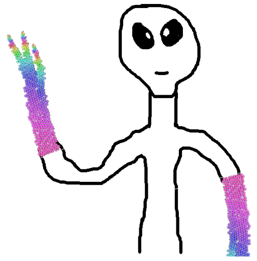
# Gestion efficace des ressources mémoire et de calcul pour l'exécution de système multi-agents sur architectures parallèles avec OpenCL



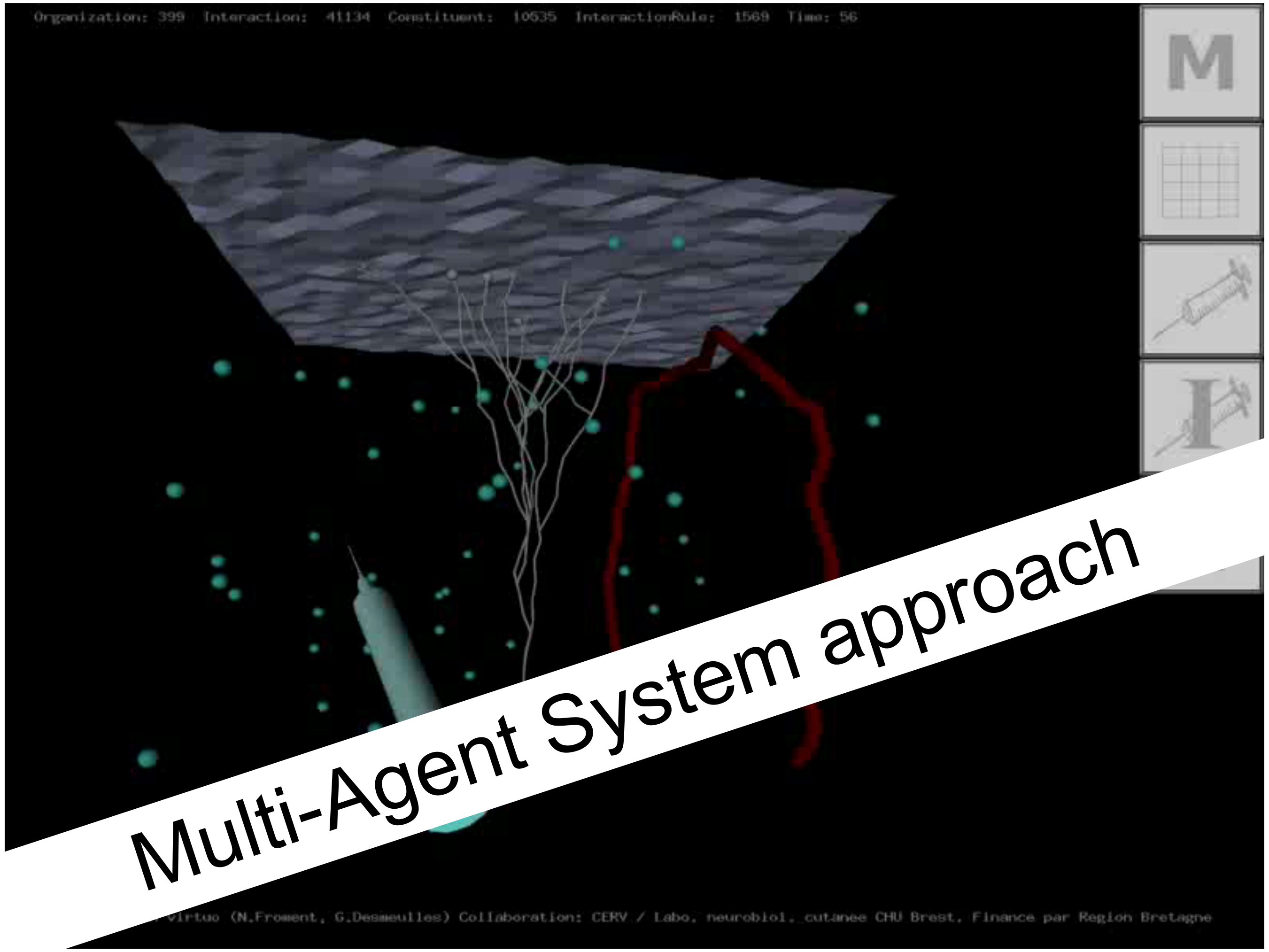
Anne Jeannin-Girardon, Vincent Rodin

Lab-STICC, UMR 6285, CNRS,

Dépt Informatique, Université de Brest

## Lab-STICC, CID, IHSEV

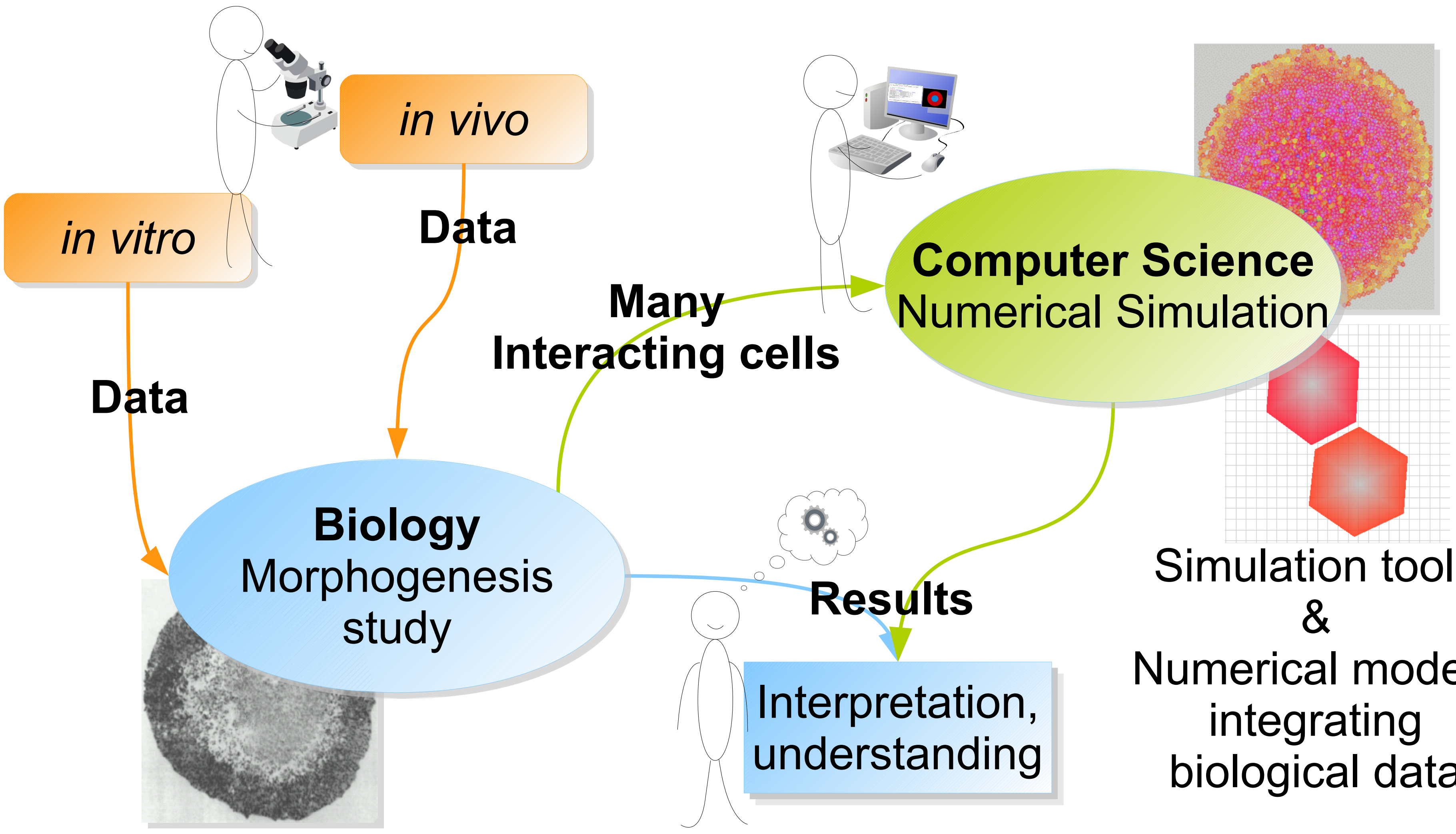
### Virtual Reality ➔ Virtual Biology


Organization: 399 Interaction: 41134 Constituent: 10535 InteractionRule: 1569 Time: 56
M
Virtuo (N.Froment, G.Desmeulles) Collaboration: CERV / Labo. neurobiol. cutanee CHU Brest. Finance par Region Bretagne

Multi-Agent System approach

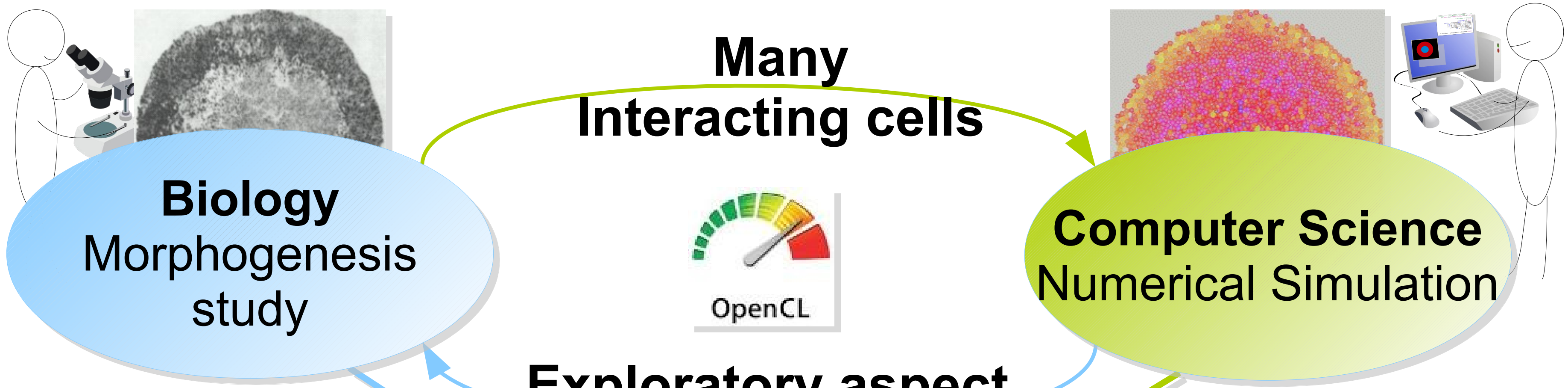
Interaction between virtual cells and/or molecules

# Context (1/3)



Simulation tools
&
Numerical models
integrating
biological data

# Context (2/3)

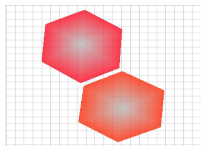


Continuous approach

$$\frac{\delta\mu_i}{\delta\tau} = \nabla^2 \mu_i + F_i(\mu)$$

Mechanisms involved in Morphogenesis process

Discrete approach

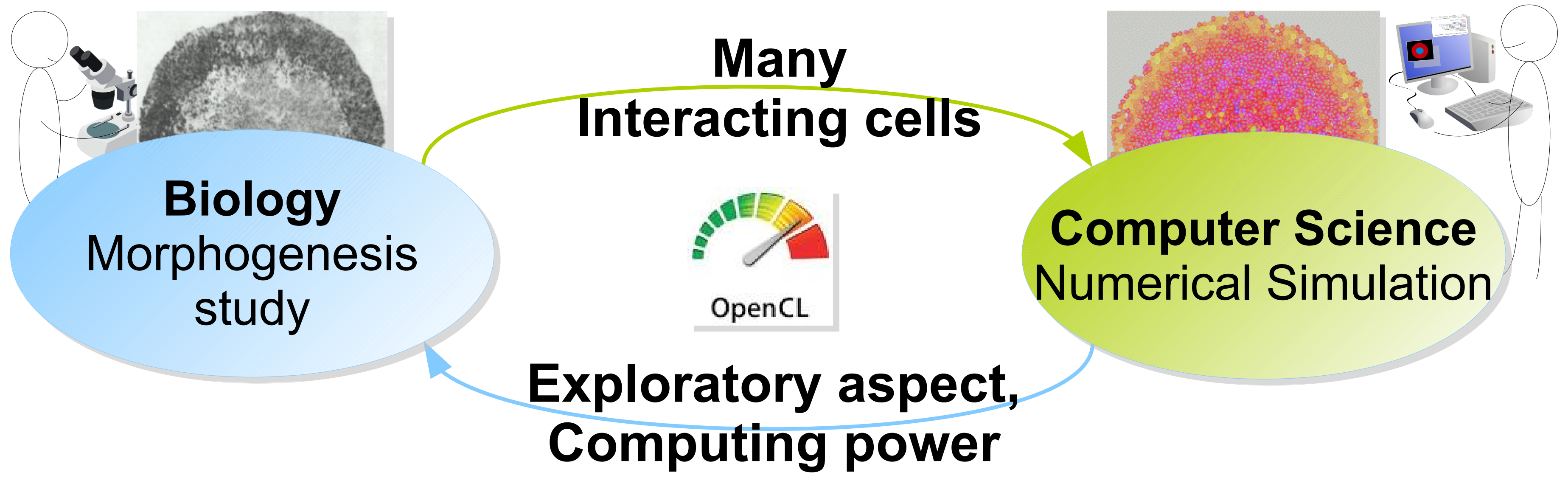


**Our approach is hybrid**

➔population level (Molecular Virtual Chemistry)

