

Une parallélisation du QuickSort

Sophie Robert

UFR ST Département informatique

Quick Sort : séquentiel

```
QuickSort(A,q,r)
  if q<r then
    pivot:=A[q];
    s:=q;
    for i from q+1 to r
      if pivot > A[i] then
        s:=s+1;
        swap(A[s],A[i])
      end if
    end for
    swap(A[q],A[s]);
    QuickSort(A,q,s);
    QuickSort(A,s+1,r);
  end if
```

33 > 21

33 21 13 54 82 31 40 72

↑

s

Quick Sort : séquentiel

```
QuickSort(A,q,r)
  if q<r then
    pivot:=A[q];
    s:=q;
    for i from q+1 to r
      if pivot > A[i] then
        s:=s+1;
        swap(A[s],A[i])
      end if
    end for
    swap(A[q],A[s]);
    QuickSort(A,q,s);
    QuickSort(A,s+1,r);
  end if
```

33 21 13 54 82 31 40 72

↑
s

33>13

Quick Sort : séquentiel

```
QuickSort(A,q,r)
```

```
  if q<r then
```

```
    pivot:=A[q];
```

```
    s:=q;
```

```
    for i from q+1 to r
```

```
      if pivot > A[i] then
```

```
        s:=s+1;
```

```
        swap(A[s],A[i])
```

```
      end if
```

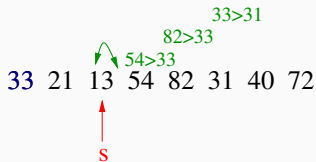
```
    end for
```

```
    swap(A[q],A[s]);
```

```
    QuickSort(A,q,s);
```

```
    QuickSort(A,s+1,r);
```

```
  end if
```



Quick Sort : séquentiel

```
QuickSort(A,q,r)
```

```
  if q<r then
```

```
    pivot:=A[q];
```

```
    s:=q;
```

```
    for i from q+1 to r
```

```
      if pivot > A[i] then
```

```
        s:=s+1;
```

```
        swap(A[s],A[i])
```

```
      end if
```

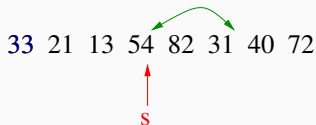
```
    end for
```

```
    swap(A[q],A[s]);
```

```
    QuickSort(A,q,s);
```

```
    QuickSort(A,s+1,r);
```

```
  end if
```



Quick Sort : séquentiel

```
QuickSort(A,q,r)
  if q<r then
    pivot:=A[q];
    s:=q;
    for i from q+1 to r
      if pivot > A[i] then
        s:=s+1;
        swap(A[s],A[i])
      end if
    end for
    swap(A[q],A[s]);
    QuickSort(A,q,s);
    QuickSort(A,s+1,r);
  end if
```

33 21 13 31 82 54 40 72

72>33
40>33

↑
s

Quick Sort : séquentiel

```
QuickSort(A,q,r)
```

```
  if q<r then
```

```
    pivot:=A[q];
```

```
    s:=q;
```

```
    for i from q+1 to r
```

```
      if pivot > A[i] then
```

```
        s:=s+1;
```

```
        swap(A[s],A[i])
```

```
      end if
```

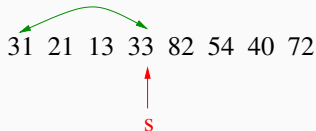
```
    end for
```

```
    swap(A[q],A[s]);
```

```
    QuickSort(A,q,s);
```

```
    QuickSort(A,s+1,r);
```

```
  end if
```



Quick Sort : séquentiel

```
QuickSort(A,q,r)
```

```
  if q<r then
```

```
    pivot:=A[q];
```

```
    s:=q;
```

```
    for i from q+1 to r
```

```
      if pivot > A[i] then
```

```
        s:=s+1;
```

```
        swap(A[s],A[i])
```

```
      end if
```

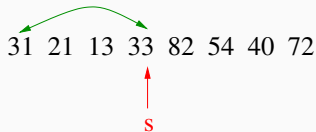
```
    end for
```

```
    swap(A[q],A[s]);
```

```
    QuickSort(A,q,s);
```

```
    QuickSort(A,s+1,r);
```

```
  end if
```



Complexité

Si on suppose que le découpage est bien équilibré

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = 2^{\log_2 n} + (\log_2 n)O(n)$$

$$T(n) = O(n \log_2 n)$$

Si on suppose le pire cas

$$T(n) = T(n-1) + O(n)$$

$$T(n) = O(n^2)$$

Le découpage

La complexité vient du découpage : comment paralléliser le découpage ?

