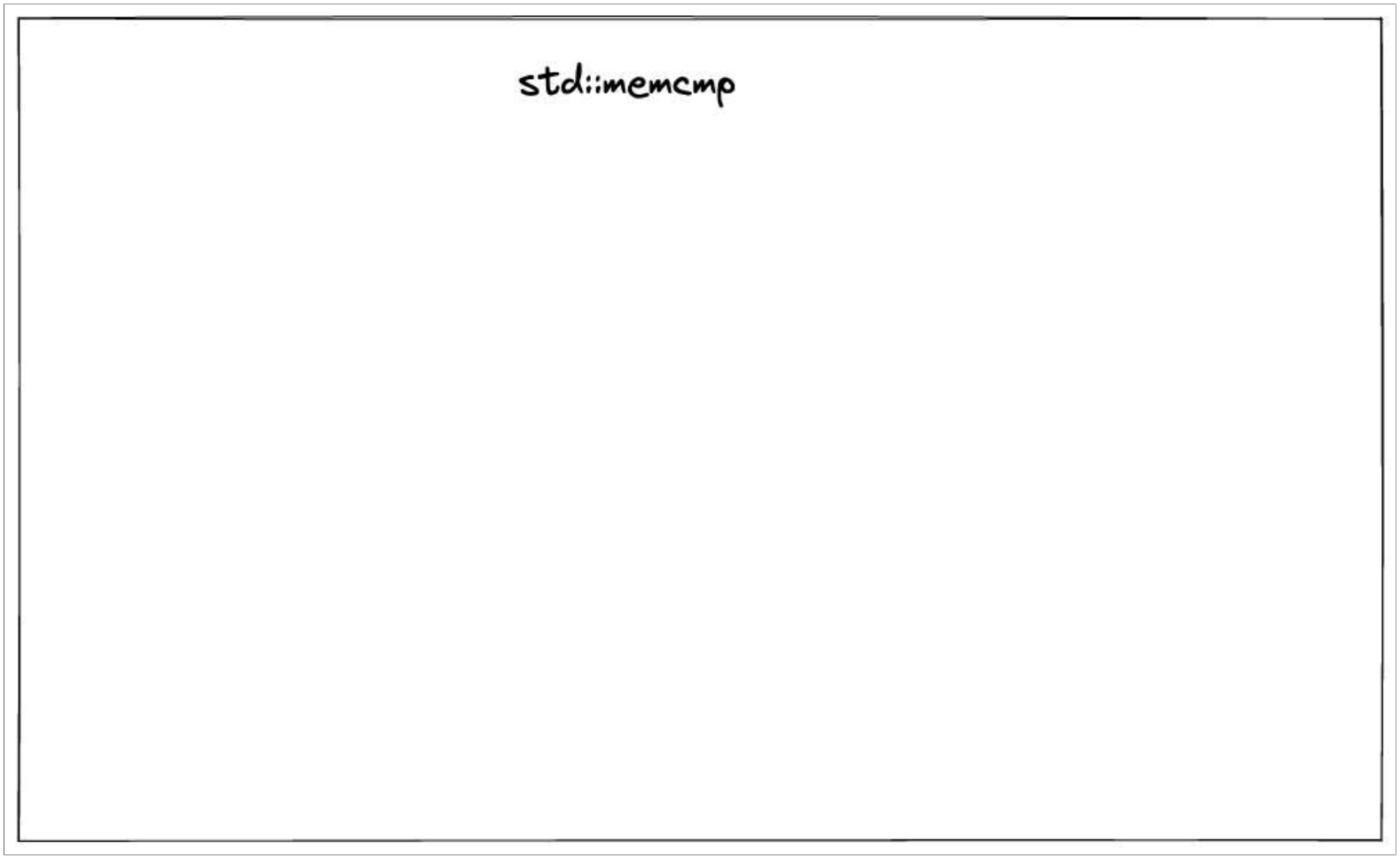
ADVANCED SIMD ALGORITHMS IN PICTURES

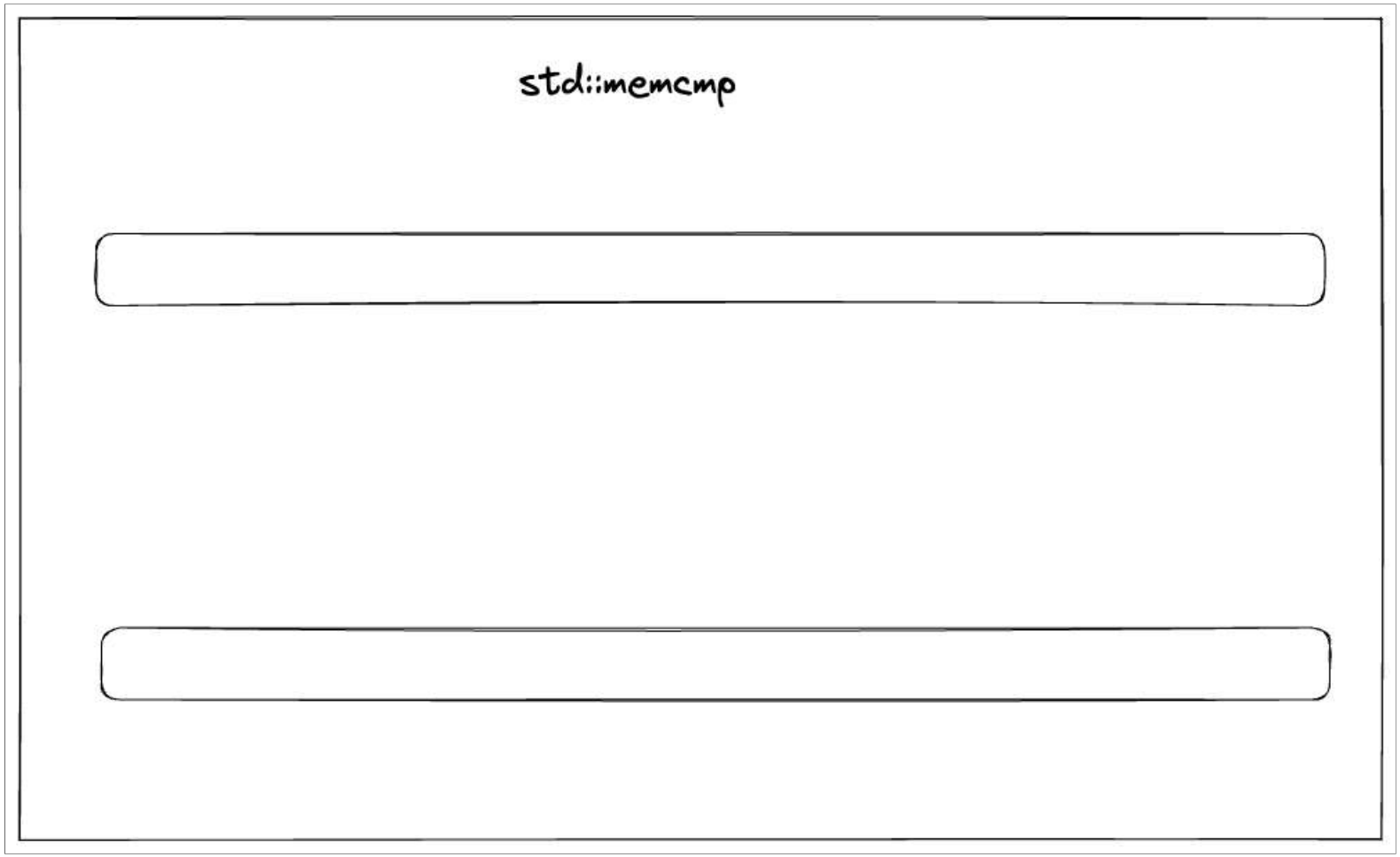
Denis Yaroshevskiy

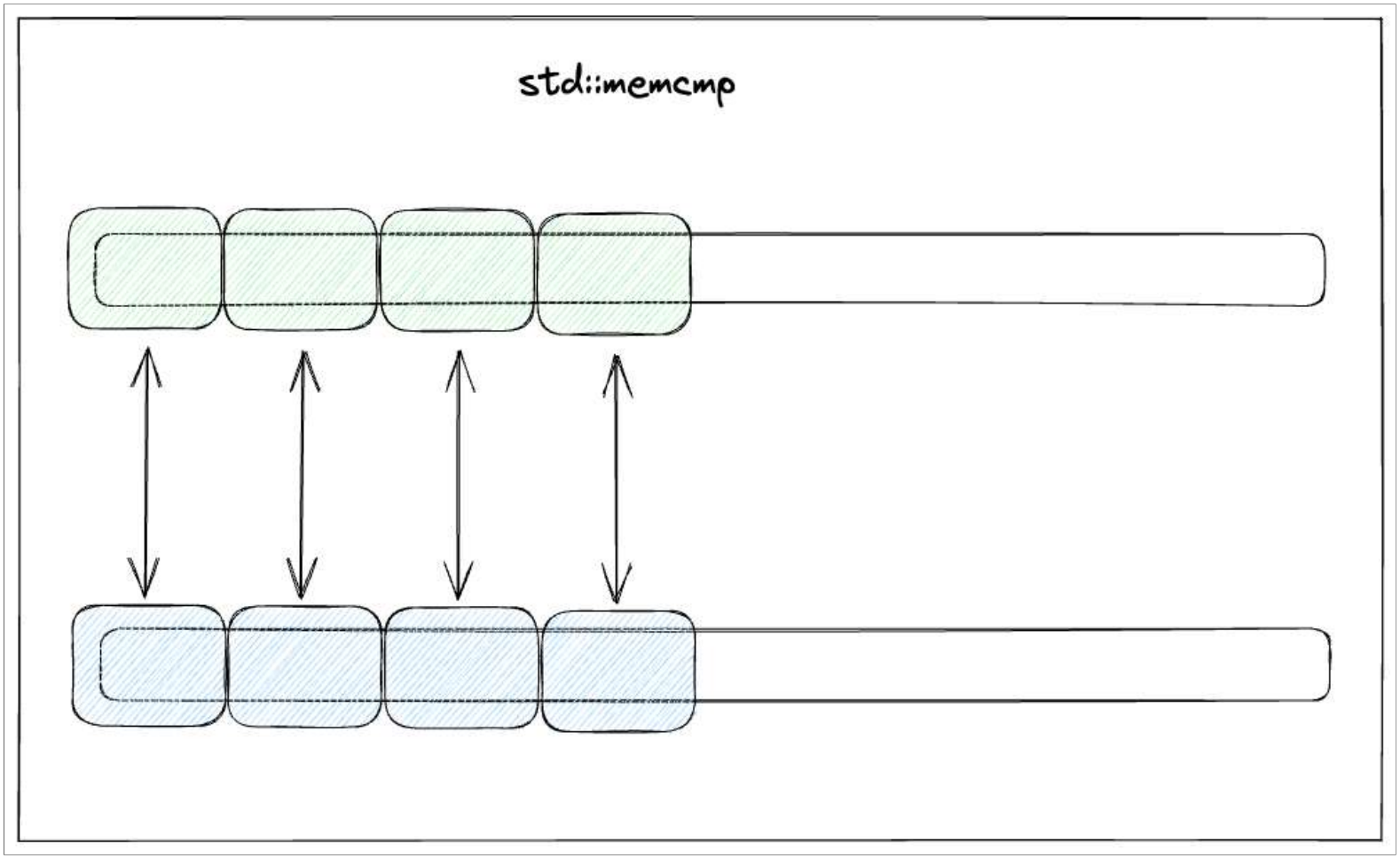
THIS TALK

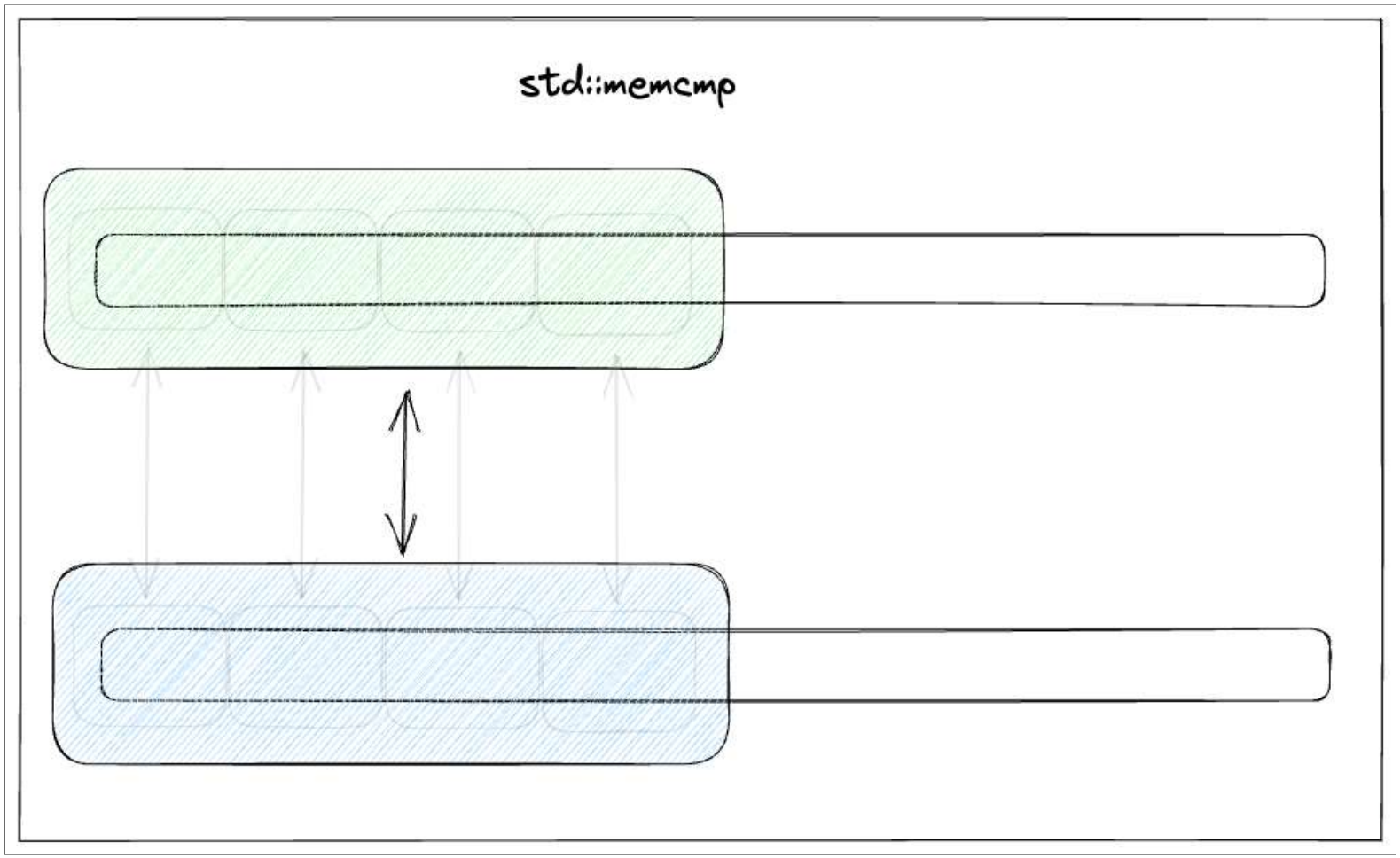
- tinyurl.com/dycppcon2023
- memcmp
- copy_if
- set_intersection