➔individual level (Virtual Cells)

# Context (3/3)



From a computational point of view, the work presented today is an improvement of

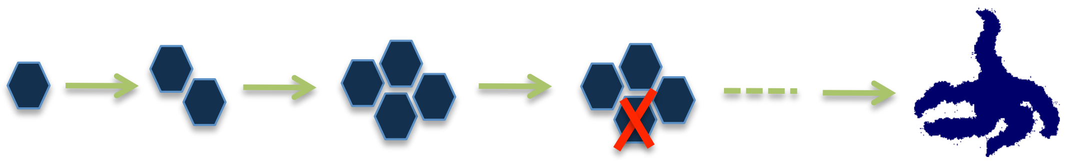
- ➔ Anne Jeannin-Girardon Ph.D thesis, 2014
- ➔ Anne Jeannin-Girardon et al, IEEE/ACM Transaction on Computational Biology and Bioinformatics, 2015

# Outline

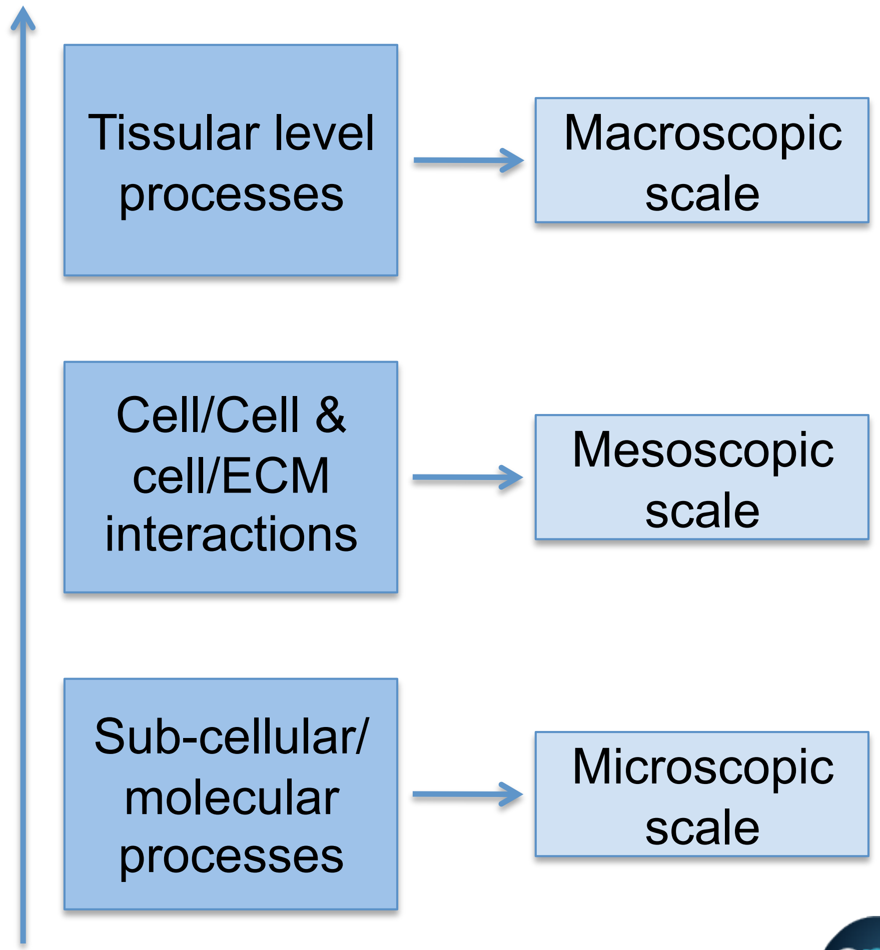
- Introduction
  - ➔ Morphogenesis & Dynamicity
- Virtual Biological Model
  - ➔ Virtual Cell, Molecular Virtual Chemistry, Virtual Growth
- Parallel implementation
  - ➔ OpenCL, model coupled with a MAS
- How to get a new Id for a new Virtual Cell?
- Results

# Introduction (1/1)

**Tissue morphogenesis:**

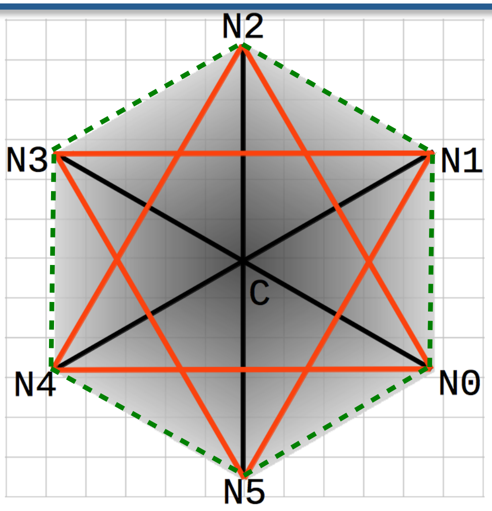


- Is a multi-scale phenomenon
- Can be addressed through continuous & discrete models
- Involves many of interacting entities (cells, molecules, etc.)
- Implies birth and death of cells ➔ dynamicity

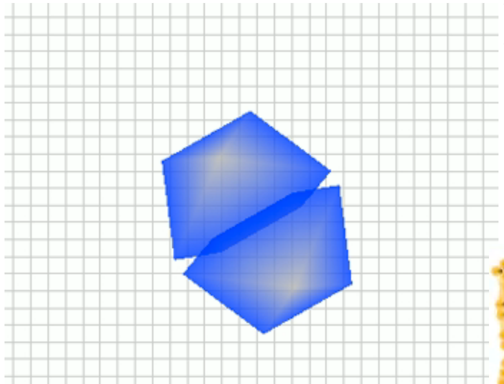


# Virtual Biological Model (1/3)

## Virtual Cell

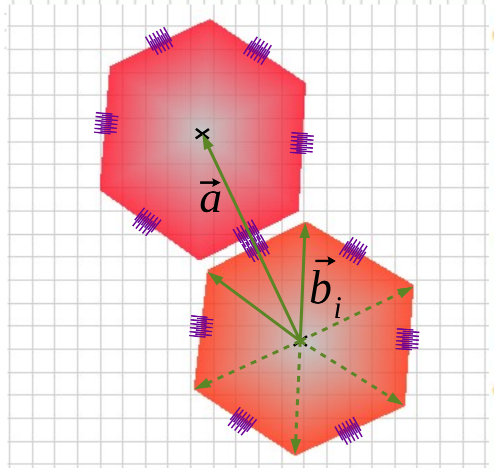


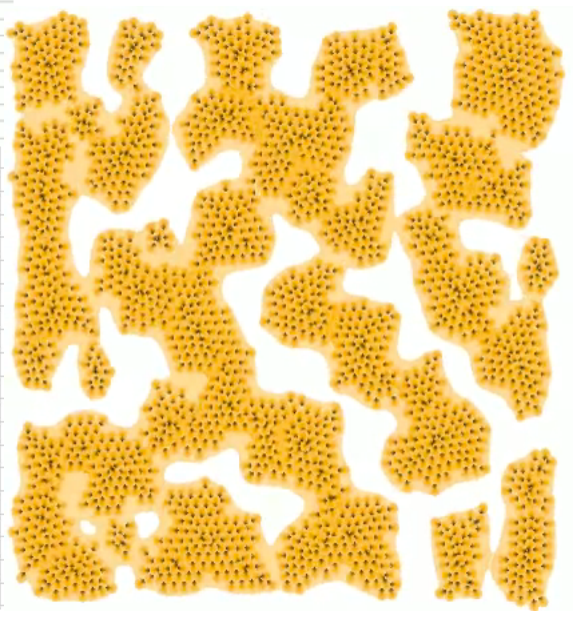
- Structure: mass/spring system
  - ➔ n+1 nodes
  - ➔ membrane, cytoskeleton, cortex
  - ➔ cell deformation



- Mitose
  - ➔ orientated mitosis given an axis

- Cell adhesion/repulsion
  - ➔ differential interaction





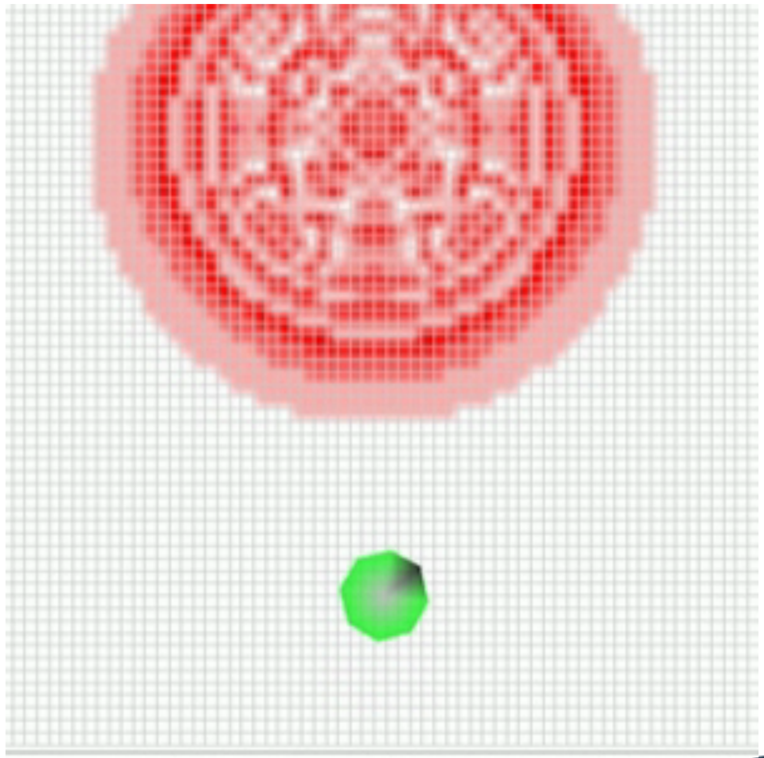
# Virtual Biological Model (2/3)

## Molecular Virtual Chemistry

**Molecular level modelled with diffusion/reaction equations**

$$\frac{\delta_i(x,t)}{\delta t} = D_i \, \Delta_i(x,t) - R_i(x,t)$$

- Set of molecules. Ex: {A, B,C}
- Set of reactions. Ex: $\{2A + B \rightarrow C\}$
- Set of 2D discrete layers.
  One grid layer per molecule type
- Equations solved in 2 steps:
  1) diffusion
  2) reaction



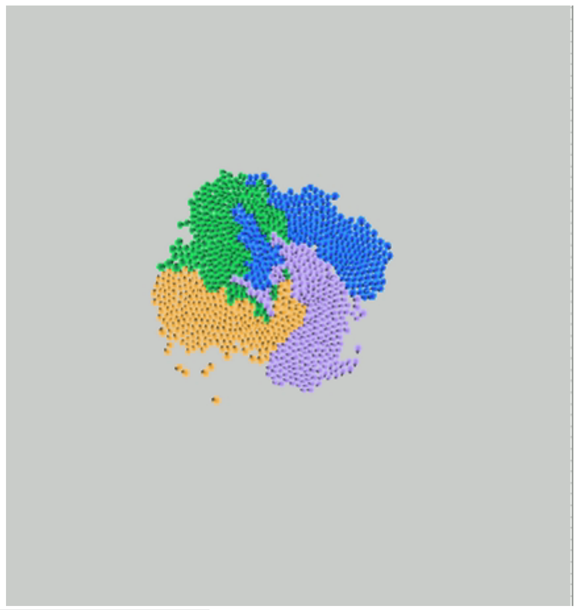
# Virtual Biological Model (3/3)