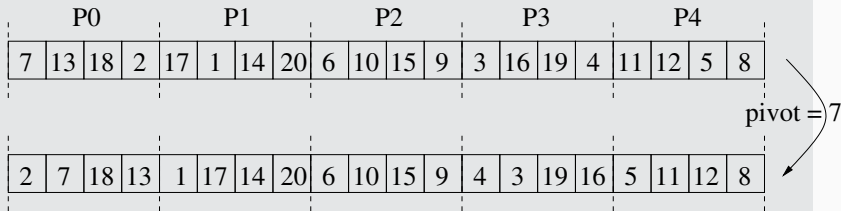
Quick Sort : Machine à mémoire partagée

Vers une version OpenMP

7	13	18	2	17	1	14	20	6	10	15	9	3	16	19	4	11	12	5	8
---	----	----	---	----	---	----	----	---	----	----	---	---	----	----	---	----	----	---	---

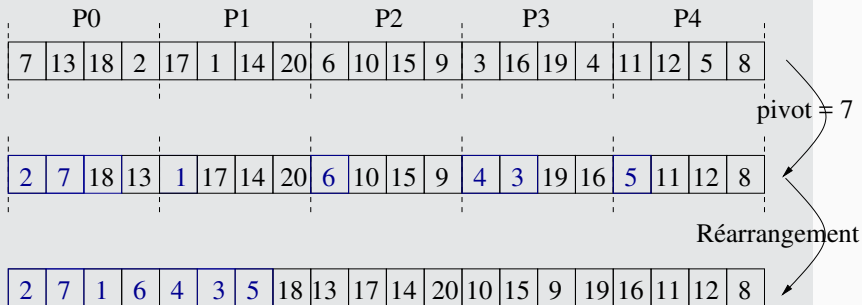
Quick Sort : Machine à mémoire partagée

Vers une version OpenMP



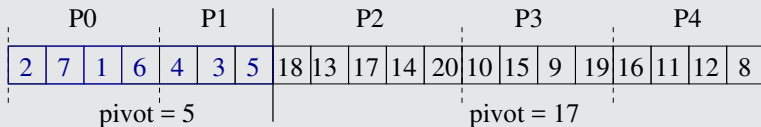
Quick Sort : Machine à mémoire partagée

Vers une version OpenMP



Quick Sort : Machine à mémoire partagée

Vers une version OpenMP



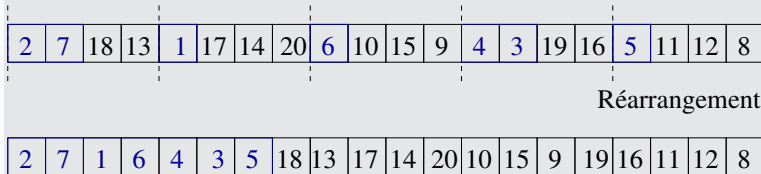
Quick Sort : Machine à mémoire partagée

Les étapes de l'algorithme avec t threads

- $\log_2(t)$ étapes
 - * partition pour un pivot donné
 - * participation au réarrangement global (?)
- quick sort local

Quick Sort : Machine à mémoire partagée

Le réarrangement global



Quick Sort : Machine à mémoire partagée

Le réarrangement global

2	7	18	13	1	17	14	20	6	10	15	9	4	3	19	16	5	11	12	8
---	---	----	----	---	----	----	----	---	----	----	---	---	---	----	----	---	----	----	---

(Smaller :)|Si| : 2 1 1 2 1

2 3 3 2 3 |Li| (: Larger)

Somme des préfixes Q : 0 2 3 4 6 7 (gauche)

R : 0 2 5 8 10 13 (droit)

2	7	18	13	1	17	14	20	6	10	15	9	4	3	19	16	5	11	12	8
0		2		3		3		4		4		6		6					

2	7	18	13	1	17	14	20	6	10	15	9	4	3	19	16	5	11	12	8
	7+0		7+2		7+5				7+8		7+10								

2	7	1	6	4	3	5	18	13	17	14	20	10	15	9	19	16	11	12	8
---	---	---	---	---	---	---	----	----	----	----	----	----	----	---	----	----	----	----	---

