SYNACKTIV

# Spatial Frinet

SSTIC 2026 - Théo EMERIAU

# Contexte & Historique

SYNACKTIV

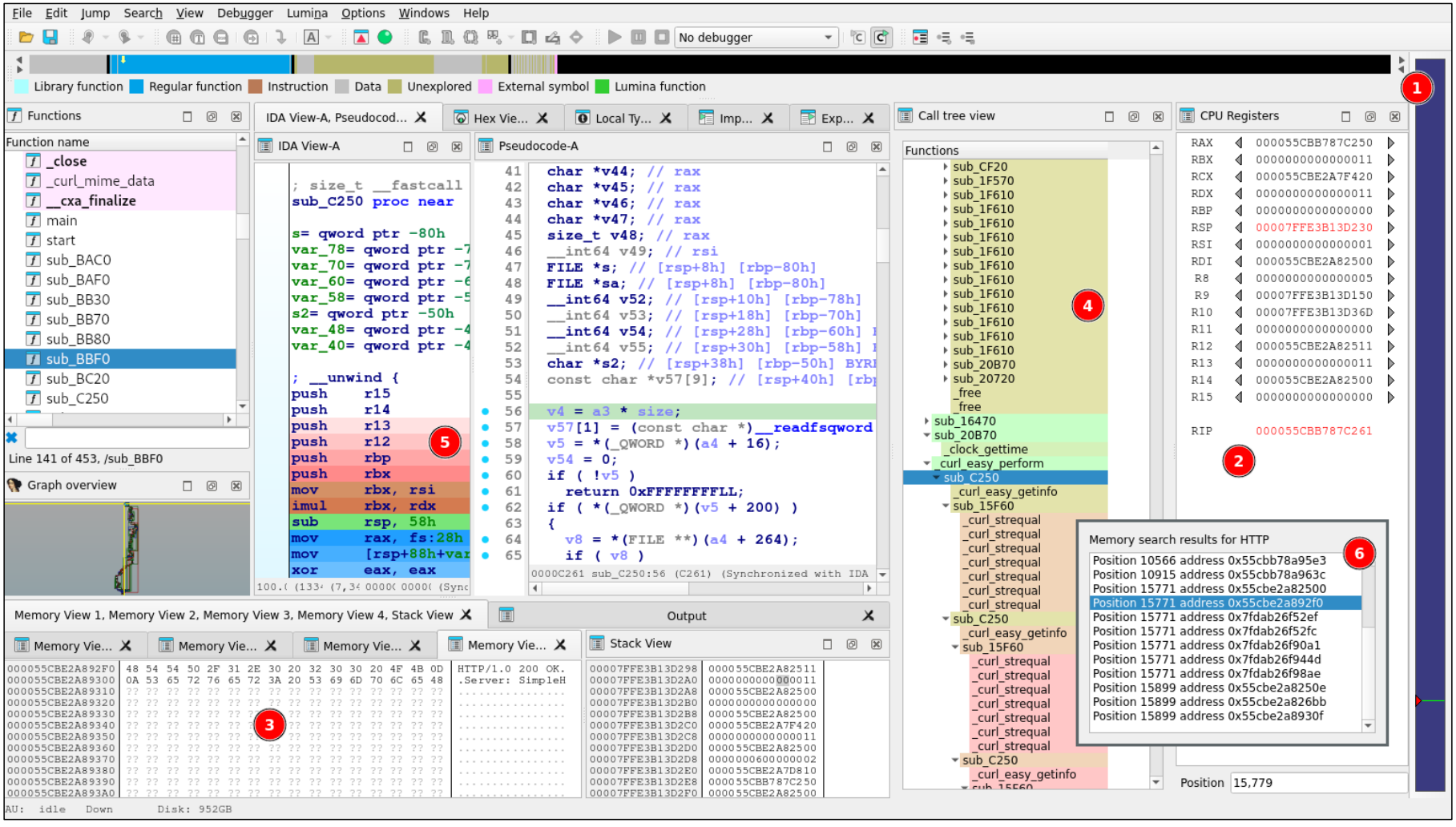
- **Combiner l'analyse statique & dynamique**
  - Traces d'exécution → IDA
- **gaasedelen/tenet (2021)**
- **synacktiv/frinet (SSTIC 2024)**
  - Traceur basé sur Frida
  - Vue « Callgraph »
  - Recherche dans la mémoire

# Trace d'exécution

SYNACKTIV

```
$ python3 trace.py spawn [BIN_PATH] [MODULE_NAME] [FUNC_ADDR] -a [ARGS]
```

```
rip=0x7fdf256e2494,rdi=0x7fdf24291df0,mr=0x7ffc857caeb8:f01d2924df7f0000
rip=0x7fdf256e249b,mw=0x7ffc857caea8:00000000
rip=0x7fdf256e24a5,mw=0x7ffc857caeb8:0000000000000000
rip=0x7fdf256e24b0
rip=0x7fdf256e24b3
rip=0x7fdf256e24b5
rip=0x7fdf256e24ba,rdi=0x7fdf2428a890,mr=0x7ffc857caeb0:90a82824df7f0000
rip=0x7fdf256e24c1,mw=0x7ffc857caeb0:0000000000000000
rip=0x7fdf256e24cc
...
```


File Edit Jump Search View Debugger Lumina Options Windows Help
No debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions
Function name
_close
_curl_mime_data
__cxa_finalize
main
start
sub_BAC0
sub_BAF0
sub_BB30
sub_BB70
sub_BB80
sub_BBF0
sub_BC20
sub_C250
Line 141 of 453, /sub_BBF0
Graph overview
IDA View-A, Pseudocod...
Hex Vie...
Local Ty...
Imp...
Exp...
IDA View-A
; size_t __fastcall
sub_C250 proc near
s= qword ptr -80h
var_78= qword ptr -7
var_70= qword ptr -7
var_60= qword ptr -6
var_58= qword ptr -5
s2= qword ptr -50h
var_48= qword ptr -4
var_40= qword ptr -4
; __unwind {
push r15
push r14
push r13
push r12
push rbp
push rbx
mov rbx, rsi
imul rbx, rdx
sub rsp, 58h
mov rax, fs:28h
mov [rsp+88h+var
xor eax, eax
Pseudocode-A
char *v44; // rax
char *v45; // rax
char *v46; // rax
char *v47; // rax
size_t v48; // rax
__int64 v49; // rsi
FILE *s; // [rsp+8h] [rbp-80h]
FILE *sa; // [rsp+8h] [rbp-80h]
__int64 v52; // [rsp+10h] [rbp-78h]
__int64 v53; // [rsp+18h] [rbp-70h]
__int64 v54; // [rsp+28h] [rbp-60h]
__int64 v55; // [rsp+30h] [rbp-58h]
char *s2; // [rsp+38h] [rbp-50h] BYR
const char *v57[9]; // [rsp+40h] [rbp
v4 = a3 * size;
v57[1] = (const char *)__readfsqword
v5 = *(_QWORD *)(a4 + 16);
v54 = 0;
if ( !v5 )
return 0xFFFFFFFFLL;
if ( *(_QWORD *)(v5 + 200) )
{
v8 = *(FILE **)(a4 + 264);
if ( v8 )
0000C261 sub_C250:56 (C261) (Synchronized with IDA
Memory View 1, Memory View 2, Memory View 3, Memory View 4, Stack View
Memory Vie...
Memory Vie...
Memory Vie...
Memory Vie...
000055CBE2A892F0 48 54 54 50 2F 31 2E 30 20 32 30 30 20 4F 4B 0D HTTP/1.0 200 OK.
000055CBE2A89300 0A 53 65 72 76 65 72 3A 20 53 69 6D 70 6C 65 48 .Server: SimpleH
Output
Stack View
00007FFE3B13D298 000055CBE2A82511
00007FFE3B13D2A0 0000000000000011
00007FFE3B13D2A8 000055CBE2A82500
00007FFE3B13D2B0 0000000000000000
00007FFE3B13D2B8 000055CBE2A82500
00007FFE3B13D2C0 000055CBE2A7F420
00007FFE3B13D2C8 0000000000000011
00007FFE3B13D2D0 000055CBE2A82500
00007FFE3B13D2D8 0000000600000002
00007FFE3B13D2E0 000055CBE2A7D810
00007FFE3B13D2E8 000055CBB787C250
00007FFE3B13D2F0 000055CBE2A82500
AU: idle Down Disk: 952GB
Call tree view
Functions
sub_CF20
sub_1F570
sub_1F610
sub_20B70
sub_20720
_free
sub_16470
_clock_gettime
_curl_easy_perform
sub_C250
_curl_easy_getinfo
sub_15F60
_curl_strequal
sub_C250
sub_15F60
sub_C250
_curl_easy_getinfo
CPU Registers
RAX 000055CBB787C250
RBX 0000000000000011
RCX 000055CBE2A7F420
RDX 0000000000000011
RBP 0000000000000000
RSP 00007FFE3B13D230
RSI 0000000000000001
RDI 000055CBE2A82500
R8 0000000000000005
R9 00007FFE3B13D150
R10 00007FFE3B13D36D
R11 0000000000000000
R12 000055CBE2A82511
R13 0000000000000011
R14 000055CBE2A82500
R15 0000000000000000
RIP 000055CBB787C261
Memory search results for HTTP
Position 10566 address 0x55cbb78a95e3
Position 10915 address 0x55cbb78a963c
Position 15771 address 0x55cbe2a82500
Position 15771 address 0x55cbe2a892f0
Position 15771 address 0x7fdab26f52ef
Position 15771 address 0x7fdab26f52fc
Position 15771 address 0x7fdab26f90a1
Position 15771 address 0x7fdab26f944d
Position 15771 address 0x7fdab26f98ae
Position 15899 address 0x55cbe2a8250e
Position 15899 address 0x55cbe2a826bb
Position 15899 address 0x55cbe2a8930f
Position 15,779
1
2
3
4
5
6

# Cas d'utilisation

SYNACKTIV

- **Identifier le code lié à un log particulier**
  - Rechercher la construction du message dans la mémoire
  - Puis remonter le callgraph
- **Suivre l'évolution d'une allocation**
  - Breakpoint en R/W sur la zone mémoire
  - Ctrl-scroll
- **Time Travel Debugger**

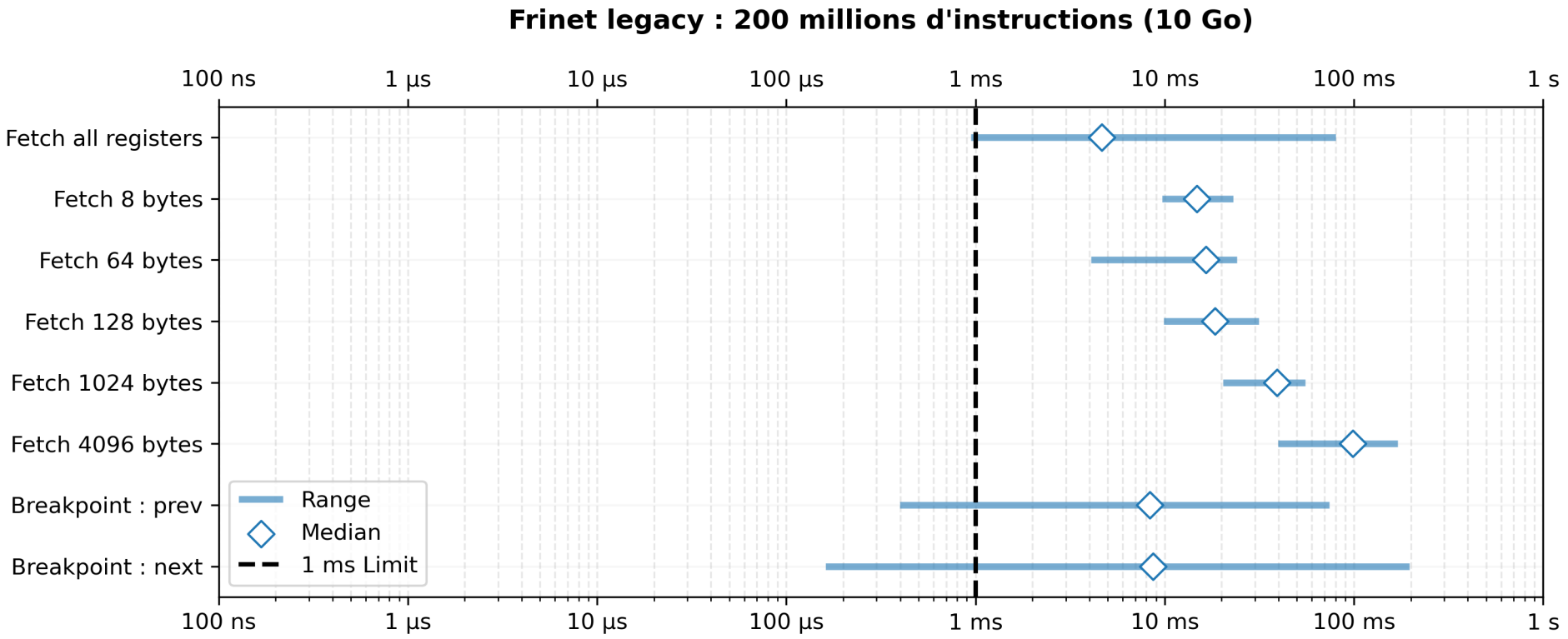
# Performance

SYNACKTIV

- **Un obstacle persiste …**
- **200 millions d'instructions (10 Go)**
  - Chargement initial : 23 minutes / 20 Go de RAM
  - Fluidité de l'interface dégradée
- **Besoin communs des équipes Synacktiv**
  - Entre 50 et 500 millions d'instructions