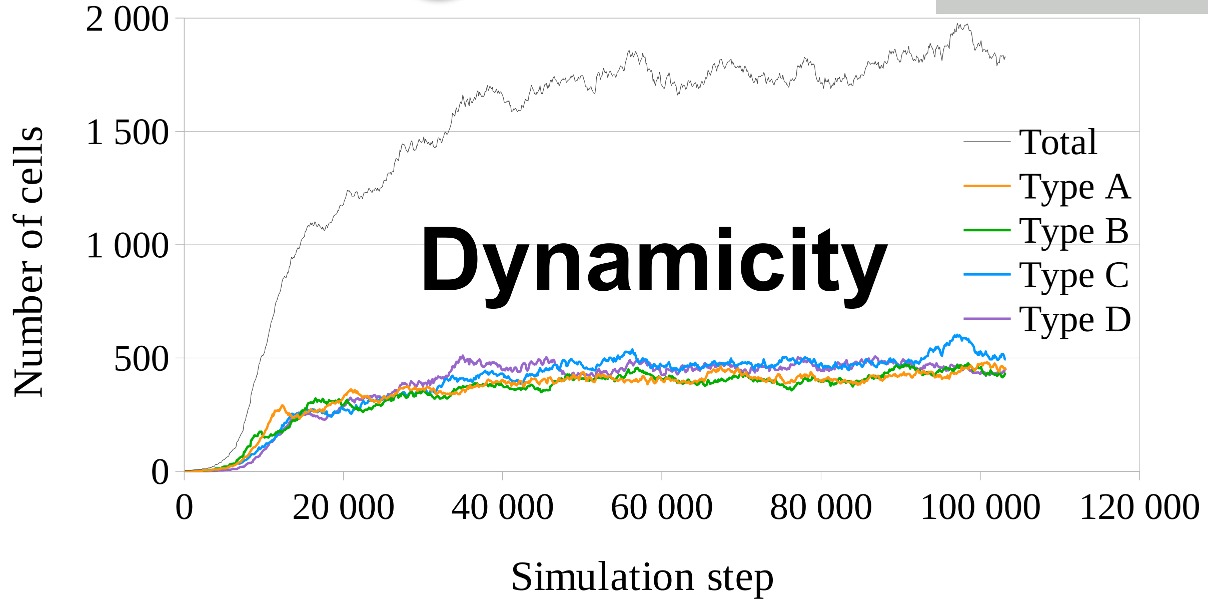
## Virtual Growth

$$\begin{tikzcd}
T_A \arrow[r, dashed] \arrow[d] & B & T_B \arrow[l] \arrow[d, dashed] \\
A & & C \\
T_D \arrow[u, dashed] \arrow[r] & D & T_C \arrow[l, dashed] \arrow[u]
\end{tikzcd}$$

$\longrightarrow$ produces

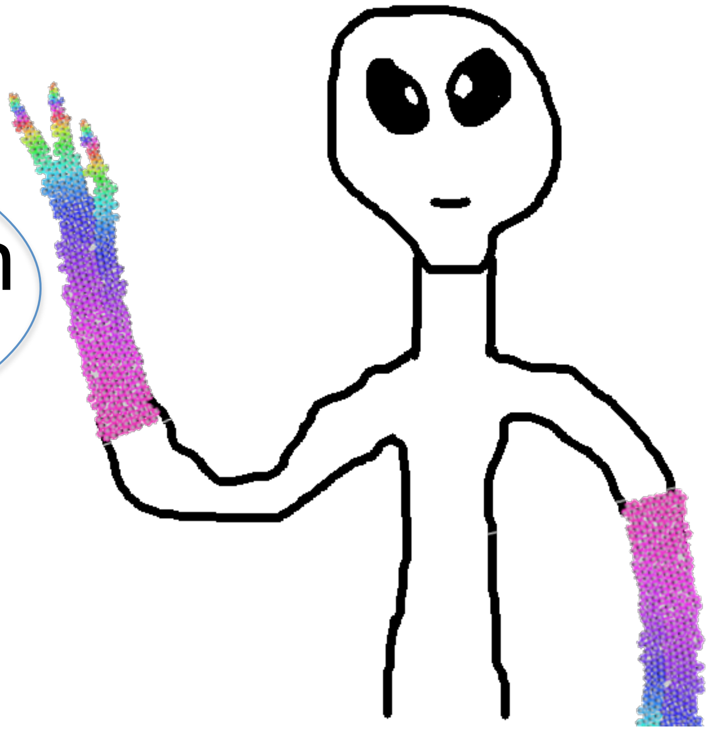
$\dashrightarrow$ consumes





**Dynamicity**

Hey Vincent,
do you know that you are not in a bioinformatics conference today?



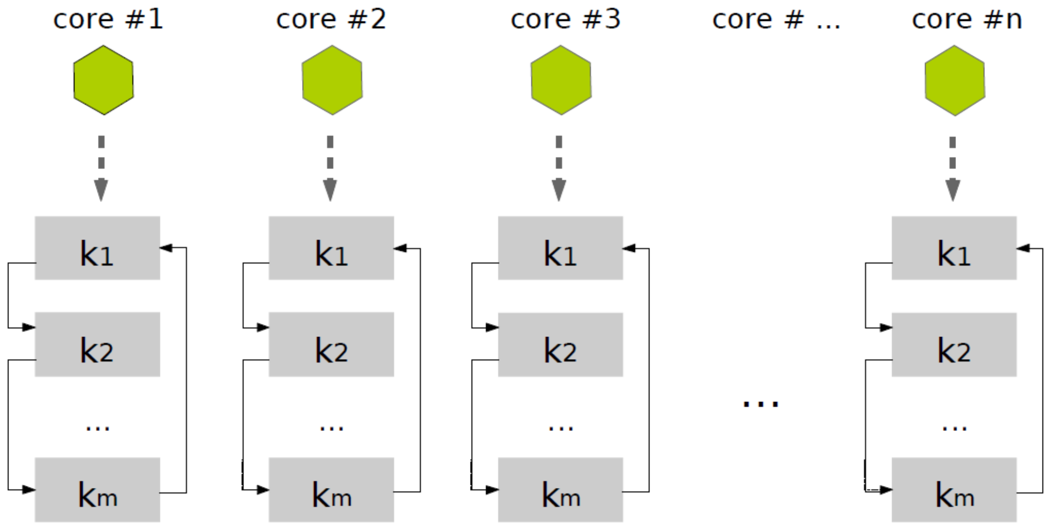
# Parallel implementation (1/2)

- Parallel hardware and device are everywhere
- Parallel programing gets easier
- Numerous parallel frameworks are available

- Our model seems well adapted to parallel implementation
- We choose to use the **OpenCL framework** to implement it
  ➔ we can use CPUs, GPUs, FPGAs, etc.

# Parallel implementation (2/2)

- Fine grained implementation: a cell = an OpenCL core
  - ➔ model coupled with a Multi-Agent System



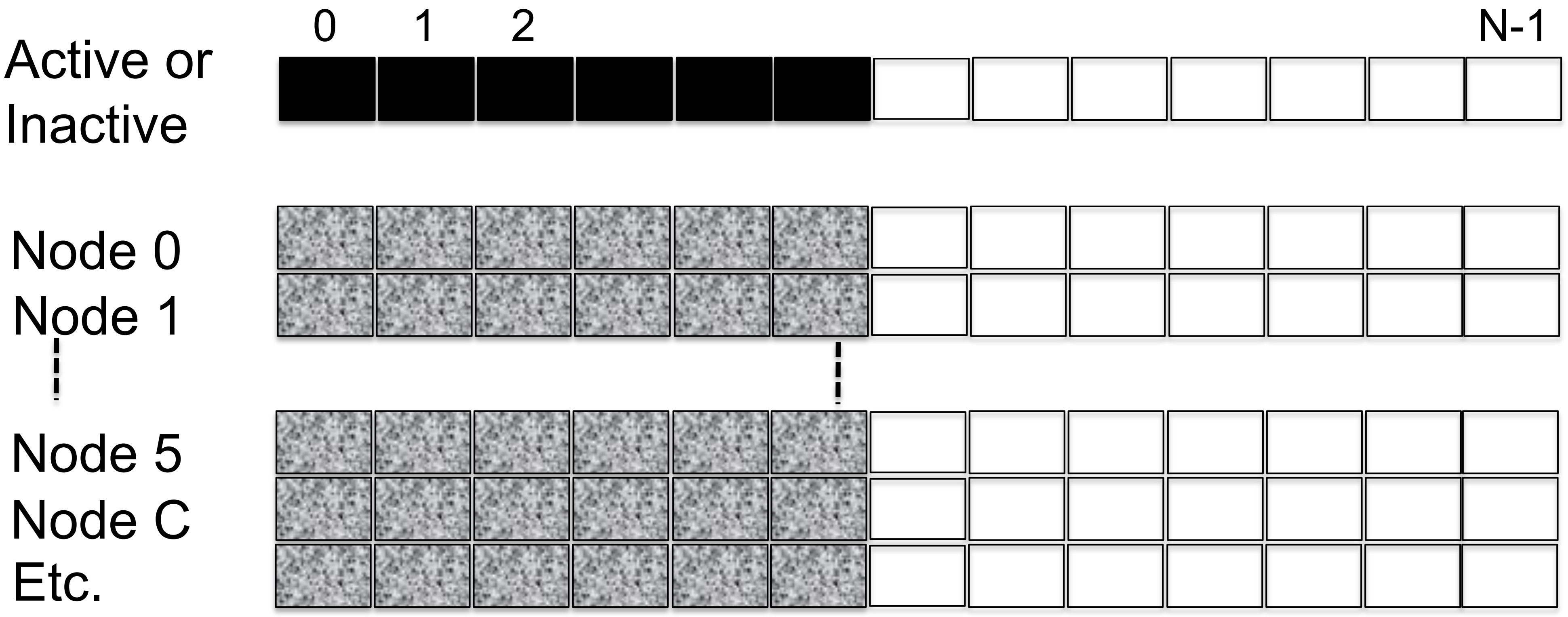
Kernel k1: computes forces
Kernel k2: integrates forces (Euler method)

- Data stored into structures of arrays: nodes, etc.
  - ➔ adapted data structure for OpenCL: a cell = an id

Question: How to find a new Id for a new Virtual Cell?

# How to get a new Id ? (1/3)

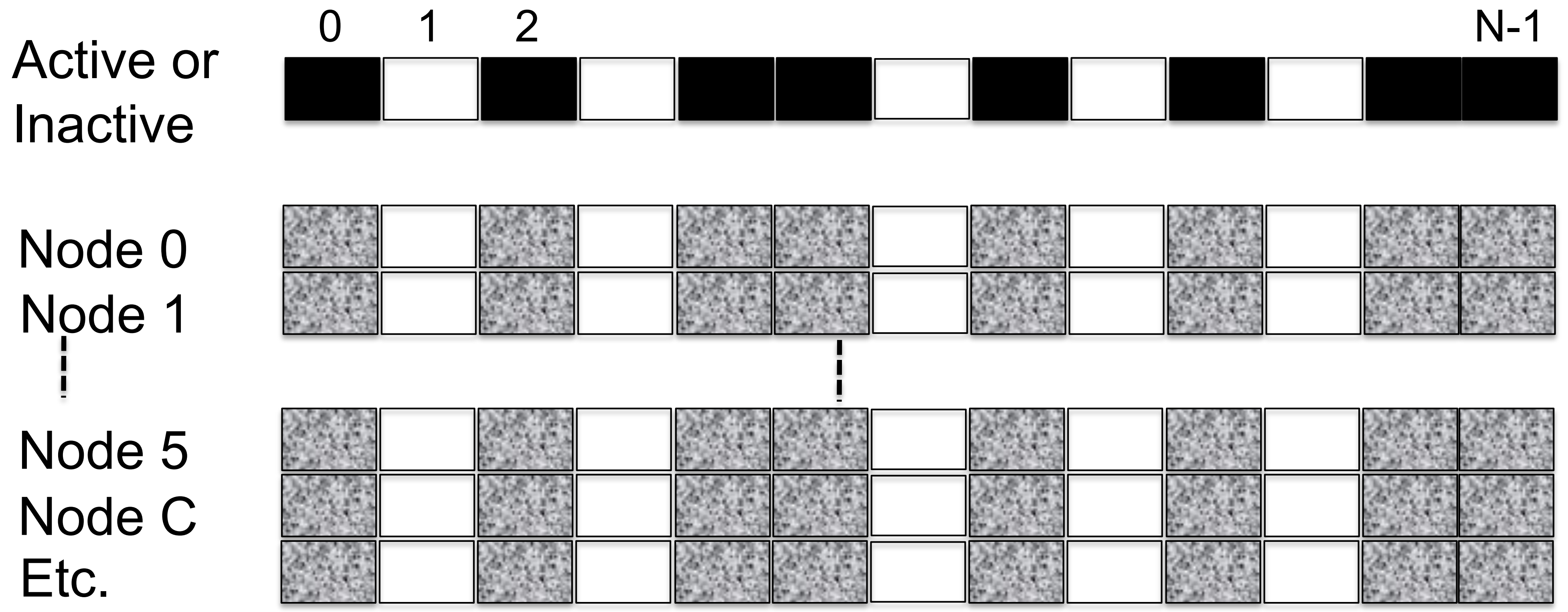
N (± $10^6$) Virtual Cells : structures of arrays



At the beginning of the simulation… easy!

