Université Sidi Mohamed Ben Abdellah

Faculté des sciences Dhar El Mahrez Fès

Master Big Data Analytics & Smart Systems

Traitement Parallèle

TP6 : Déroulement des algorithmes parallèle

Réalisé par (grp 6) :

Ayoub BERRAG

Mohamed EL BAGHDADI

Encadré par :

Professeur

Mohamed MEKNASSI

SOMMAIRE

INTRODUCTION

Dans ce travail pratique, on va dérouler des algorithmes qu'on vu dans le cours, ces algorithmes sont : Tri séquentiel (QuickSort), Tri splite and merge, EREW Sort et CRCW Sort. On va voir l'algorithme de chaque procédure et on va voir son fonctionnement dans un exemple qu'on va traiter. Ce genre des algorithmes permet d'exploiter les capacités multitâches de la machine, Certains algorithmes permettent d'exploiter les capacités multitâches de la machine. Notons également que certains algorithmes, notamment ceux qui fonctionnent par insertion, peuvent être lancés sans connaître l'intégralité des données à trier ; on peut alors trier et produire les données à trier en parallèle.

TRI SEQUENTIEL (QUICKSORT)

Définition de QuickSort

Le tri rapide (en anglais quicksort) est un algorithme de tri inventé par C.A.R. Hoare en 19611 et fondé sur la méthode de conception diviser pour régner. Il est généralement utilisé sur des tableaux, mais peut aussi être adapté aux listes. Dans le cas des tableaux, c'est un tri en place mais non stable.

La complexité moyenne du tri rapide pour n éléments est proportionnelle à n log n, ce qui est optimal pour un tri par comparaison, mais la complexité dans le pire des cas est quadratique. Malgré ce désavantage théorique, c'est en pratique un des tris les plus rapides, et donc un des plus utilisés. Le cas le pire est en effet peu probable lorsque l'algorithme est correctement mis en œuvre et il est possible de s'en prémunir définitivement avec la variante Introsort.

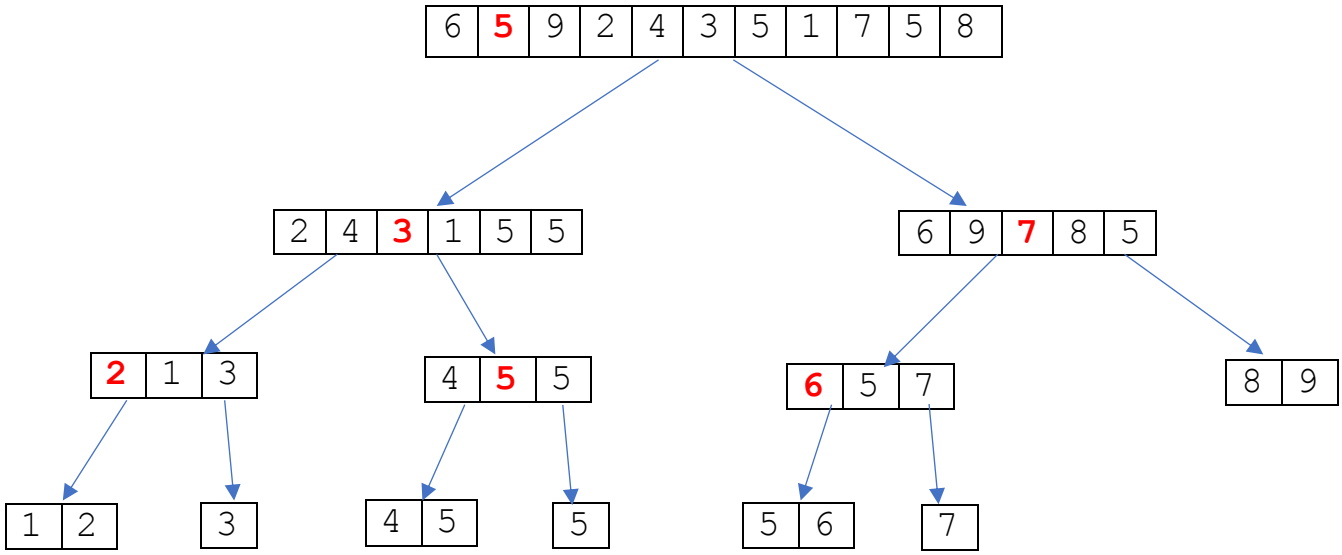
Le tri rapide ne peut cependant pas tirer avantage du fait que l'entrée est déjà presque triée. Dans ce cas particulier, il est plus avantageux d'utiliser le tri par insertion ou l'algorithme smoothsort.