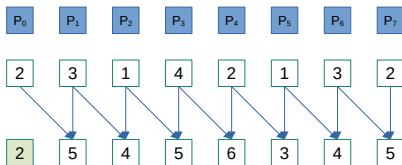
Quick Sort : Machine à mémoire partagée

La somme des préfixes

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
2	3	1	4	2	1	3	2

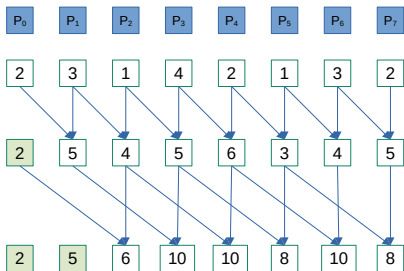
Quick Sort : Machine à mémoire partagée

La somme des préfixes



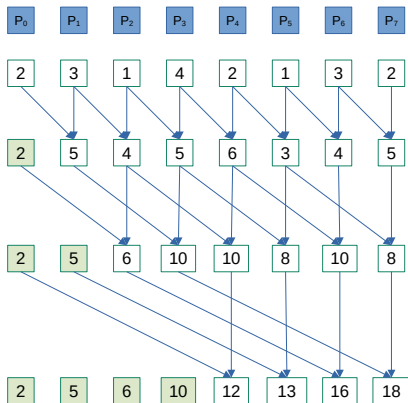
Quick Sort : Machine à mémoire partagée

La somme des préfixes



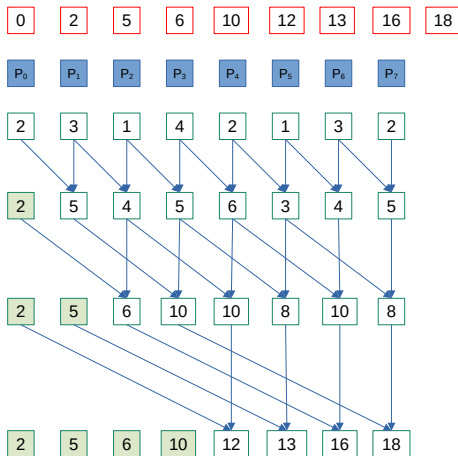
Quick Sort : Machine à mémoire partagée

La somme des préfixes



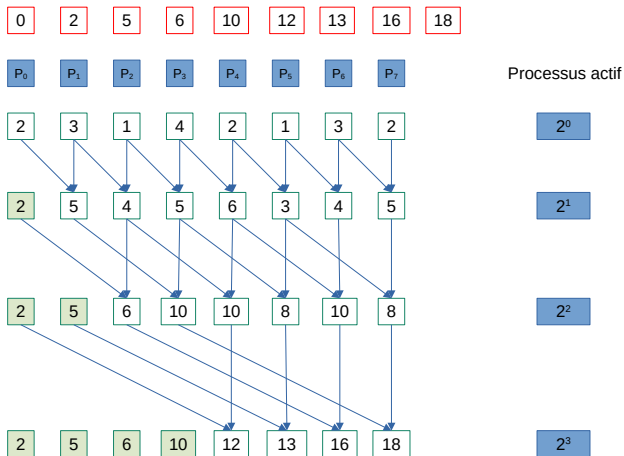
Quick Sort : Machine à mémoire partagée

La somme des préfixes



Quick Sort : Machine à mémoire partagée

La somme des préfixes



Quick Sort : Machine à mémoire partagée

Algorithme final sur t threads

Si on note $\text{QuickSortElt}(P_i, \dots, P_j, q, r)$ une étape élémentaire

1. $\text{QuickSortElt}(P_0, \dots, P_{p-1}, 0, n-1)$
2. $\text{QuickSortElt}(P_0, \dots, P_A, 0, Q[p]) +$
 $\text{QuickSortElt}(P_{A+1}, \dots, P_p, Q[p+1], n)$
3. ...
4. Si $\text{QuickSortElt}(P_i, q, r)$ alors QuickSort séquentiel et exit.