# How to get a new Id ? (1/3)

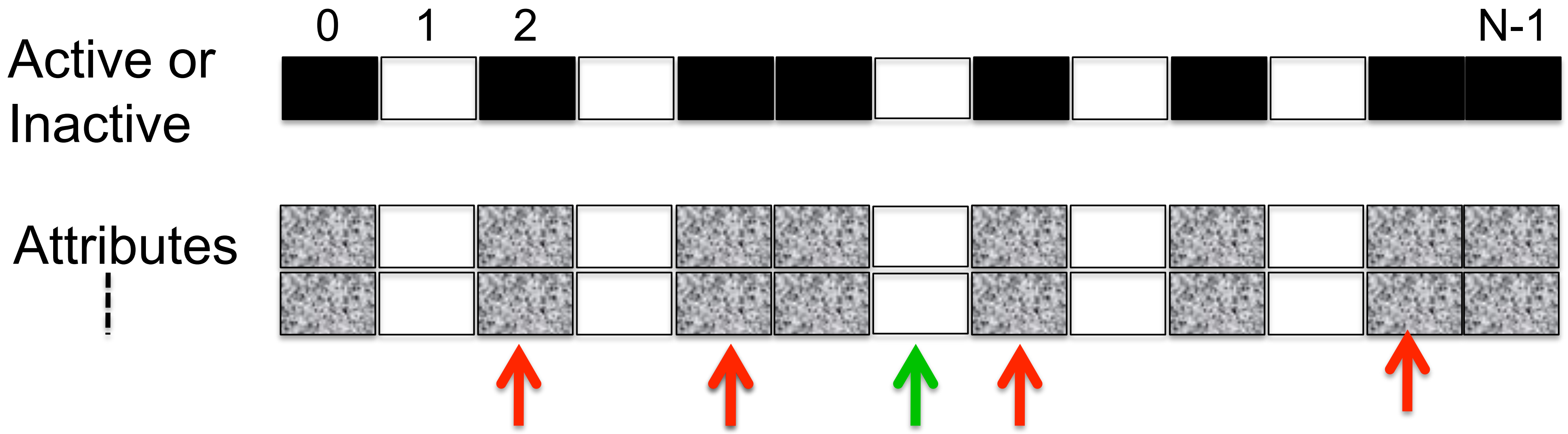
N (± $10^6$) Virtual Cells : structures of arrays



After births and deaths….?

# How to get a new Id ? (2/3)

Some previous works (1): [Lysenko & D'Souza, 2008]



A stochastic method:

A random choise happens…

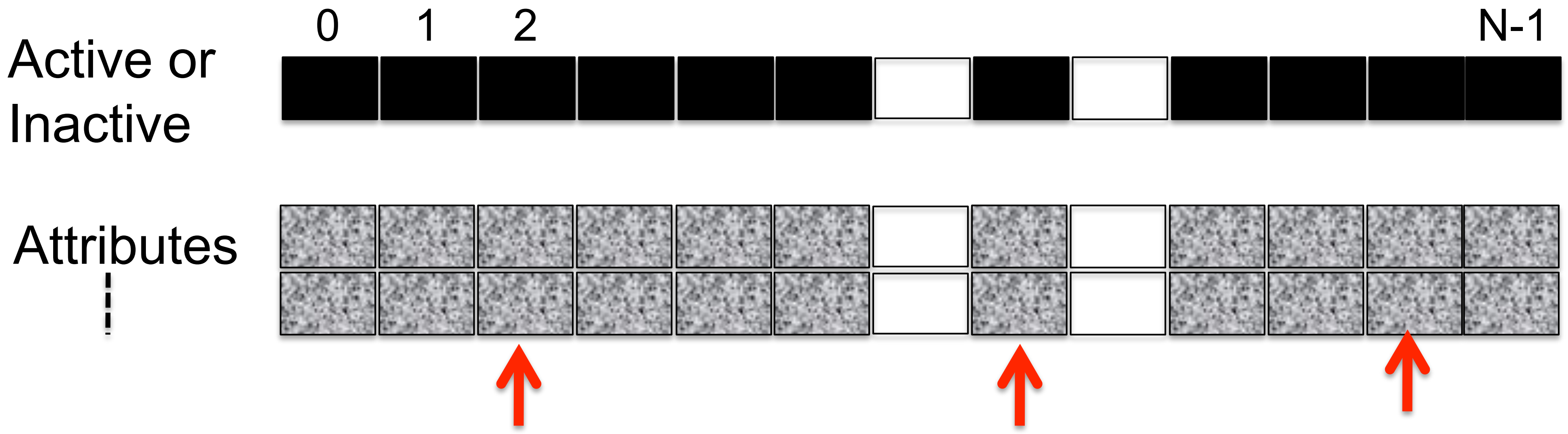
…until an inactive element is obtained!

Main drawback:

What appends if the memory is nearly full?

# How to get a new Id ? (2/3)

Some previous works (2): [Jeannin-Girardon, Ph.D, 2014]
[Jeannin-Girardon et al, TCBB, 2015]

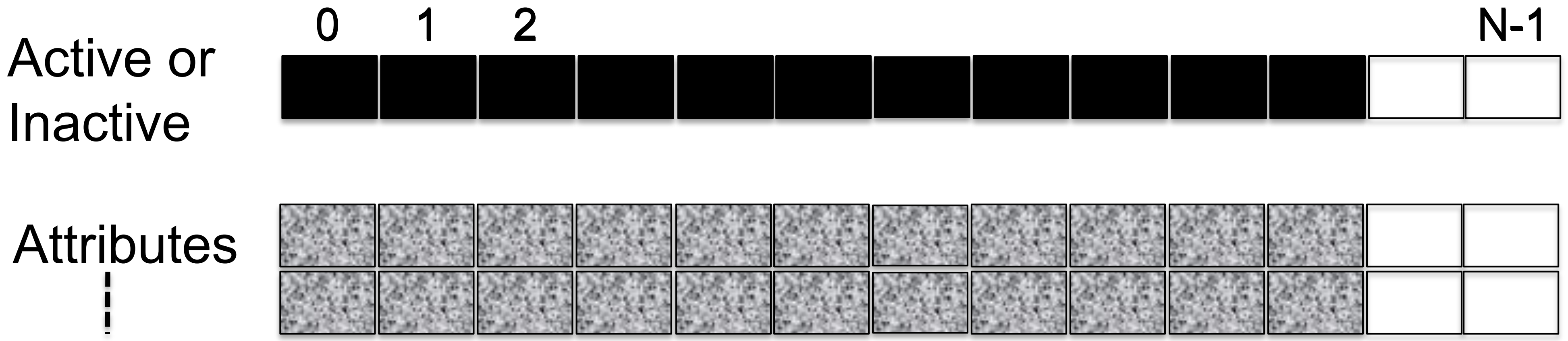


A hybrid method:

Dynamic switch : Stochastic Method & Parallel Sort

## How to get a new Id ? (2/3)

Some previous works (2): [Jeannin-Girardon, Ph.D, 2014]
[Jeannin-Girardon et al, TCBB, 2015]



A hybrid method:

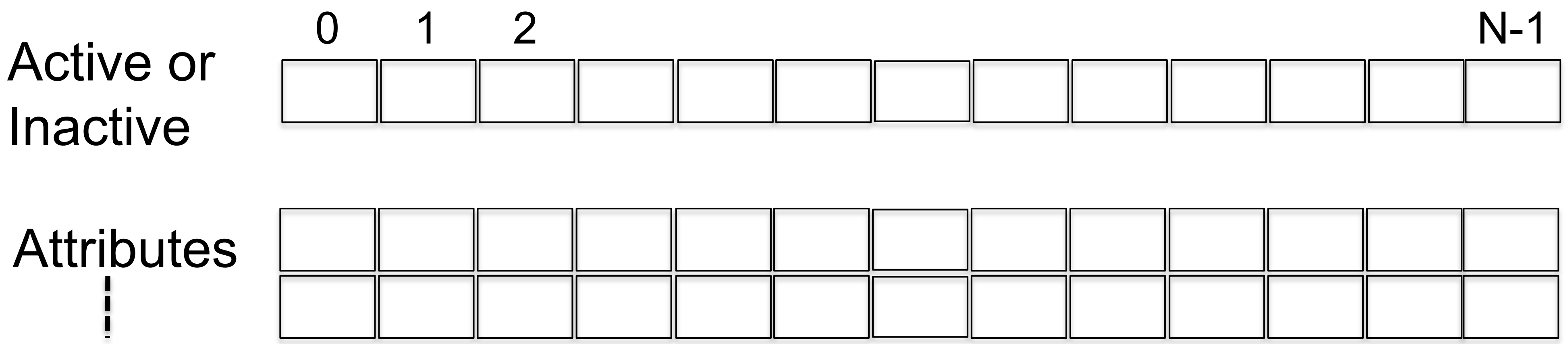
Dynamic switch : Stochastic Method & Parallel Sort

Main drawbacks:

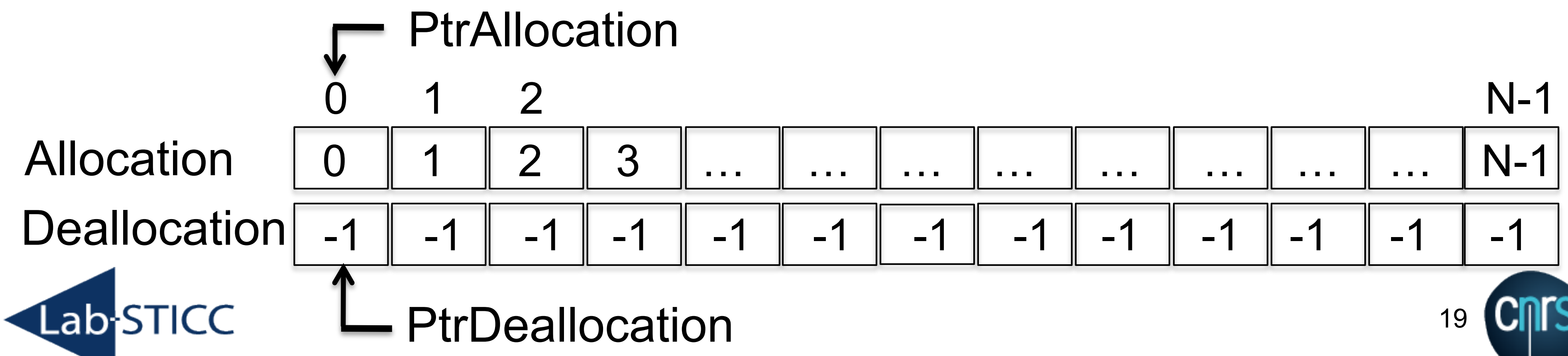
- How to choose the thresholds to switch?
- Parallel sort… data transfers

# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]