Algorithme

```
procedure QUICKSORT (S)
   if ( |S| = 2 and s2 < s1 )
         then s1 ↔ s2
   else if ( |S| > 2 ) then
         (1) {Determine m, the median of S}
               m = SEQUENTIAL-SELECT(S,
(⌈|S|/2⌉)
         (2) {Split S into two subsequence
S1 and S2}
               (2.1) S1 ← {si : si ≤ m} and
|S1| = ⌈|S|/2⌉
               (2.2) S2 ← {si : si ≥ m} and
|S2| = ⌊|S|/2⌋
         (3) QUICKSORT(S1)
         (4) QUICKSORT(S2)
   end if
```

Jeu d'essai

On applique maintenant ce tri, on prend un tableau qui contient 11 entiers, on visualise le déroulement de notre algorithme comme suit.



TRI SPLITE AND MERGE

Algorithme

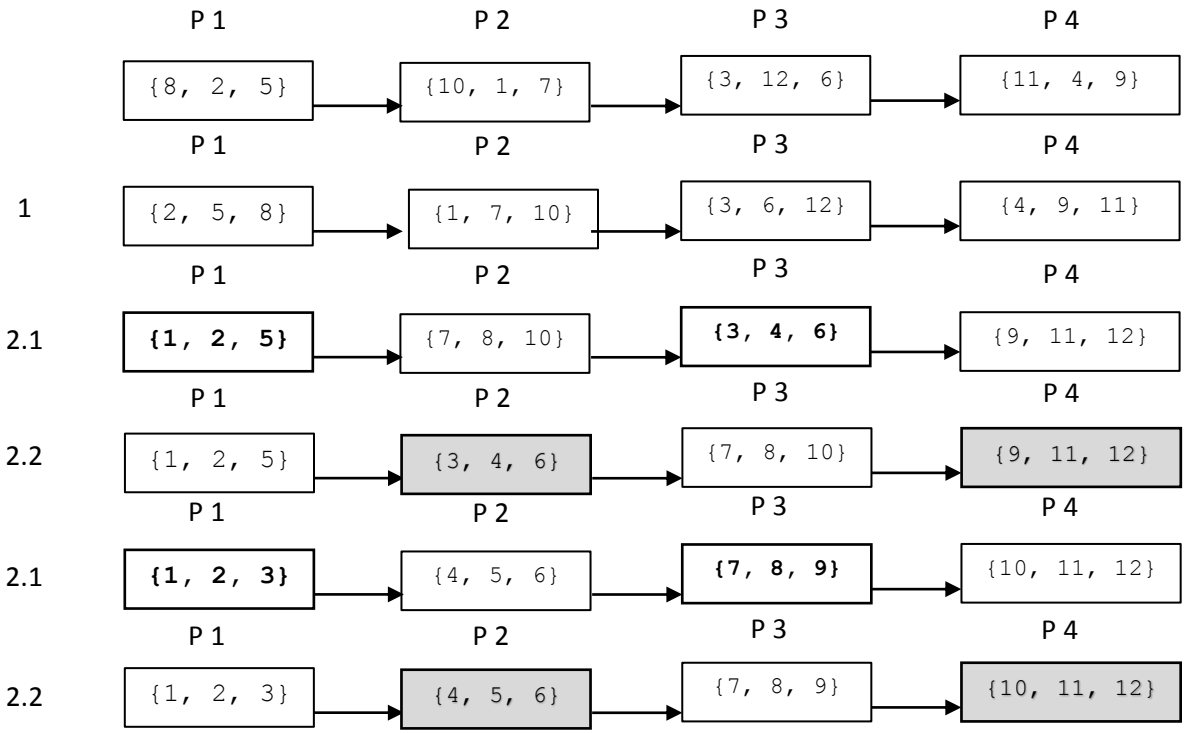
```
procedure MERGE SPLIT(S)

Step 1: for i = 1 to N do in parallel
            QUICKSORT(Si)
        end for
Step 2: for j = 1 to ⌈n/2⌉ do
       (2.1) for i = 1, 3, …, 2⌊n/2⌋ - 1 do in parallel
                (i) SEQUENTIAL-MERGE(Si, Si+1, S'i)
                (ii) Si ← {S'1, S'2, …, S'n/N }
                (iii) Si+1 ← {S'(n/N)+1, S'(n/N)+2, …, S'2n/N }
             end for
       (2.2) for i = 2, 4, …, 2⌊(n-1)/2⌋ do in parallel
                (i) SEQUENTIAL-MERGE(Si, Si+1, S'i)
                (ii) Si ← {S'1, S'2, …, S'n/N }
                (iii) Si+1 ← {S'(n/N)+1, S'(n/N)+2, …, S'2n/N }end
for
```

```
            end for
        end for
```

Jeu d'essai

L'exemple suivant illustre le fonctionnement de l'algorithme.