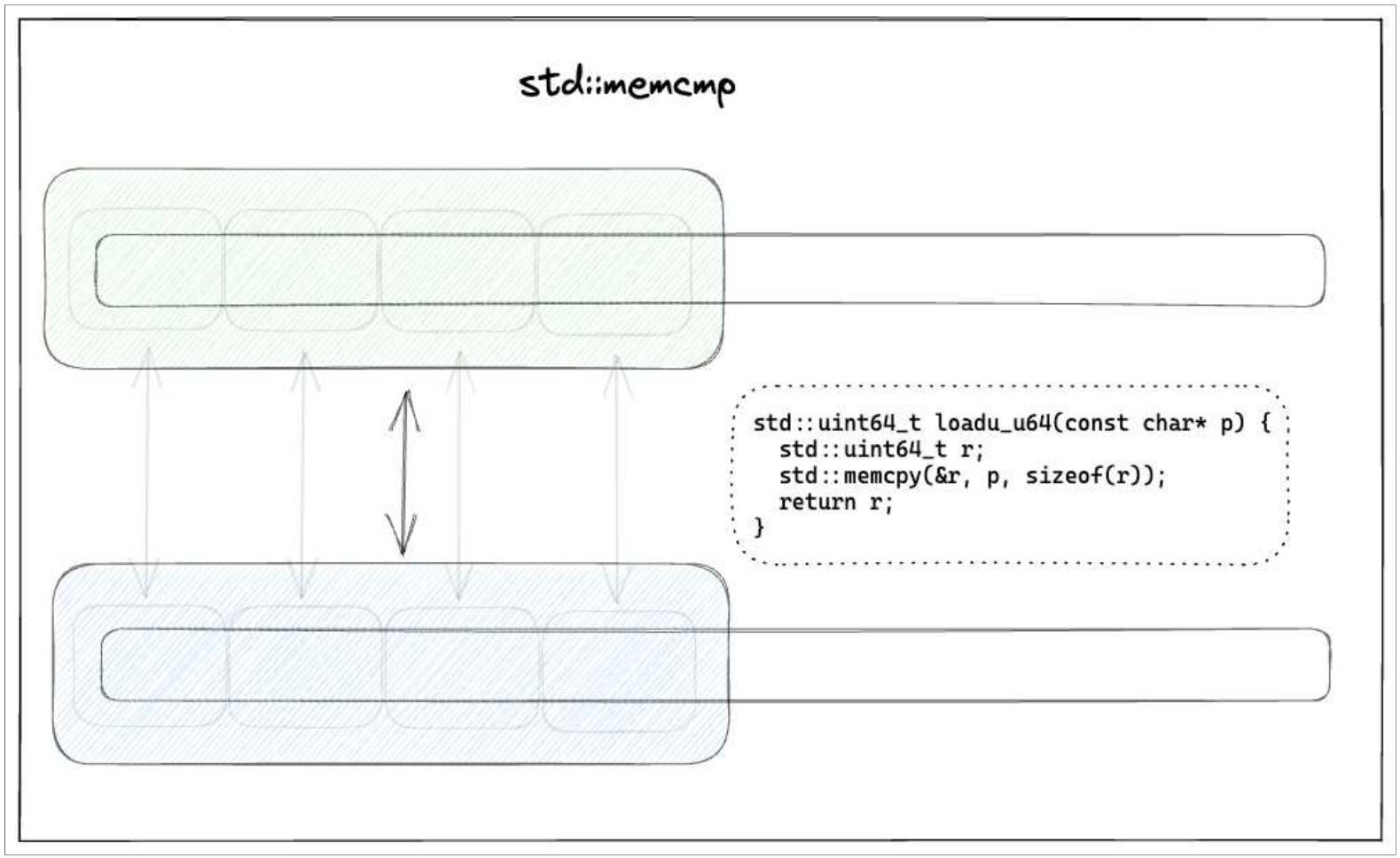
MEMCMP


std::memcmp

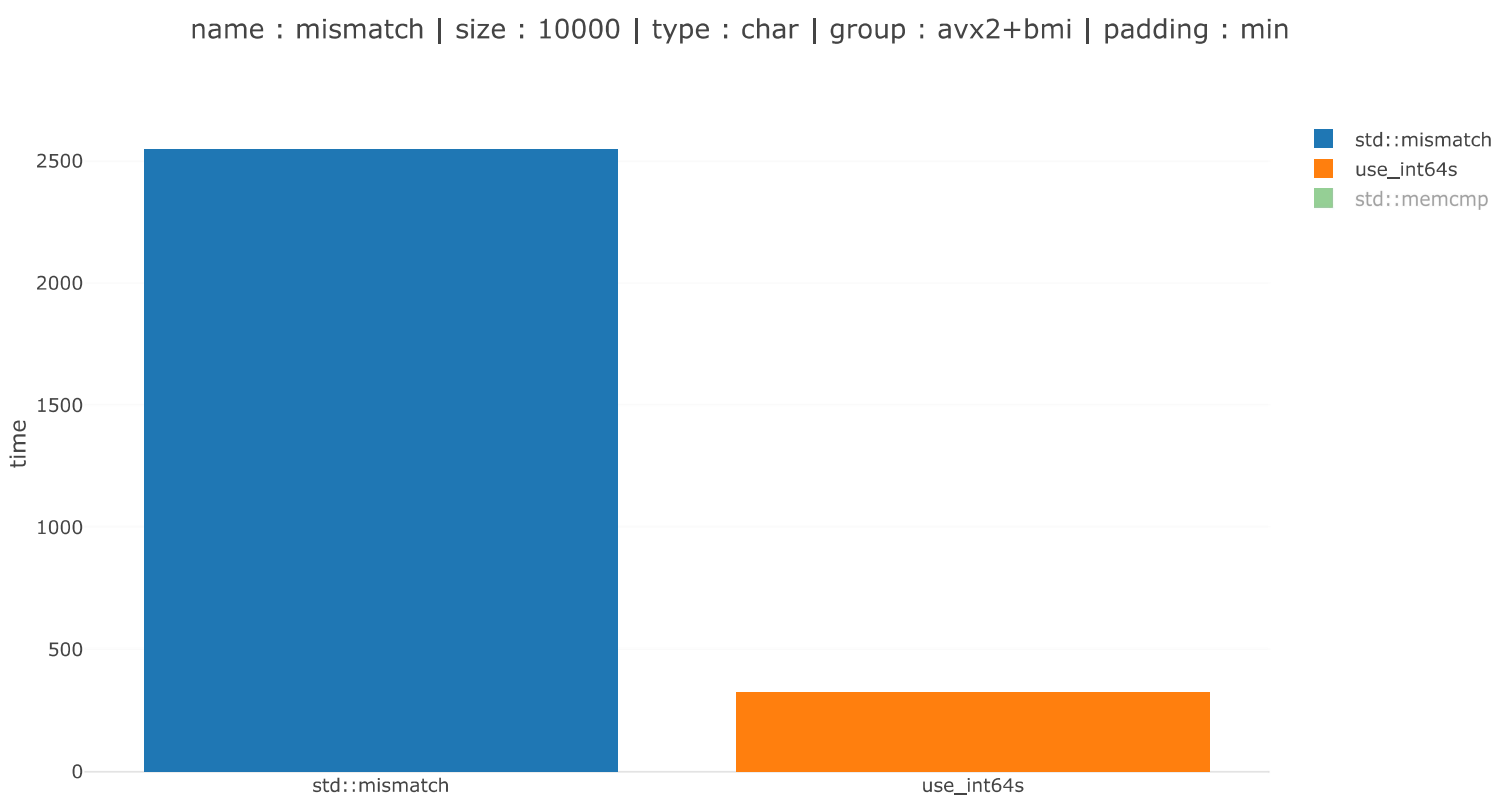

std::memcmp


std::memcmp


std::memcmp


std::memcmp
std::uint64_t loadu_u64(const char* p) {
std::uint64_t r;
std::memcpy(&r, p, sizeof(r));
return r;
}

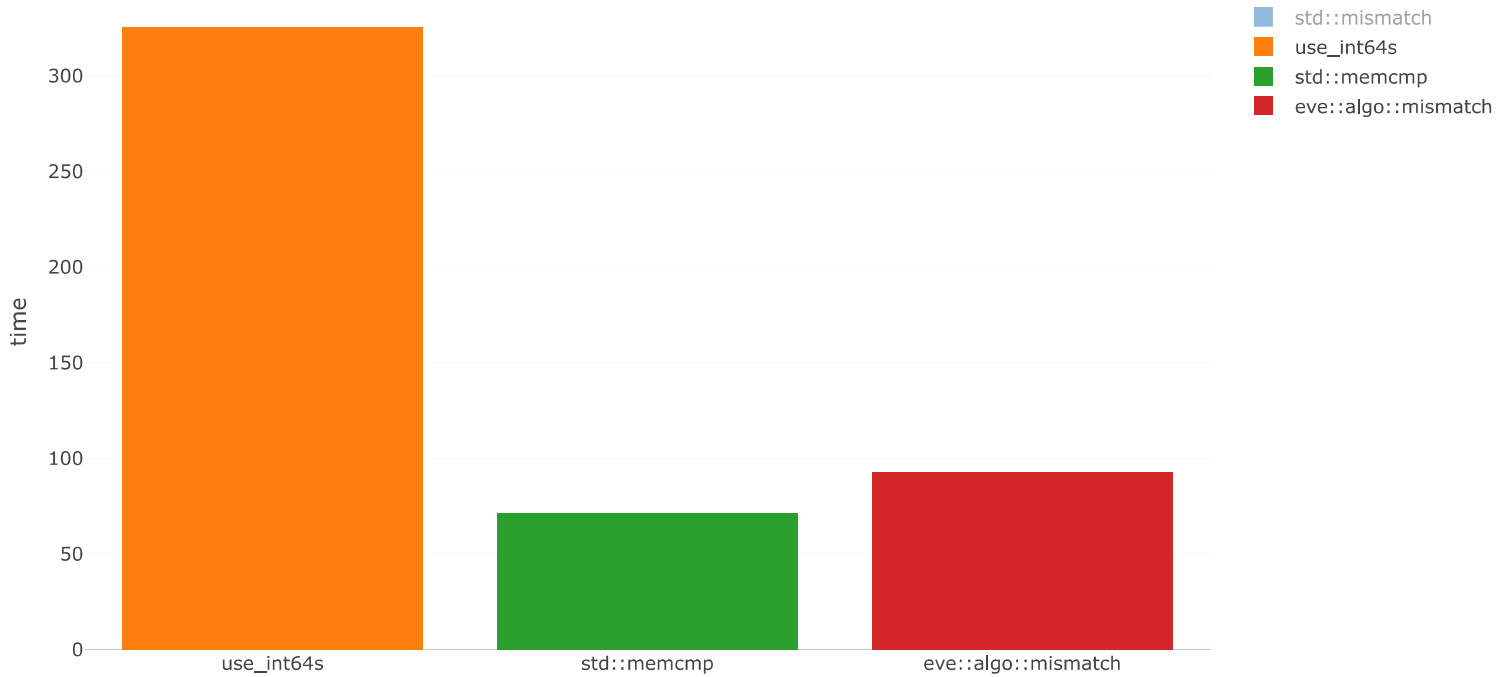
MEMCMP RESULTS, X86



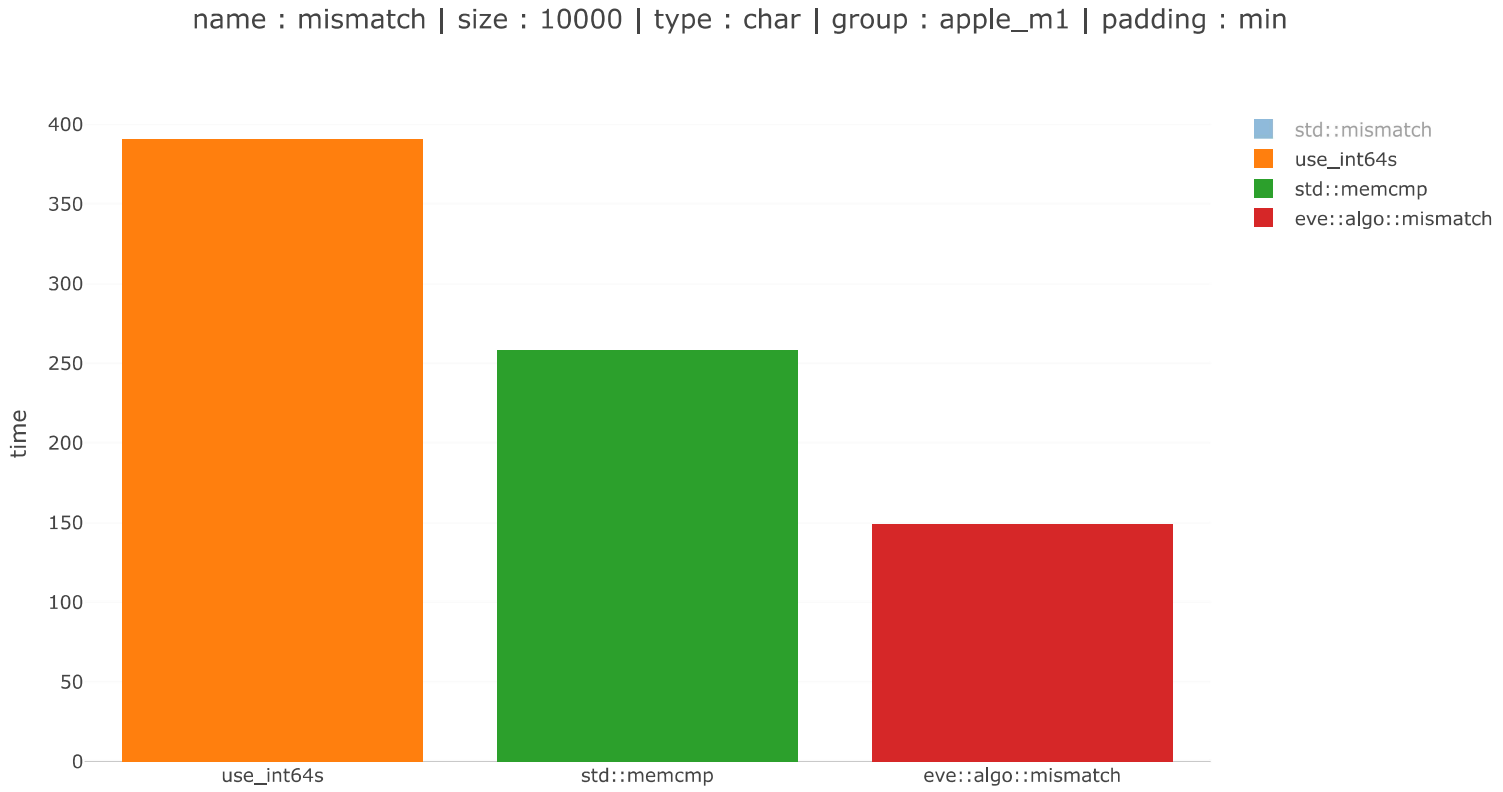
EVE LIBRARY

- **github**
- Joel Falcou, Jean-Thierry Lapresté, Denis Yaroshevskiy
- **SIMD in C++20: EVE of a new Era**
- eve::algo::mismatch

MEMCMP RESULTS, X86


name : mismatch | size : 10000 | type : char | group : avx2+bmi | padding : min
std::mismatch
use_int64s
std::memcmp
eve::algo::mismatch
time
300
250
200
150
100
50
0
use_int64s
std::memcmp
eve::algo::mismatch

