Parallel Merge Sort

Nguyễn Tuấn Anh, Nguyễn Trần Việt Anh*

Ngày 13 tháng 3 năm 2023

Tóm tắt nội dung

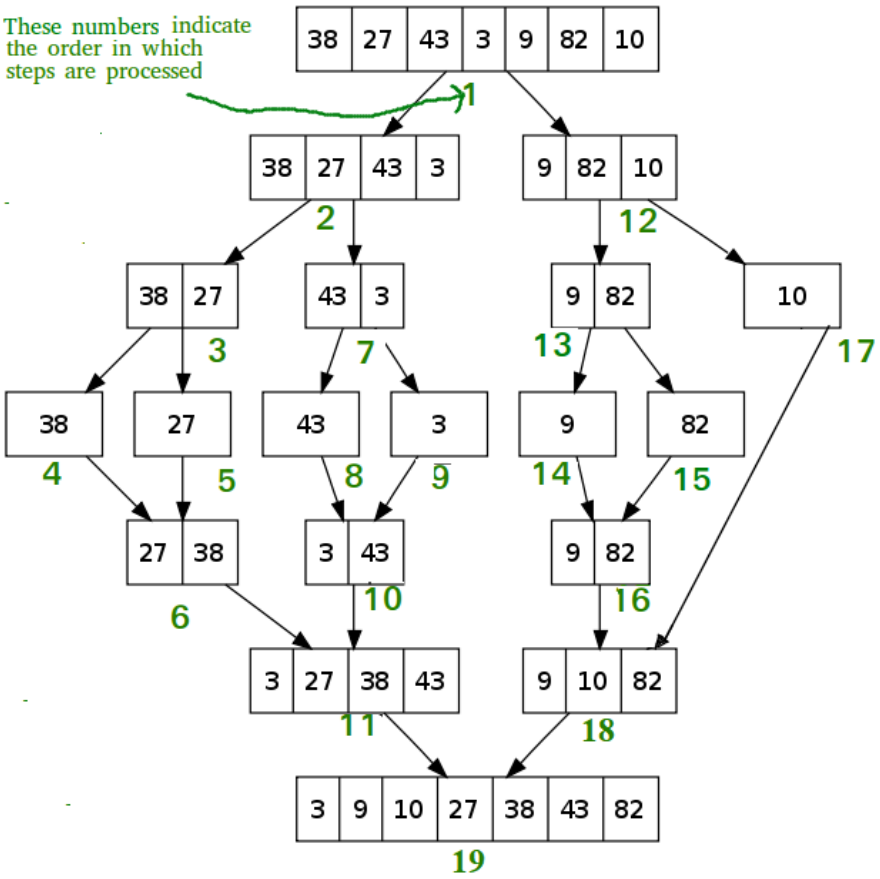
Lập trình song song trên các máy tính nhiều nhân nhằm giải quyết vấn đề nhanh hơn bằng cách khai thác nhiều bộ vi xử lý chạy cùng thời điểm. Các bài toán sắp xếp có ứng dụng rất lớn trong tin học. Việc ứng dụng phương pháp song song có thể tăng tốc chương trình khá nhiều. Ở đây chúng ta xem xét thuật toán Merge Sort.

Mục lục

*Sinh viên lớp KHTN2021, MSSV: 21520142 - 21520006

1. Nhắc lại về Merge Sort

Merge Sort là một thuật toán sắp xếp sử dụng ý tưởng chia để trị. Ý tưởng của thuật toán này là chia một mảng chưa được sắp xếp thành các mảng con, đã được sắp xếp, sau đó hợp chúng lại để tạo ra mảng ban đầu được sắp xếp.



Hình 1: Minh họa Merge Sort

Các bước để sắp xếp có thể được tóm gọn như sau:

1. **Nếu mảng con đủ nhỏ, return.**
2. **Liên tục chia mảng thành hai nửa, trái và phải.**
3. **Sau đó sắp xếp chúng.**
4. **Hợp hai mảng đã sắp xếp đến khi có được đầu ra.**

Chúng ta có nhận xét quan trọng sau.

