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 varaible intermedier result. Cette modification petmer au compilateur de vectoriser car il peut maintenet facilement voir que les différante lige peuve être calculer en paralelle.*

Nouas avons modifier c'est lignes :

```
int ssandPile_do_tile_opt(int x, int y, int width, int height)
{
  int diff = 0;

  for (int i = y; i < y + height; i++)
    for (int j = x; j < x + width; j++)
    {
-       table(out, i, j) = table(in, i, j) % 4;
+       int result = table(in, i, j) % 4;
-       table(out, i, j) += table(in, i + 1, j) / 4;
+       result += table(in, i + 1, j) / 4;
-       table(out, i, j) += table(in, i - 1, j) / 4;
+       result += table(in, i - 1, j) / 4;
-       table(out, i, j) += table(in, i, j + 1) / 4;
+       result += table(in, i, j + 1) / 4;
-       table(out, i, j) += table(in, i, j - 1) / 4;
+       result += table(in, i, j - 1) / 4;
+       table(out, i, j) = result;
-       if (table(out, i, j) >= 4)
-         diff = 1;
+       diff |= result >= 4;
    }

  return diff;
}
```

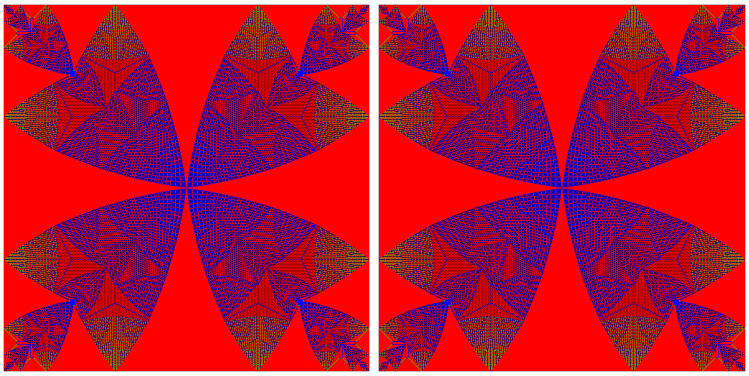
Le code de la fonction final :

```
int ssandPile_do_tile_opt(int x, int y, int width, int height)
{
  int diff = 0;

  for (int i = y; i < y + height; i++)
    for (int j = x; j < x + width; j++)
    {
      int result = table(in, i, j) % 4;
      result += table(in, i + 1, j) / 4;
      result += table(in, i - 1, j) / 4;
      result += table(in, i, j + 1) / 4;
      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 verifier 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...
Nouas 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;
```

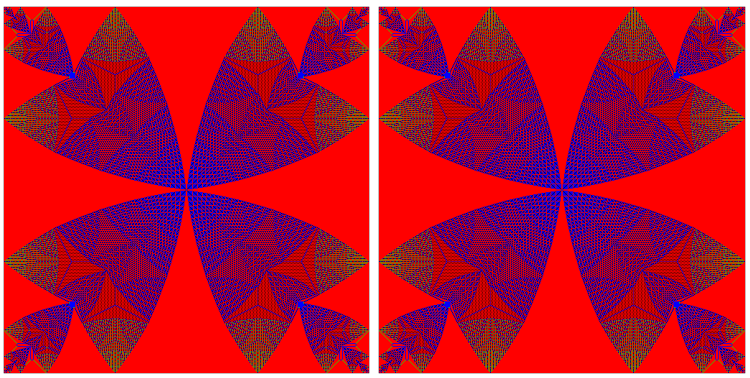
```
-         atable(i + 1, j) += atable(i, j) / 4;
+         atable(i + 1, j) += result;
          atable(i, j) %= 4;
          change = 1;
        }
  return change;
}
```

Le code de la fonction final :

```
int asandPile_do_tile_opt(int x, int y, int width, int height)
{
  int change = 0;

  for (int i = y; i < y + height; i++)
    for (int j = x; j < x + width; j++)
    {
      int result = atable(i, j);
      if (result >= 4)
      {
        result/=4;
        atable(i, j - 1) += result;
        atable(i, j + 1) += result;
        atable(i - 1, j) += result;
        atable(i + 1, j) += result;
        atable(i, j) %= 4;
        change = 1;
      }
    }
  return change;
}
```