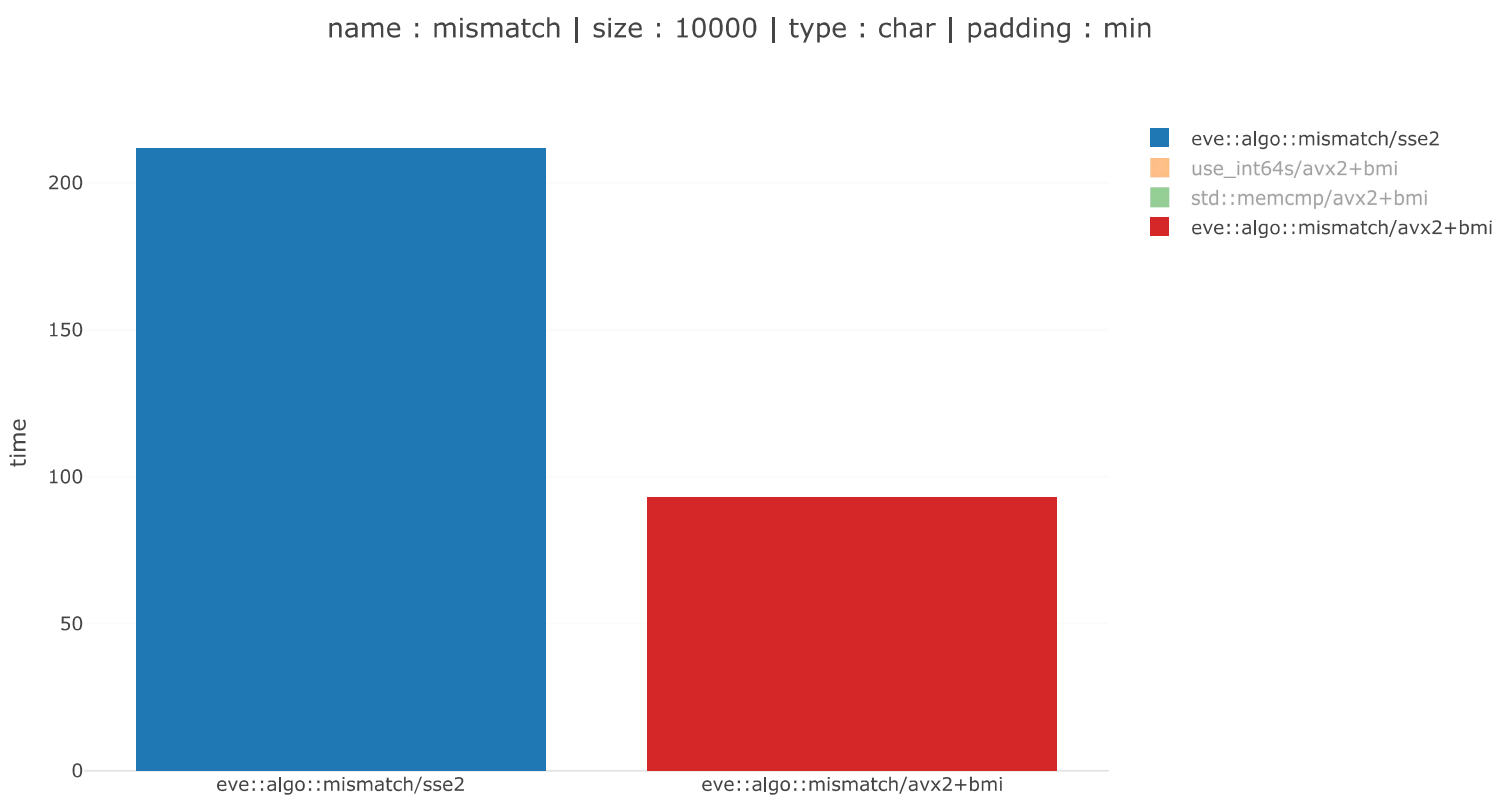
MEMCMP RESULTS, M1


name : mismatch | size : 10000 | type : char | group : apple_m1 | padding : min
time
400
350
300
250
200
150
100
50
0
use_int64s
std::memcmp
eve::algo::mismatch
std::mismatch
use_int64s
std::memcmp
eve::algo::mismatch

VECTOR PROCESSOR EXTENSIONS

- x86
  - 128 bits: SSE2, SSE3, SSSE3, SSE4, SSE4.1, SSE4.2
  - 256 bits: AVX, AVX2, XOP
  - 512 bits: AVX512 and its myriad of sub-genre
- ARM
  - 128 bits: NEON, ASIMD
  - SVE (VLS/VLA)
- PowerPC
- WASM

MARCHING


name : mismatch | size : 10000 | type : char | padding : min
time
200
150
100
50
0
eve::algo::mismatch/sse2
eve::algo::mismatch/avx2+bmi
eve::algo::mismatch/sse2
use_int64s/avx2+bmi
std::memcmp/avx2+bmi
eve::algo::mismatch/avx2+bmi

WHAT'S INSIDE MISMATCH?