> **Nhận xét.** Sự nhàn rỗi khi hợp các mảng là quá nhiều!!

2. Tiến trình trong Python

2.1. Python Multiprocessing Pool

Python Multiprocessing Pool cho phép tạo và quản lý các tiến trình trong Python.
Một tiến trình liên hệ đến một chương trình máy tính.
Mỗi chương trình Python là một tiến trình và có một luồng được gọi là luồng chính được sử dụng để thực hiện các chỉ thị. Thực tế, mỗi tiến trình là một phiên bản của trình thông dịch Python thực thi các lệnh Python.
Đôi khi chúng ta có thể cần tạo các tiến trình mới để chạy đồng thời các tác vụ. Python hỗ trợ điều này thông qua lớp Multiprocessing.
Tạo một tiến trình cơ bản

```python
# SuperFastPython.com
# example of running a function in a new process
from multiprocessing import Process

# a task to execute in another process
def task():
    print('This is another process', flush=True)

# entry point for the program
if __name__ == '__main__':
    # define a task to run in a new process
    p = Process(target=task)
    # start the task in a new process
    p.start()
    # wait for the task to complete
    p.join()
```

Hình 2: Tạo tiến trình

2.2. Process Pool

Process Pool là một **pattern** có chức năng tự động quản lý các tiến trình. Hỗ trợ trong một số tiến trình cố định. Kiểm soát thời điểm chúng được tạo ra, quản lý thời gian rảnh của tiến trình.
Pool có thể cung cấp giao diện chung để thực hiện các tác vụ với số lượng đối số không đổi nhưng không yêu cầu chọn tiến trình để chạy tác vụ, bắt đầu tiến trình khác hoặc đợi tác vụ hoàn thành.
Python cung cấp process pool thông qua lớp **multiprocessing.Pool**.

Một số tác vụ với lớp pool.

```
...
# create a process pool
pool = multiprocessing.pool.Pool(...)
```

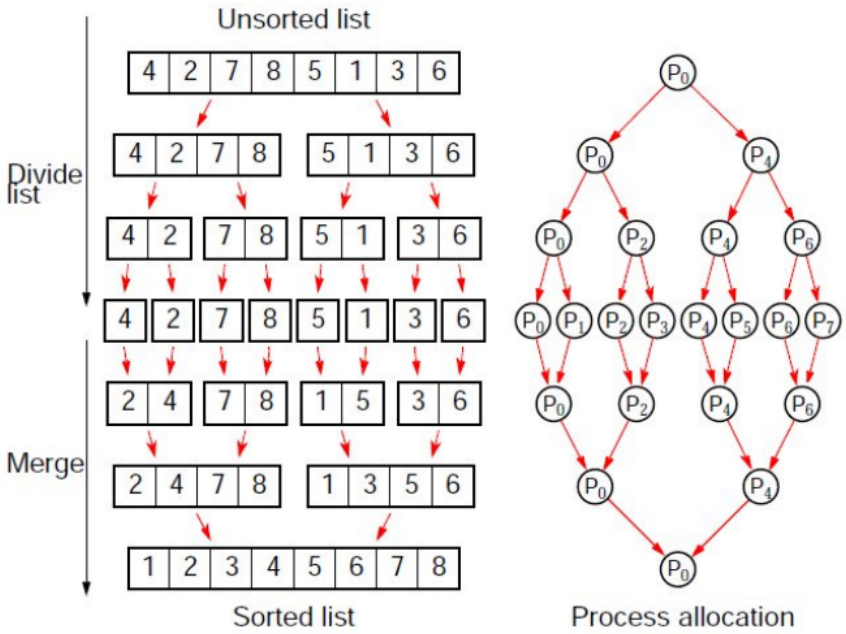
Hình 3: Tạo process pool

```
...
# issues tasks for execution
results = pool.map(task, items)
```

