

INF7235 : Laboratoire #2
Parallélisme itératif à granularité grossière :
Analyse des résultats

30 janvier 2017
PK-4665

1 Les paramètres d'exécution des *benchmarks*

Voici un premier exemple d'exécution, des versions séquentielle et avec parallélisme de boucles, pour un tableau de 64 000 éléments et différents nombre de *threads* — mesures effectuées sur `japet.labunix.uqam.ca`.

```
# NB_WARMPUP = 0
# NB_REPETITIONS = 1
# AVEC_GC = true
# TAILLE = 64000
#          nb.th.          seq    par_boucles
          1          0.064      0.098
          1          0.024      0.050
          1          0.057      0.046

          2          0.023      0.064
          2          0.022      0.051
          2          0.022      0.053

          4          0.021      0.048
          4          0.049      0.045
          4          0.022      0.044

          8          0.022      0.057
          8          0.022      0.049
          8          0.024      0.066

          16         0.022      0.068
          16         0.022      0.068
          16         0.021      0.062
```

Donc : On ne peut pas se fier à une seule exécution, car le temps varie d'une exécution à une autre ⇒ On doit exécuter plusieurs fois puis faire la moyenne des temps obtenus.

Voici un exemple avec cinq (5) répétitions, et différents nombres de *threads*.

```
# NB_WARMUP = 0
# NB_REPETITIONS = 5
# AVEC_GC = true
# TAILLE = 64000
#          nb.th.          seq    par_boucles
          1          0.040          0.056
          2          0.028          0.052
          4          0.028          0.050
          8          0.028          0.048
         16          0.028          0.051
         32          0.027          0.063
         64          0.031          0.076
        128          0.022          0.082
        256          0.026          0.102
```

Bizarre : Le temps pour `seq` semble plus long avec 1 *thread*?

```
# NB_WARMUP = 0
# NB_REPETITIONS = 5
# AVEC_GC = true
# TAILLE = 64000
#      nb.th.      seq  par_boucles
      32      0.040      0.082
       8      0.027      0.044
      256      0.026      0.113
       16      0.028      0.059
       64      0.025      0.066
        2      0.026      0.049
        1      0.026      0.042
      128      0.023      0.086
        4      0.023      0.049
```

NON : Cela ne dépend pas du nombre de *threads* : c'est **toujours** la première exécution qui prend plus de temps !?

```
# NB_WARMPUP = 10
# NB_REPETITIONS = 5
# AVEC_GC = true
# TAILLE = 64000
#          nb.th.          seq    par_boucles
          1          0.023      0.043
          2          0.025      0.052
          4          0.023      0.051
          8          0.013      0.037
         16          0.017      0.031
         32          0.016      0.034
         64          0.013      0.036
        128          0.017      0.058
        256          0.013      0.085
```

```
# NB_WARMPUP = 20
# NB_REPETITIONS = 5
# AVEC_GC = true
# TAILLE = 64000
#          nb.th.          seq    par_boucles
          1          0.019      0.046
          2          0.022      0.058
          4          0.014      0.041
          8          0.012      0.037
         16          0.016      0.034
         32          0.012      0.031
         64          0.012      0.038
        128          0.012      0.064
        256          0.012      0.084
```

```
# NB_WARMPUP = 60
# NB_REPETITIONS = 5
# AVEC_GC = true
# TAILLE = 64000
```

#	nb.th.	seq	par_boucles
	1	0.012	0.015
	2	0.015	0.041
	4	0.012	0.042
	8	0.015	0.044
	16	0.013	0.037
	32	0.016	0.031
	64	0.013	0.042
	128	0.016	0.066
	256	0.013	0.091

Un autre facteur...

```
# NB_WARMPUP = 10
# NB_REPETITIONS = 5
# AVEC_GC = true
# TAILLE = 64000
#      nb.th.      seq  par_boucles
      1      0.023    0.043
      2      0.025    0.052
      4      0.023    0.051
      8      0.013    0.037
     16      0.017    0.031
     32      0.016    0.034
     64      0.013    0.036
    128      0.017    0.058
    256      0.013    0.085
```

```
# NB_WARMPUP = 5
# NB_REPETITIONS = 5
# AVEC_GC = true
# TAILLE = 64000
#      nb.th.      seq  par_boucles
      1      0.031    0.054
      2      0.029    0.055
      4      0.029    0.054
      8      0.029    0.051
     16      0.031    0.057
     32      0.029    0.063
     64      0.025    0.070
    128      0.024    0.084
    256      0.024    0.104
```

Quelle est la différence?