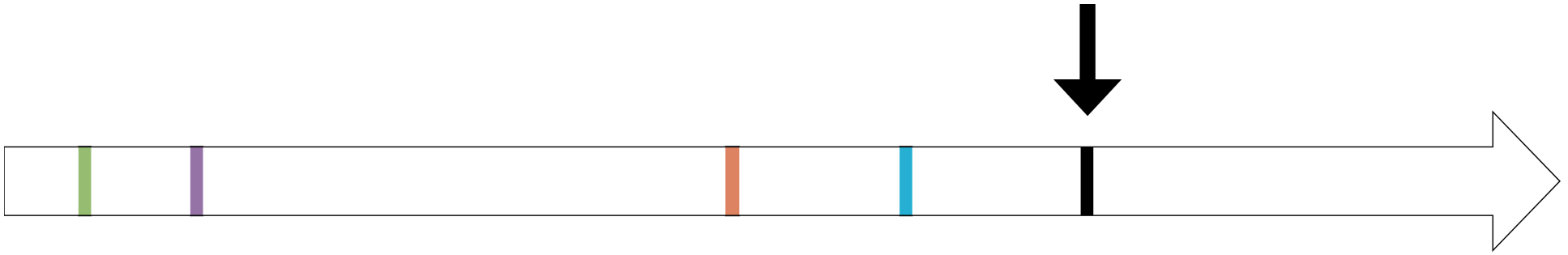
# Benchmark : Frinet Legacy

SYNACKTIV



# Informations dispersées

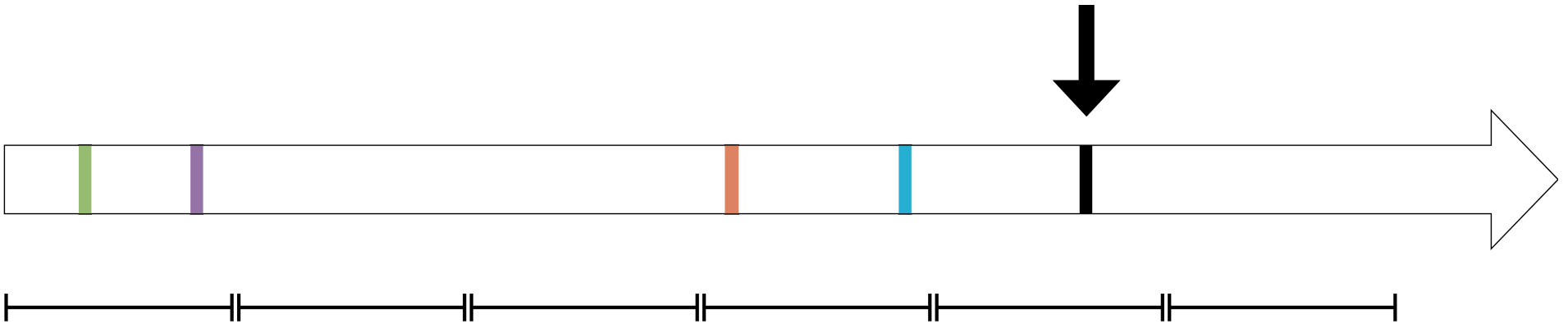
SYNACKTIV



```
00007FFCB2C30B50  E8 03 00 00 00 00 00 00 00 97 9D FC C9 80 16 A8  ................
00007FFCB2C30B60  80 0B C3 B2 FC 7F 00 00 60 71 82 BF 51 56 00 00  ........`q..QV..
00007FFCB2C30B70  B9 B9 82 BF 51 56 00 00 68 51 44 AB 51 56 00 00  ....QV..hQD.QV..
00007FFCB2C30B80  00 00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00  ................
00007FFCB2C30B90  00 00 00 00 00 00 00 00 58 08 46 AB 51 56 00 00  ........X.F.QV..
00007FFCB2C30BA0  68 51 44 AB 51 56 00 00 67 0C C3 B2 FC 7F 00 00  hQD.QV..g.......
00007FFCB2C30BB0  F0 CA 80 BF 51 56 00 00 B0 0D C3 B2 FC 7F 00 00  ....QV..........
00007FFCB2C30BC0  02 00 00 00 01 00 00 00 58 0F C3 00 00 7F 00 00  ........X.......
00007FFCB2C30BD0  F0 CA 80 BF 51 56 00 00 B0 0D C3 B2 FC 7F 00 00  ....QV..........
```

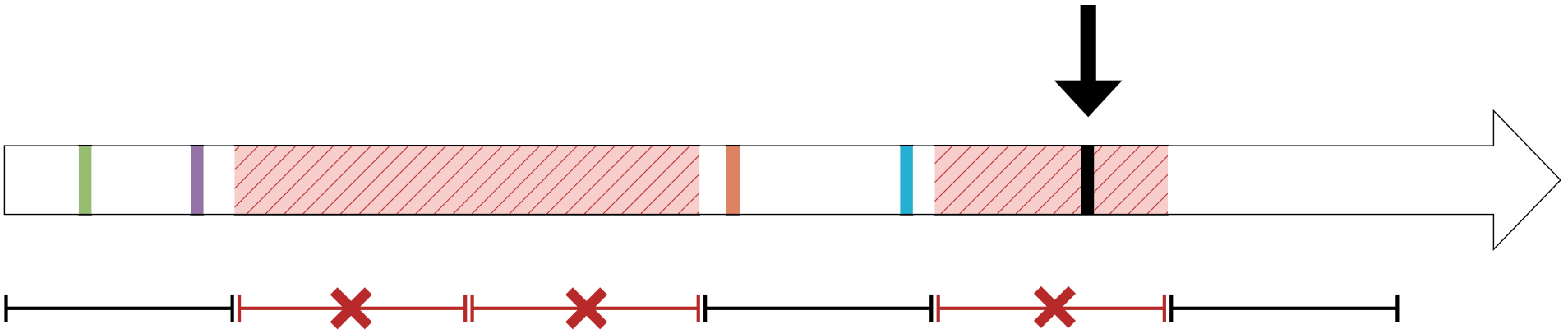
# Pré-calculs par segments

SYNACKTIV



# Pré-calculs par segments

SYNACKTIV



# Réécriture du backend

SYNACKTIV

- **3 Composants**
  - Indexeur (Rust) : CLI externe
  - Backend (Rust) : algorithmes de recherches
  - Frontend (Python) : plugin IDA

- **Objectif cible : minimiser la latence**
  - Idéalement ~20ms par frame / ~1ms par requête

# Index : liste au père noël

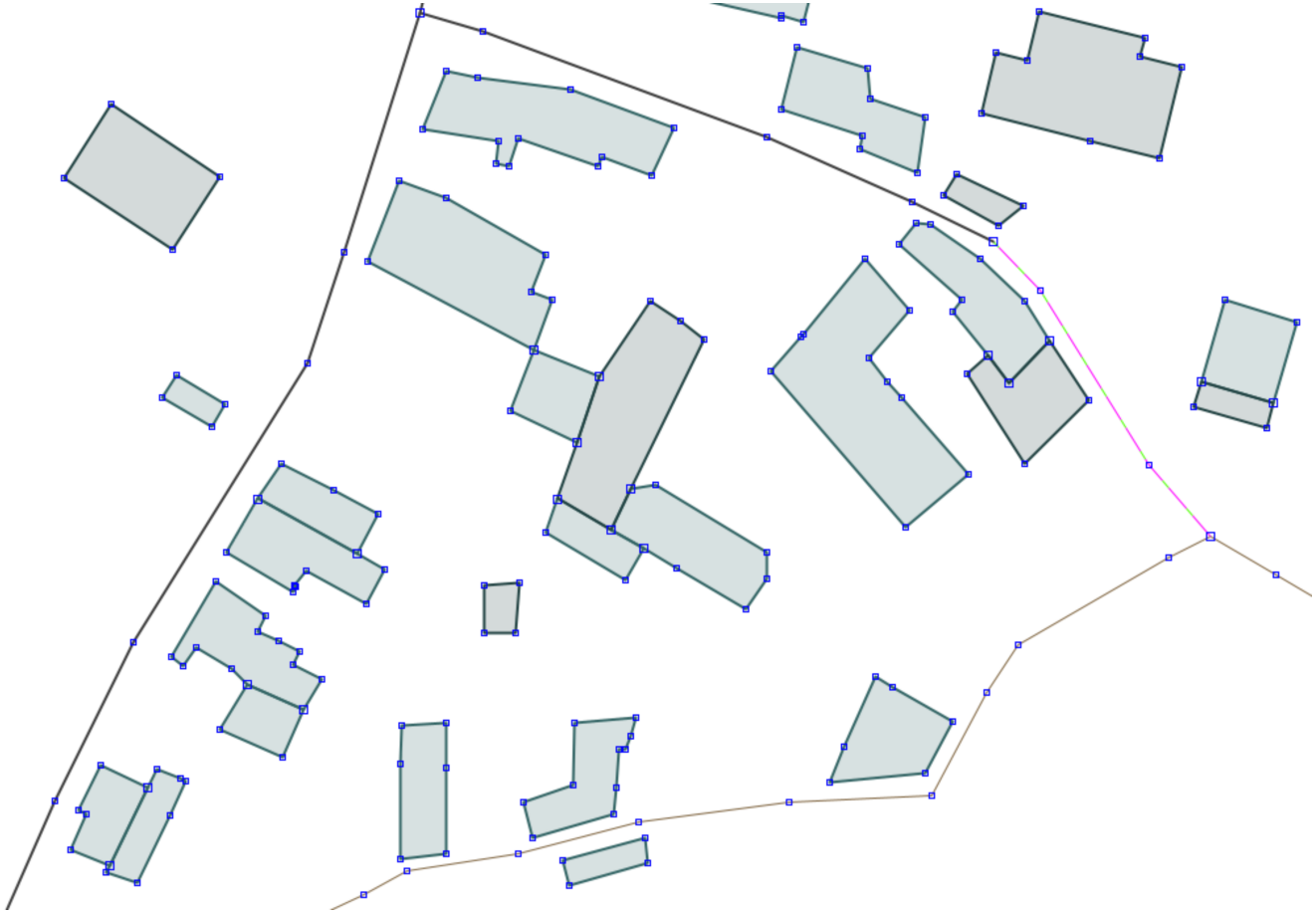
SYNACKTIV

- **Multi-dimensionnel**
  - Mémoire : Temps + Adresse
  - Registre : Temps + Valeur
- **Construction sur un PC portable lambda**
  - Temps & RAM
- **Stockage**
  - Fichier auto-porteur
  - Chargement instantané (zero-copy)