Quick Sort : Machine à mémoire partagée

La répartition des QuickSortElt

$$\left| \begin{array}{cc} n & p \\ |S| & \frac{|S|p}{n} \end{array} \right|$$
$$A = \lceil |S| \frac{p}{n} + 0.5 \rceil$$

Quick Sort : Machine à mémoire partagée

Algorithme final sur t threads

Si on note $\text{QuickSortElt}(P_i, \dots, P_j, q, r)$ une étape élémentaire

1. $\text{QuickSortElt}(P_0, \dots, P_{p-1}, 0, n-1)$
2. $\text{QuickSortElt}(P_0, \dots, P_A, 0, Q[p]) +$
 $\text{QuickSortElt}(P_{A+1}, \dots, P_p, Q[p+1], n)$
3. ...
4. Si $\text{QuickSortElt}(P_i, q, r)$ alors QuickSort séquentiel et exit.

Complexité

Quick Sort : Machine à mémoire partagée

Algorithme final sur t threads

Si on note $\text{QuickSortElt}(P_i, \dots, P_j, q, r)$ une étape élémentaire

1. $\text{QuickSortElt}(P_0, \dots, P_{p-1}, 0, n-1)$
2. $\text{QuickSortElt}(P_0, \dots, P_A, 0, Q[p]) +$
 $\text{QuickSortElt}(P_{A+1}, \dots, P_p, Q[p+1], n)$
3. ...
4. Si $\text{QuickSortElt}(P_i, q, r)$ alors QuickSort séquentiel et exit.

Complexité

- Nombre d'étapes

$$\log_2(t)(\dots)$$

Quick Sort : Machine à mémoire partagée

Algorithme final sur t threads

Si on note $\text{QuickSortElt}(P_i, \dots, P_j, q, r)$ une étape élémentaire

1. $\text{QuickSortElt}(P_0, \dots, P_{p-1}, 0, n-1)$
2. $\text{QuickSortElt}(P_0, \dots, P_A, 0, Q[p]) +$
 $\text{QuickSortElt}(P_{A+1}, \dots, P_p, Q[p+1], n)$
3. ...
4. Si $\text{QuickSortElt}(P_i, q, r)$ alors QuickSort séquentiel et exit.

Complexité

- Partition

$$\log_2(t) \left(O\left(\frac{n}{t}\right) + \dots \right)$$

Quick Sort : Machine à mémoire partagée

Algorithme final sur t threads

Si on note $\text{QuickSortElt}(P_i, \dots, P_j, q, r)$ une étape élémentaire

1. $\text{QuickSortElt}(P_0, \dots, P_{p-1}, 0, n-1)$
2. $\text{QuickSortElt}(P_0, \dots, P_A, 0, Q[p]) +$
 $\text{QuickSortElt}(P_{A+1}, \dots, P_p, Q[p+1], n)$
3. ...
4. Si $\text{QuickSortElt}(P_i, q, r)$ alors QuickSort séquentiel et exit.

Complexité

- Réarrangement : somme des préfixes

$$\log_2(t) \left(O\left(\frac{n}{t}\right) + O(\log_2(t)) + \dots \right)$$

Quick Sort : Machine à mémoire partagée

Algorithme final sur t threads

Si on note $\text{QuickSortElt}(P_i, \dots, P_j, q, r)$ une étape élémentaire

1. $\text{QuickSortElt}(P_0, \dots, P_{p-1}, 0, n-1)$
2. $\text{QuickSortElt}(P_0, \dots, P_A, 0, Q[p]) +$
 $\text{QuickSortElt}(P_{A+1}, \dots, P_p, Q[p+1], n)$
3. ...
4. Si $\text{QuickSortElt}(P_i, q, r)$ alors QuickSort séquentiel et exit.

Complexité

- Réarrangement : copie

$$\log_2(t) \left(O\left(\frac{n}{t}\right) + O(\log_2(t)) + O\left(\frac{n}{t}\right) \right)$$

Quick Sort : Machine à mémoire partagée

Algorithme final sur t threads

Si on note $\text{QuickSortElt}(P_i, \dots, P_j, q, r)$ une étape élémentaire

1. $\text{QuickSortElt}(P_0, \dots, P_{p-1}, 0, n-1)$
2. $\text{QuickSortElt}(P_0, \dots, P_A, 0, Q[p]) +$
 $\text{QuickSortElt}(P_{A+1}, \dots, P_p, Q[p+1], n)$
3. ...
4. Si $\text{QuickSortElt}(P_i, q, r)$ alors QuickSort séquentiel et exit.

