

PaP

Projet : rapport2

4TIN804U

BERASATEGUY Tanguy, GOEDEFROIT Charles

Table des matières

1 ILP optimization (4.1)

On as fait les modifications :

Pour `ssandPile_do_tile_opt()` on a retirer les appelles à `table(out, i, j)` pour passer par une variable intermedier `result`. Cette modification petmer au compilateur de vectoriser car il peut maintenet facilement voir que les différente lige peuve être calculer en parallele.

Nouas avons modifier c'est lignes :

```
1  int ssandPile_do_tile_opt(int x, int y, int width, int height)
2  {
3      int diff = 0;
4
5      for (int i = y; i < y + height; i++)
6          for (int j = x; j < x + width; j++)
7              {
8                  -   table(out, i, j) = table(in, i, j) % 4;
9                  +   int result = table(in, i, j) % 4;
10                 -   table(out, i, j) += table(in, i + 1, j) / 4;
11                 +   result += table(in, i + 1, j) / 4;
12                 -   table(out, i, j) += table(in, i - 1, j) / 4;
13                 +   result += table(in, i - 1, j) / 4;
14                 -   table(out, i, j) += table(in, i, j + 1) / 4;
15                 +   result += table(in, i, j + 1) / 4;
16                 -   table(out, i, j) += table(in, i, j - 1) / 4;
17                 +   result += table(in, i, j - 1) / 4;
18                 +   table(out, i, j) = result;
19                 -   if (table(out, i, j) >= 4)
20                 -       diff = 1;
21                 +   diff |= result >= 4;
22             }
23
24     return diff;
25 }
```

Le code de la fonction final :

```
27 int ssandPile_do_tile_opt(int x, int y, int width, int height)
28 {
29     int diff = 0;
30
31     for (int i = y; i < y + height; i++)
32         for (int j = x; j < x + width; j++)
33             {
34                 int result = table(in, i, j) % 4;
35                 result += table(in, i + 1, j) / 4;
36                 result += table(in, i - 1, j) / 4;
37                 result += table(in, i, j + 1) / 4;
38                 result += table(in, i, j - 1) / 4;
```

```

39     table(out, i, j) = result;
40     diff /= result >= 4;
41 }
42
43 return diff;
44 }

```

Nous avons verifïer et on obtient le même résultats et le même nombre d'iterations (69190) avec la version par défaut et la version opt.



(a) avec do_tile_default

(b) avec do_tile_opt

FIGURE 1 – Verification du résultats pour ssandPile

Le gain de performance est de 2.37 car $\frac{178812}{75408}$

Pour asandPile_do_tile_opt() on a retirer...
 Nous avons modifier c'est lignes :

```

1  int asandPile_do_tile_default(int x, int y, int width, int height)
2  {
3      int change = 0;
4
5      for (int i = y; i < y + height; i++)
6          for (int j = x; j < x + width; j++)
7              - if (atable(i, j) >= 4)
8                  + int result = atable(i, j);
9                  + if (result >= 4)
10                 {
11                     + result/=4;
12                     - atable(i, j - 1) += atable(i, j) / 4;
13                     + atable(i, j - 1) += result;
14                     - atable(i, j + 1) += atable(i, j) / 4;
15                     + atable(i, j + 1) += result;
16                     - atable(i - 1, j) += atable(i, j) / 4;
17                     + atable(i - 1, j) += result;

```

```

18 -     atable(i + 1, j) += atable(i, j) / 4;
19 +     atable(i + 1, j) += result;
20     atable(i, j) %= 4;
21     change = 1;
22 }
23 return change;
24 }