# OpenStreetMap

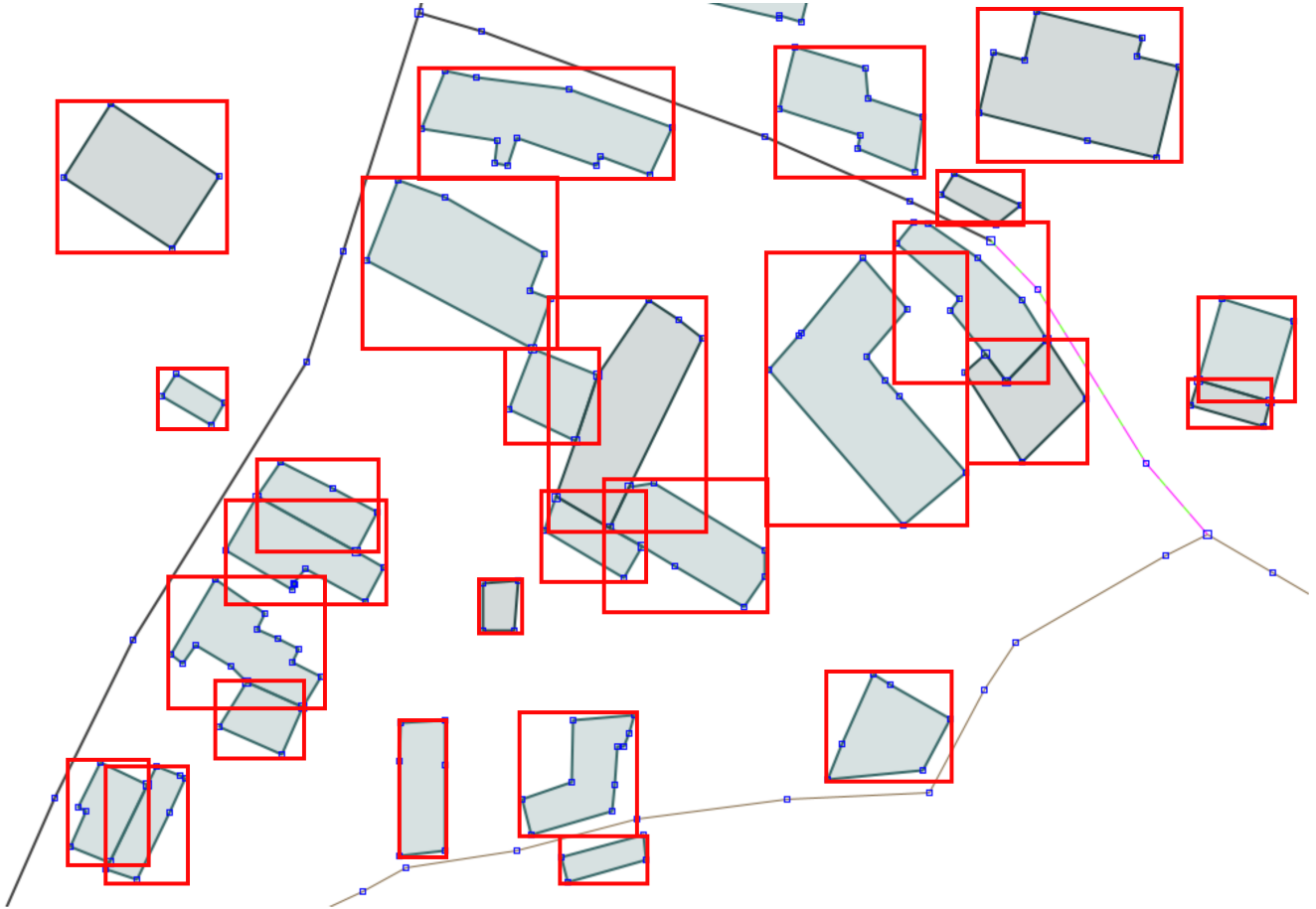
# OpenStreetMap

SYNACKTIV



# OpenStreetMap

SYNACKTIV

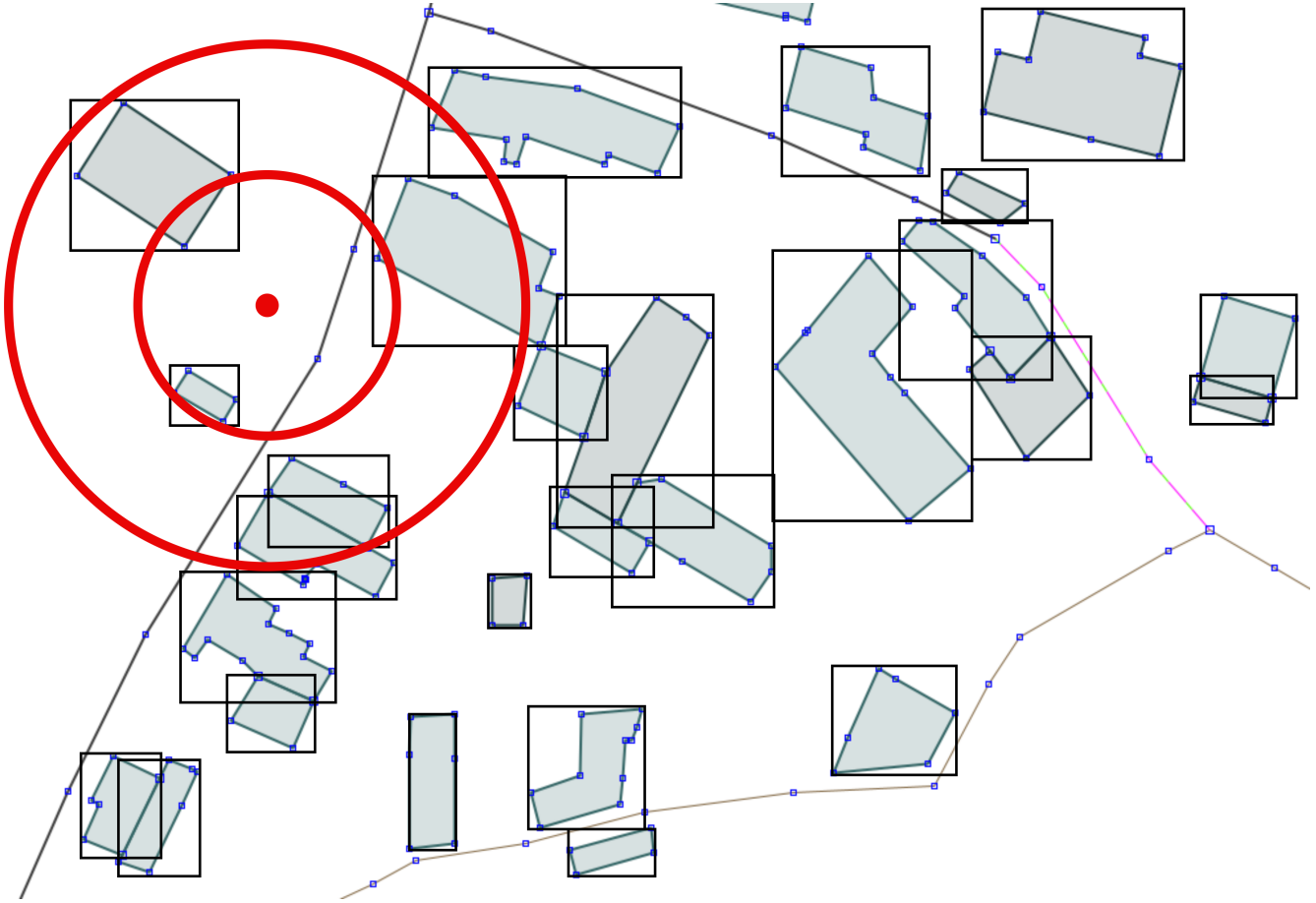


# OpenStreetMap : intersection

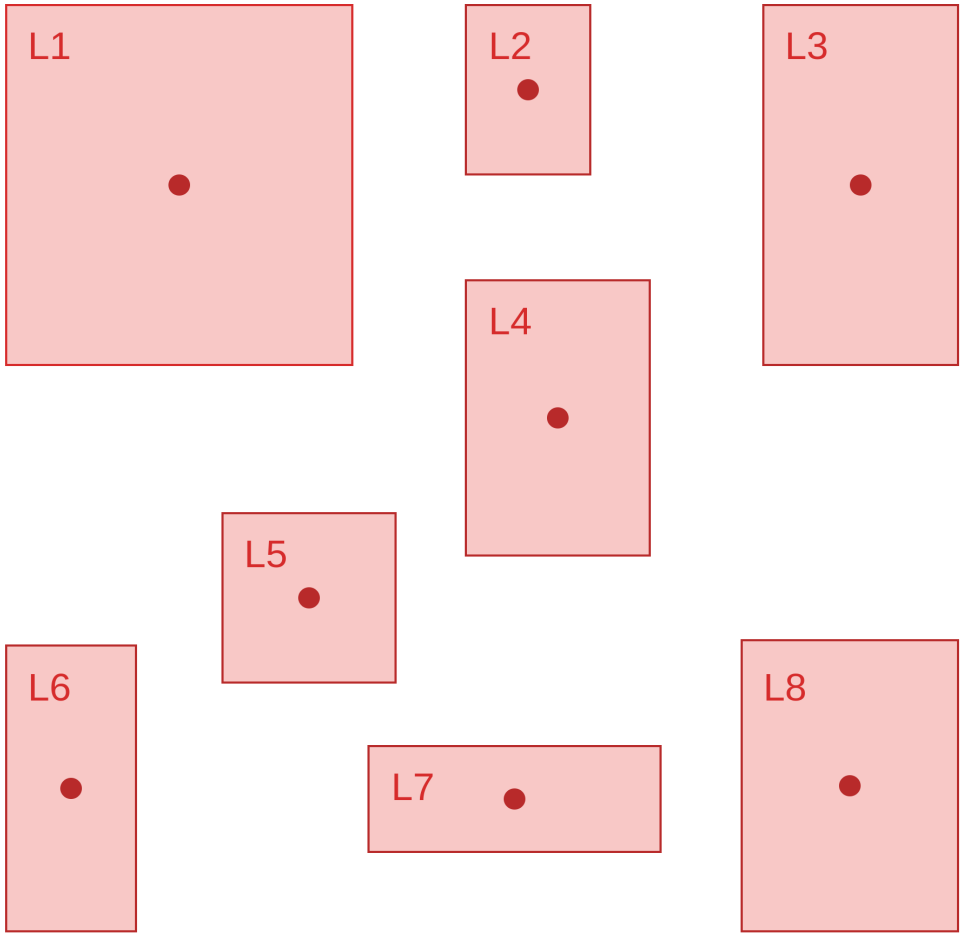
SYNACKTIV

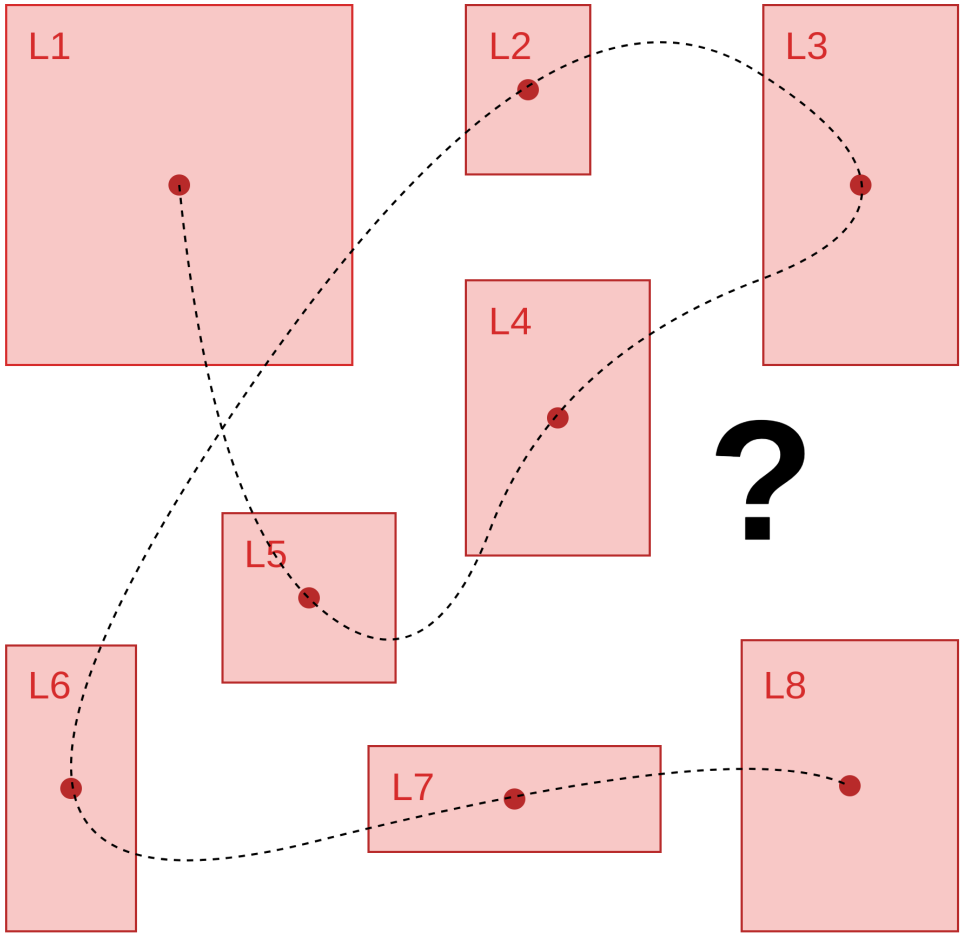
# OpenStreetMap : nearest neighbour

SYNACKTIV



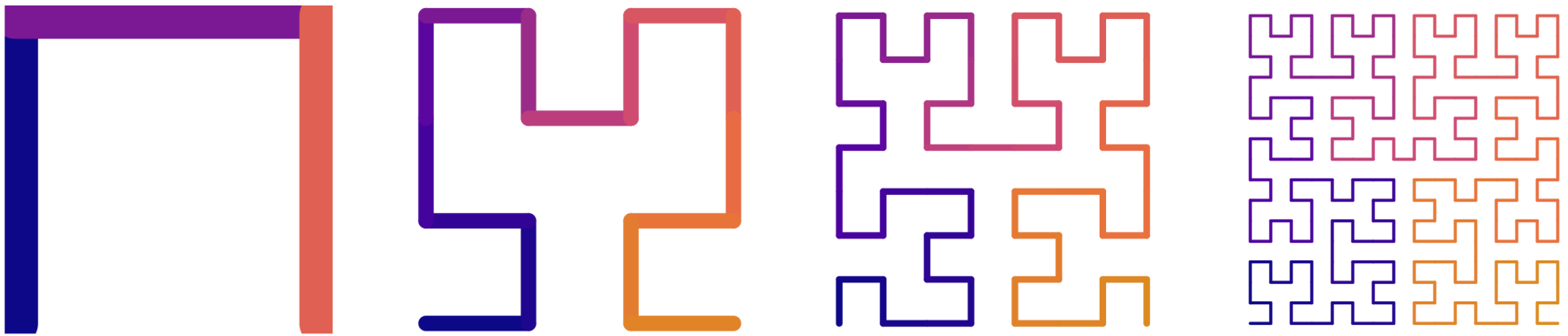
# Hilbert Packed R-Tree


L1
L2
L3
L4
L5
L6
L7
L8

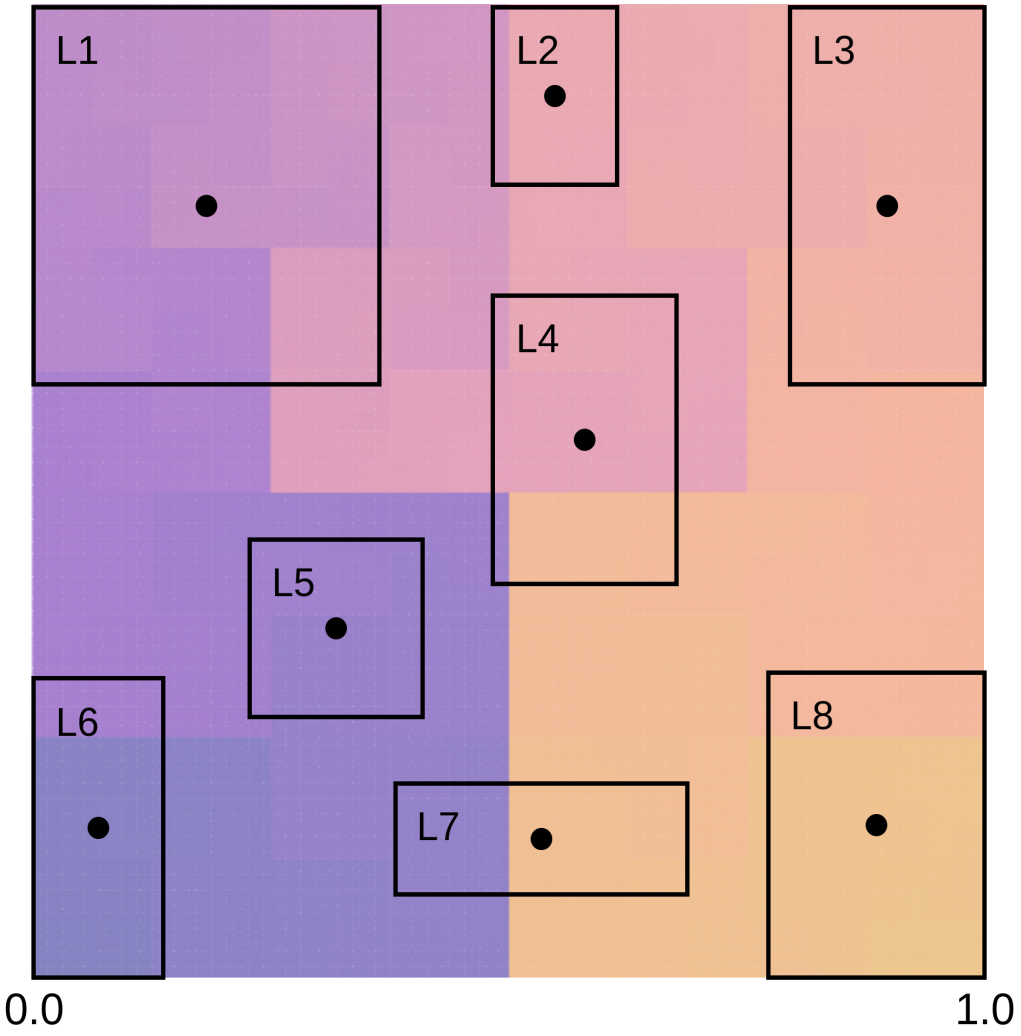

L1
L2
L3
L4
?
L5
L6
L7
L8

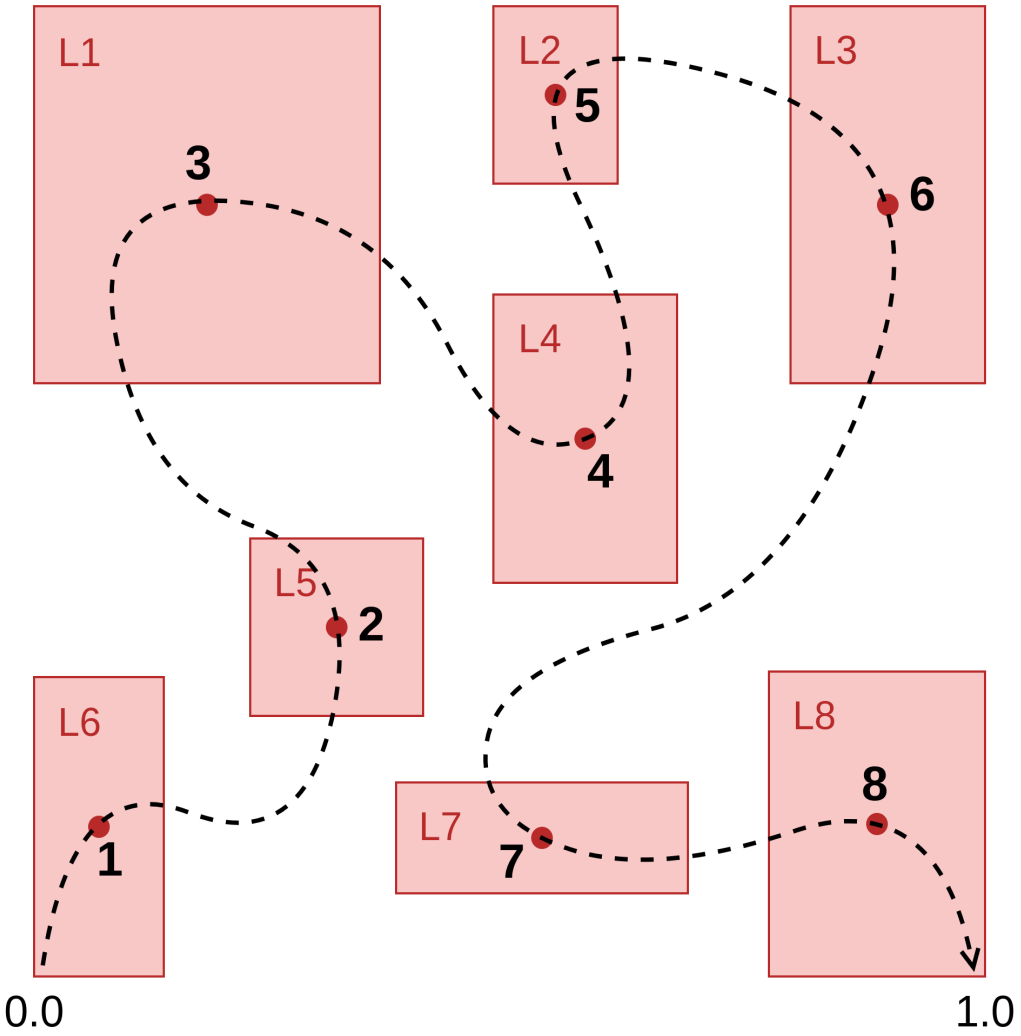
# Courbe de Hilbert

SYNACKTIV



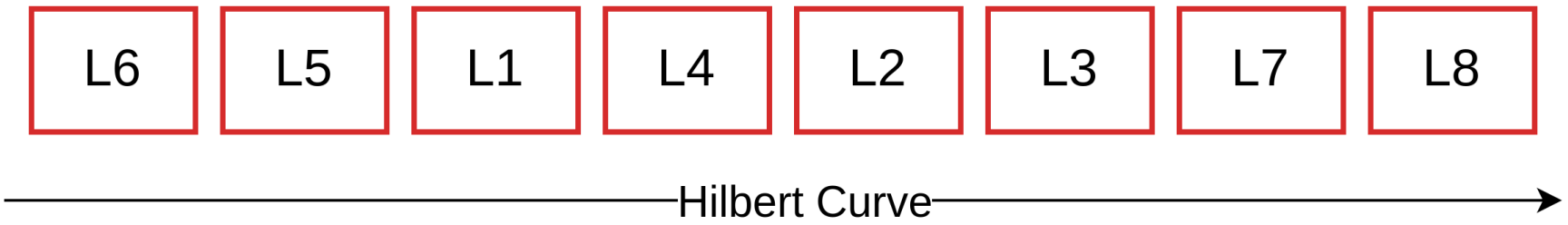