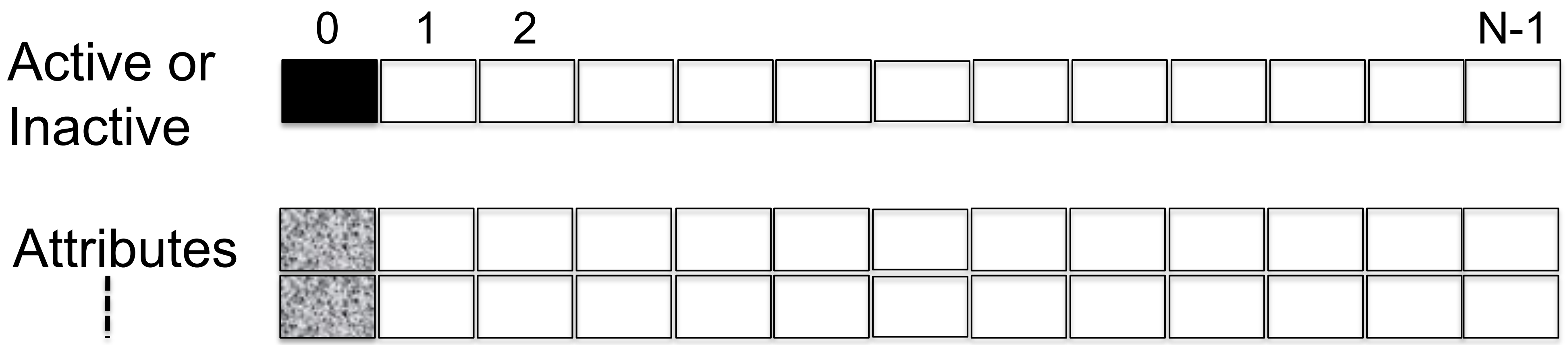


**Available Indexes are stored in two buffers & atomic_inc**

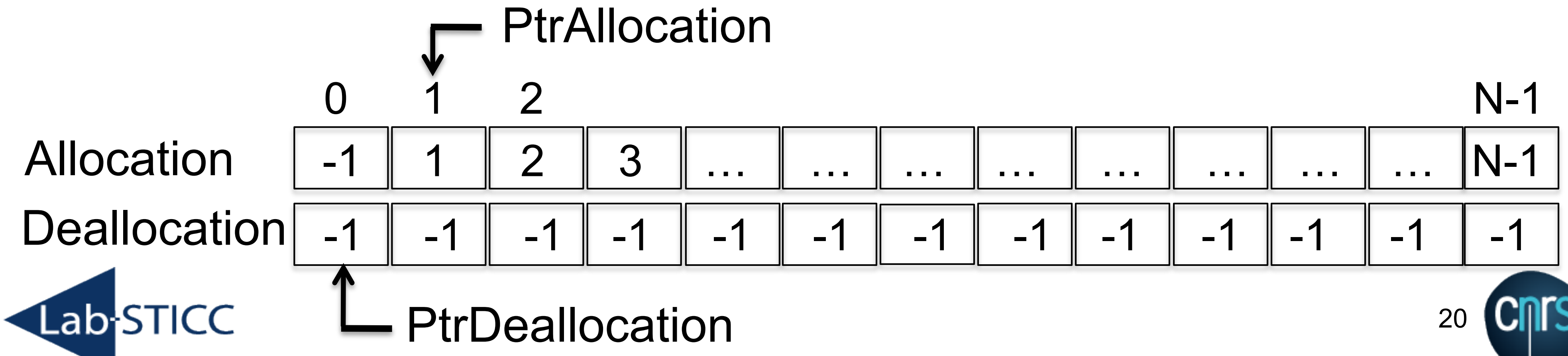


# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]



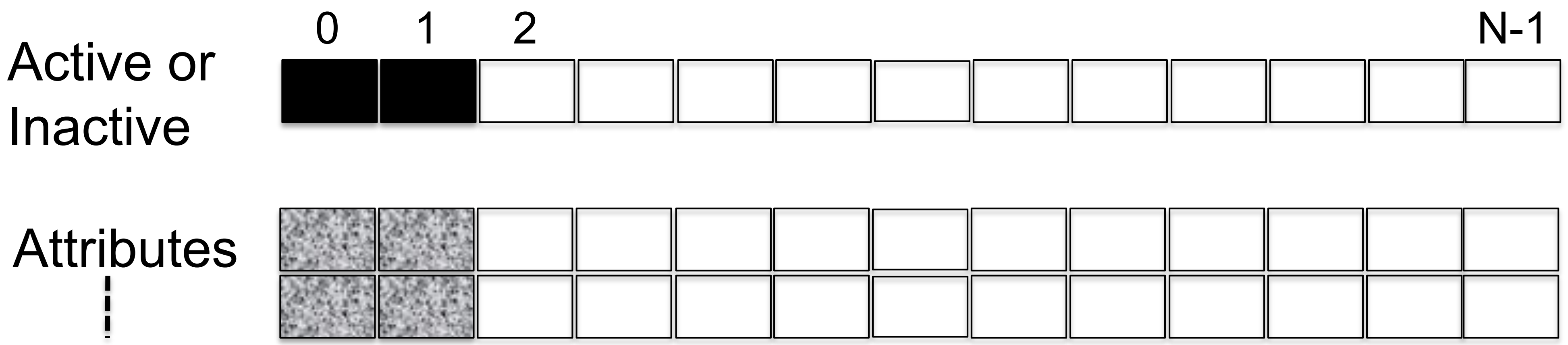
**Available Indexes are stored in two buffers & atomic_inc**



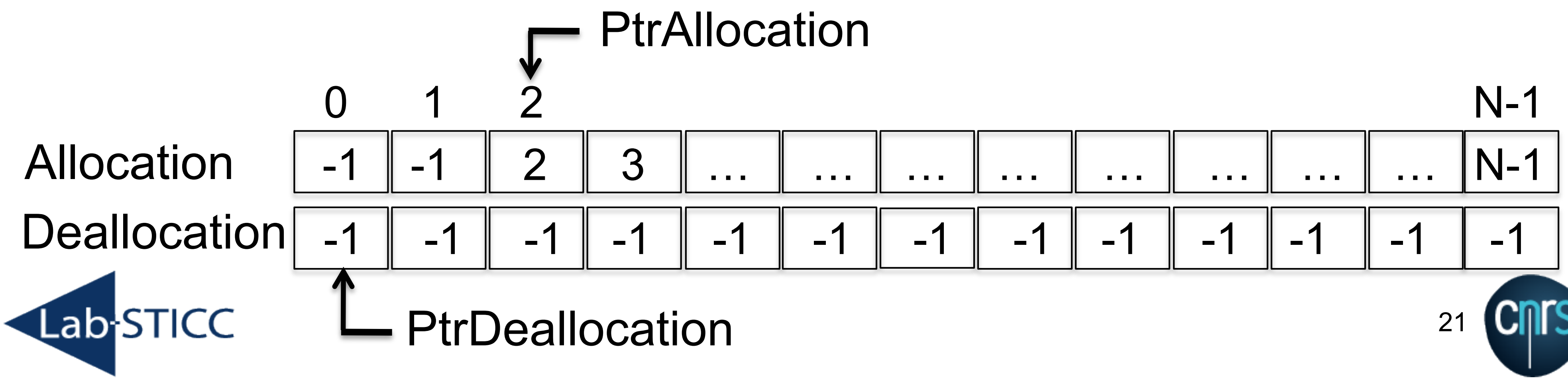
| | 0 | 1 | 2 | | | | | | | | | | N-1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Allocation | -1 | 1 | 2 | 3 | … | … | … | … | … | … | … | … | N-1 |
| Deallocation | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]



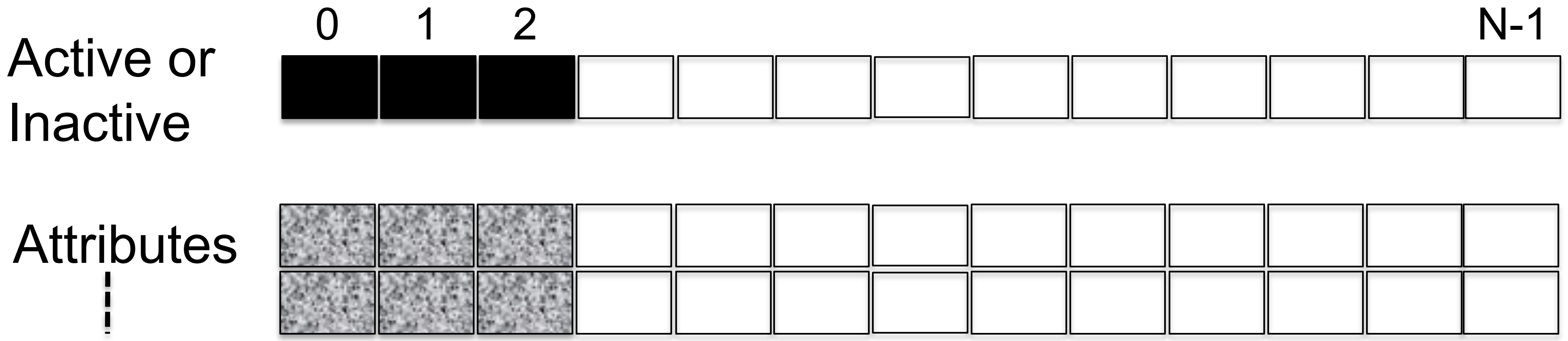
**Available Indexes are stored in two buffers & atomic_inc**



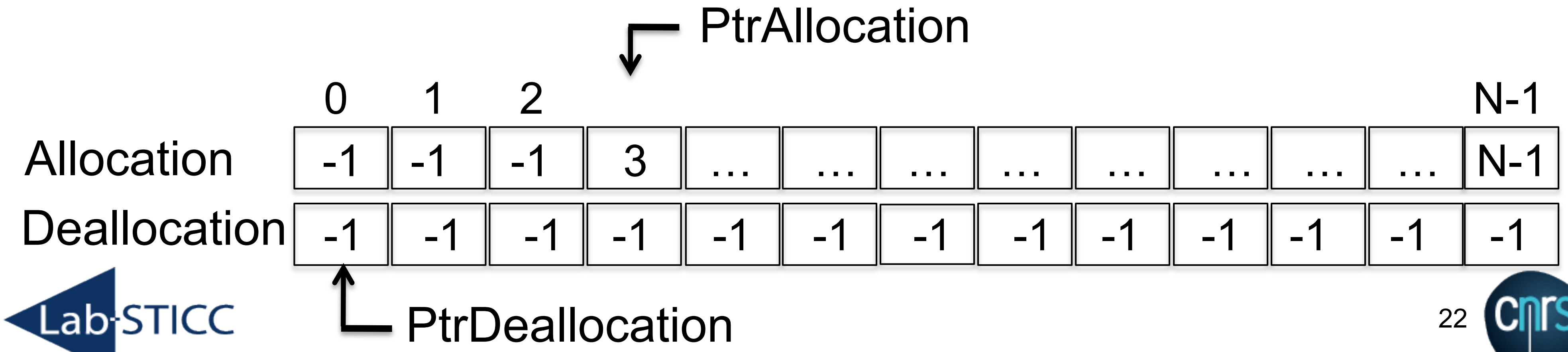
|  | 0 | 1 | 2 |  |  |  |  |  |  |  |  |  | N-1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Allocation | -1 | -1 | 2 | 3 | … | … | … | … | … | … | … | … | N-1 |
| Deallocation | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]



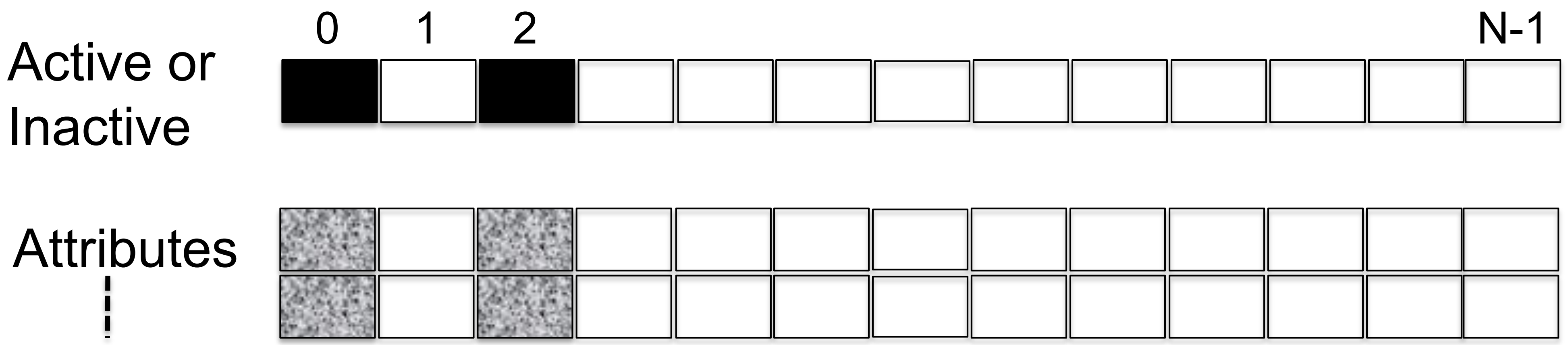
**Available Indexes are stored in two buffers & atomic_inc**