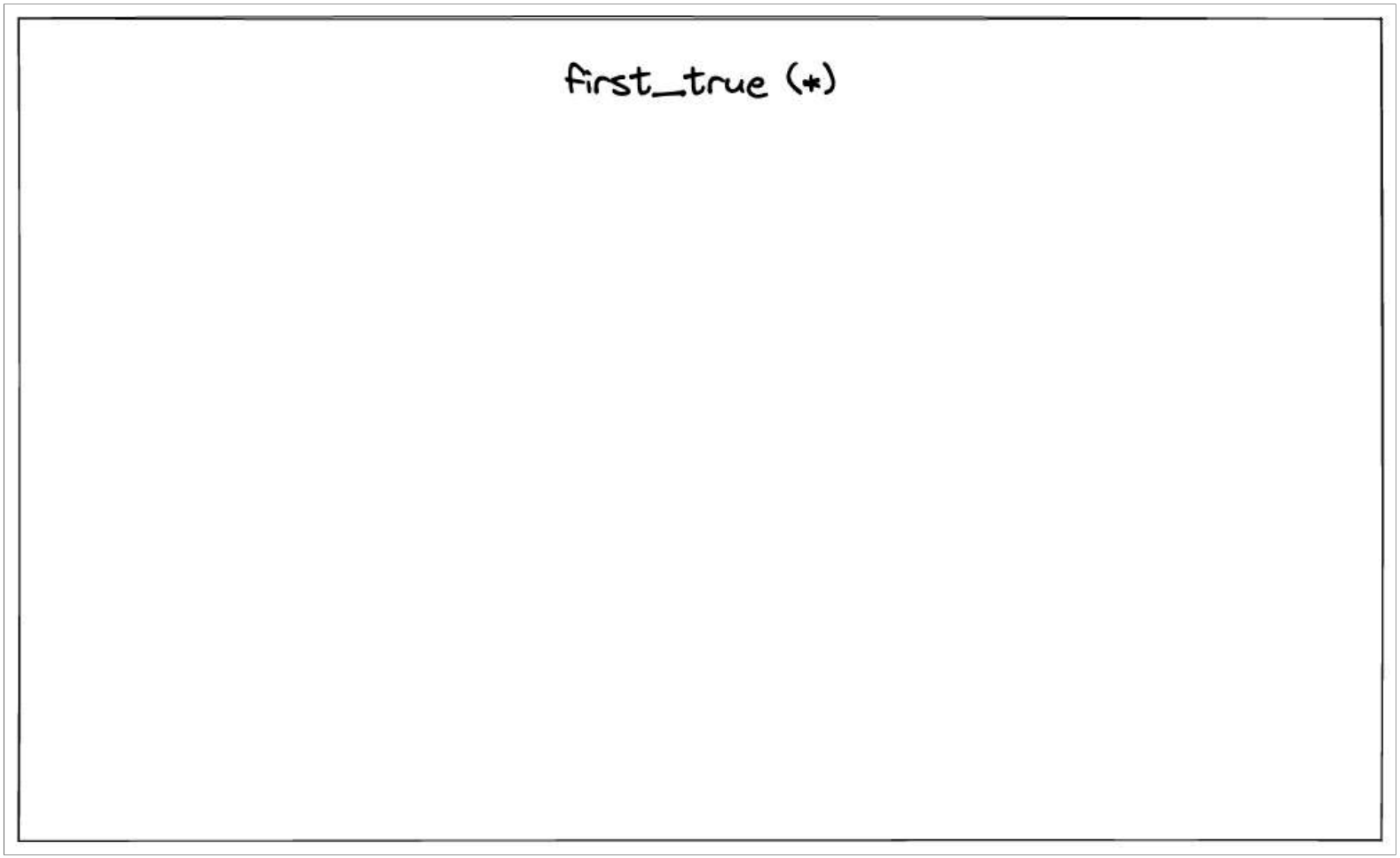
- find_if + zip

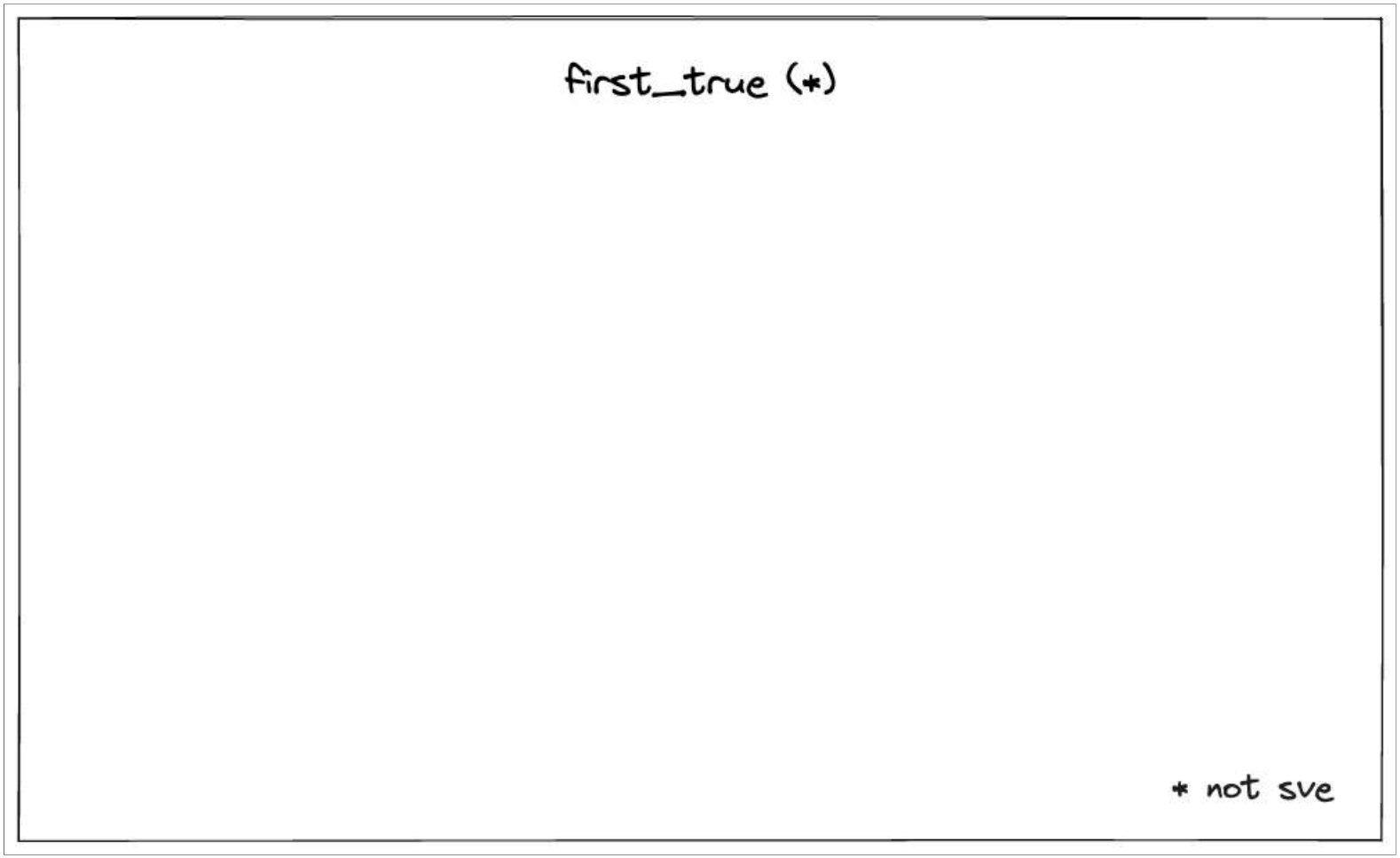
MISMATCH FROM SCRATCH

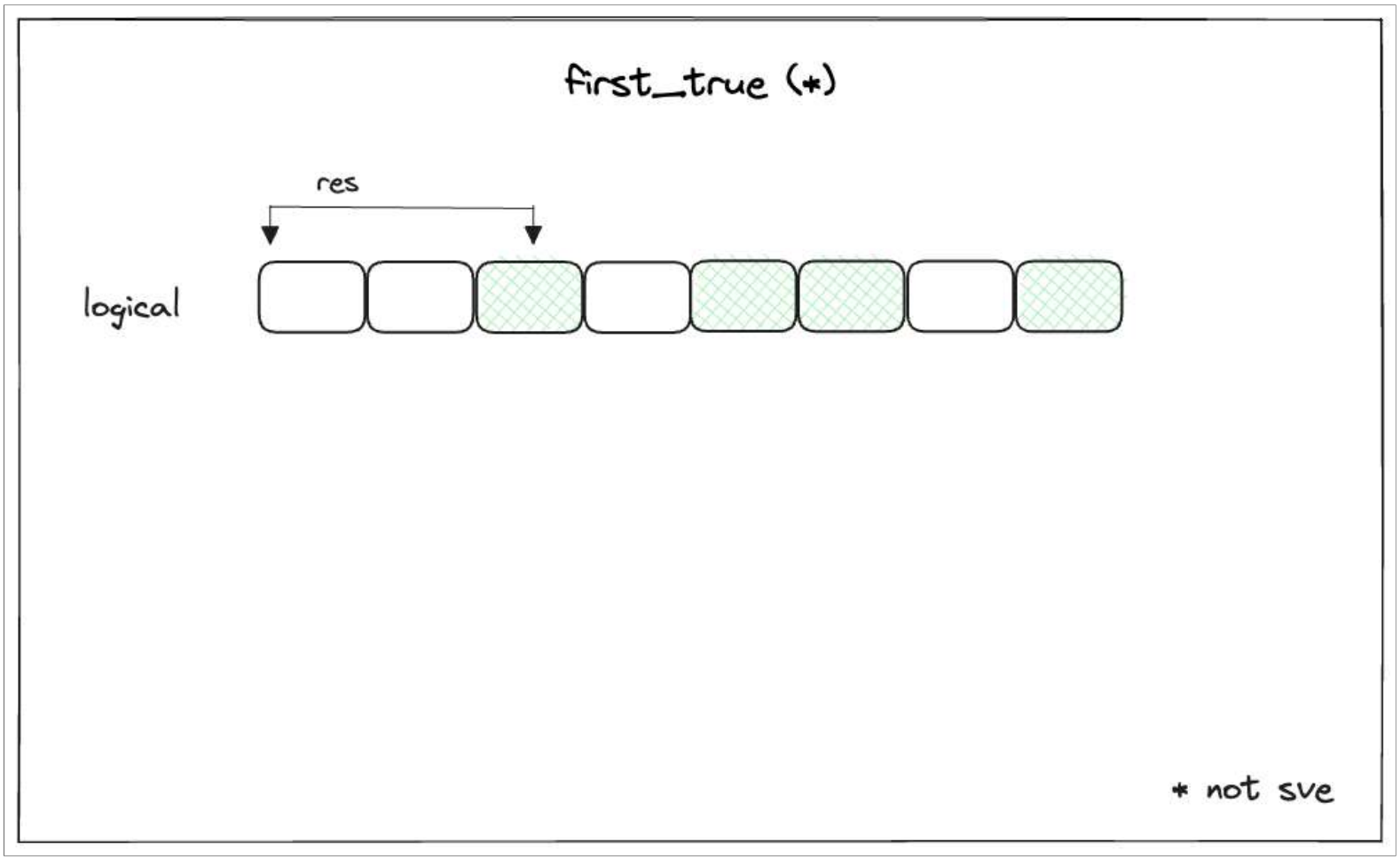
Source Editor: C++ source #1

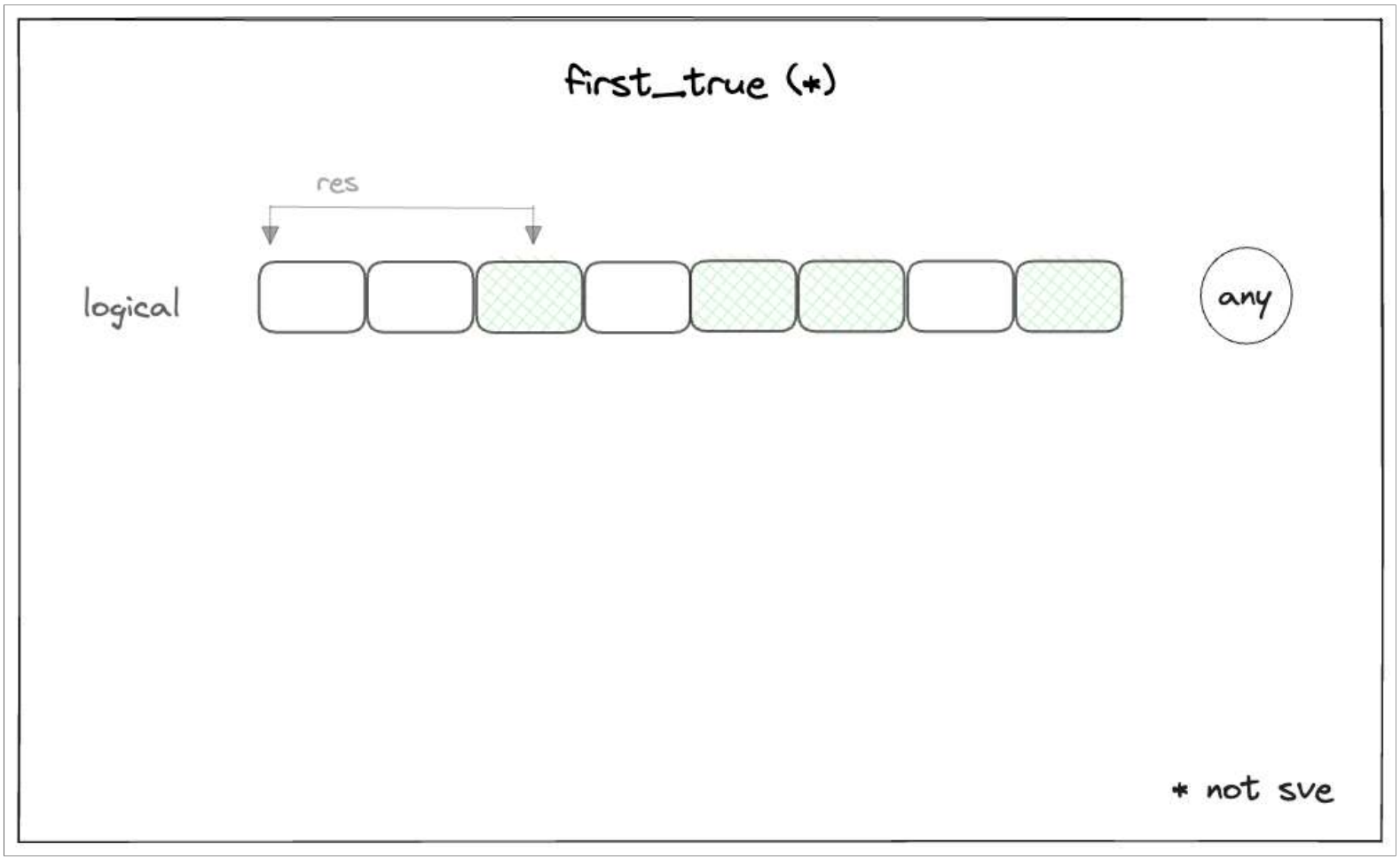
```cpp
#include <eve/module/core.hpp>

bool core_of_mmatch(const std::uint8_t*& a, const std::uint8_t*& b)
{
  using wide = eve::wide<std::uint8_t>;

  wide wide_a = eve::load(a);  // [0, 1, 2, 3]
  wide wide_b = eve::load(b);  // [0, 9, 2, 3]

  eve::logical<wide> test = wide_a != wide_b;  // [ F, T, F, F ]
  std::optional<std::ptrdiff_t> match = eve::first_true(test); // {1}

  if (!match) {
    a += wide::size();
    b += wide::size();
    return false;
  }

  a += *match;
  b += *match;
  return true;
}
```