2x2 4x4 8x8 16x16


L1
L2
L3
L4
L5
L6
L7
L8
0.0
1.0


L1
3
L2
5
L3
6
L4
4
L5
2
L6
1
L7
7
L8
8
0.0
1.0

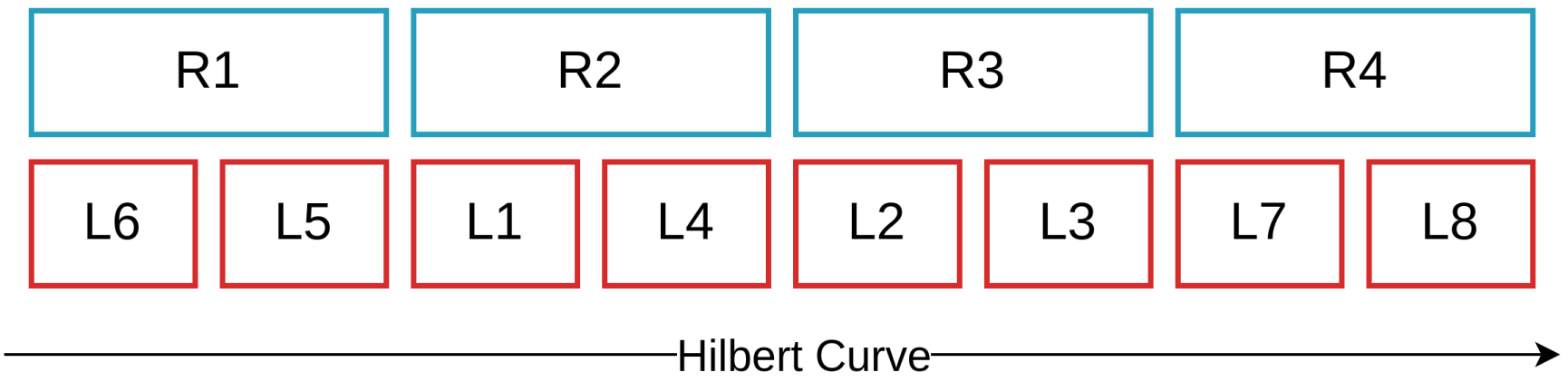
# Hilbert Packed RTree

SYNACKTIV



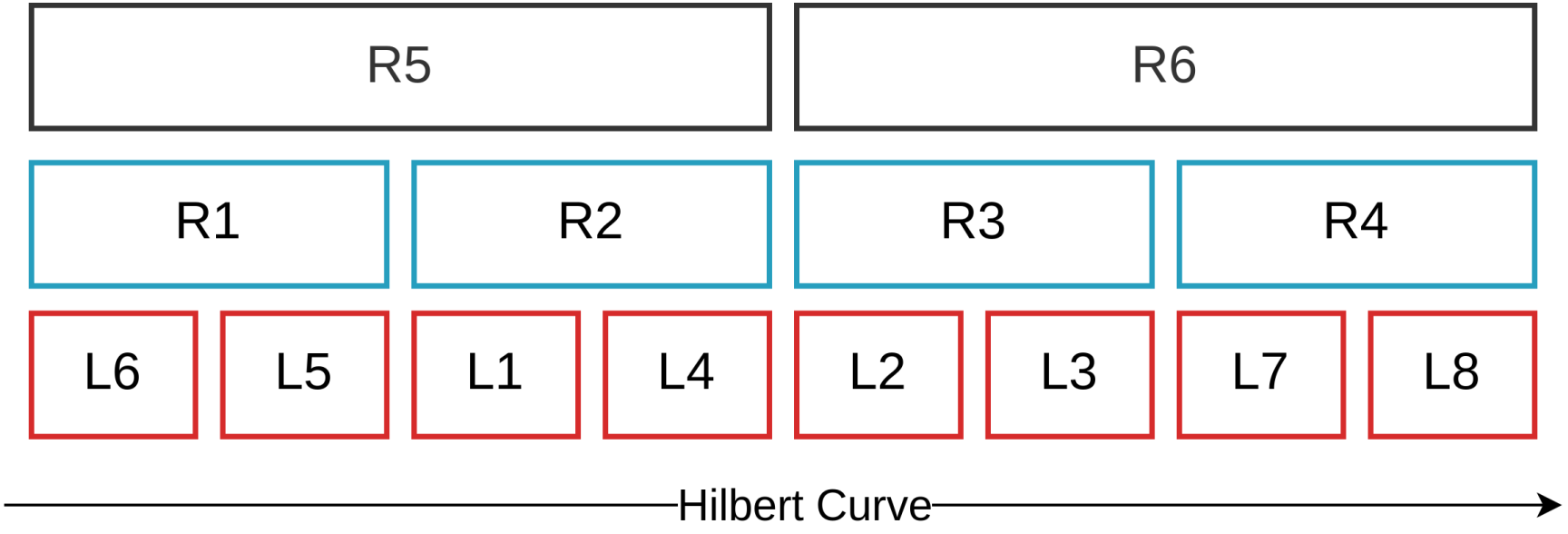
# Hilbert Packed RTree

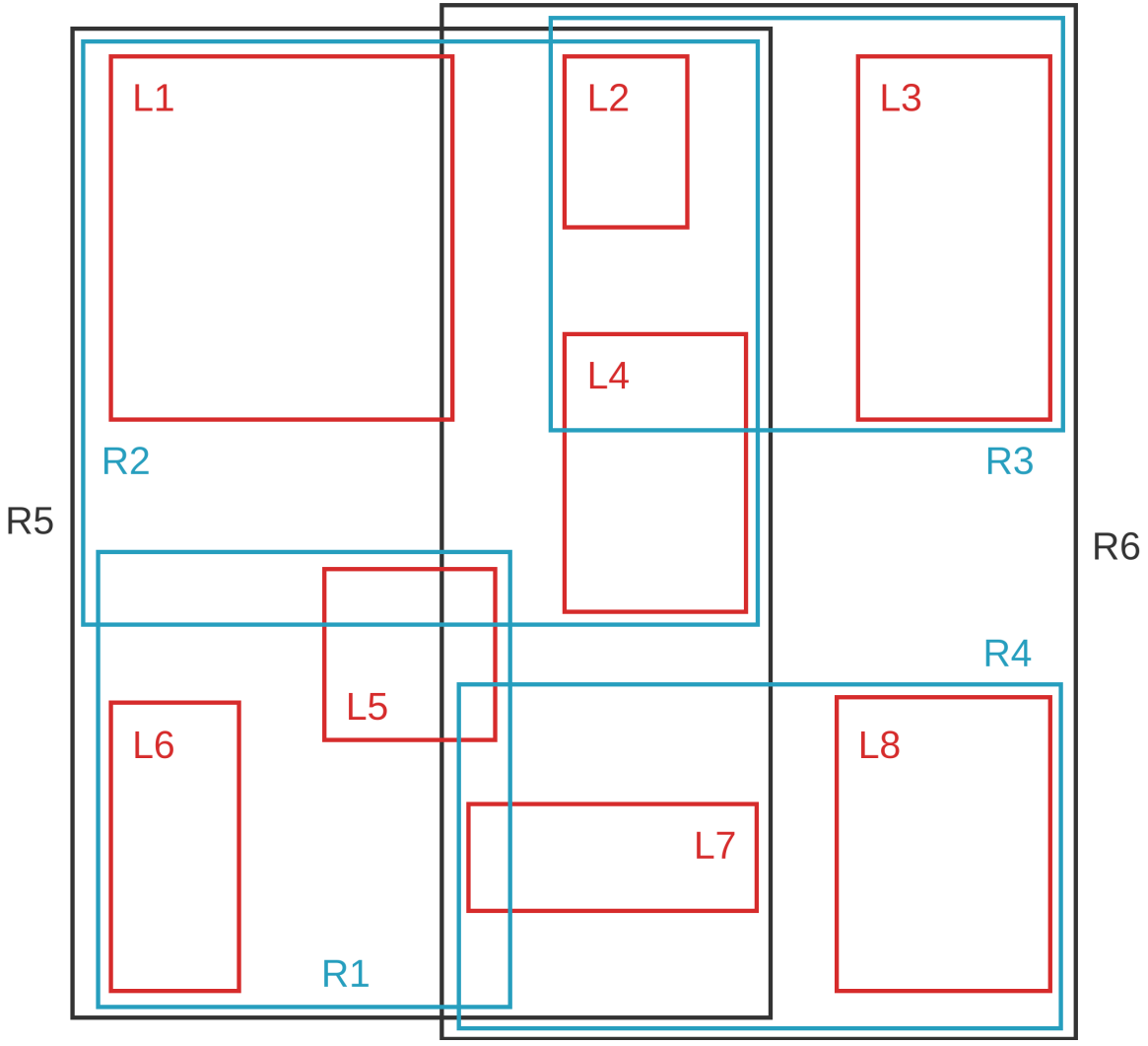
SYNACKTIV

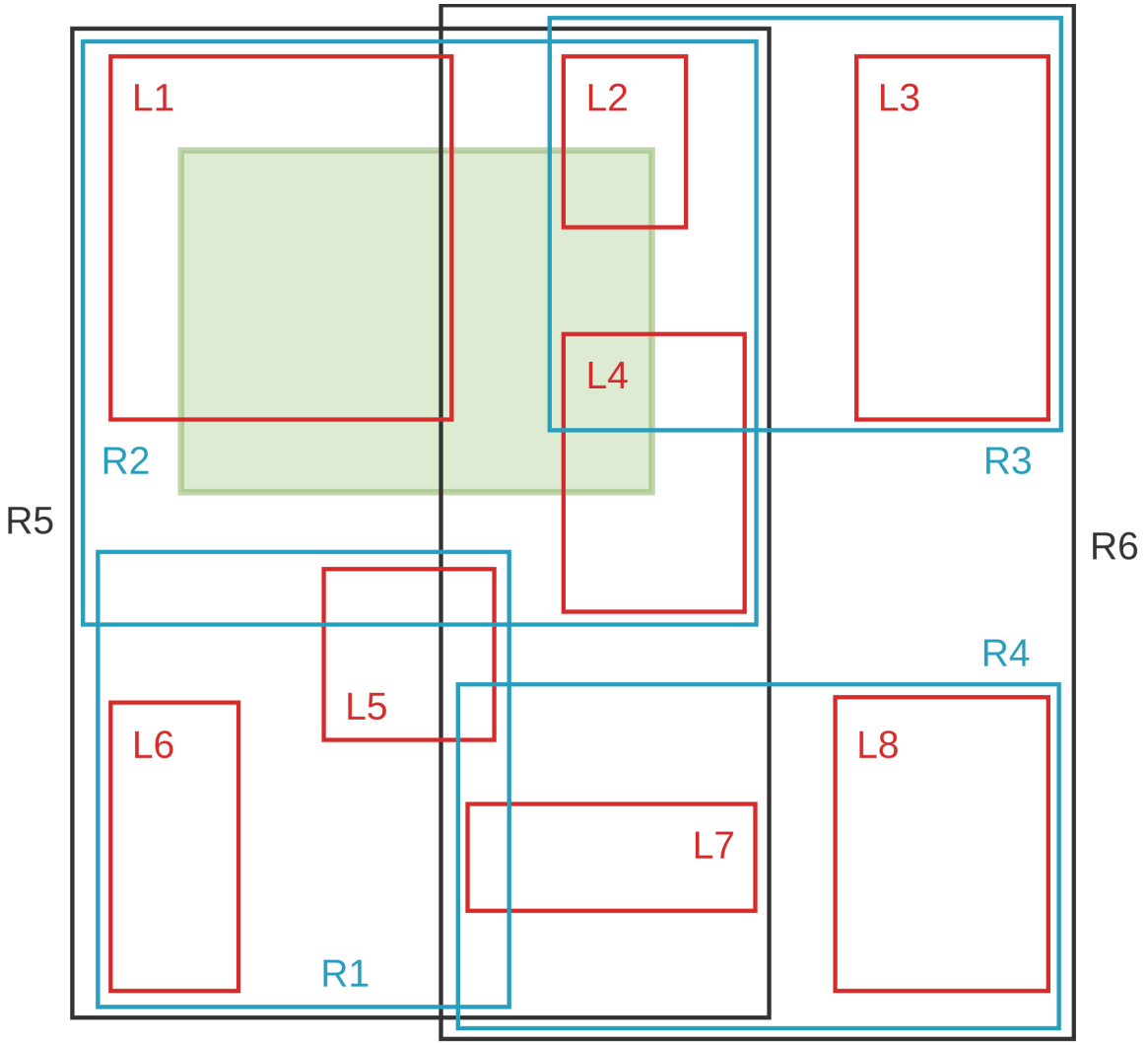


# Hilbert Packed RTree

SYNACKTIV



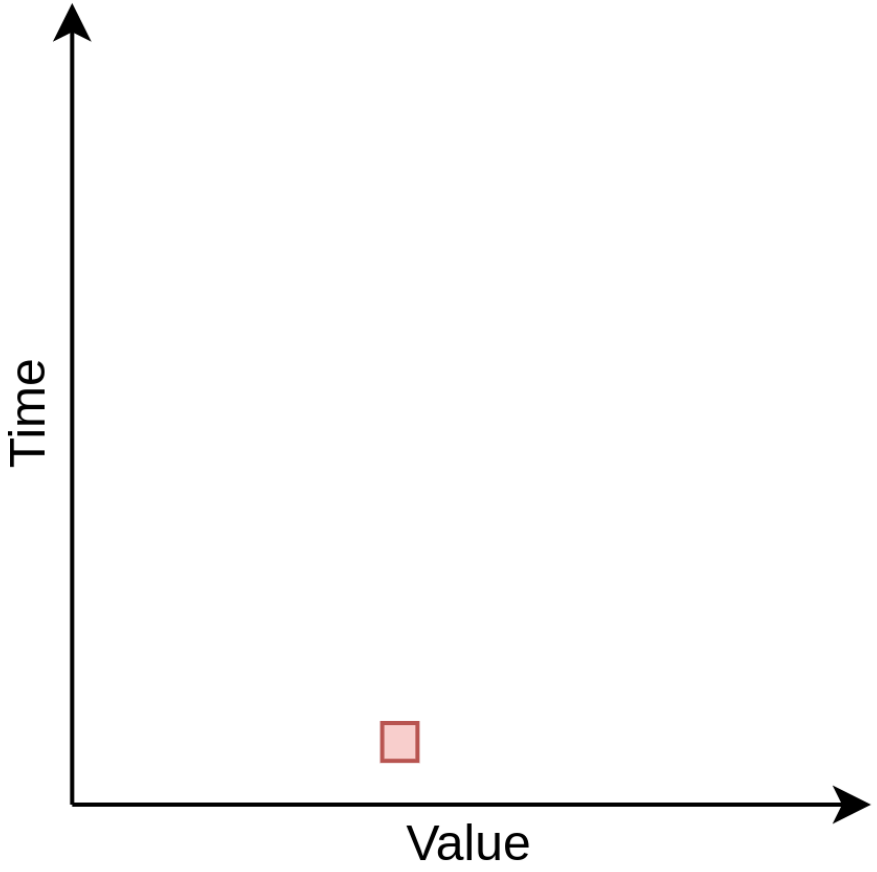

L1
L2
L3
L4
R2
R3
R5
R6
L5
R4
L6
L8
L7
R1


L1
L2
L3
L4
R2
R3
R5
R6
L5
R4
L6
L8
L7
R1

# Trace d'exécution → Rectangles

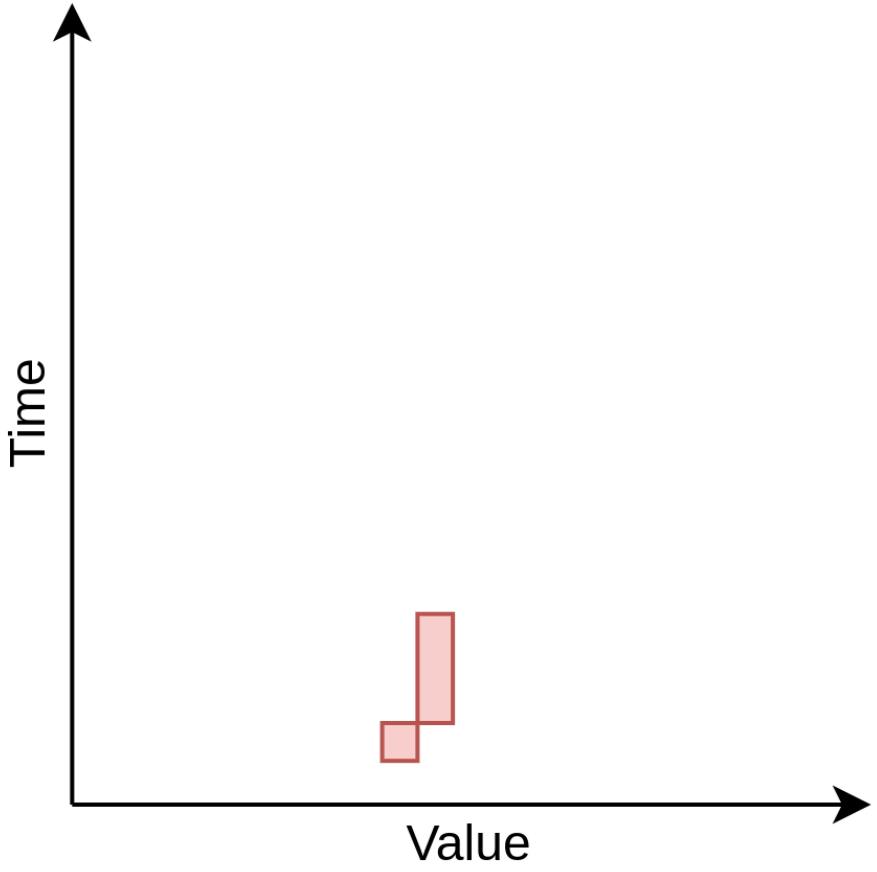
# Registres : RAX





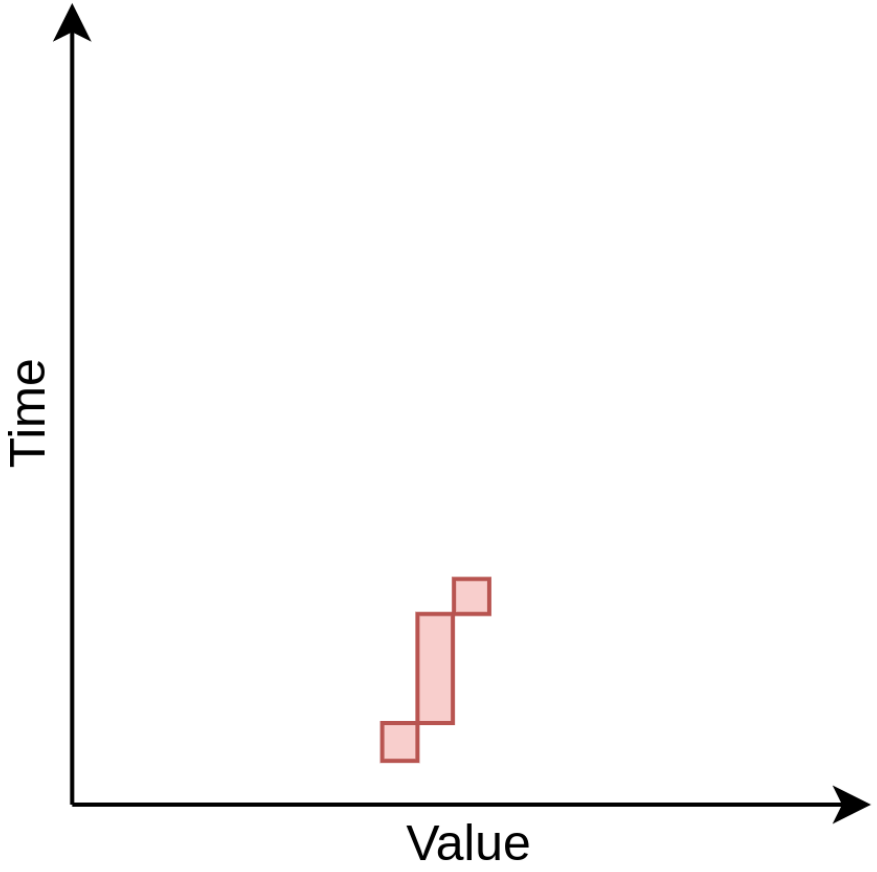