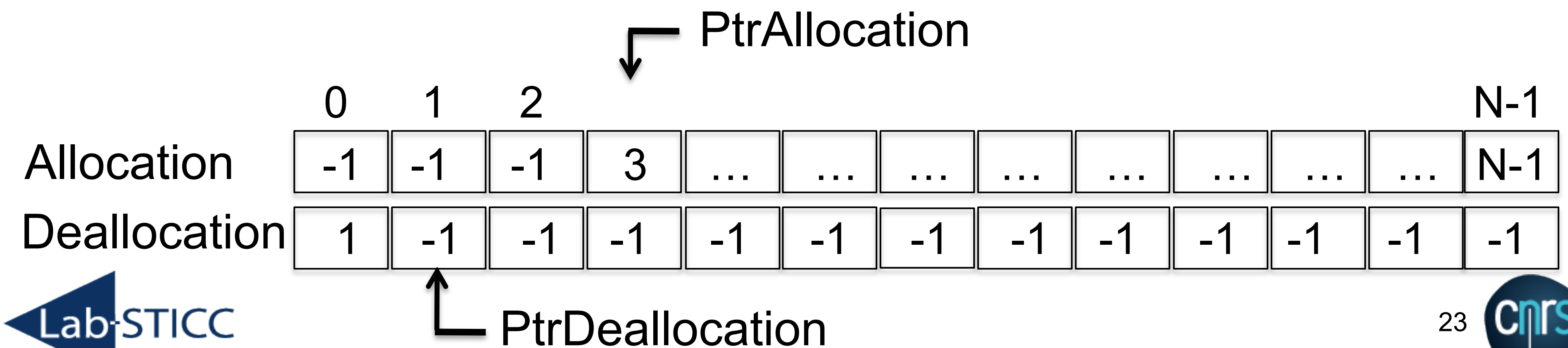
| | 0 | 1 | 2 | | | | | | | | | | N-1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Allocation | -1 | -1 | -1 | 3 | … | … | … | … | … | … | … | … | N-1 |
| Deallocation | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]



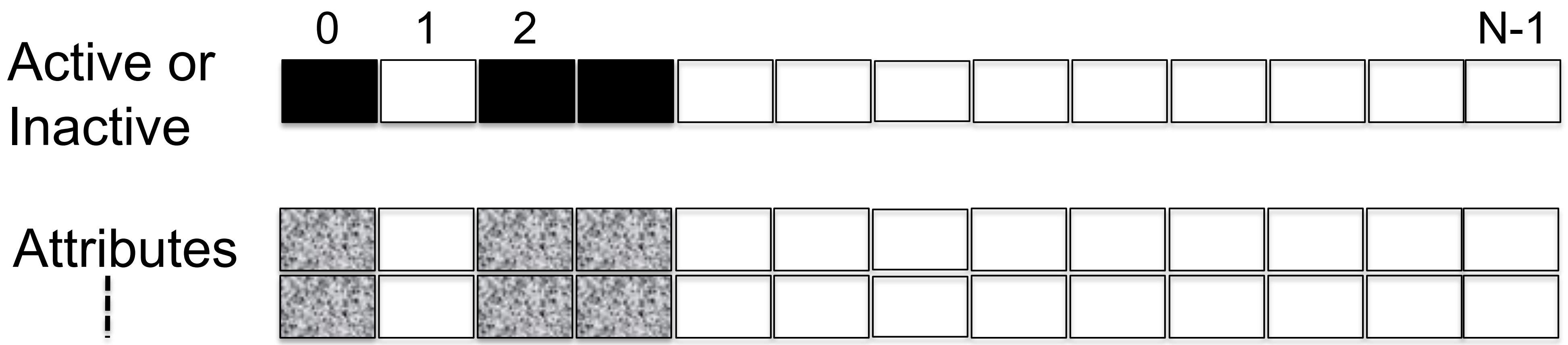
**Available Indexes are stored in two buffers & atomic_inc**



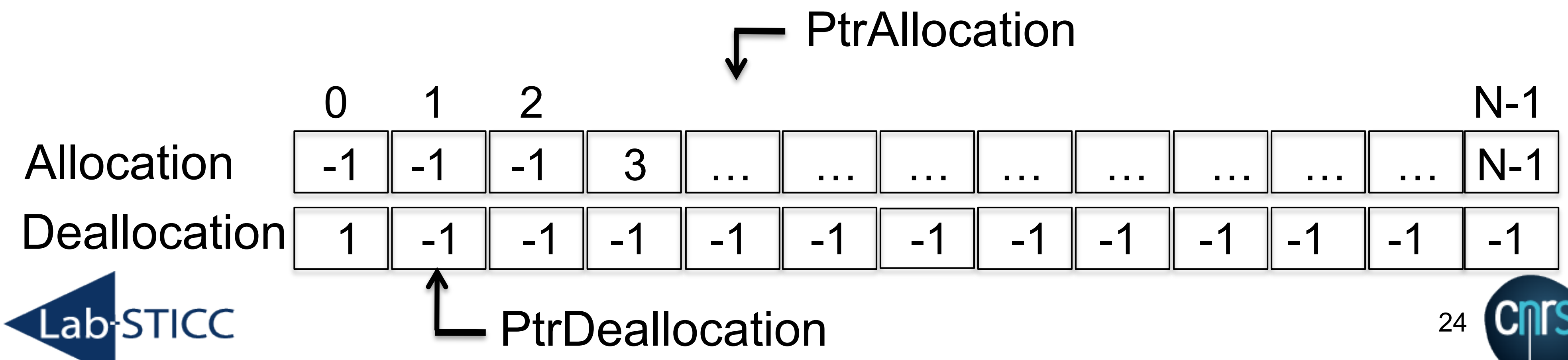
| | 0 | 1 | 2 | | | | | | | | | | N-1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Allocation | -1 | -1 | -1 | 3 | … | … | … | … | … | … | … | … | N-1 |
| Deallocation | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]

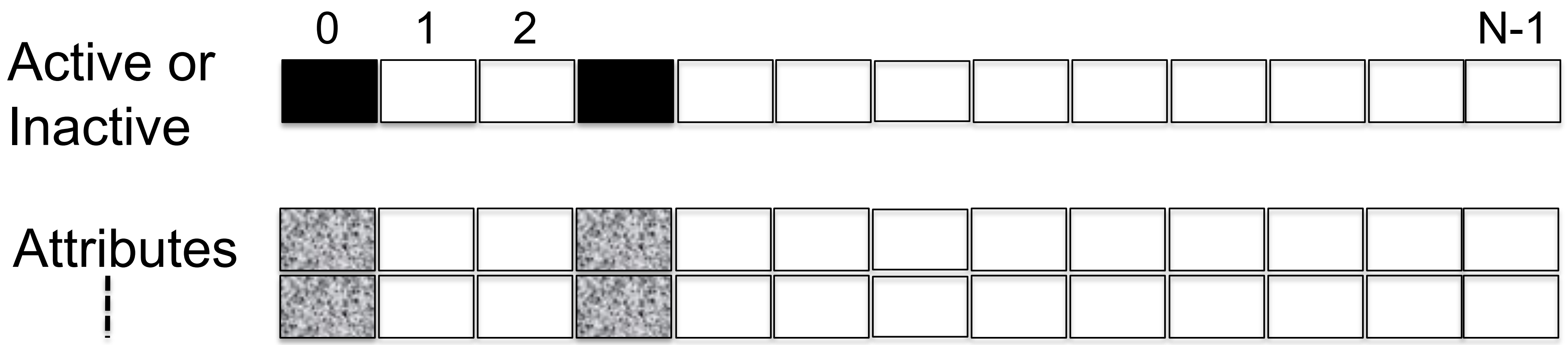


**Available Indexes are stored in two buffers & atomic_inc**

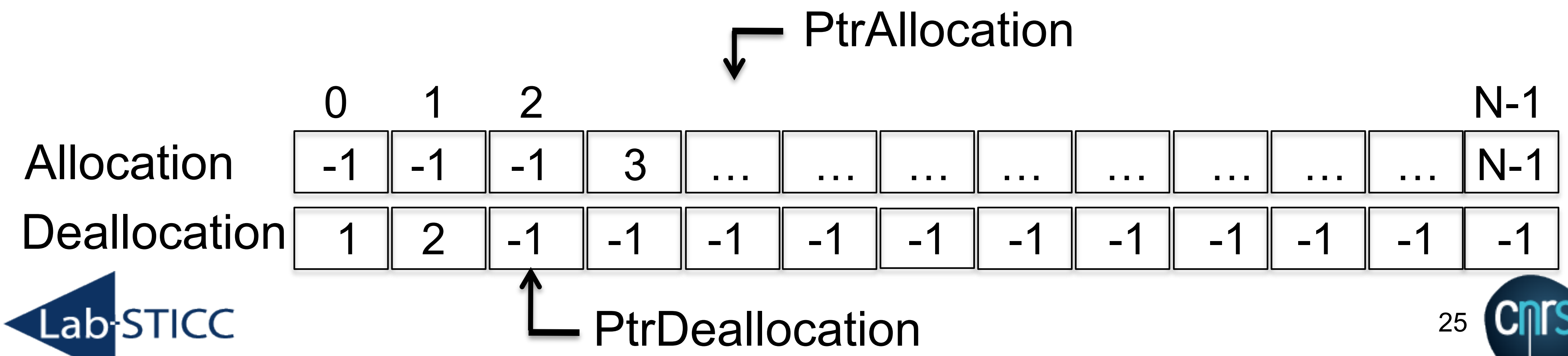


# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]



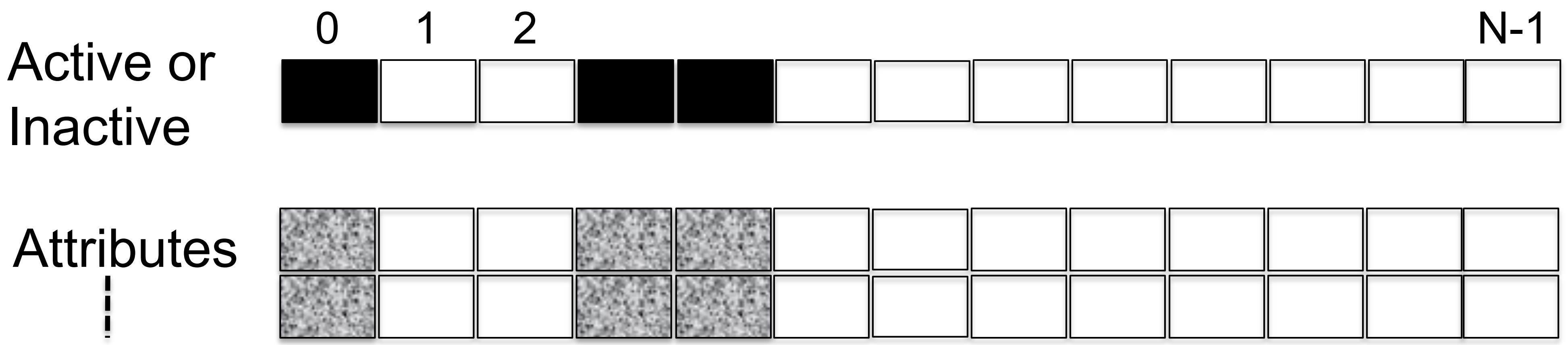
**Available Indexes are stored in two buffers & atomic_inc**



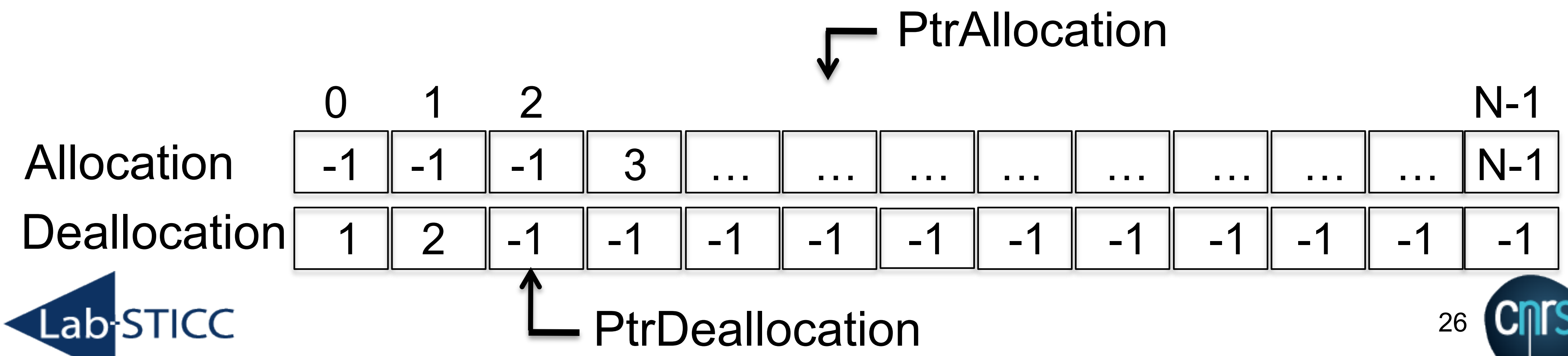
|  | 0 | 1 | 2 |  |  |  |  |  |  |  |  |  | N-1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Allocation | -1 | -1 | -1 | 3 | … | … | … | … | … | … | … | … | N-1 |
| Deallocation | 1 | 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]