Dans le 2^e cas — où les temps d'exécution sont plus longs (exécutions plus lentes) — la compilation JIT a été désactivée :

```
$ jruby -J-Djruby.compile.mode=OFF nb_inversions_bm.rb
```

Par contre, NB_WARMUP est encore trop faible...

Encore un autre facteur...

```
# NB_WARMPUP = 10
# NB_REPETITIONS = 5
# AVEC_GC = true
# TAILLE = 64000
#      nb.th.      seq  par_boucles
      1      0.025    0.049
      2      0.025    0.056
      4      0.026    0.057
      8      0.026    0.051
     16      0.030    0.058
     32      0.030    0.067
     64      0.027    0.070
    128      0.027    0.081
    256      0.026    0.101
```

```
# NB_WARMPUP = 10
# NB_REPETITIONS = 5
# AVEC_GC = false
# TAILLE = 64000
#      nb.th.      seq  par_boucles
      1      0.023    0.054
      2      0.022    0.054
      4      0.022    0.053
      8      0.021    0.051
     16      0.025    0.066
     32      0.026    0.071
     64      0.026    0.080
    128      0.025    0.092
    256      0.023    0.111
```

Quelle est la différence?

Collecte des résidus!?

2 Extraits du programme de *benchmark* : nb_inversions_bm.rb

```
require 'benchmark'

[1, 2, 4, 8, 16, 32, 64, 128, 256].each do |nb_threads|
  ENV['NB_THREADS'] = nb_threads.to_s

  # On execute la version sequentielle.
  ENV['METHODE'] = "seq"
  NB_WARMUP.times { a.nb_inversions }
  GC.start
  temps_seq = temps_moyen(NB_REPETITIONS) { a.nb_inversions }
  print "%#{largeur}.3f" % temps_seq

  # On execute les versions paralleles.
  METHODES.each do |methode|
    ENV['METHODE'] = methode
    NB_WARMUP.times { a.nb_inversions }
    GC.start
    temps_par = temps_moyen(NB_REPETITIONS) { a.nb_inversions }
    print "%#{largeur}.3f" % temps_par
  end
end

def temps_moyen( nb_fois, &block )
  return 0.0 if nb_fois == 0

  tot = 0
  nb_fois.times do
    GC.start
    GC.disable unless AVEC_GC
    tot += (Benchmark.measure &block).real
    GC.enable
  end
  tot / nb_fois
end
```

3 Les temps d'exécution effectifs pour **différentes tailles** de tableaux et **différents nombres de *threads***

Note : Le JIT n'a pas été désactivé!

```
# NB_WARMUP = 10
# NB_REPETITIONS = 5
# AVEC_GC = false
# TAILLE = 6400
#          nb.th.          seq    par_forkjoin    par_boucles
          1          0.003      0.005      0.007
          2          0.003      0.003      0.007
          4          0.003      0.003      0.006
          8          0.003      0.007      0.010
         16          0.003      0.004      0.008
         32          0.003      0.014      0.016
         64          0.002      0.020      0.022
        128          0.003      0.043      0.036
        256          0.003      0.062      0.052
```

```

# NB_WARMPUP = 10
# NB_REPETITIONS = 5
# AVEC_GC = false
# TAILLE = 64000
#           nb.th.           seq    par_forkjoin    par_boucles
           1             0.023      0.028          0.050
           2             0.023      0.021          0.051
           4             0.026      0.013          0.050
           8             0.027      0.007          0.049
          16             0.024      0.011          0.062
          32             0.024      0.012          0.066
          64             0.025      0.020          0.075
         128             0.031      0.040          0.088
         256             0.030      0.059          0.114