# Registres : RAX

SYNACKTIV



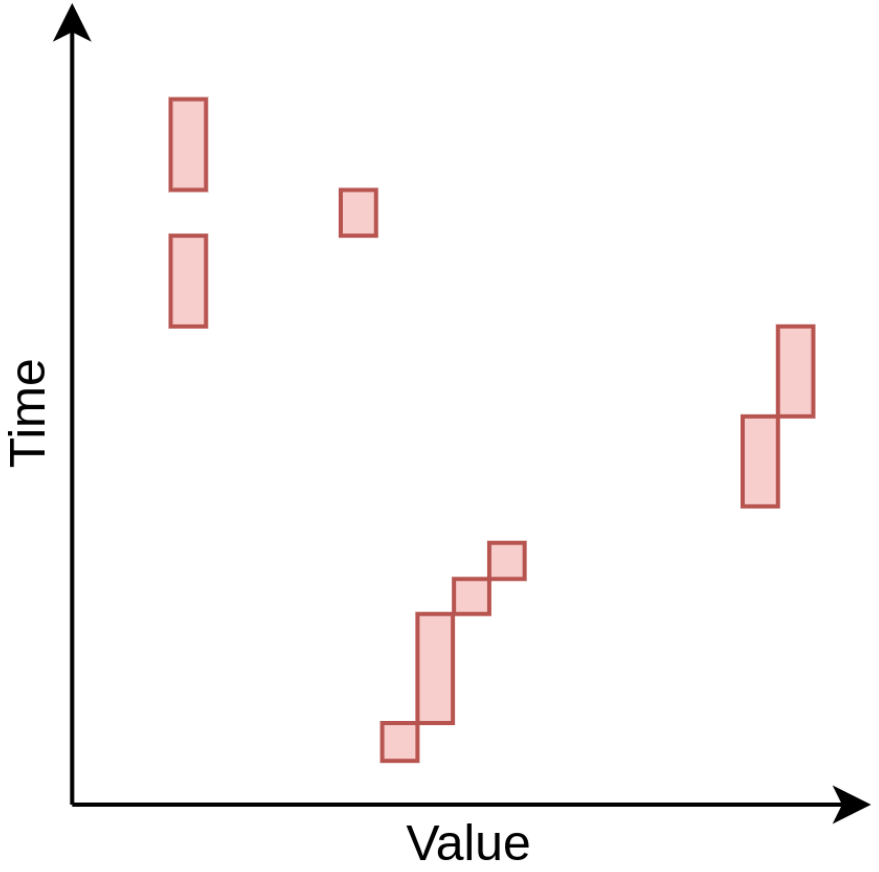
# Registres : RAX

SYNACKTIV



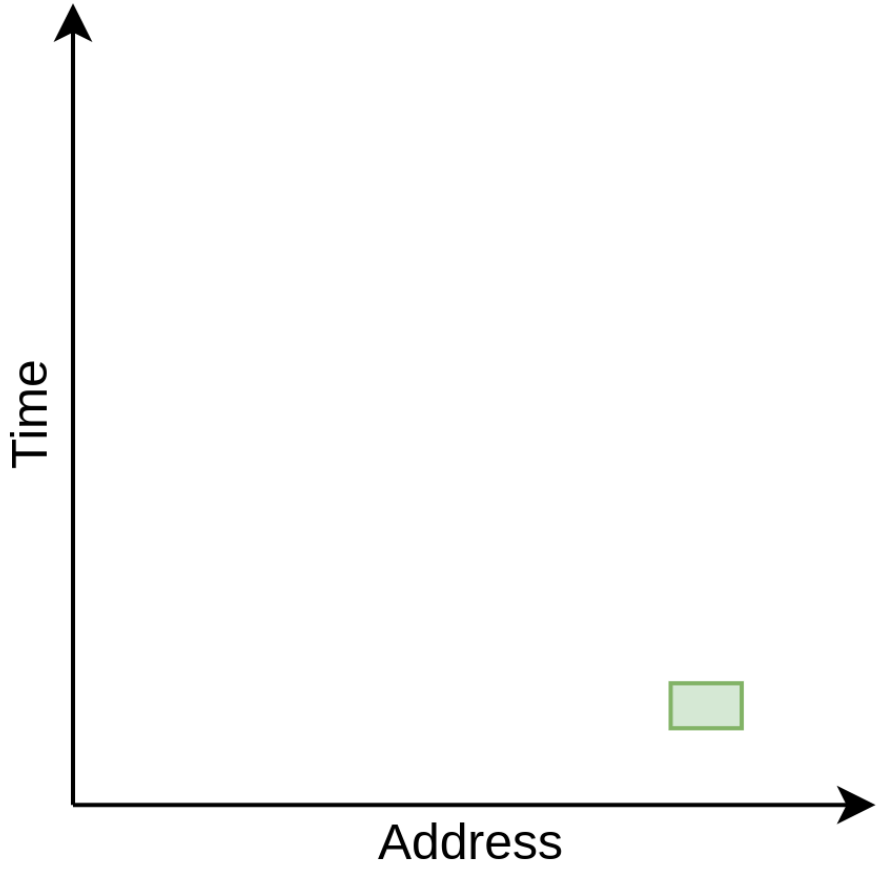
# Registres : RAX

SYNACKTIV



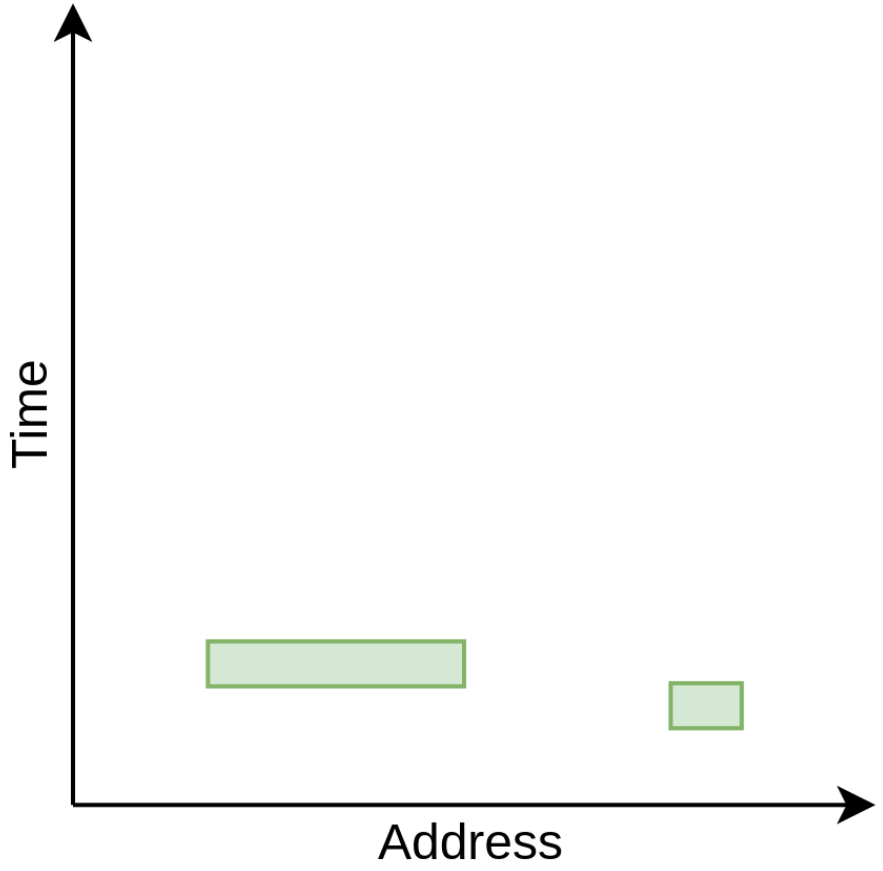
# Lectures mémoire

SYNACKTIV



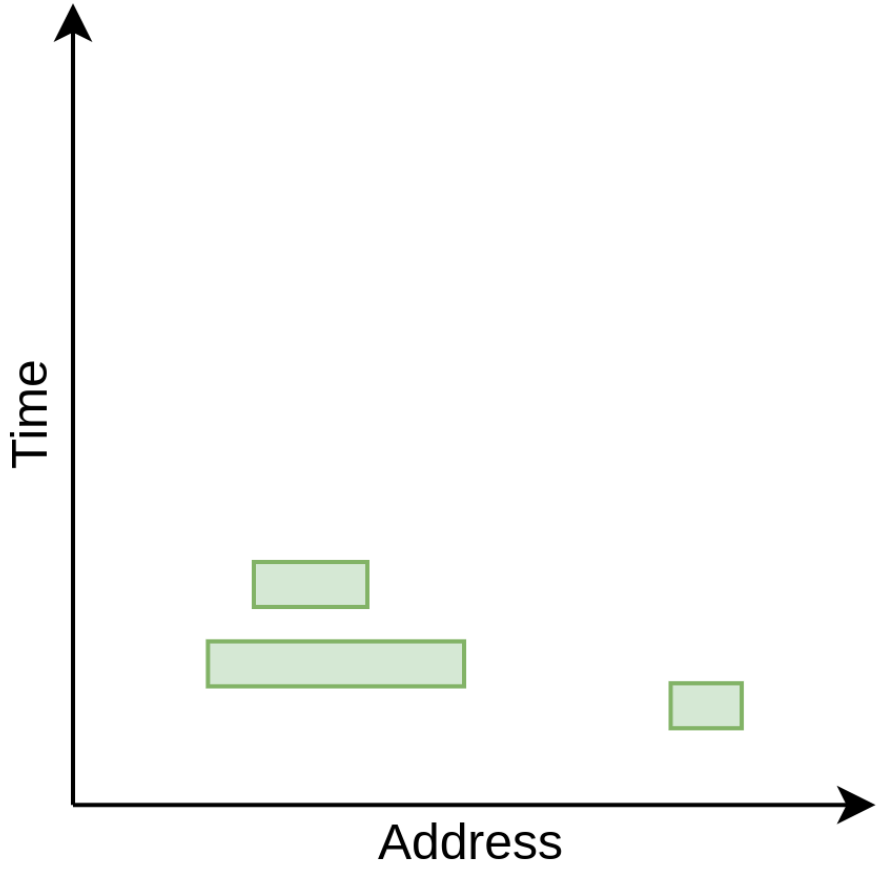
# Lectures mémoire

SYNACKTIV



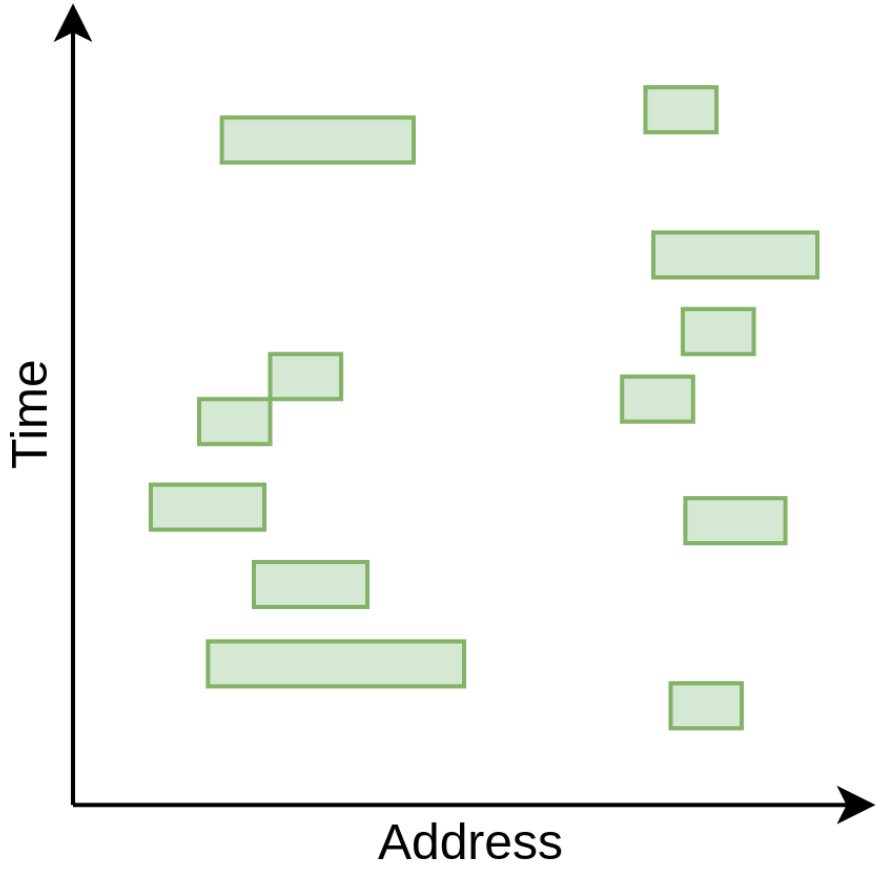
# Lectures mémoire

SYNACKTIV



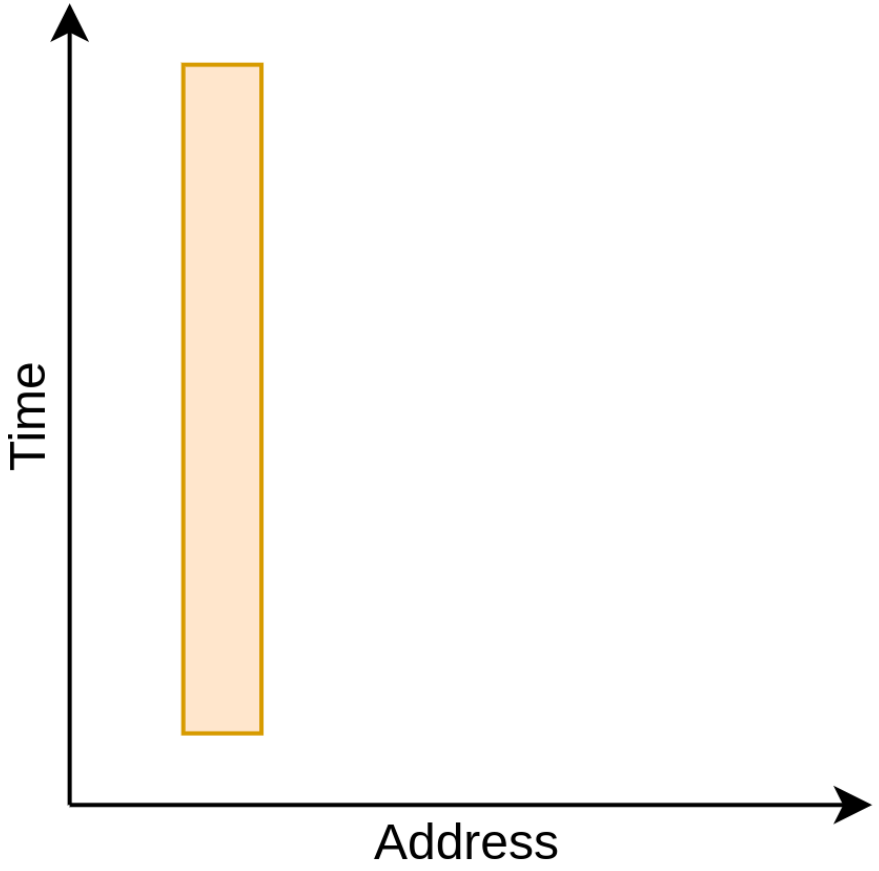
# Lectures mémoire

SYNACKTIV



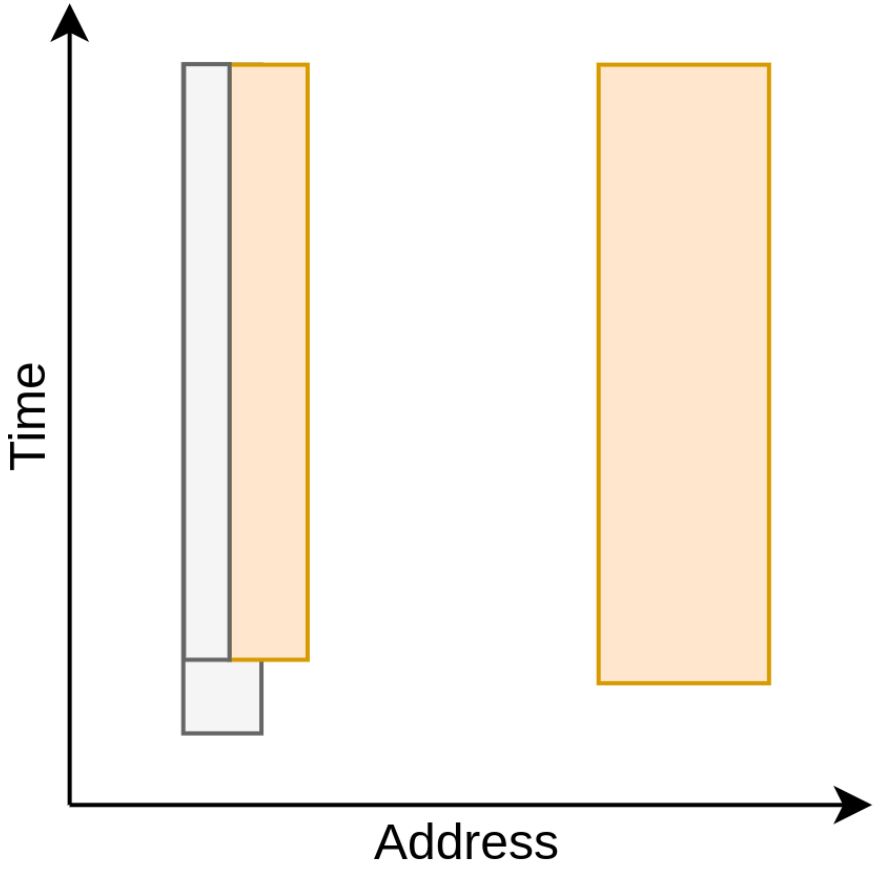
# Écritures mémoire

SYNACKTIV



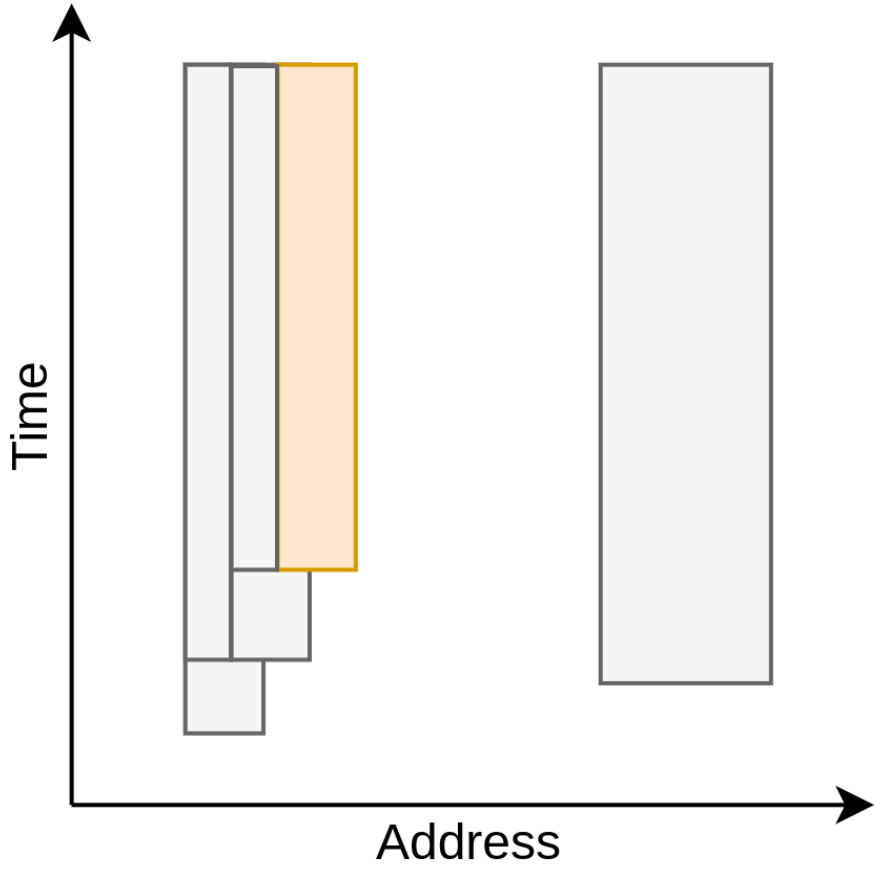