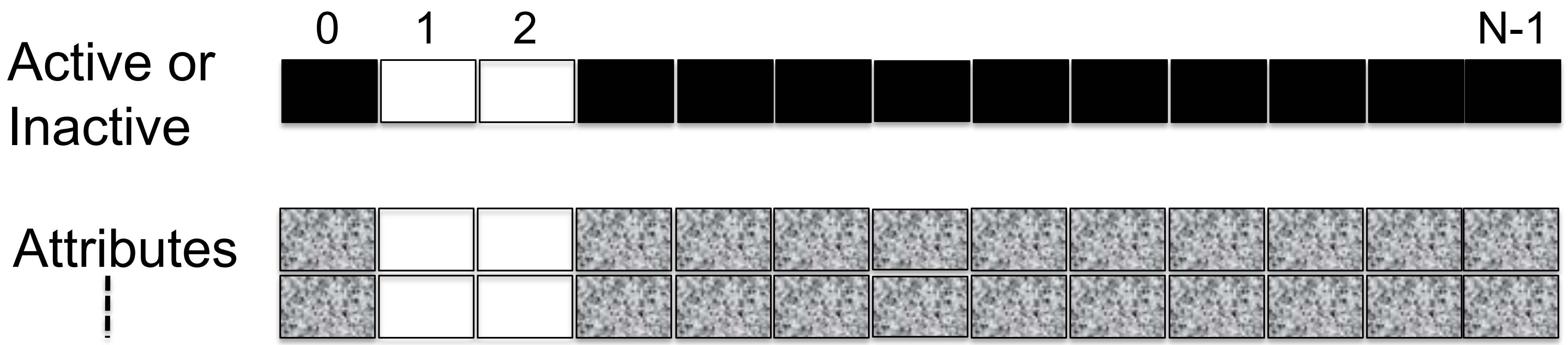


**Available Indexes are stored in two buffers & atomic_inc**

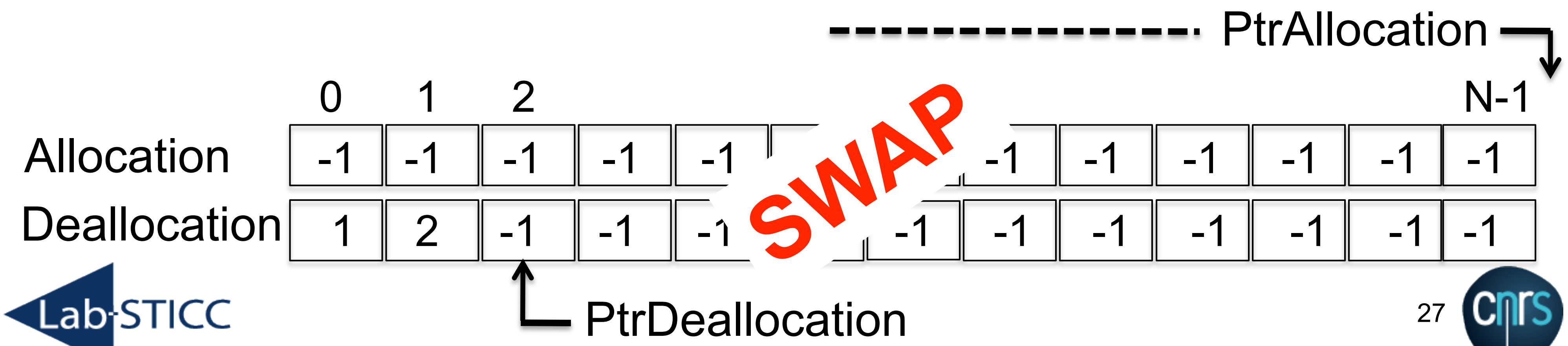


# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]

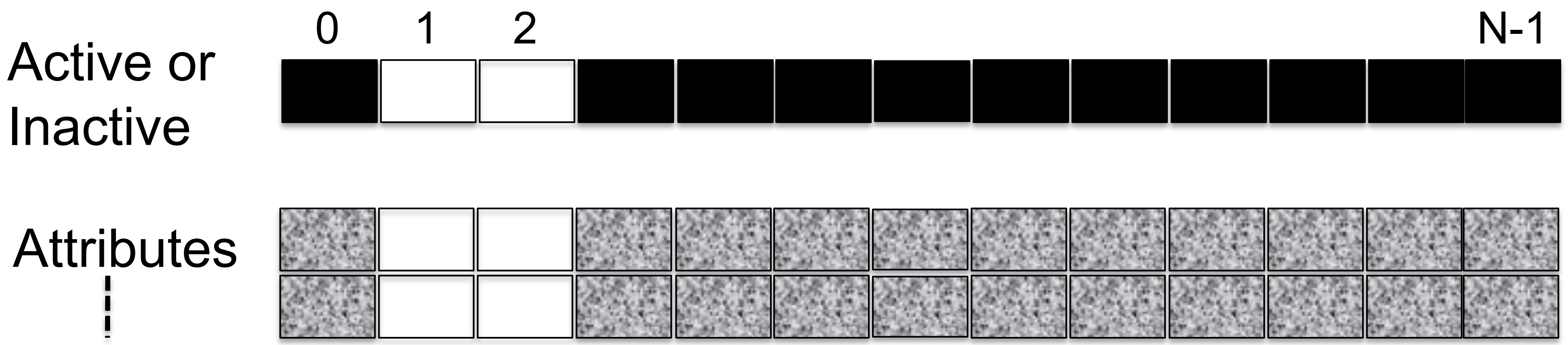


**Available Indexes are stored in two buffers & atomic_inc**

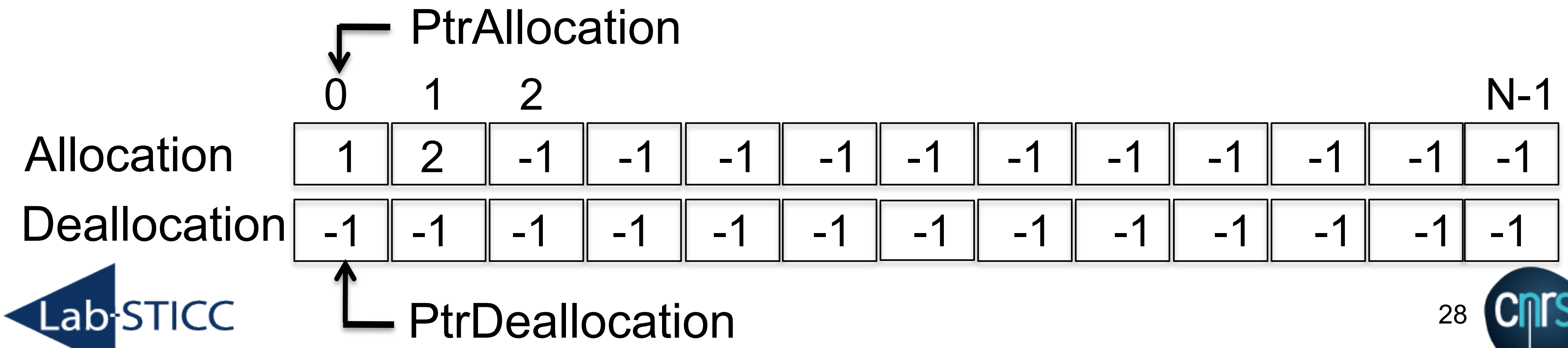


# How to get a new Id ? (3/3)

Our proposition (1): [Jeannin-Girardon et al, Compas, 2016]



**Available Indexes are stored in two buffers & atomic_inc**



# How to get a new Id ? (3/3)

Our proposition (2): [Jeannin-Girardon et al, Compas, 2016]

**res = atomic_inc(var):** OpenCL atomic operation

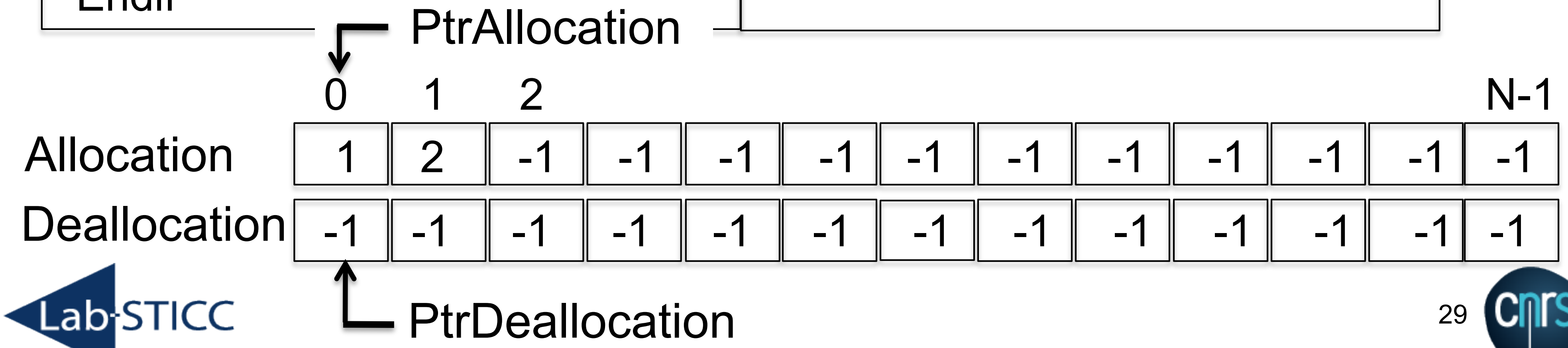
➔ tmp=var; var++; return tmp

```
Allocation:

Ptr = atomic_inc(PtrAllocation)
If Ptr < N and Allocation[Ptr] != -1
Then    Id = Allocation[Ptr]
        Allocation[Ptr] = -1
        Return Id
Endif
```

```
Deallocation:

Ptr = atomic_inc(PtrDeallocation)
Desallocation[Ptr] = Id
```

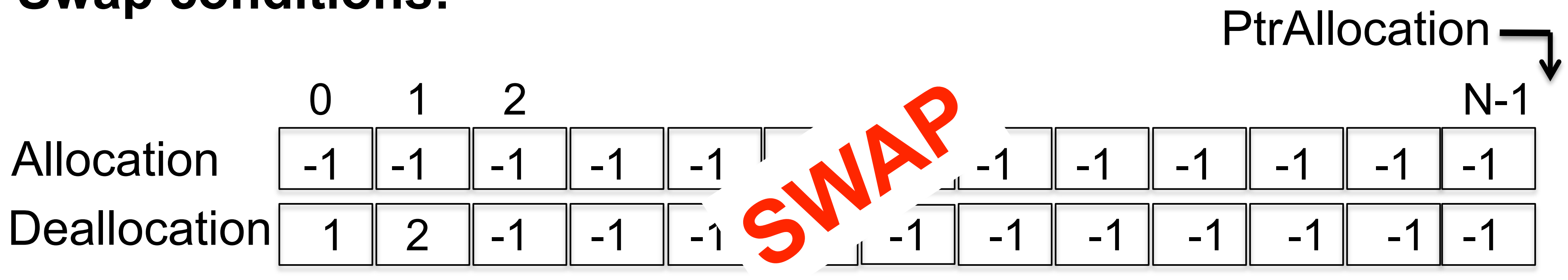


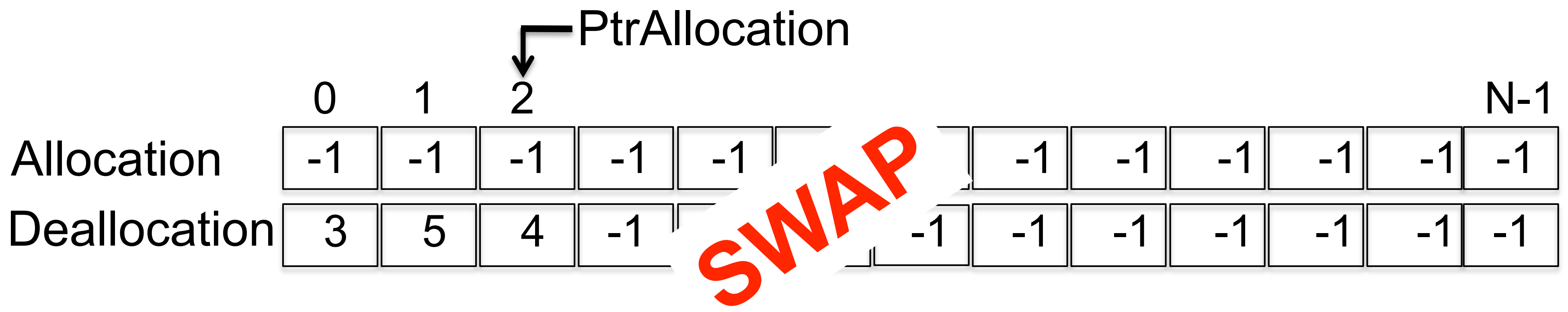
| | 0 | 1 | 2 | | | | | | | | | | N-1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Allocation | 1 | 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Deallocation | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

# How to get a new Id ? (3/3)

Our proposition (3): [Jeannin-Girardon et al, Compas, 2016]

