# Bee: Buffer Sizing and Allocation under Scheduling Constraints

Tool Demo

Christophe Alias

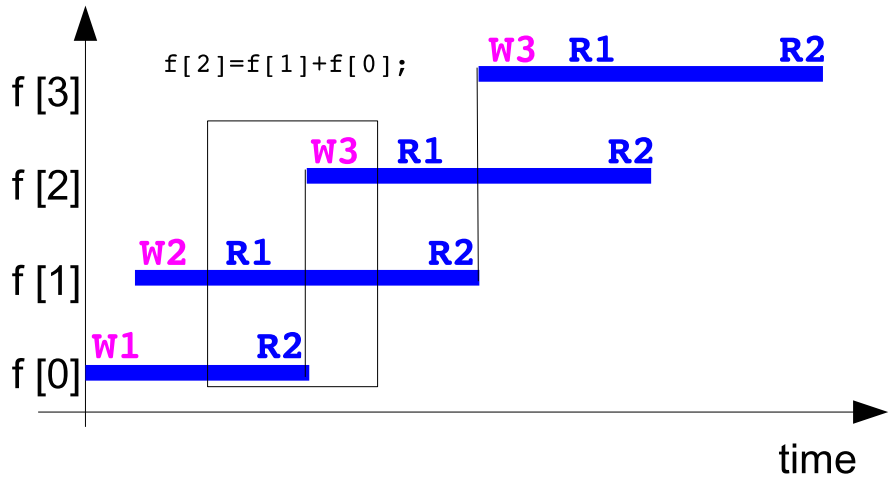
http://perso.ens-lyon.fr/christophe.alias

INRIA, ENS Lyon
Laboratoire de l'Informatique du Parallélisme

First Spring School on Polyhedral Code Analysis and Optimizations, May 2013

# Array Contraction at a Glance

```
int fib(int N)
{
  int* f = malloc(N*sizeof(int));
  f[0] = 0; //W1
  f[1] = 1; //W2
  for(i=2; i<N; i++)
    f[i] = f[i-1] + f[i-2];
  // W3       R1        R2
  return f[N-1];
}
```




- Collapse array cells that do not conflict.
- Linear allocation functions :

$$\sigma(\vec{i}) = A\vec{i} \mod \vec{b}$$

- Here : $f[i] \rightarrow f_{compacted}[i\%2]$.

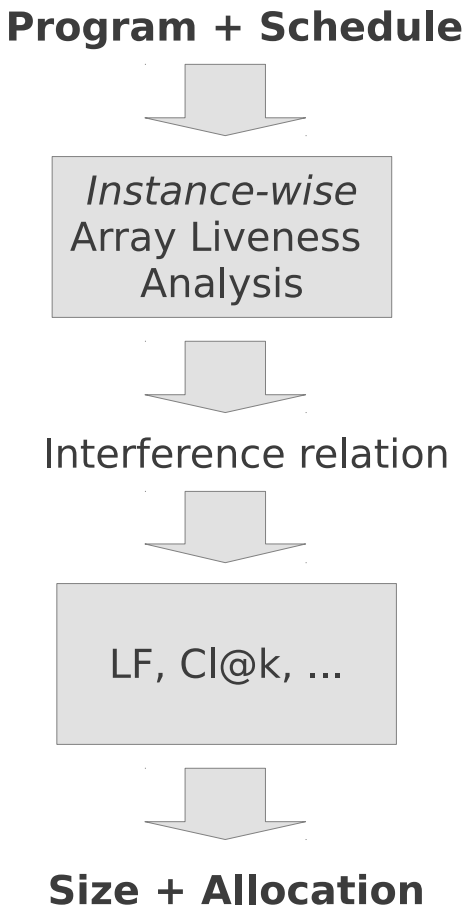
# Bee

## Step 1. Interference Analysis

- Array liveness analysis
- Interference relation $\bowtie_a$ :
  $DS = \{\vec{i} - \vec{j},\ a[\vec{i}] \bowtie_a a[\vec{j}]\}$