# Écritures mémoire

SYNACKTIV



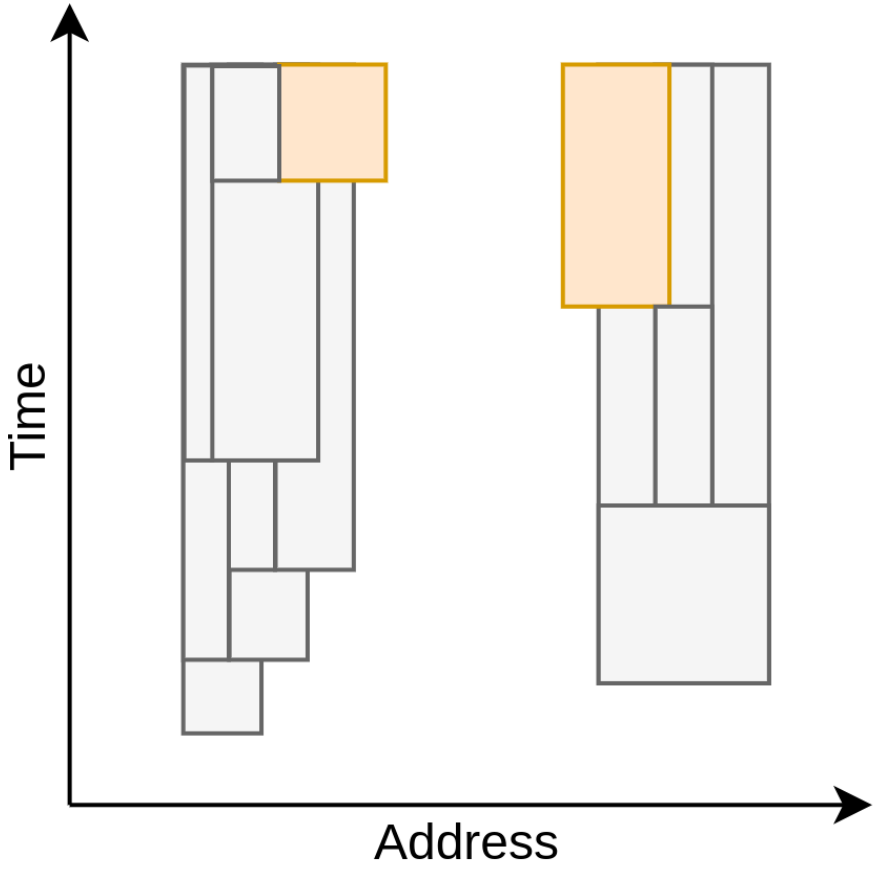
# Écritures mémoire

SYNACKTIV



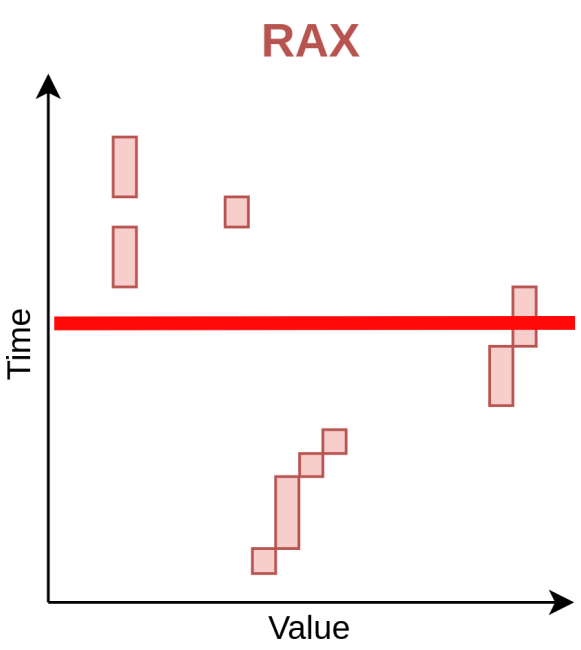
# Écritures mémoire

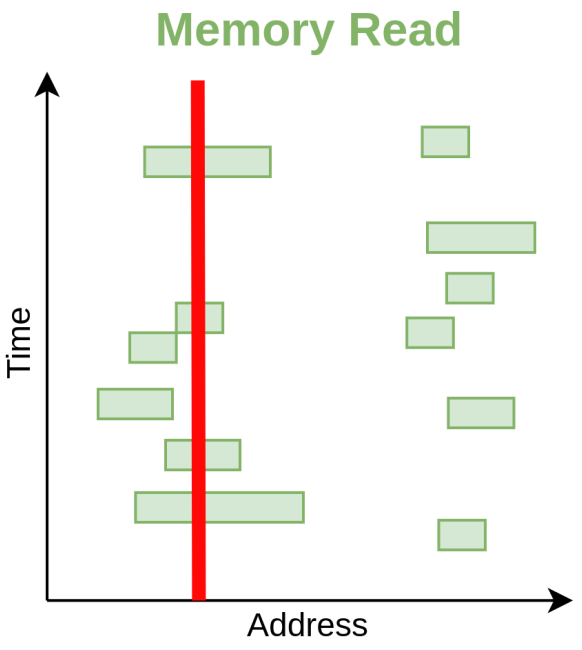
SYNACKTIV

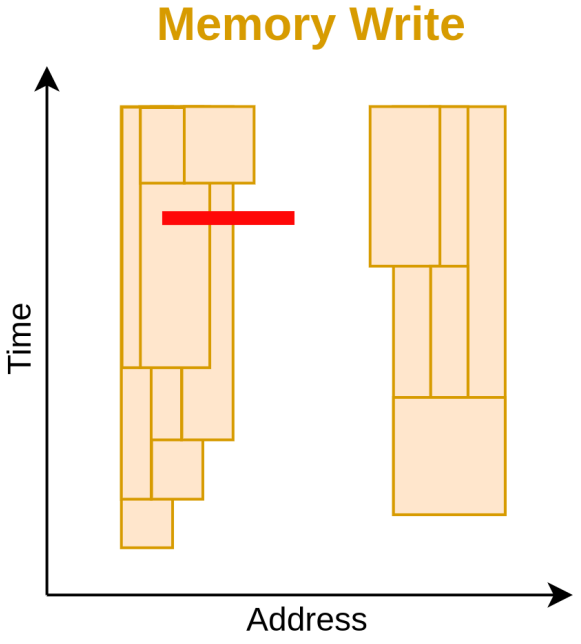


# Requêtes → Rectangles

SYNACKTIV



# Benchmark : comparaison

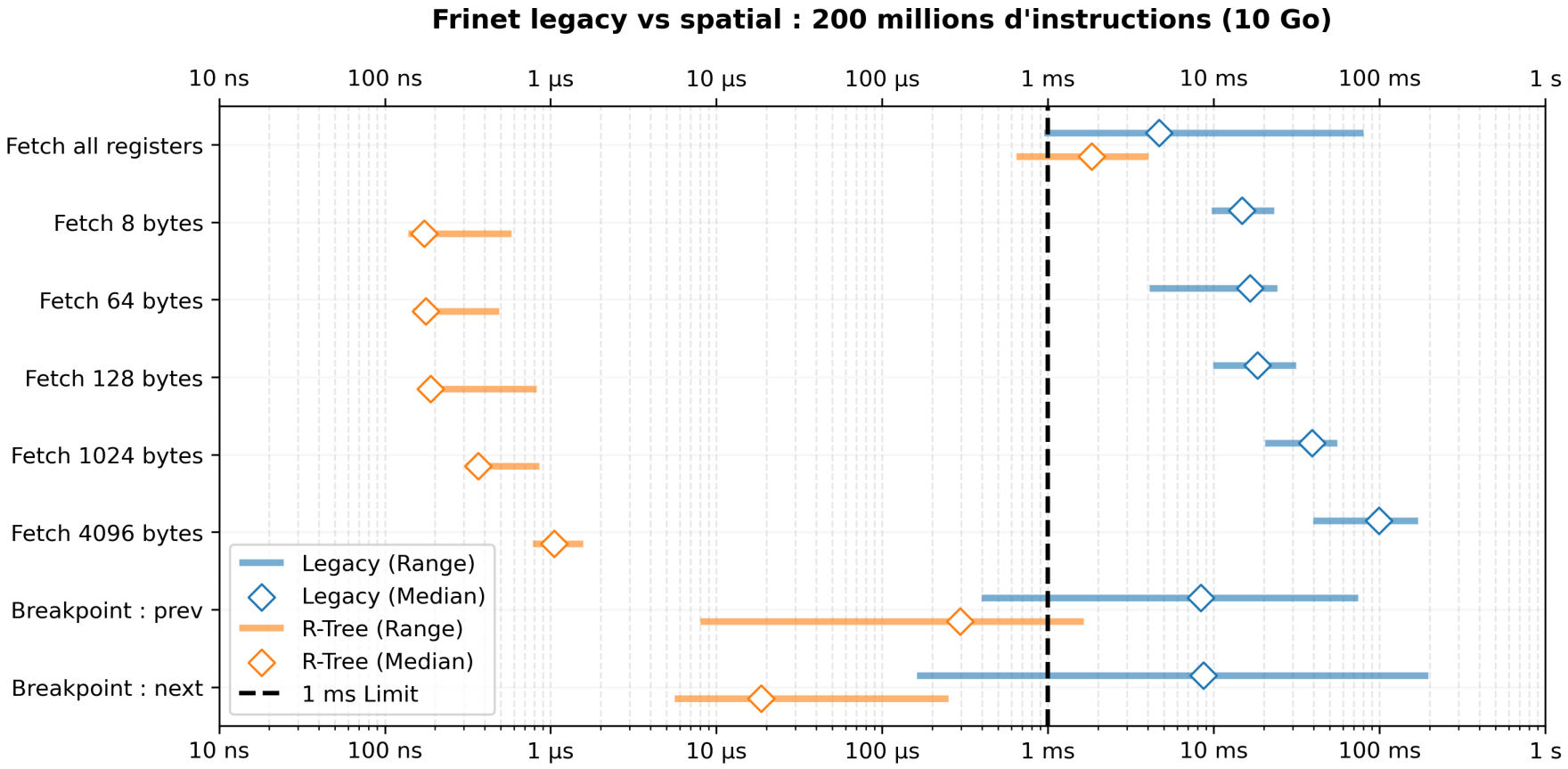
SYNACKTIV

```
$ frinet index --format tenet little_wget.tenet little_wget.db
     Scout [00:01:02] ######################################## 10.08 GiB/10.08 GiB
Partitions [00:01:01] ######################################## 10.08 GiB/10.08 GiB