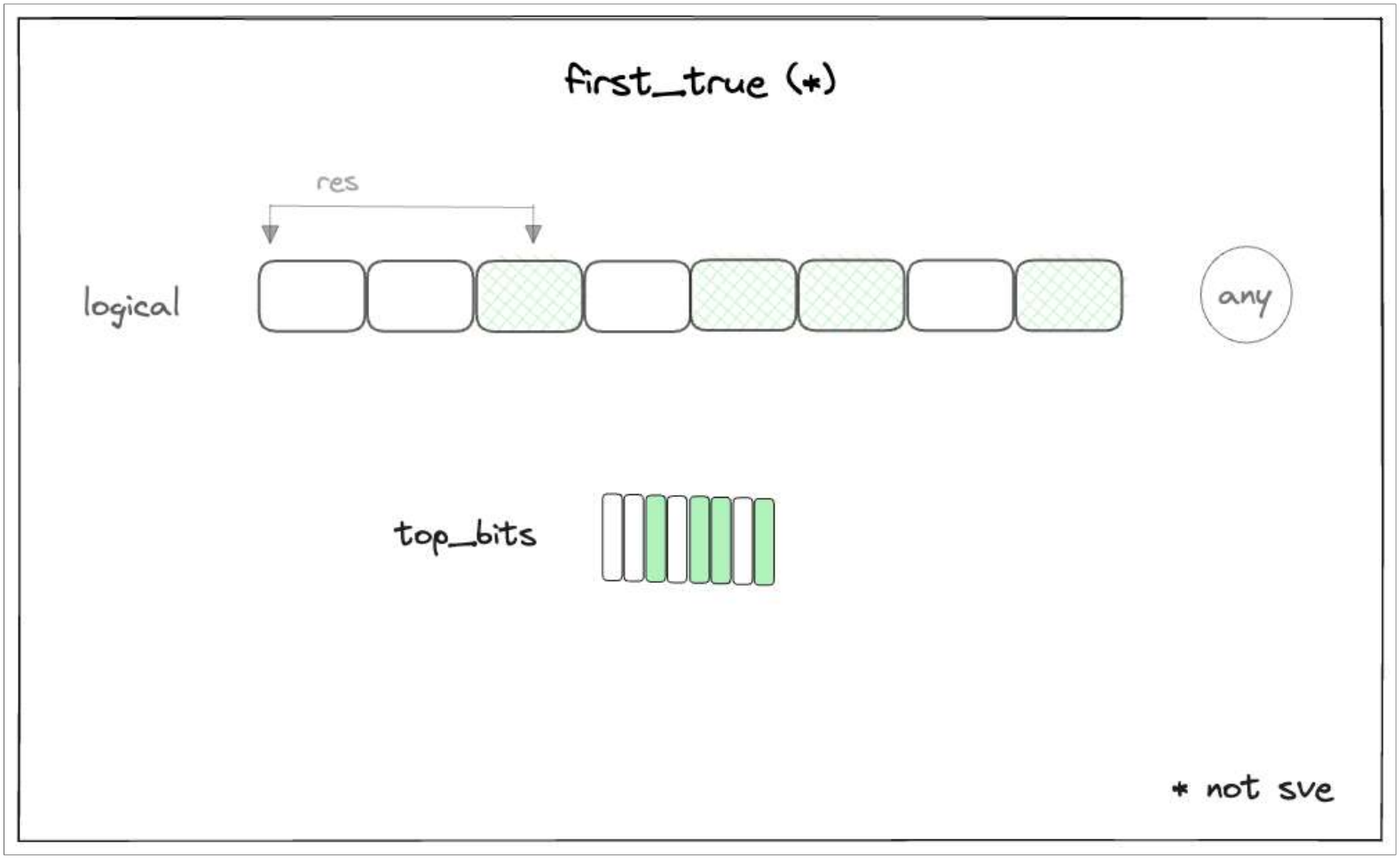
Edit on Compiler Explorer

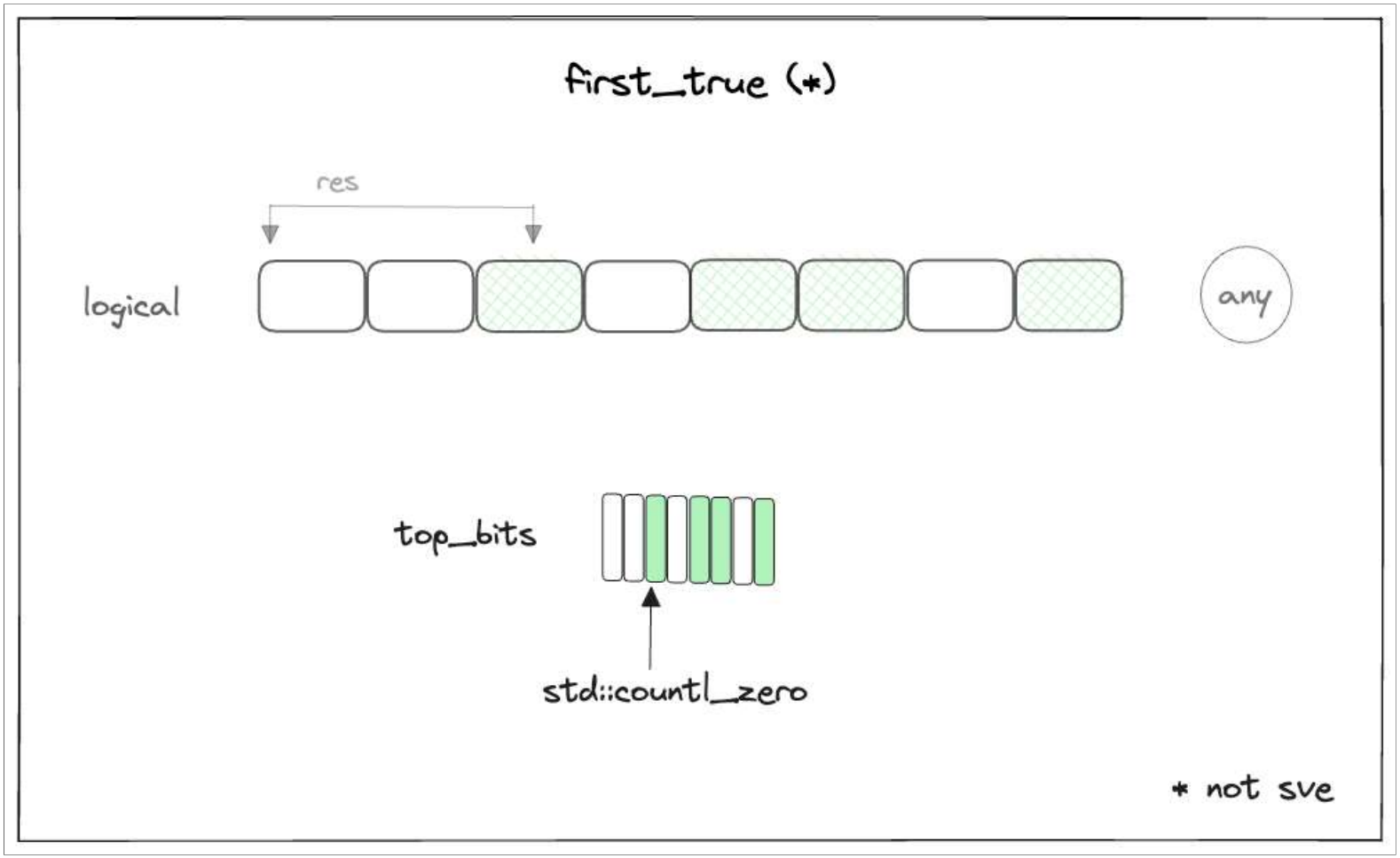

first_true (*)