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



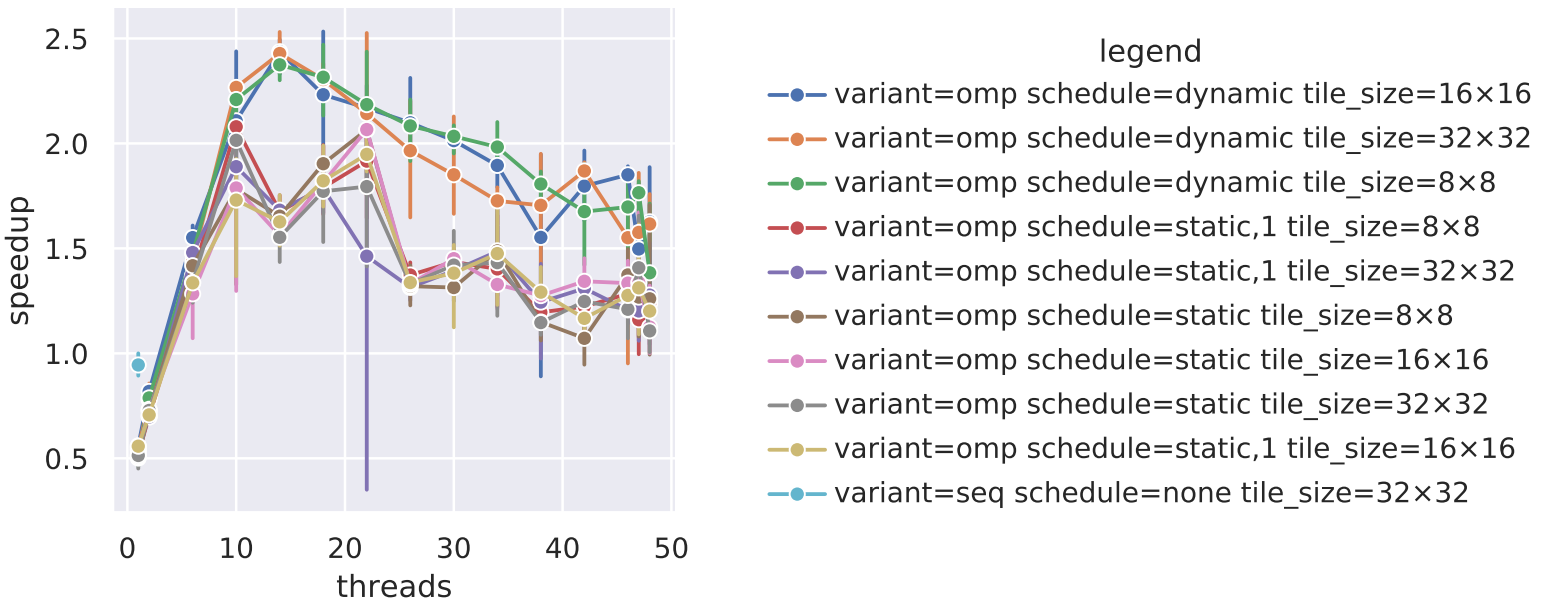
(a) avec do_tile_default (b) avec do_tile_opt