      Fill [00:01:24] ######################################## 10.08 GiB/10.08 GiB
[INFO  frinet] Total time : 3 minutes
[INFO  frinet] Maximum memory usage : 2.89 GiB
```

- **Parsing & Indexing**
  - 23 min → 3 min
  - 20 Go → 3 Go

# Benchmark : comparaison

SYNACKTIV



# Passage à l'échelle

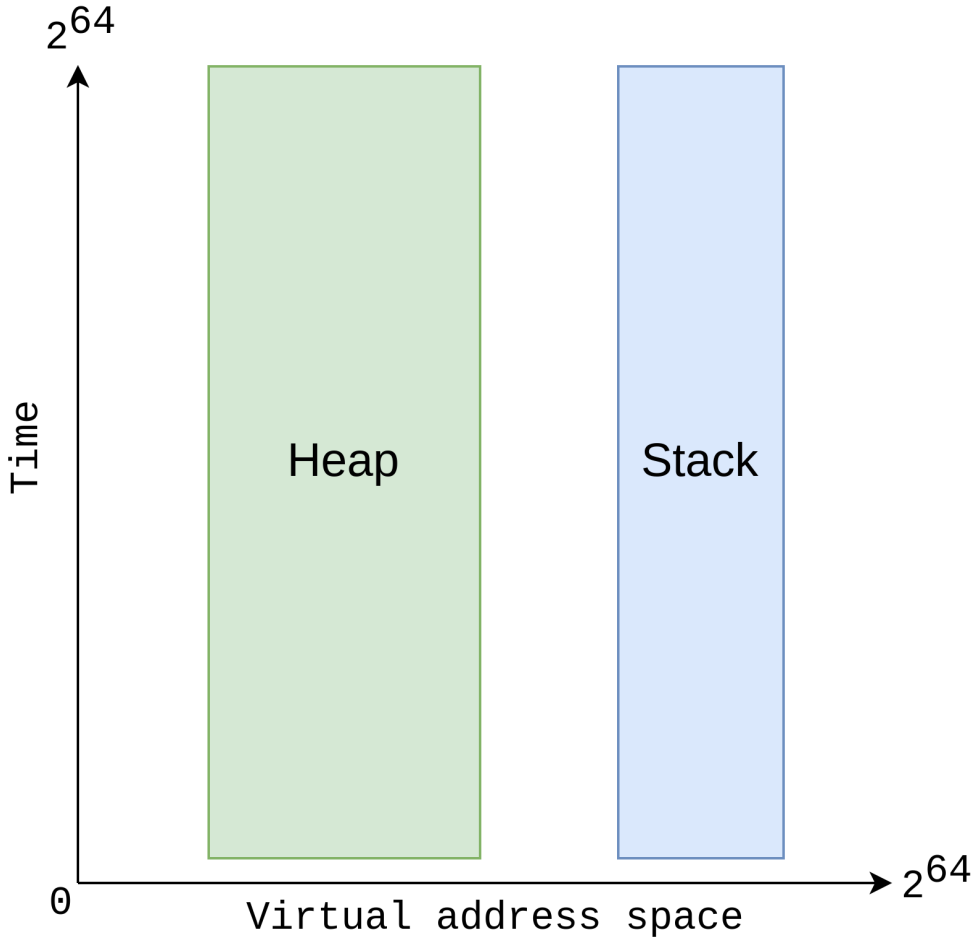
# Passage à l'échelle

SYNACKTIV

- **>500 millions d'instructions**
- **Tri en mémoire**
  - 32 octets par feuille
  - → Tri en 2 phases
- **Index inefficace**
  - Latence se dégrade très rapidement
  - → Densité des feuilles trop hétérogène
- **1 → 3 passes**