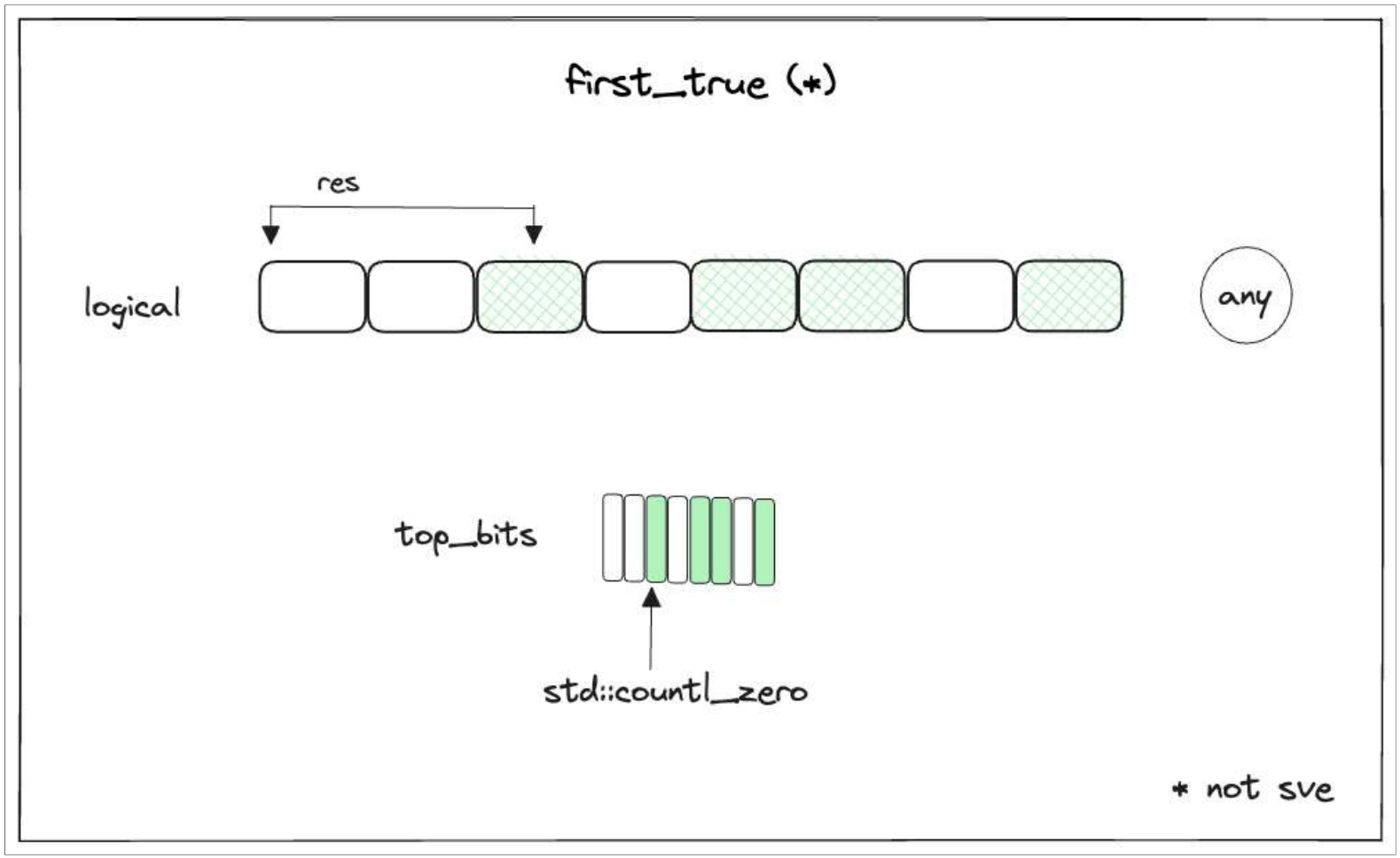

first_true (*)
* not sve


first_true (*)
res
logical
* not sve


first_true (*)
res
logical
any
* not sve


first_true (*)
res
logical
any
top_bits
* not sve

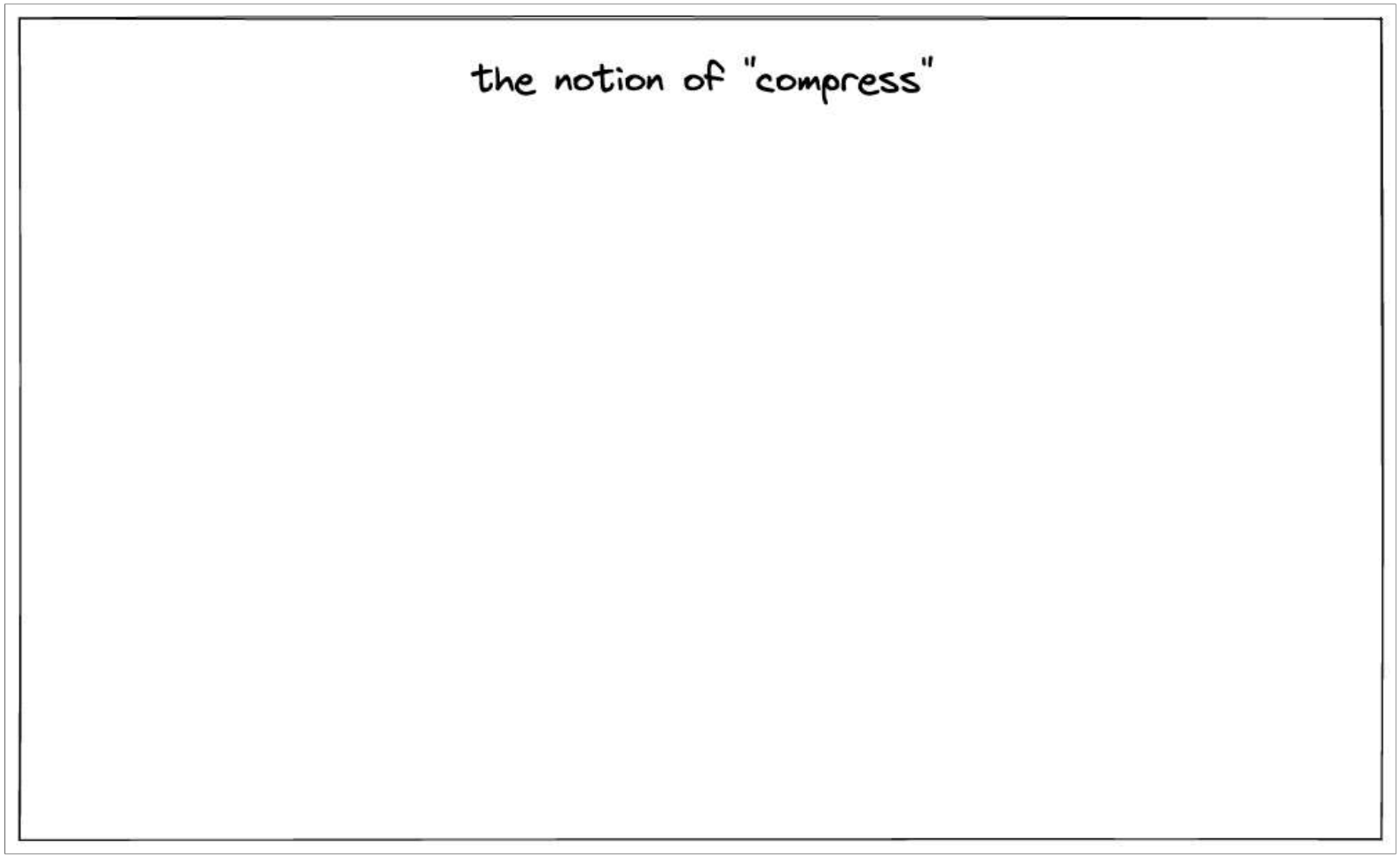

first_true (*)
res
logical
any
top_bits
std::countl_zero
* not sve


first_true (*)
res
logical
any
top_bits
std::countl_zero
* not sve

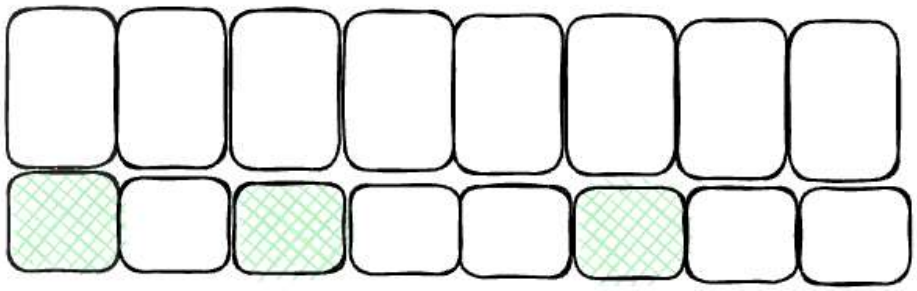
COMPARE WITH PROPOSAL

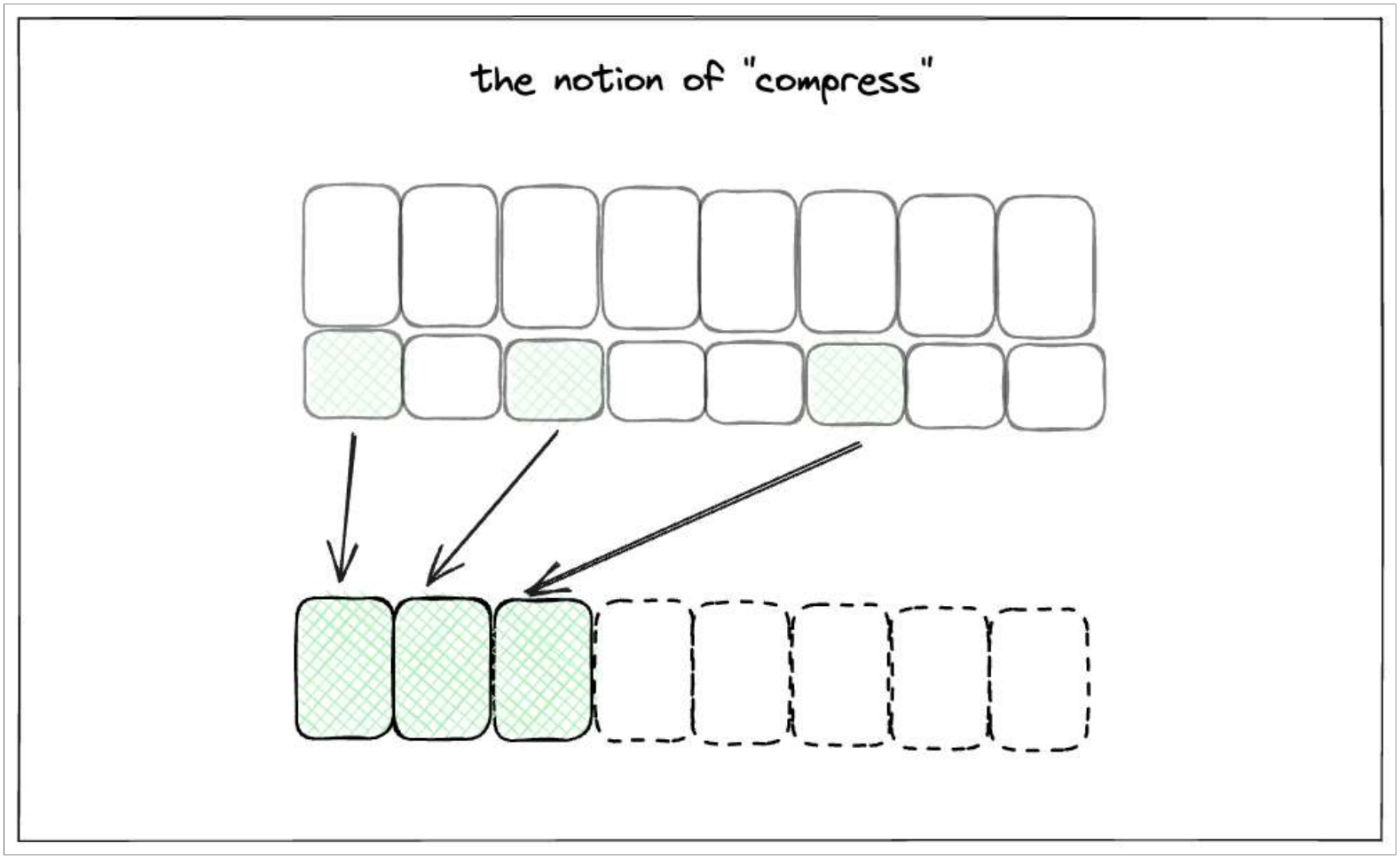
eve	p1928(r6)
wide	std::simd
load	.copy_from
logical	std::simd_mask
!=	!=
first_true	any_of + reduce_min_index

