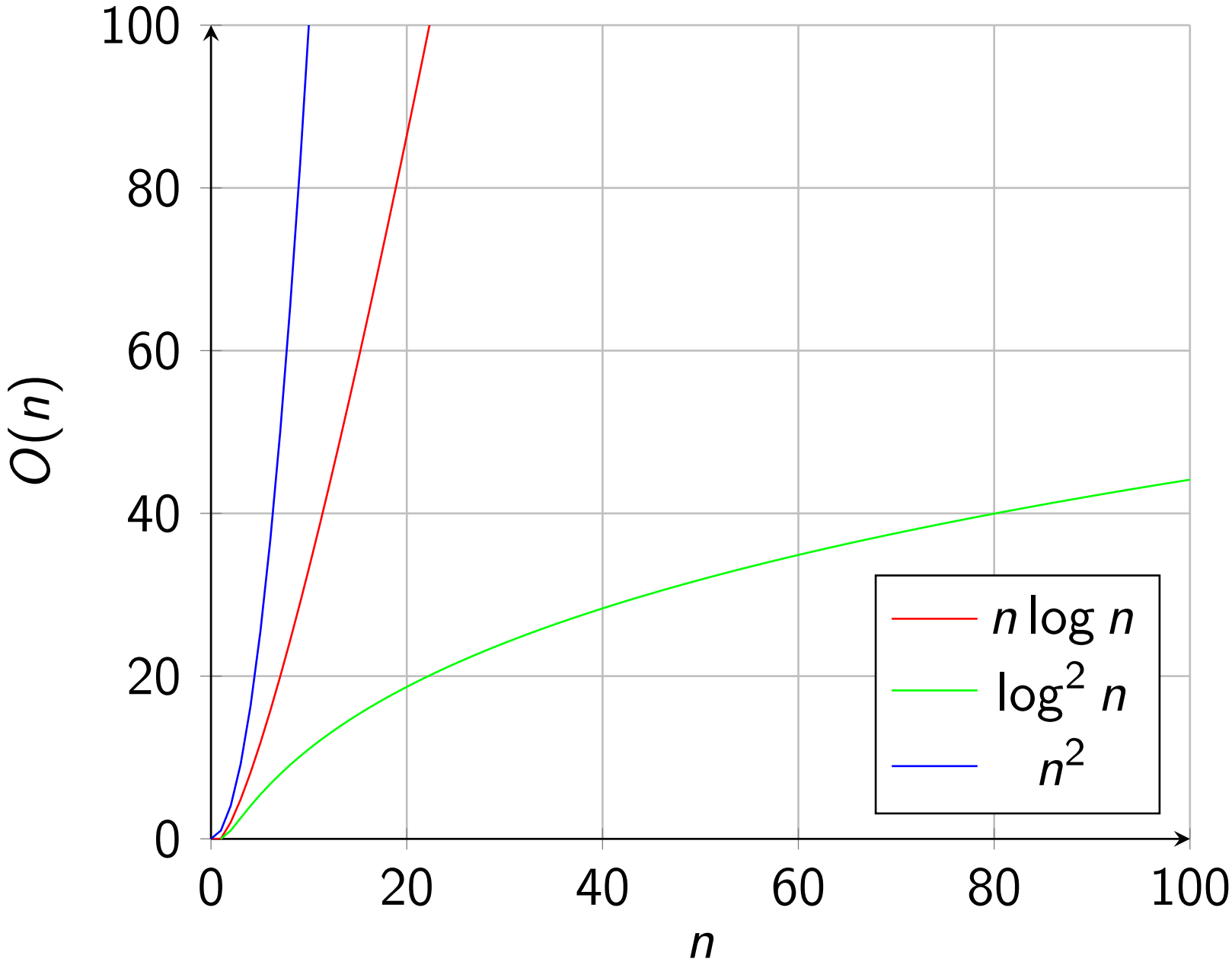
$$O(n \log n) \tag{10}$$

Средняя выч. сложность:

$$O(n \log n) \tag{11}$$

Временная сложность

$$T^{sort}(n) = 2T^{sort}\left(\frac{n}{2}\right) + T^{merge}(n) = \\ = T^{sort}\left(\frac{n}{2}\right) + \Theta(\log^2 n) \tag{12}$$


100
80
60
40
20
0
$O(n)$
0
20
40
60
80
100
n
$n \log n$
$\log^2 n$
n^2

Параллельные вычисления в CPython (side-stepping the GIL (Global Interpreter Lock))

Рис.: Иллюстрация GIL

- `multiprocessing.Process`, `multiprocessing.Pool` `https://docs.python.org/3/library/multiprocessing.html`,
- C-расширение, расширение на Cython
- `os.system("python␣child.py")`,

или...

Многопроцессная реализация алгоритма слияния

```
import multiprocessing
import math


def parallel_merge_sort(arr: list[Number]) -> list[Number]:
        processes = multiprocessing.cpu_count()
        pool = multiprocessing.Pool(processes = processes)
        size = math.ceil(len(arr) / processes)
        arr = [arr[i * size:(i + 1) * size]
                for i in range(processes)]
        arr = pool.map(merge_sort, arr)
        while len(arr) > 1:
                extra = arr.pop() if len(arr) % 2 == 1
                                  else None
                arr = [[arr[i], arr[i + 1]]
                        for i in range(0, len(arr), 2)]
                arr = pool.map(merge, arr) + \
                        ([extra] if extra else [])
        return arr[0]
```

Временная сложность

$$T^{sort}(n) = T^{sort}\left(\frac{n}{2}\right) + T^{merge}(n) = T^{sort}\left(\frac{n}{2}\right) + \Theta(\log^2 n) \quad (13)$$


The Art of
Computer
Programming.
Volume 3.
P. 158, 159