```

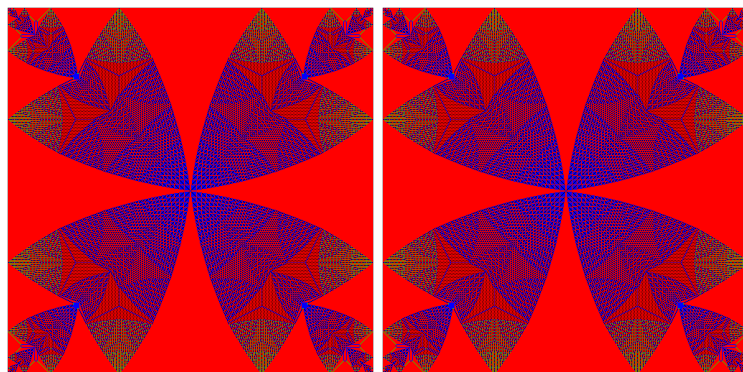
Le code de la fonction final :

```

26 int asandPile_do_tile_opt(int x, int y, int width, int height)
27 {
28     int change = 0;
29
30     for (int i = y; i < y + height; i++)
31         for (int j = x; j < x + width; j++)
32         {
33             int result = atable(i, j);
34             if (result >= 4)
35             {
36                 result/=4;
37                 atable(i, j - 1) += result;
38                 atable(i, j + 1) += result;
39                 atable(i - 1, j) += result;
40                 atable(i + 1, j) += result;
41                 atable(i, j) %= 4;
42                 change = 1;
43             }
44         }
45     return change;
46 }

```

Nous avons verifie et on obtient le même résultats et le même nombre d'iterations (34938) avec la version par défaut et la version opt.



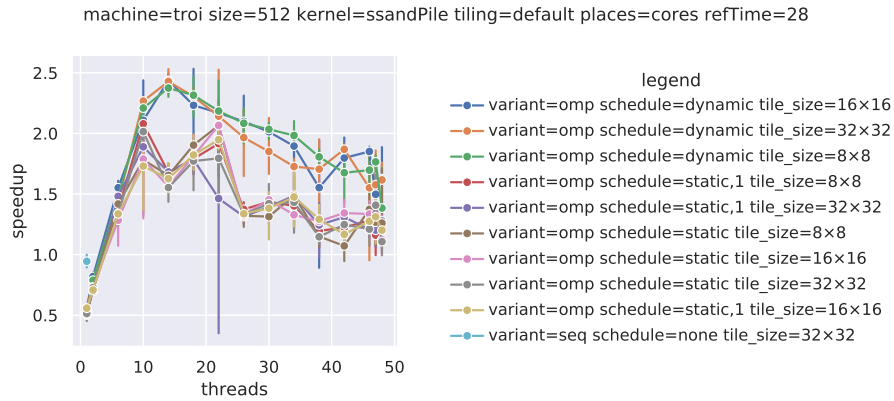
(a) avec do_tile_default

(b) avec do_tile_opt

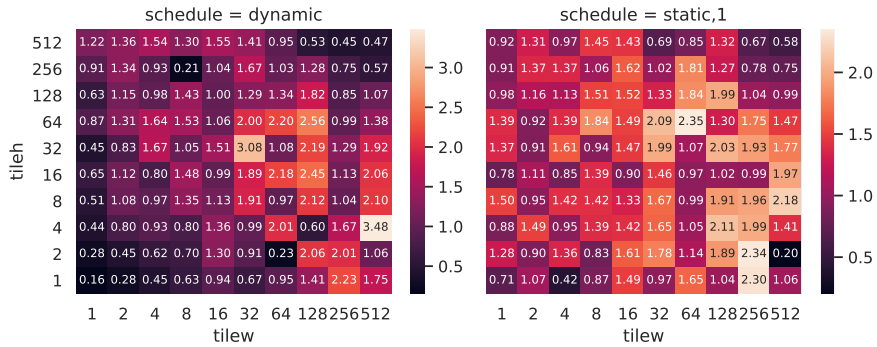
FIGURE 2 – Verification du résultats pour asandPile

Le gain de performance est de 1.2 car $\frac{37990}{31405}$

2 OpenMP implementation of the synchronous version (4.2)



machine=data size=512 threads=24 kernel=ssandPile variant=omp_tiled tiling=default places=cores refTime=28



3 OpenMP implementation of the asynchronous version (4.3)

4 Lazy OpenMP implementations (4.4)