EREW SORT

Définition de EREW

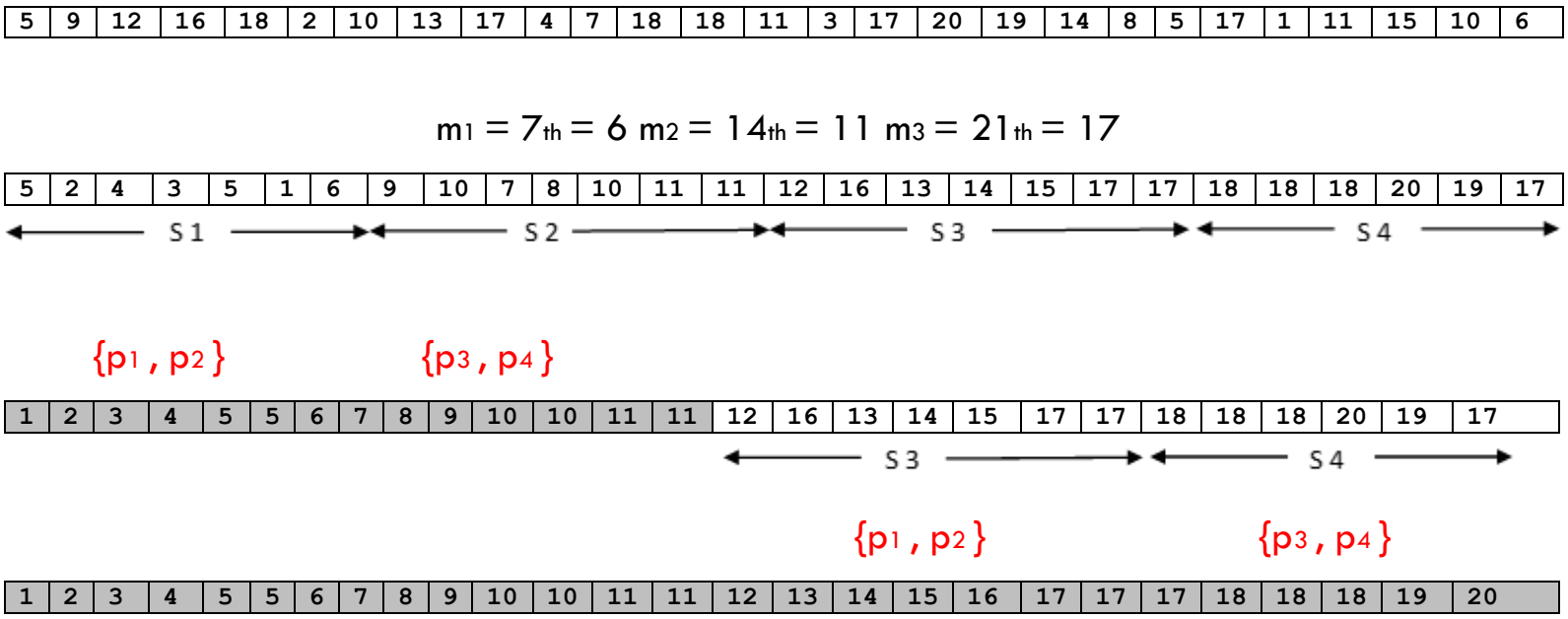
EREW : Exclusive Read, Exclusive Write : chaque processeur ne peut lire ou écrire à un endroit de la mémoire que si aucun autre processeur n'y accède à ce moment-là.

Algorithme

```
procedure EREW SORT(S)
   if |S| ≤ k
   then QUICKSORT(S)
   else (1) for i = 1 to k − 1 do
            PARALLEL SELECT(S, ⌈i|S|/k⌉) {obtain mi}
            end for
        (2) S1 ← {s ∈ S: s ≤ m1}
        (3) for i = 2 to k − 1 do
            Si ← {s ∈ S: mi-1 ≤ s ≤ mi}
            end for
        (4) Sk ← {s ∈ S: s ≥ mk-1}
        (5) for i = 1 to k/2 do in parallel
                EREW SORT(Si)
            end for
        (6) for i = (k/2) + 1 to k do in parallel
                EREW SORT(Si)
            end for
  end if
```

Jeu d'essai

Ce qui suit est un exemple d'application de l'algorithme donné au-dessous :



CRCW SORT

Définition

CRCW : Concurrent Read, Concurrent Write : chaque processeur peut lire et écrire n'importe où dans la mémoire à tout moment.

Algorithme

```
procedure CRCW SORT(S)

Step 1: for i = 1 to n do in parallel
            for j = 1 to n do in parallel
                if (si > sj) or (si = sj and i > j)
                then P(i, j) writes 1 in ci
                else P(i, j) writes 0 in ci
                end if
            end for
        end for
Step 2: for i = 1 to n do in parallel
            P(i, 1) stores si in position 1 + ci of S
        end for
```

Jeu d'essai

On va maintenant appliquer l'algorithme sur l'exemple suivant :