Hình 4: Tác vụ sẽ thực hiện

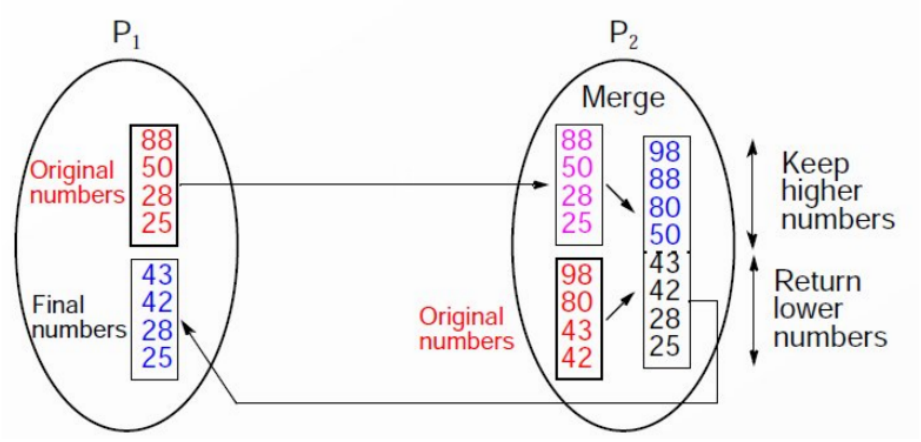
3. Parallel Merge Sort

Để ý cấu trúc dạng cây của thuật toán. Chúng ta có thể tối ưu bằng cách phân công công việc cho các tiến trình. Thực hiện bằng thuật toán song song.

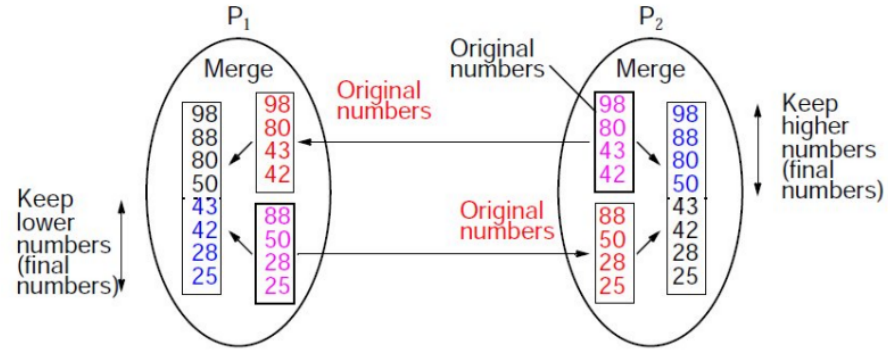


Ý tưởng song song sẽ áp dụng vào đâu? Như hình trên chính là một cây nhị phân. Ta hoàn toàn có thể sử dụng ý tưởng nhị phân. Tuy nhiên ta cũng có thể mở rộng sang cây $N-$ phân.
Về vấn đề hợp các mảng, có hai phương pháp thông dụng.

Version 1 – P_1 sends its list to P_2, which then performs the merge operation and sends back to P_1 the lower half of the merged list.



Version 2 – both processing units exchange their lists, then both perform the merge operation and P_1 keeps the lower half of the merged list and P_2 keeps the higher half of the merged list.