```

```

# NB_WARMPUP = 10
# NB_REPETITIONS = 5
# AVEC_GC = false
# TAILLE = 640000
#           nb.th.           seq    par_forkjoin    par_boucles
           1             0.259      0.262          0.455
           2             0.258      0.167          0.575
           4             0.266      0.164          0.437
           8             0.247      0.114          0.451
          16             0.252      0.085          0.647
          32             0.252      0.074          0.666
          64             0.246      0.068          0.696
         128             0.263      0.077          0.709
         256             0.257      0.102          0.755

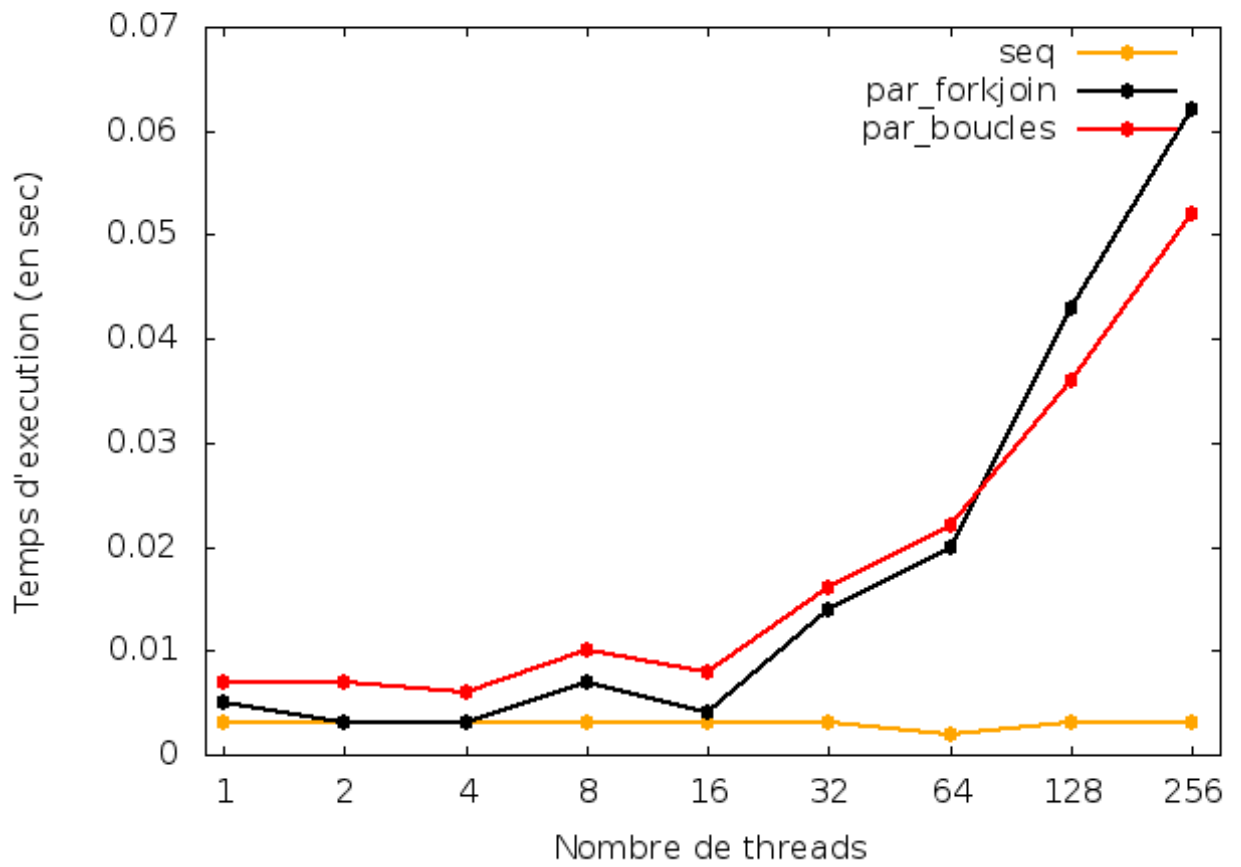
```

Que constate-t-on quant au temps d'exécution séquentiel?