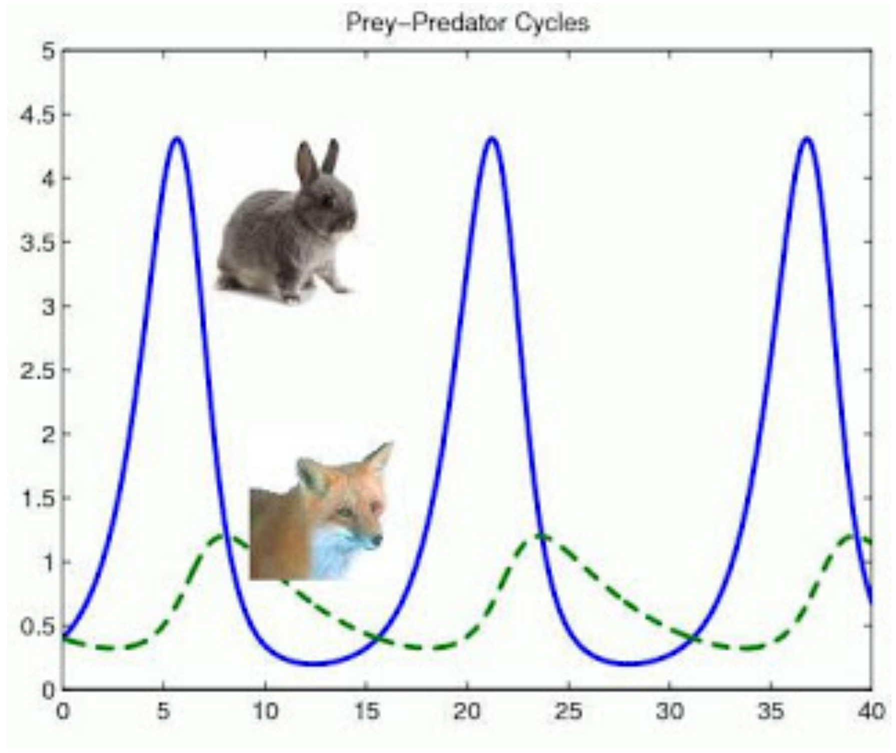
**Swap conditions:**





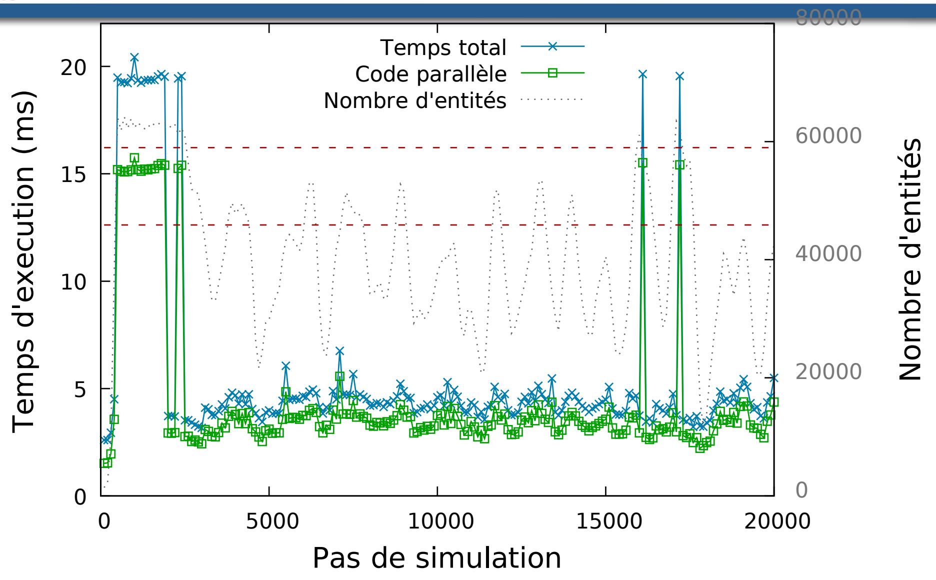
# Results: Prey-Predator model (1/2)

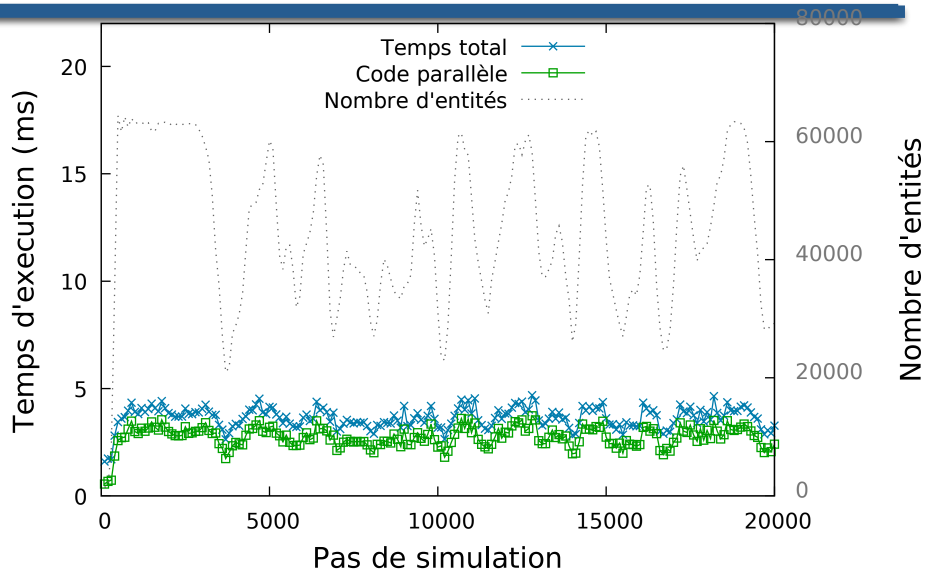
Prey-Predator model



High dynamicity…

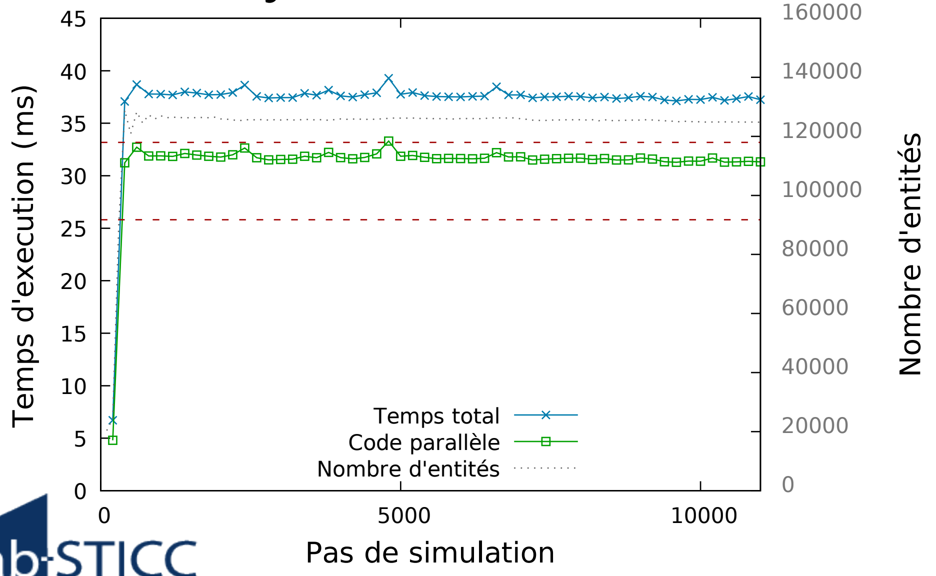
# Results: Prey-Predator model (2/2)

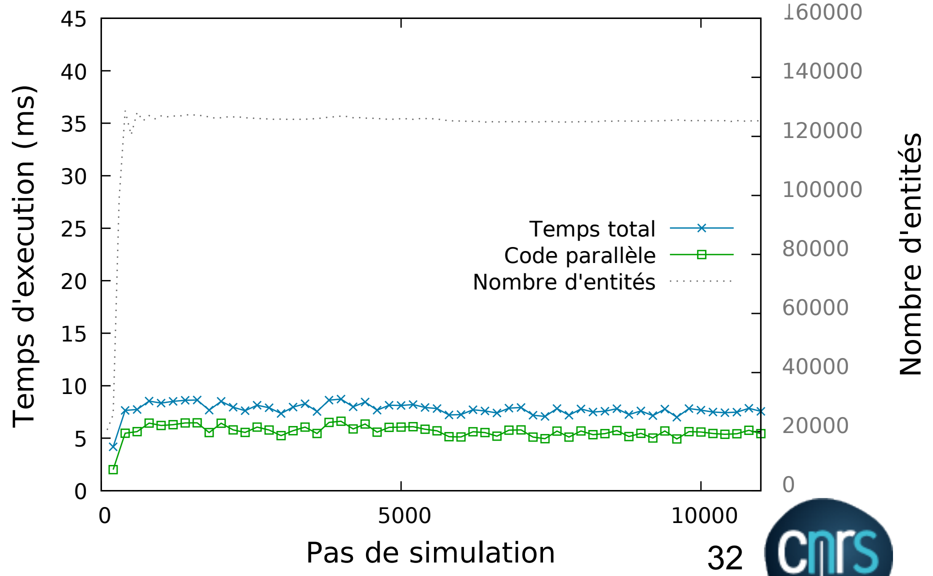


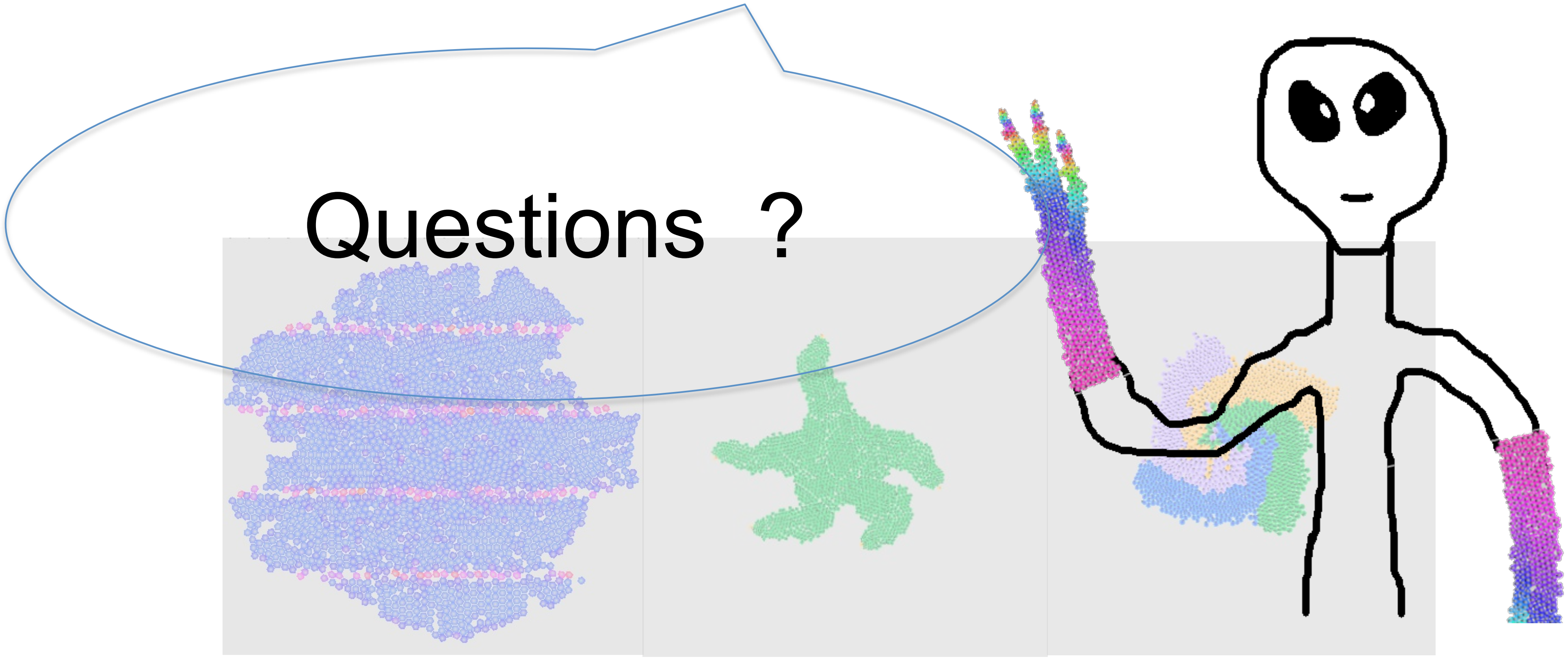


Hybrid method

Double buffer method






Questions ?