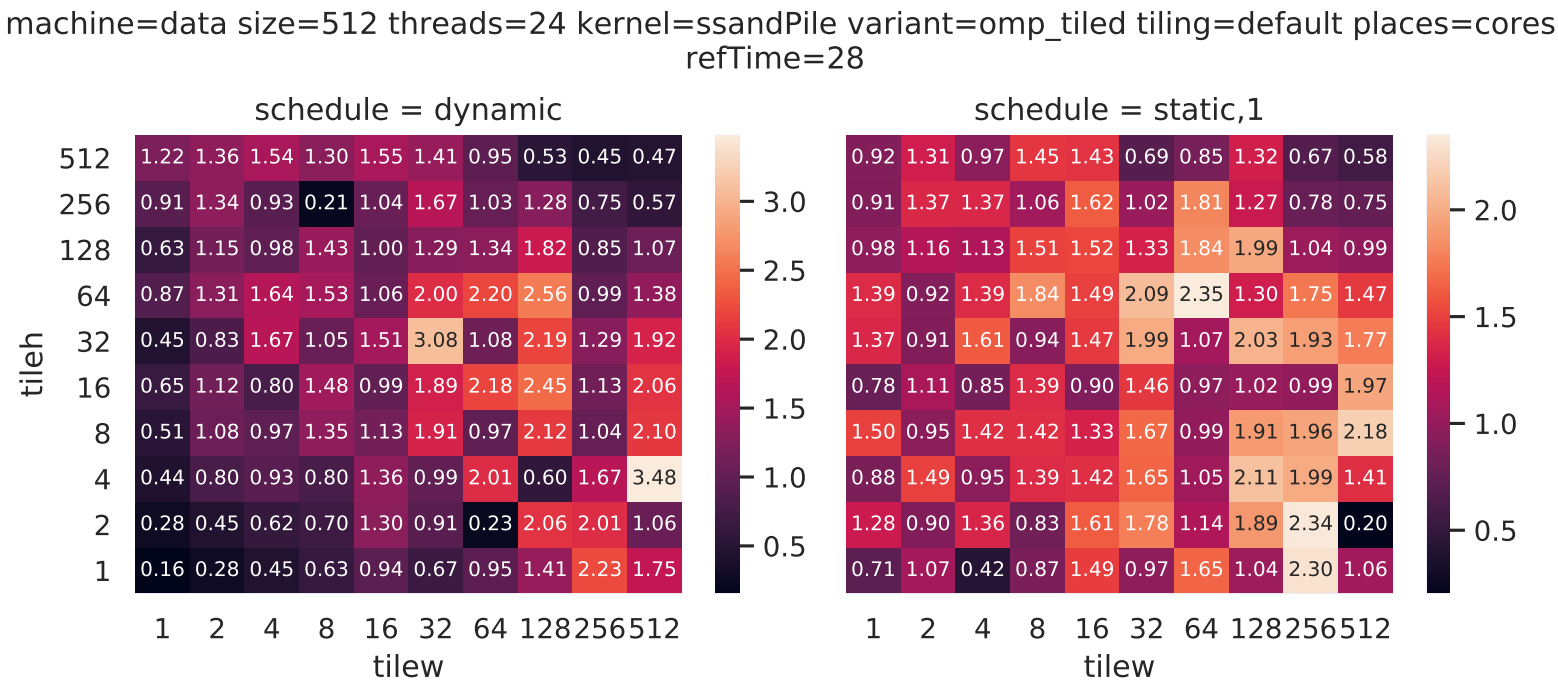
FIGURE 2 – Verification du résultats pour asandPile

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

2 OpenMP implementation of the synchronous version (4.2)

machine=troi size=512 kernel=ssandPile tiling=default places=cores refTime=28




machine=data size=512 threads=24 kernel=ssandPile variant=omp_tiled tiling=default places=cores
refTime=28
schedule = dynamic
schedule = static,1
tileh
tilew
512 256 128 64 32 16 8 4 2 1
1 2 4 8 16 32 64 128 256 512
1.22 1.36 1.54 1.30 1.55 1.41 0.95 0.53 0.45 0.47
0.91 1.34 0.93 0.21 1.04 1.67 1.03 1.28 0.75 0.57
0.63 1.15 0.98 1.43 1.00 1.29 1.34 1.82 0.85 1.07
0.87 1.31 1.64 1.53 1.06 2.00 2.20 2.56 0.99 1.38
0.45 0.83 1.67 1.05 1.51 3.08 1.08 2.19 1.29 1.92
0.65 1.12 0.80 1.48 0.99 1.89 2.18 2.45 1.13 2.06
0.51 1.08 0.97 1.35 1.13 1.91 0.97 2.12 1.04 2.10
0.44 0.80 0.93 0.80 1.36 0.99 2.01 0.60 1.67 3.48
0.28 0.45 0.62 0.70 1.30 0.91 0.23 2.06 2.01 1.06
0.16 0.28 0.45 0.63 0.94 0.67 0.95 1.41 2.23 1.75
3.0 2.5 2.0 1.5 1.0 0.5
0.92 1.31 0.97 1.45 1.43 0.69 0.85 1.32 0.67 0.58
0.91 1.37 1.37 1.06 1.62 1.02 1.81 1.27 0.78 0.75
0.98 1.16 1.13 1.51 1.52 1.33 1.84 1.99 1.04 0.99
1.39 0.92 1.39 1.84 1.49 2.09 2.35 1.30 1.75 1.47
1.37 0.91 1.61 0.94 1.47 1.99 1.07 2.03 1.93 1.77
0.78 1.11 0.85 1.39 0.90 1.46 0.97 1.02 0.99 1.97
1.50 0.95 1.42 1.42 1.33 1.67 0.99 1.91 1.96 2.18
0.88 1.49 0.95 1.39 1.42 1.65 1.05 2.11 1.99 1.41
1.28 0.90 1.36 0.83 1.61 1.78 1.14 1.89 2.34 0.20
0.71 1.07 0.42 0.87 1.49 0.97 1.65 1.04 2.30 1.06
2.0 1.5 1.0 0.5

3 OpenMP implementation of the asynchronous version (4.3)

4 Lazy OpenMP implementations (4.4)