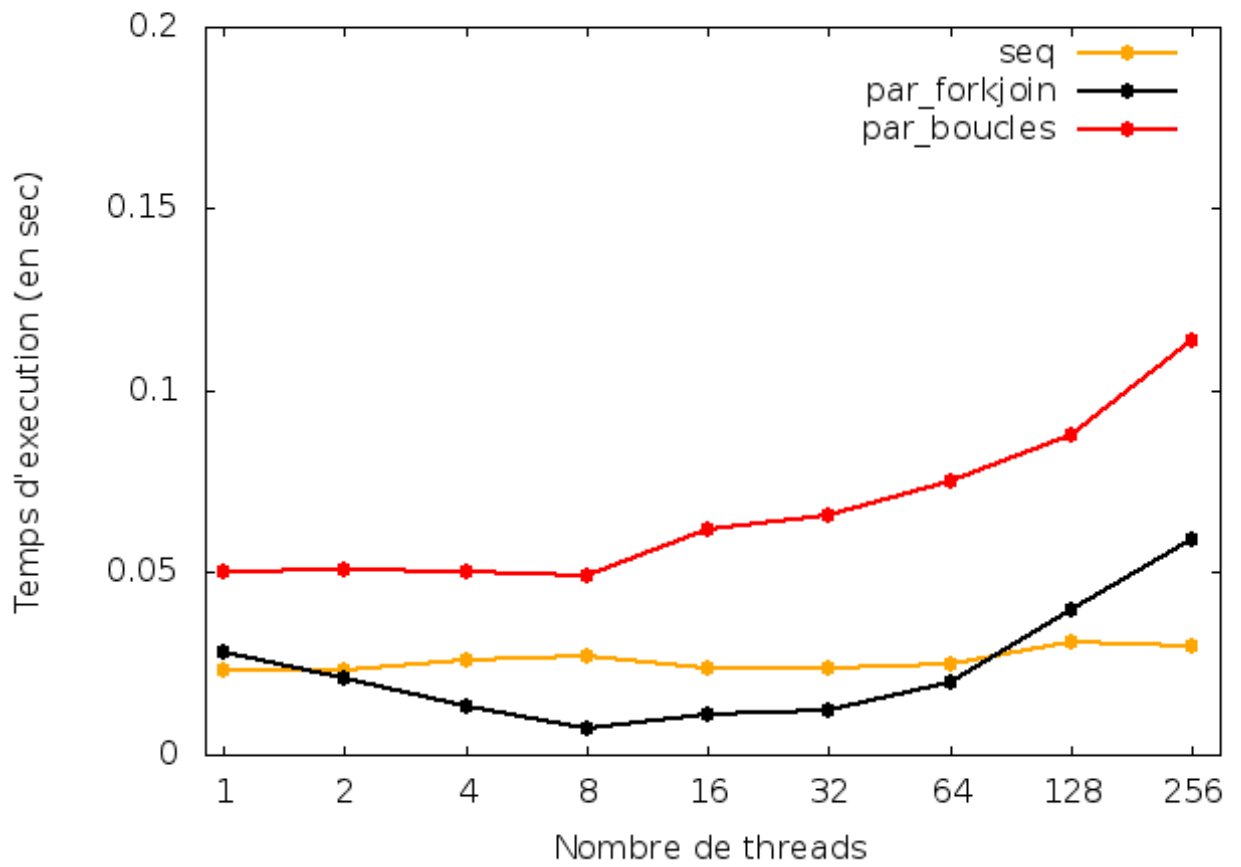
Constat : Le temps d'exécution séquentiel augmente de façon (quasi) **linéaire**!

- 4 Les graphes de temps d'exécution pour les trois versions parallèles, différents nombres de *threads* et tailles de tableaux