## Step 2. Mapping Derivation

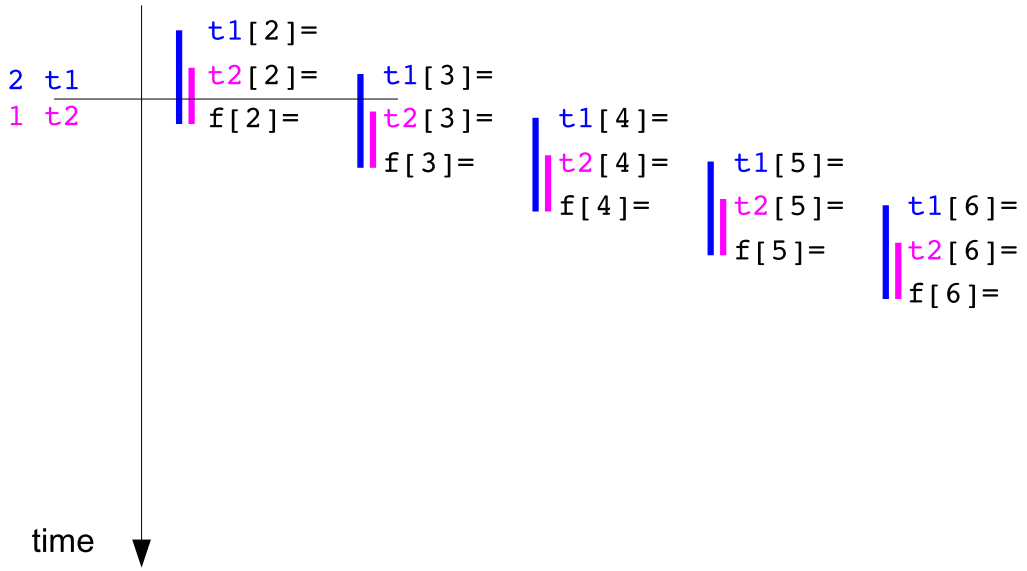
- Critical lattice method (`Cl@k`).
- LF method




`http://compsys-tools.ens-lyon.fr/bee`

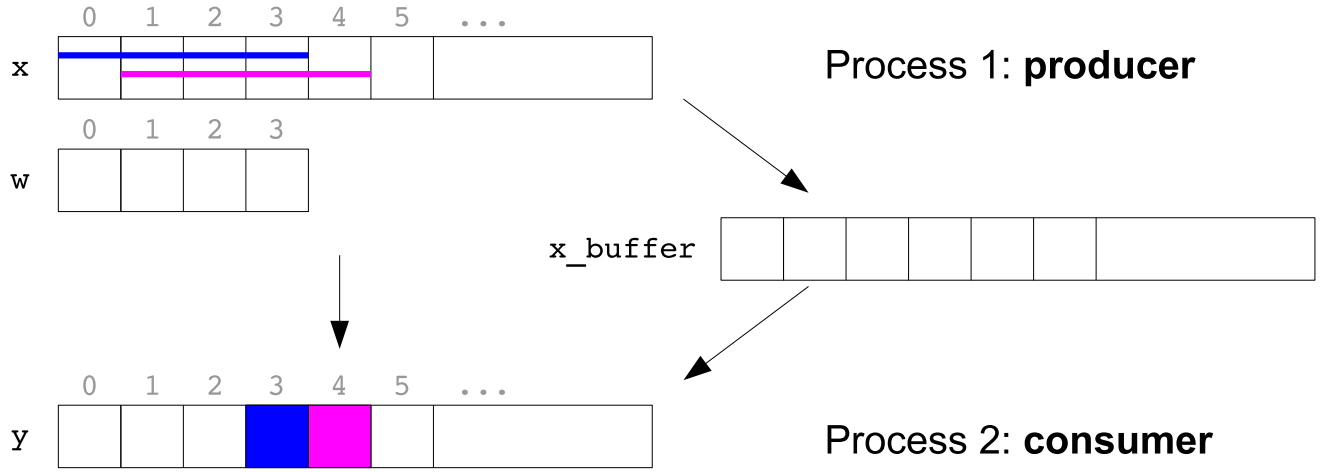
# Interplay with Scheduling, I

```
int fib(int N)
{
#pragma schedule[0]
  f[0] = 0;
#pragma schedule[1]
  f[1] = 1;
  for(i=0; i<N; i++)
    {
#pragma schedule[2][i]
      t1[i] = f[i-1];
#pragma schedule[2][i+1]
      t2[i] = f[i-2];
#pragma schedule[2][i+2]
      f[i] = t1[i] + t2[i];
    }
#pragma schedule[3]
  return f[N-1];
}
```




- Schedule specified with #pragmas.
- Here, software pipelining with $II = 1$.
  - t1[] requires 2 cells
  - t2[] requires 1 cell

# Interplay with Scheduling, II




- Process 1 sends x[] to process 2.
- Process 2 convolves x[] *on-the-fly* :

$$y[i] = \sum_{k=0}^{3} x[i-k] * w[k]$$

- Minimal size for `x_buffer` ? Allocation ?