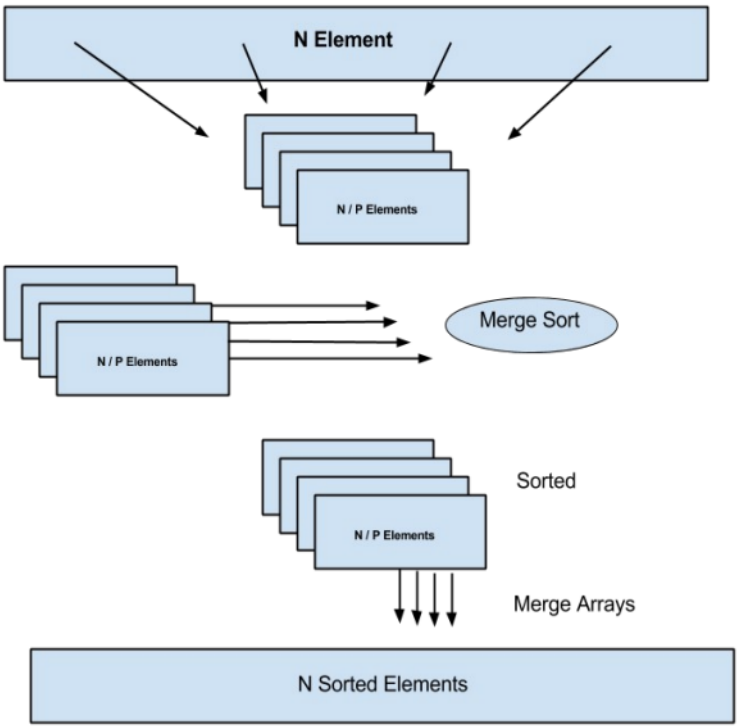
Trên thực tế version 2 nhanh hơn đáng kể so với version 1 do có thể trao đổi chậm ở P_i nào đó làm ảnh hưởng đến việc truyền lại P_j.

Vì vậy, chiến lược của chúng ta đó là

Chiến lược

Với P tiến trình được sử dụng ta chia mảng A có N phần tử thành $\frac{N}{P}$ phần, mỗi phần được quản lý bởi một tiến trình. Sau đó Merge Sort chúng và cuối cùng là hợp lại được mảng đã sắp xếp.

Hình dưới đây minh họa điều đó.



Bây giờ, chúng ta sẽ ước tính độ phức tạp của chiến lược trên. Để ý rằng nếu giả thiết rằng thời gian các tiến trình trao đổi không đáng kể, khi đó các phép tính chỉ xảy ra khi hợp các mảng. Trong trường hợp xấu nhất, phải mất tới $2s - 1$ bước để hợp các mảng con có kích thước s.
Có tổng cộng $\log(n)$ bước Merge, như vậy sẽ có

$$\sum_{i=1}^{\log(n)} (2^i - 1)$$

phép tính. Độ phức tạp này xấp xỉ $\mathrm{O}(n)$.

4. Chương trình minh họa

Đoạn code dưới đây xây dựng theo chiến lược đã trình bày.

```
def parallel_merge_sort(arr):
    # number of processes
    n_processes = 8
    # create a pool
    pool = Pool(processes=n_processes)
    # split arr into parts
    unsorted_parts = split(arr, n_processes)
    # sort each part in parallel
    sorted_parts = pool.map(merge_sort, unsorted_parts)

    while (len(sorted_parts) > 1):
        # split the array into contiguous pairs
        pairs = [[sorted_parts[i], sorted_parts[i+1]]
                 for i in range(0, len(sorted_parts)-1, 2)]
        remain = []
        # if the size of sorted_parts is odd, the last part is unmerged
        if len(sorted_parts) % 2 == 1:
            remain = sorted_parts[-1]
        # merge each pair
        sorted_parts = pool.map(merge_wrap, pairs)
        # if the last part is unmerged, append it to the final array
        if (len(remain)):
            sorted_parts.append(remain)


    return sorted_parts[0]
```

Hình 5: Hàm parallel merge sort

Toàn bộ demo code đầy đủ và báo cáo được lưu trữ tại Github.

Tài liệu

[1] Parallel Merge Sort:
https://www.sjsu.edu/people/robert.chun/courses/cs159/s3/T.pdf

[2] Parallel Sorting Algorithms:
https://www.dcc.fc.up.pt/~ricroc/aulas/1516/cp/apontamentos/slides_sorting.pdf

[3] Approaches to the Parallelization of Merge Sort in Python:
https://arxiv.org/pdf/2211.16479.pdf