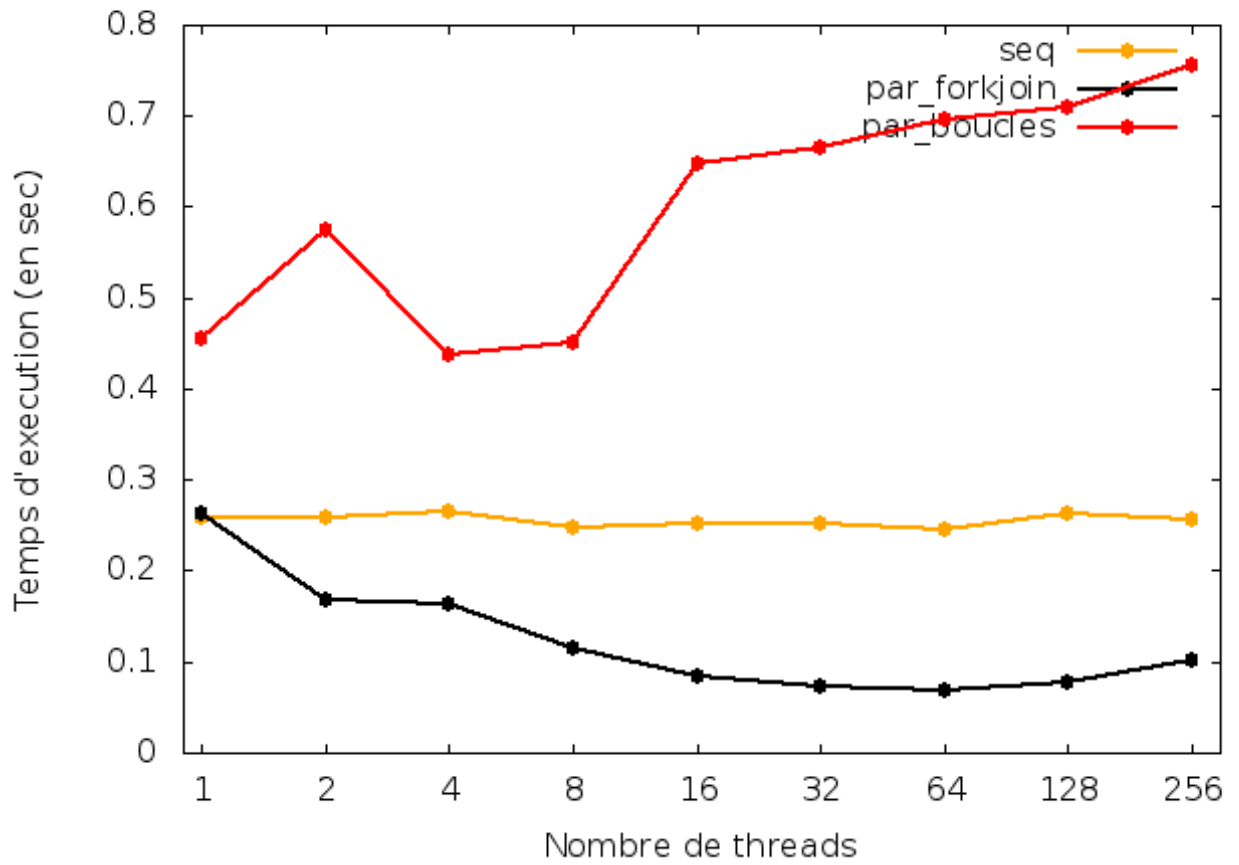
Temps d'execution en fonction du nombre de threads pour n = 6400