Complexité

- Quick sort séquentiel

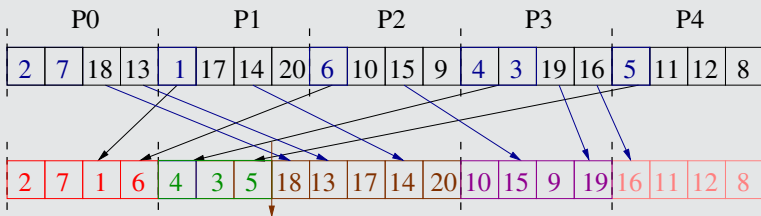
$$\log_2(t) \left(O\left(\frac{n}{t}\right) + O\left(\frac{n}{t}\right) + O(\log_2(t)) \right) + O\left(\frac{n}{t} \log_2\left(\frac{n}{t}\right)\right)$$

Quick Sort : Message Passing Paradigm

Mémoire partagée versus mémoire distribuée

- La diffusion du pivot
- Le réarrangement global

Définir les processus destinataires



Quick Sort : Message Passing Paradigm

Mémoire partagée versus mémoire distribuée

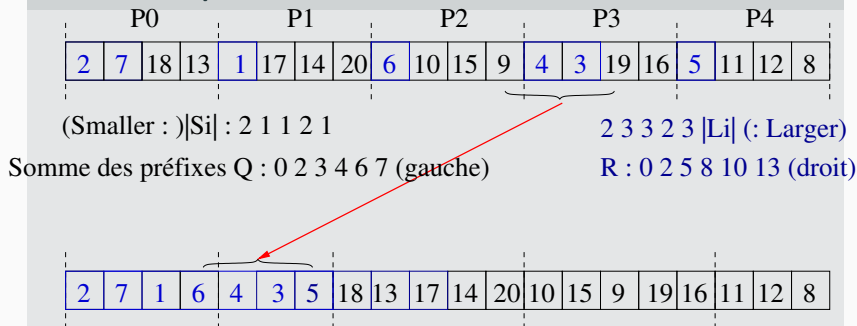
- La diffusion du pivot
- Le réarrangement global

Définir les processus destinataires

- Définir une règle à partir de S et Q pour distribuer les éléments plus petits que le pivot
- Définir une règle à partir de L et R pour distribuer les éléments plus grands que le pivot

Quick Sort : Message Passing Paradigm

Définir les processus destinataires



- 3 ou 4 éléments par processeurs ($\lceil Q[5]/2 \rceil$)
- 2 éléments à distribuer ($S[3]$)
- 4 éléments déjà distribués ($Q[3] - S[3]$)

* e_4 et e_5 sur le processeur 1 (4/4 et 5/4).

Quick Sort : Message Passing Paradigm

Définir les processus destinataires

P_i	S	Q	les éléments	distribution
P_0	2	0	$e_0 e_1$	$e_0 e_1 e_2 e_3$
P_1	1	2	e_2	$e_4 e_5 e_6$
P_2	1	3	e_3	
P_3	2	4	$e_4 e_5$	
P_4	1	6	e_6	

7

Quick Sort : Message Passing Paradigm

Complexité

- Nombre d'étapes

$$\log_2(p)(\dots)$$

Quick Sort : Message Passing Paradigm

Complexité

- Diffusion du pivot

$$\log_2(p)(O(\log_2(p)) + \dots)$$

Quick Sort : Message Passing Paradigm

Complexité

- Partition en locale

$$\log_2(p)(O(\log_2(p)) + O(\frac{n}{p}) + \dots)$$

Quick Sort : Message Passing Paradigm

Complexité

- Réarrangement : calcul des préfixes

$$\log_2(p)(O(\log_2(p)) + O(\frac{n}{p}) + O(\log_2(p)) + \dots)$$

Quick Sort : Message Passing Paradigm

Complexité

- Réarrangement : envoi/réception

$$\log_2(p)(O(\log_2(p)) + O(\frac{n}{p}) + O(\log_2(p)) + ?)$$

Quick Sort : Message Passing Paradigm

Complexité

- Quick sort séquentiel

$$\log_2(p)(O(\log_2(p)) + O(\frac{n}{p}) + O(\log_2(p)) + ?) + O(\frac{n}{p} \log_2(\frac{n}{p}))$$

Quick Sort : Message Passing Paradigm

Complexité

Pire cas :

$$\log_2(p)(O(\log_2(p)) + O(\frac{n}{p}) + O(\log_2(p)) + O(\frac{n}{p})) + O(\frac{n}{p} \log_2(\frac{n}{p}))$$

Quick Sort : Message Passing Paradigm

Complexité

Même ordre de grandeur que pour la mémoire partagée