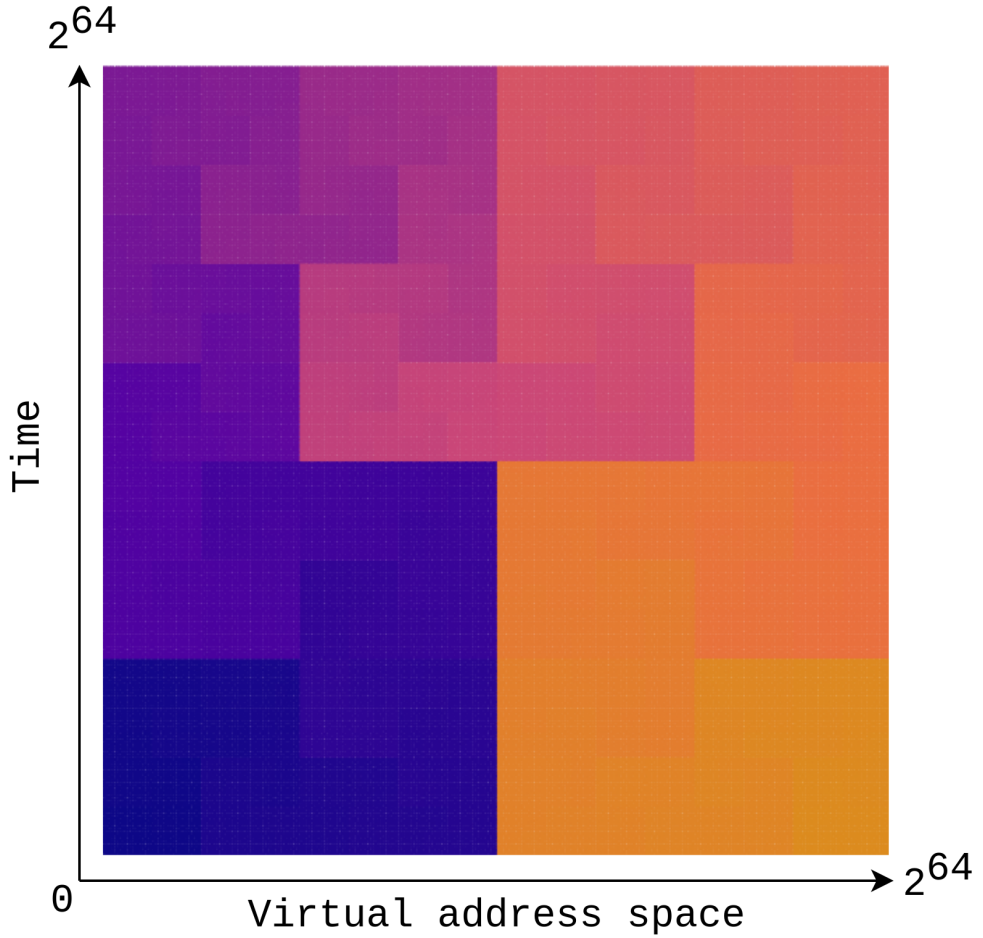
# 1ère passe : régions

SYNACKTIV



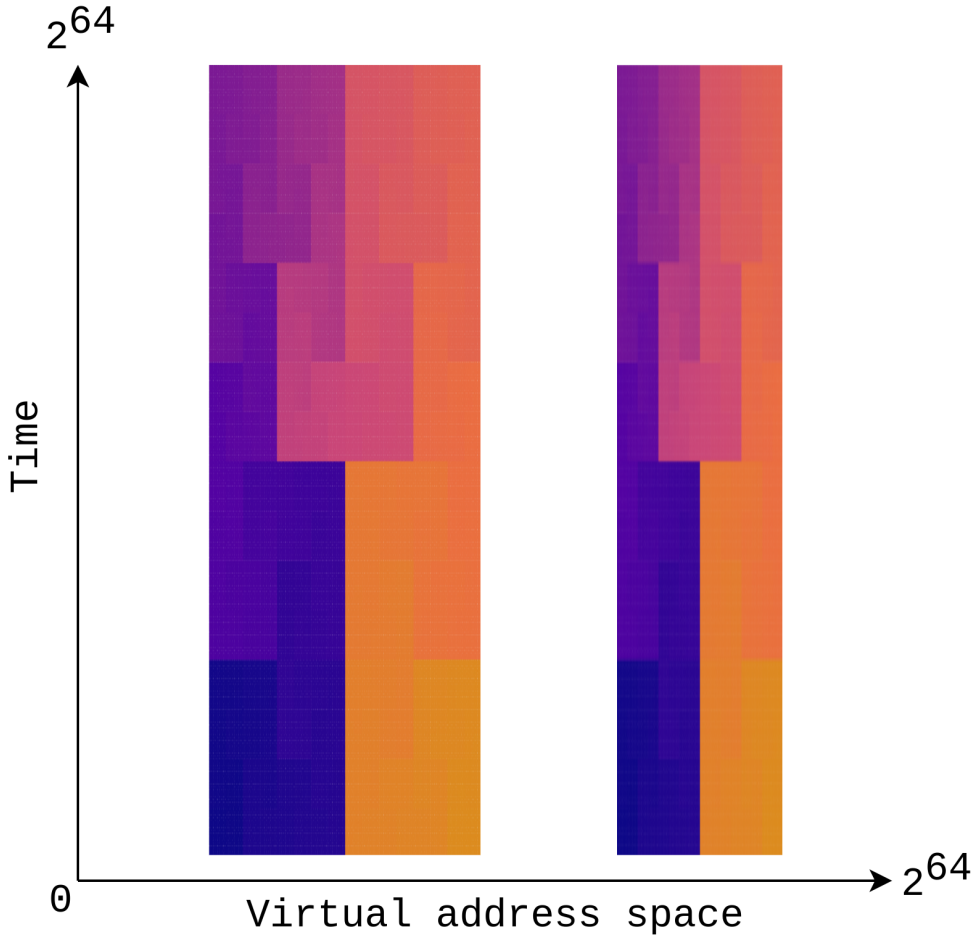
# 1ère passe : régions

SYNACKTIV



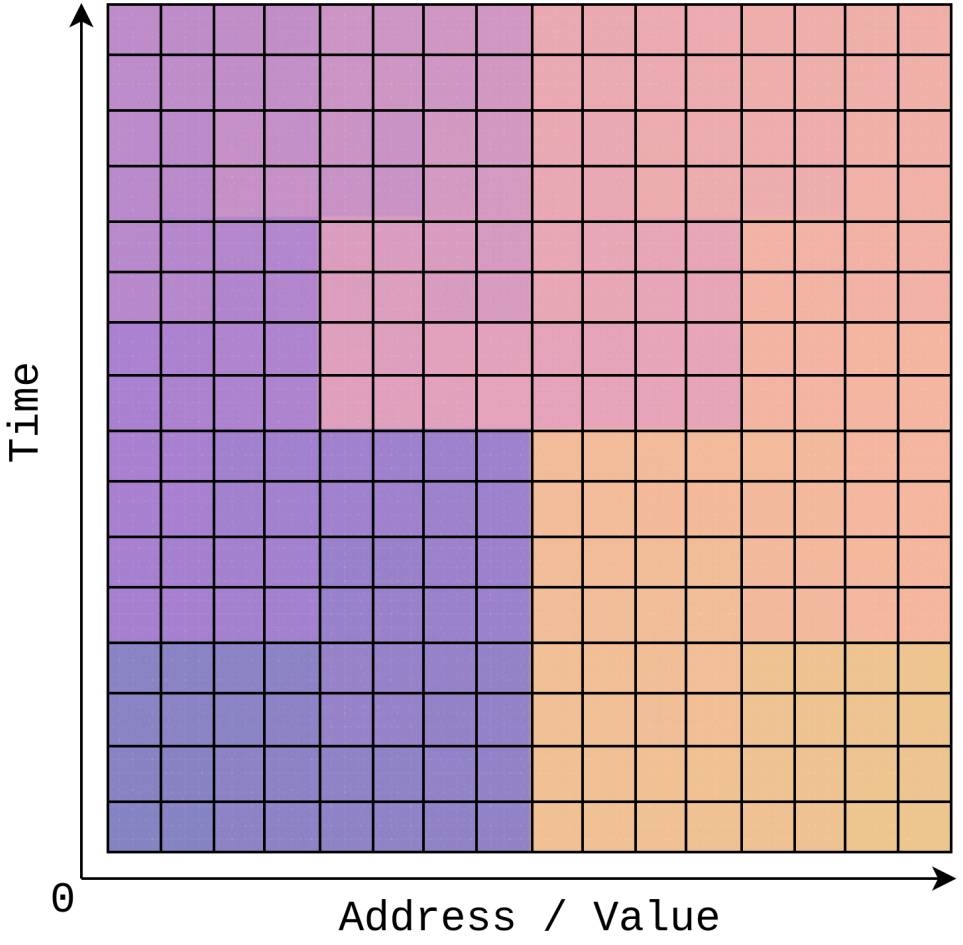
# 1ère passe : régions

SYNACKTIV



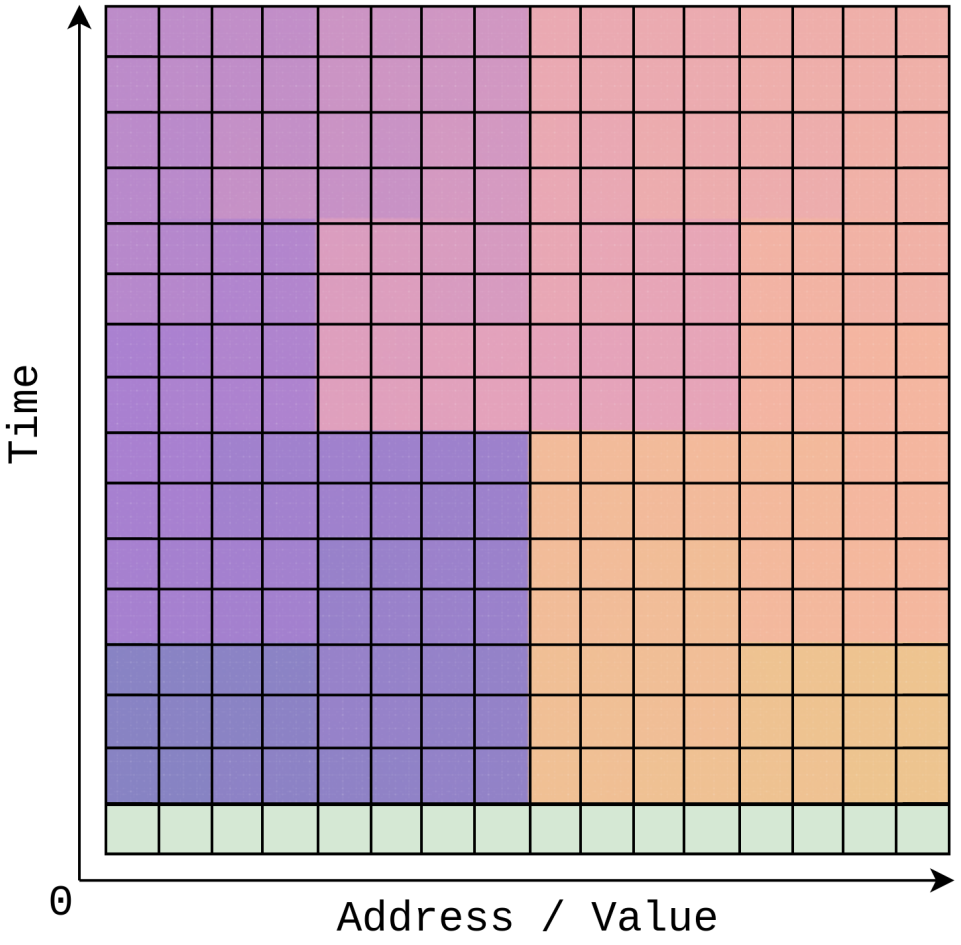
# 2ème passe : partitions

SYNACKTIV



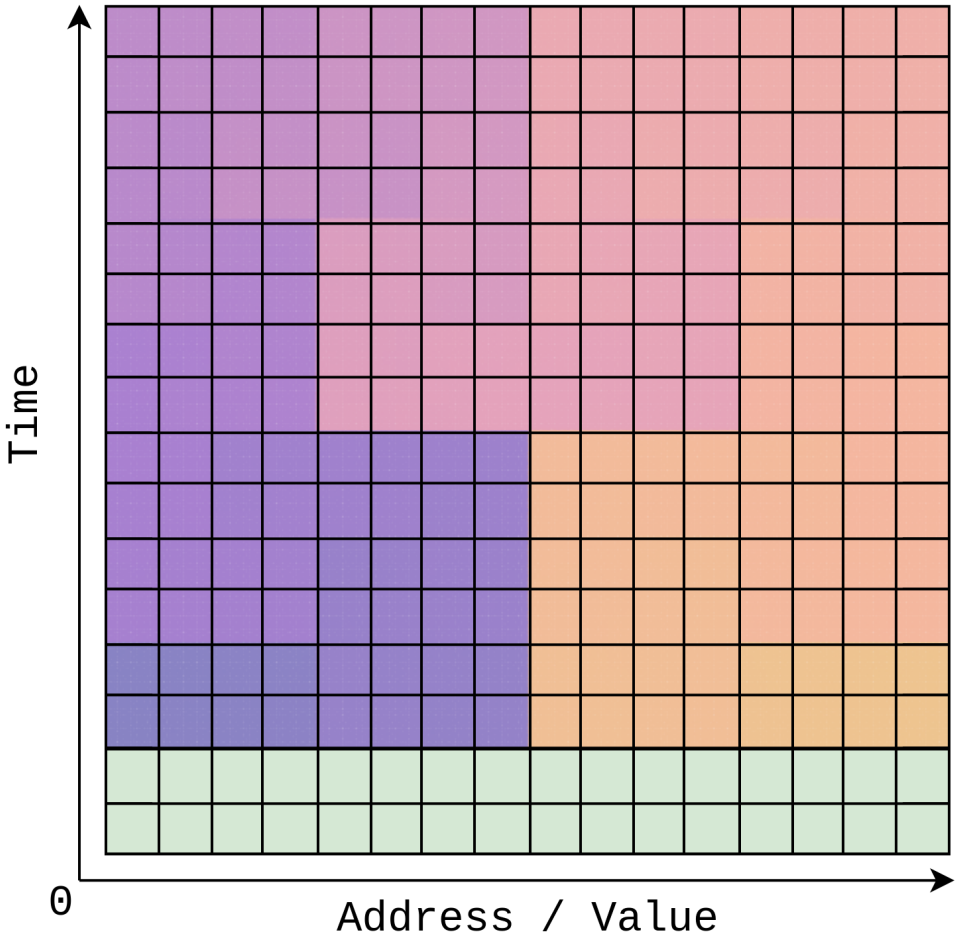
# 3ème passe : fill

SYNACKTIV



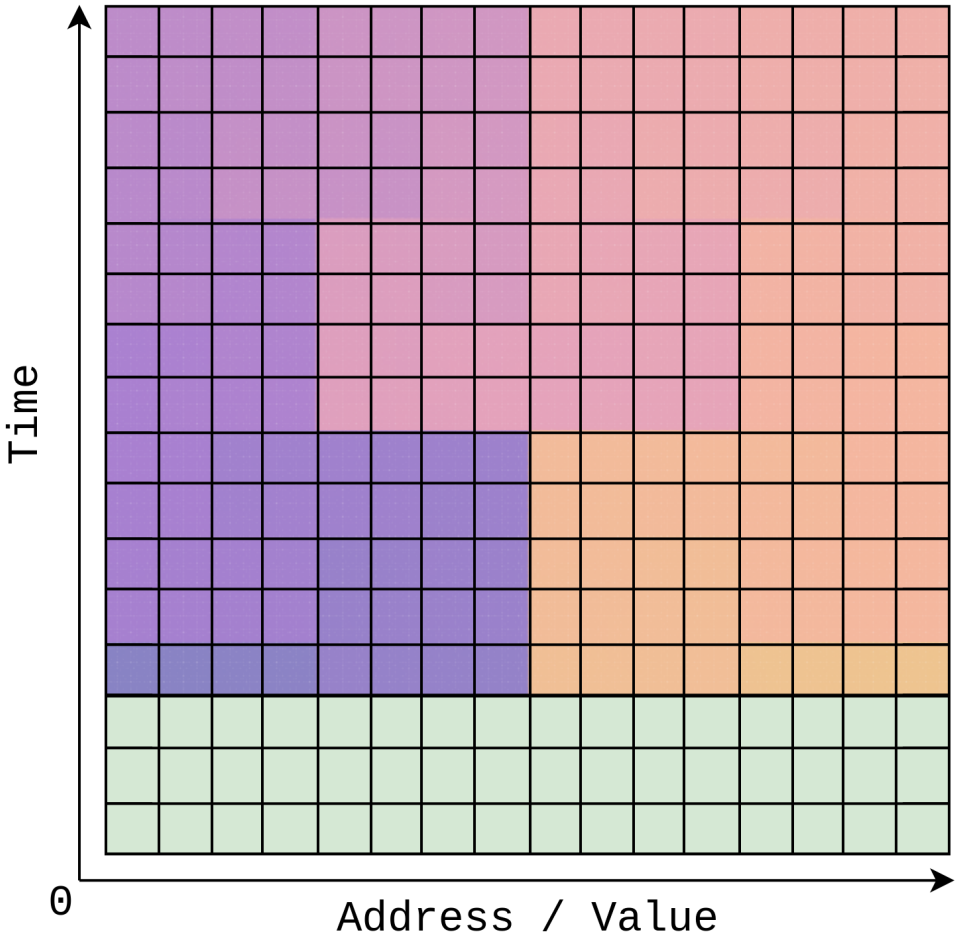
# 3ème passe : fill

SYNACKTIV



# 3ème passe : fill

SYNACKTIV



# Benchmark : Spatial Frinet

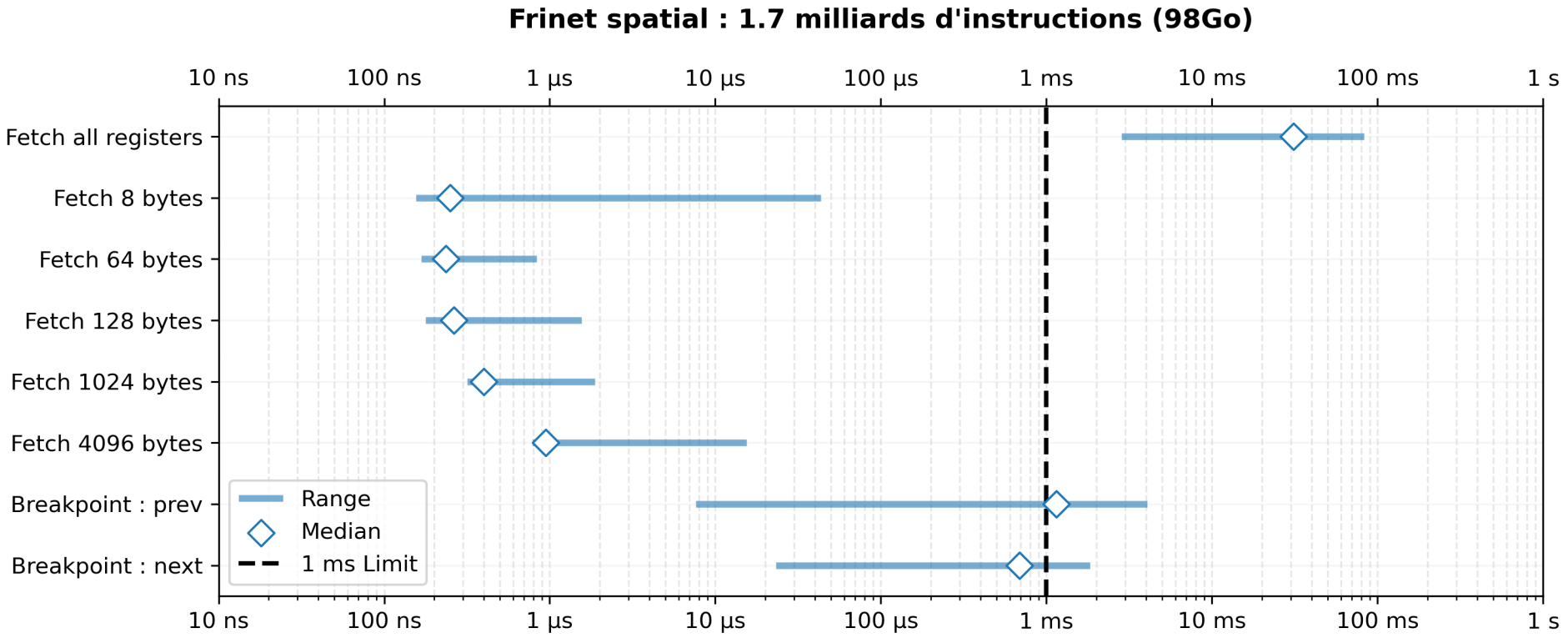
SYNACKTIV

- 1,7 milliards d'instructions / 90 Go

```
$ frinet index --format tenet wget.tenet wget.db
     Scout [00:08:45] ########################################## 90.98 GiB/90.98 GiB
Partitions [00:09:03] ########################################## 90.98 GiB/90.98 GiB
      Fill [00:12:47] ########################################## 90.98 GiB/90.98 GiB
[INFO  frinet] Total time : 31 minutes
[INFO  frinet] Maximum memory usage : 22.16 GiB
$
```

# Benchmark : passage à l'échelle

SYNACKTIV



# Conclusion

SYNACKTIV

- **2 ans de recherche et d'expérimentation**
- **Réécriture complète du backend & frontend**
  - Vue « Callgraph » pas encore implémentée
  - L'interface mérite plus de travail
- **Open-source : https://github.com/synacktiv/frinet**

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com