Temps d'execution en fonction du nombre de threads pour n = 64000

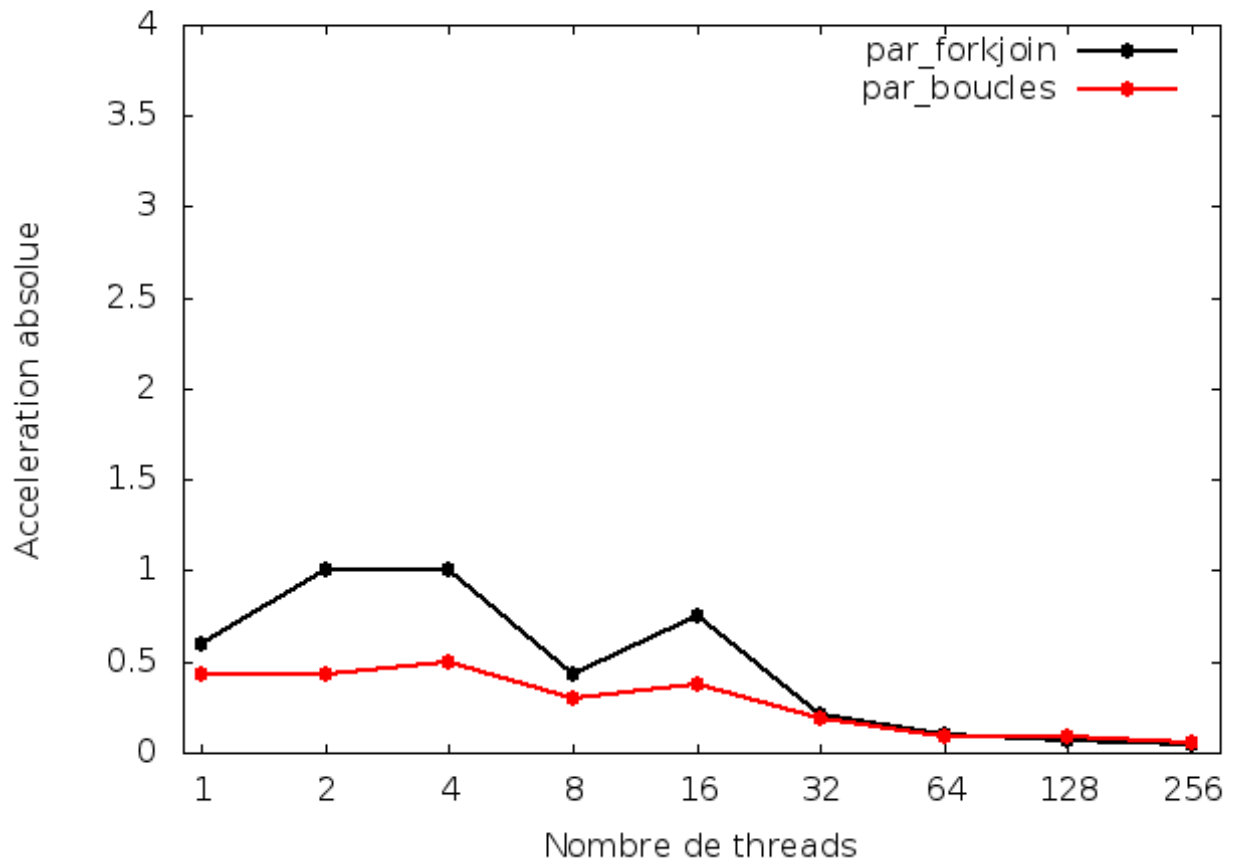


Temps d'execution en fonction du nombre de threads pour n = 640000

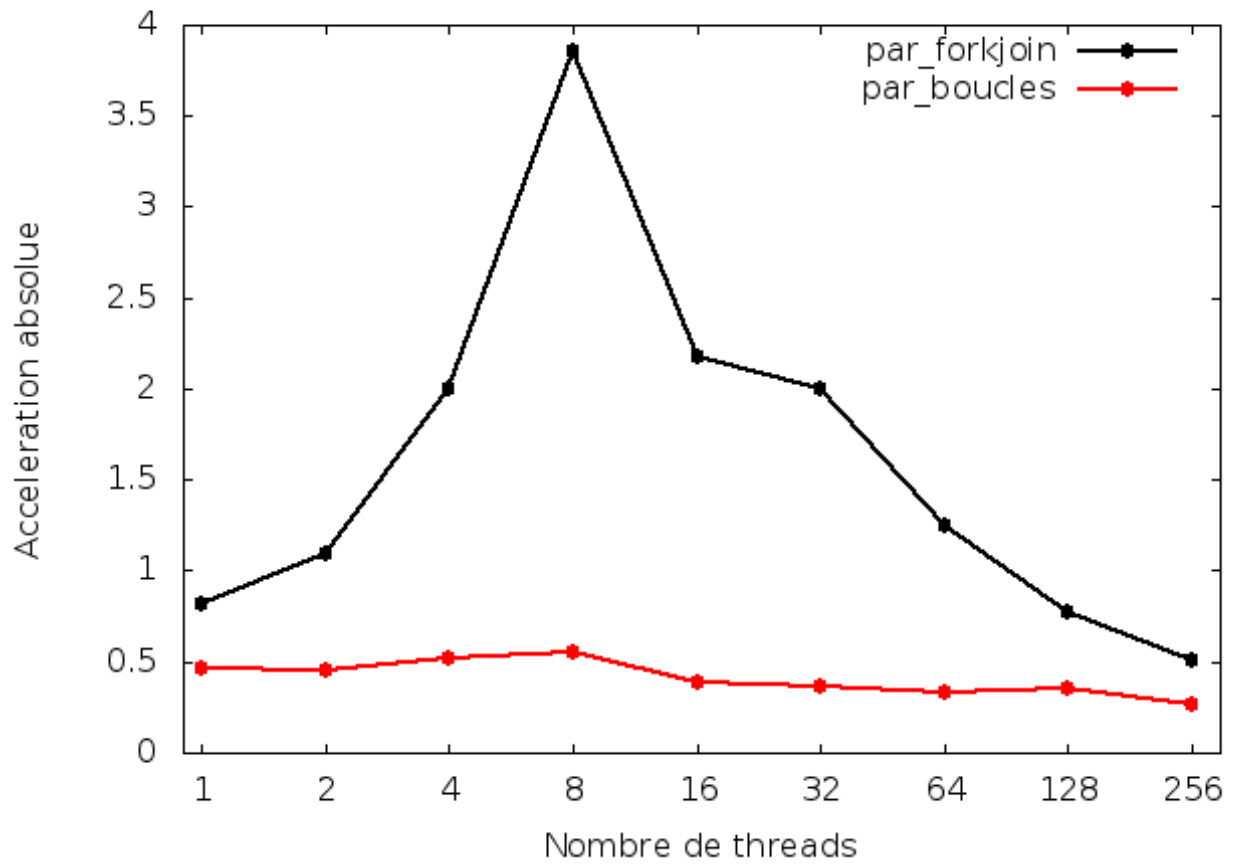


5 Les graphes d'accélération pour les trois versions parallèles, différents nombres de *threads* et tailles de tableaux

