Parallel Mergesort

documentație

Acest document prezintă evaluarea performanței algoritmului de Mergesort implementat în limbajul Java, folosind Threads pentru a-l rula concurent.

Specificații

Fiind dat un şir de valori, numerele sunt sortate folosind o strategie de Divide and Conquer, care partiționeaza şirul inițial în două partiții de dimensiuni (aproape) egale, urmând ca la fiecare pas recursiv cele două partiții la rândul lor să fie împărțite în alte două de dimensiuni (aproape) egale, iar când acestea au dimensiunea 2, se vor păstra sortate, urmând ca la revenirea din recursie să se facă procedura de *merge* a celor două porțiuni sortate.

```
MERGE(X, Y)
    L_X ← LENGTH(X)
    L_Y ← LENGTH(Y)
    L_Z ← L_X + L_Y
    Z ← [L_Z]
    i, j, k ← 0, 0, 0

    while  k < L_Y
        if  i < L_X ∧ j ≥ L_Y
                Z[k] ← X[i]
                i ← i + 1
        else-if  i ≥ L_X ∧ j < L_Y
                Z[k] ← Y[j]
                j ← j + 1
        else-if  i < L_X ∧ j < L_Y
            if  X[i] ≤ Y[j]
                Z[k] ← X[i]
                i ← i + 1
            else
                Z[k] ← Y[j]
                j ← j + 1
        k ← k + 1

    return  Z
```

```
MERGE-SORT(X)
    L ← LENGTH(X)
    if  L ≤ 1
        return  X

    return  MERGE(
        MERGE-SORT(
            PARTITION(X, 0, ⌊ L/2⌋ + L  mod 2)
        ),
        MERGE-SORT(
            PARTITION(X, ⌊ L/2⌋ + L  mod 2, ⌊ L/2⌋)
        )
    )
```

```
PARTITION(X, s, L)
    Y ← [L]
    k ← 0

    while  k < L
        Y[k] ← X[s + k]
        k ← k + 1

    return  Y
```

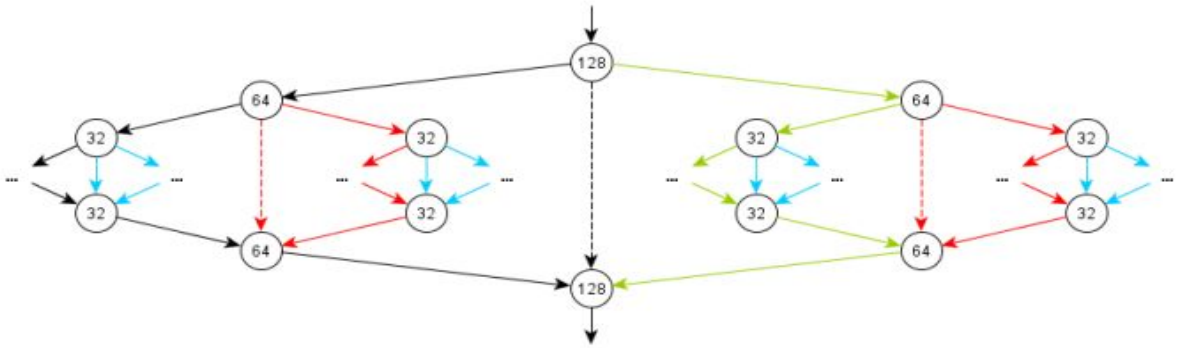
[1]

Time complexity: O(NlogN)
Space complexity: O(N)

Abordarea paralelă

Implementarea paralelă, folosind Threads, se foloseşte de faptul că porțiunea de *Divide* poate fi usor paralelizată, sortând o partiție în thread-ul curent, şi cealaltă partiție într-un alt thread. În momentul în care cel de-al doilea thread şi-a terminat treaba, apare operaţiunea de *join* a celor două thread-uri, şi cele două porţiuni sunt supuse operaţiei de *merge*.

Ilustrăm procesul descris mai sus printr-o imagine.

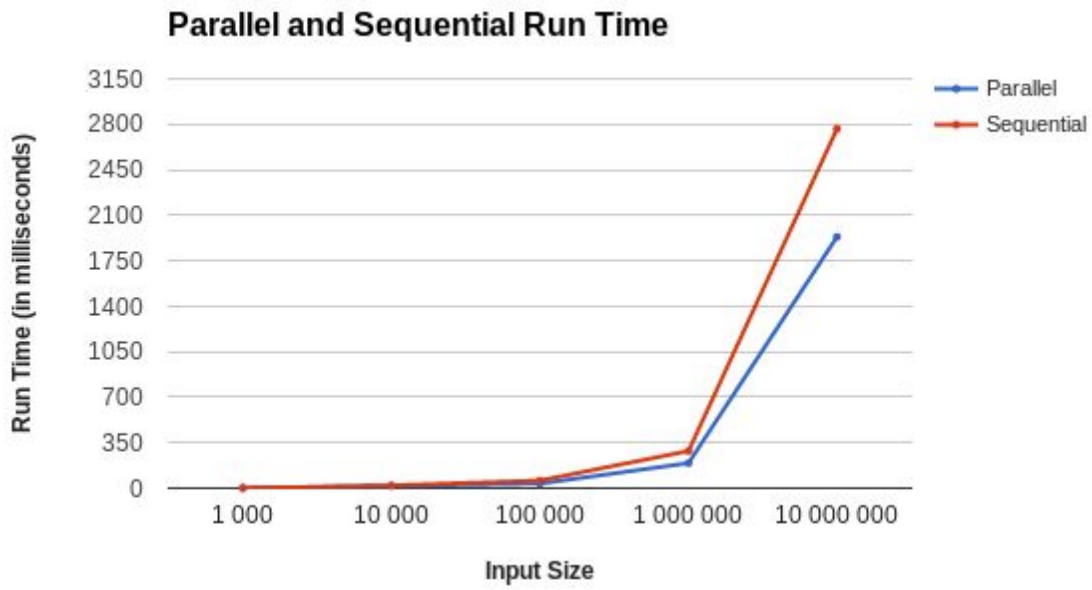


[2]

Menţionăm că în cazul versiunii paralele, complexitatea timp a devenit O(N), considerându-se că instrucţiunile executate în cele două thread-uri se execută simultan, fiind privită ca timp liniar.

Performance analysis

Input Size	**Parallel Run Time** (ms)	**Sequential Run Time** (ms)
1 000	1	1
10 000	17	21
100 000	35	59
1 000 000	193	286
10 000 000	1934	2766



Bibliografie

[1] https://en.wikipedia.org/wiki/Merge_sort

[2] https://antimatroid.wordpress.com/2012/12/01/parallel-merge-sort-in-java/