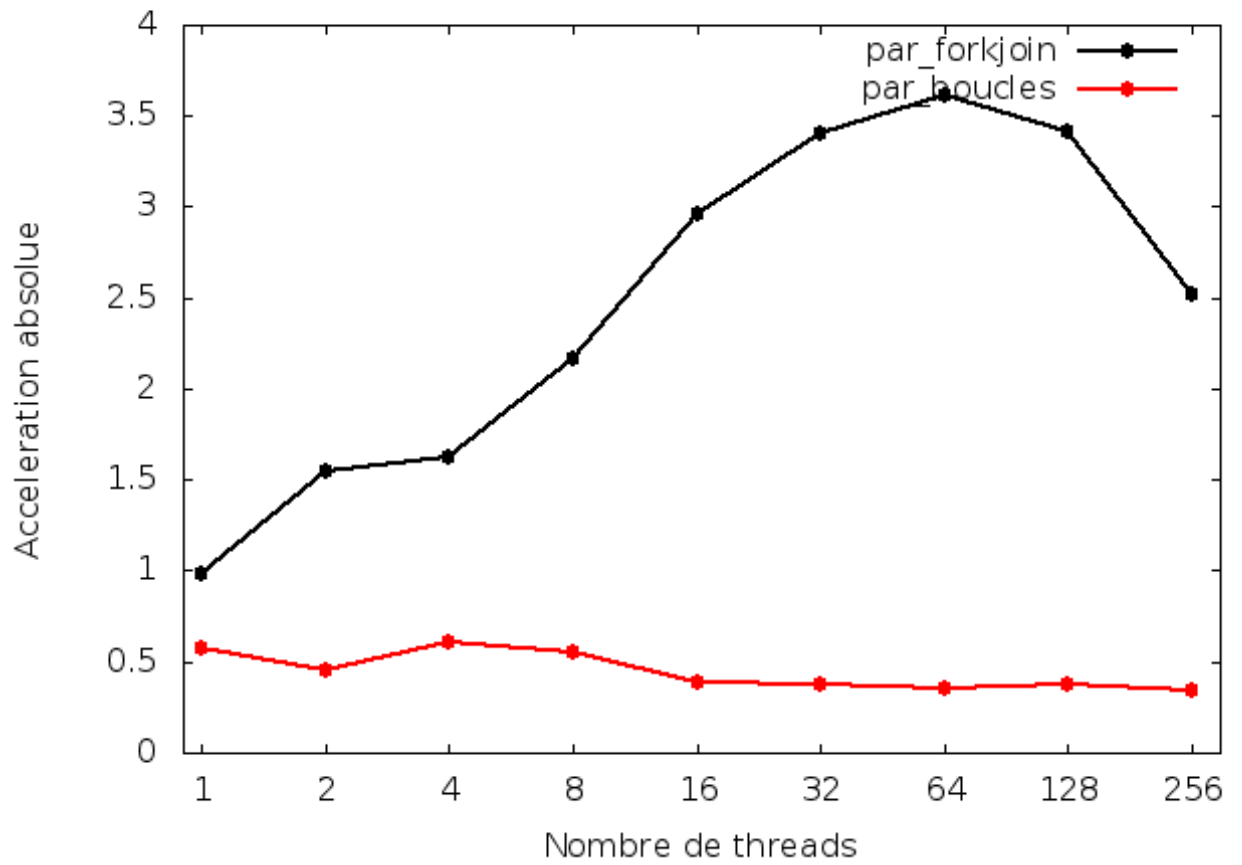
Accelération absolue en fonction du nombre de threads pour $n = 6400$



Accelération absolue en fonction du nombre de threads pour $n = 64000$



Accelération absolue en fonction du nombre de threads pour $n = 640000$



À noter : Courbes en U inversés.