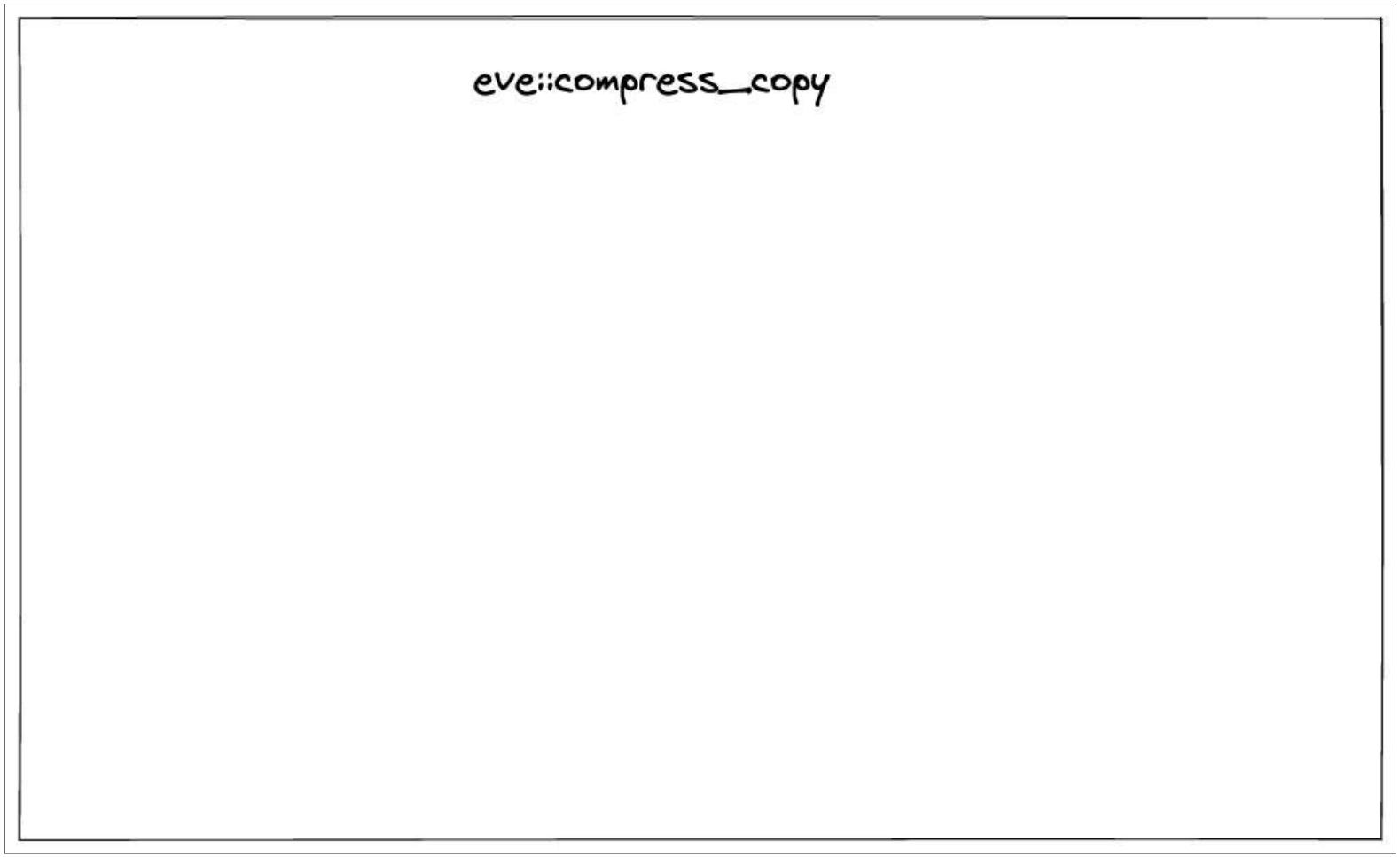
COPY_IF

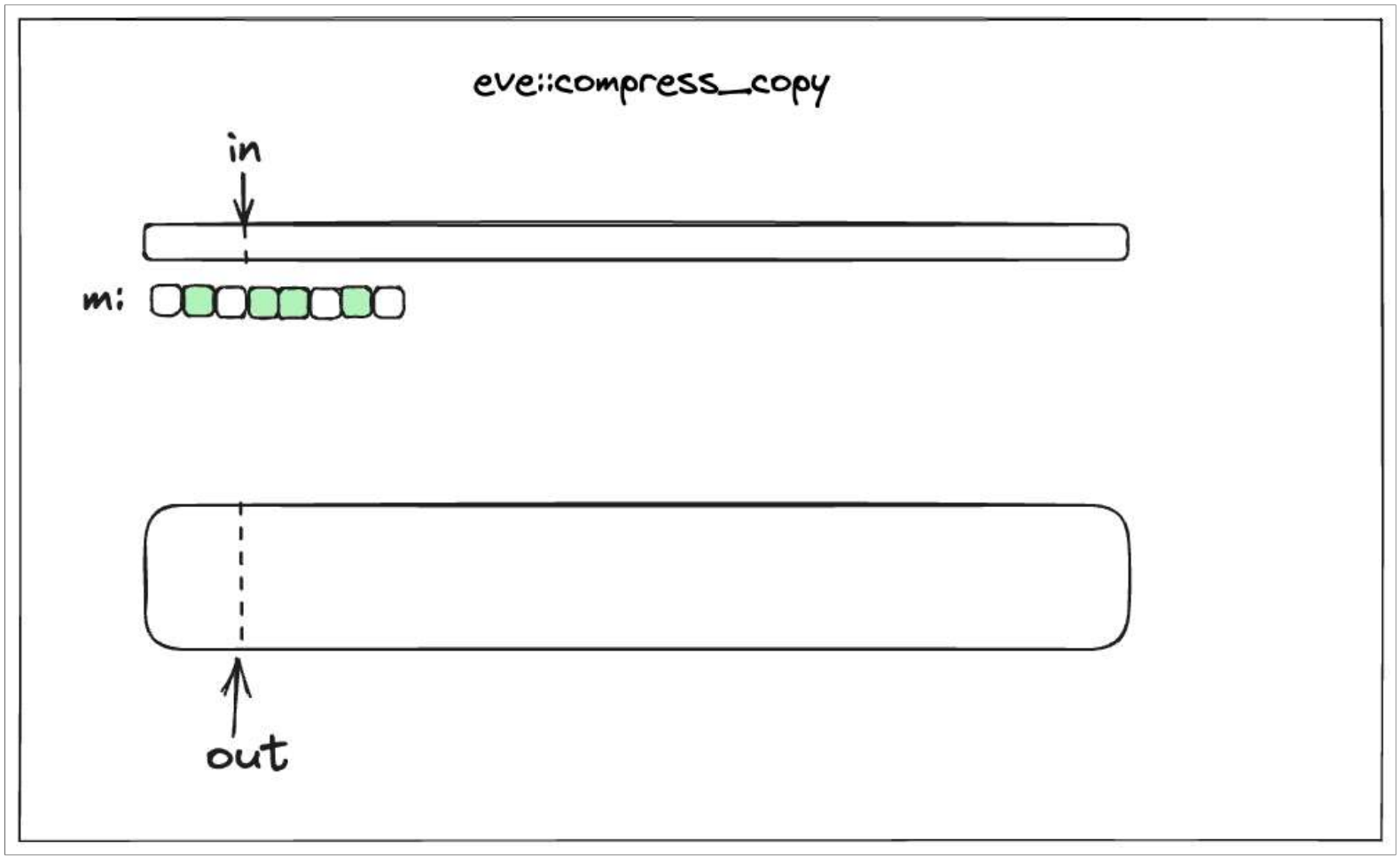

the notion of "compress"

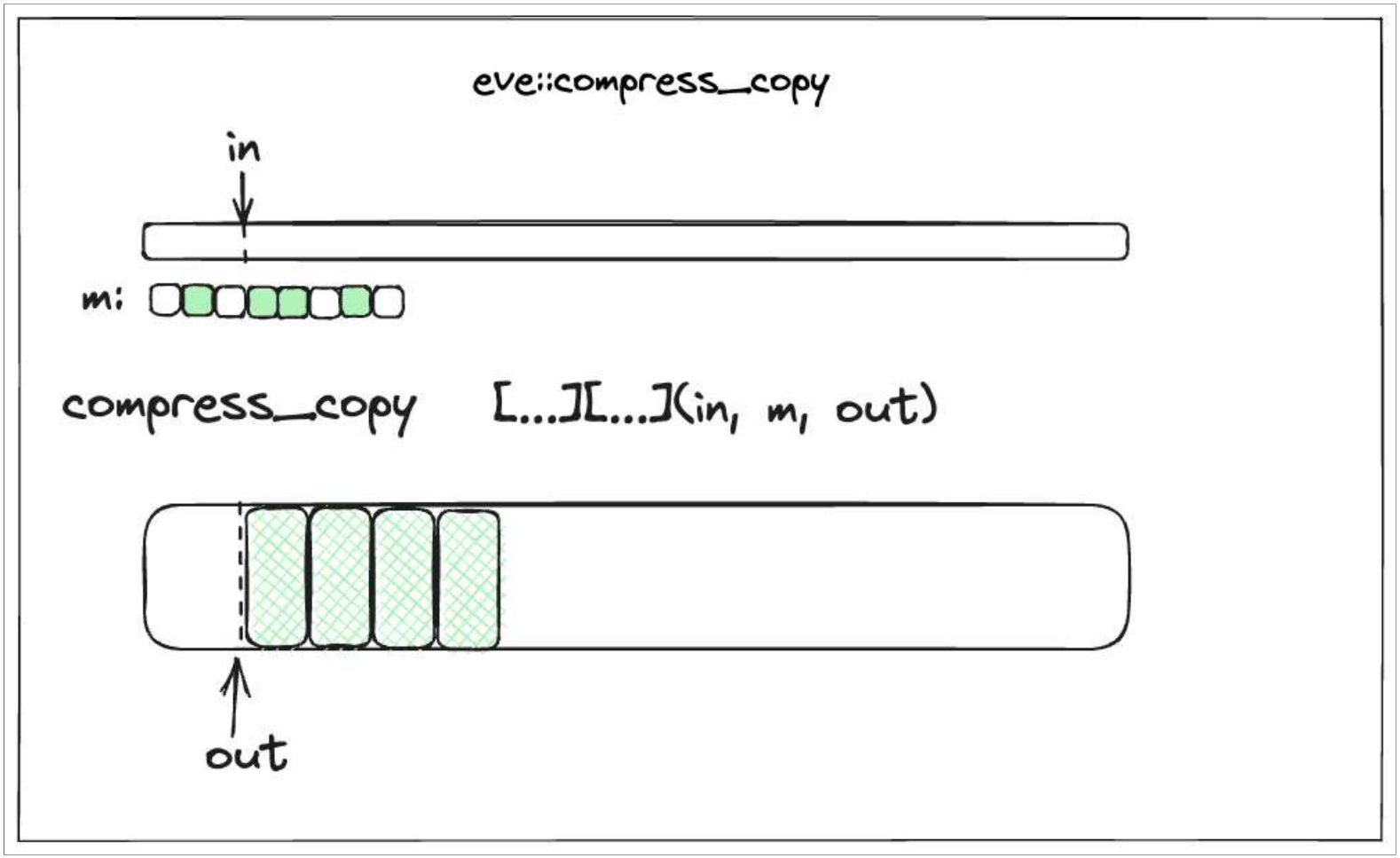
the notion of "compress"

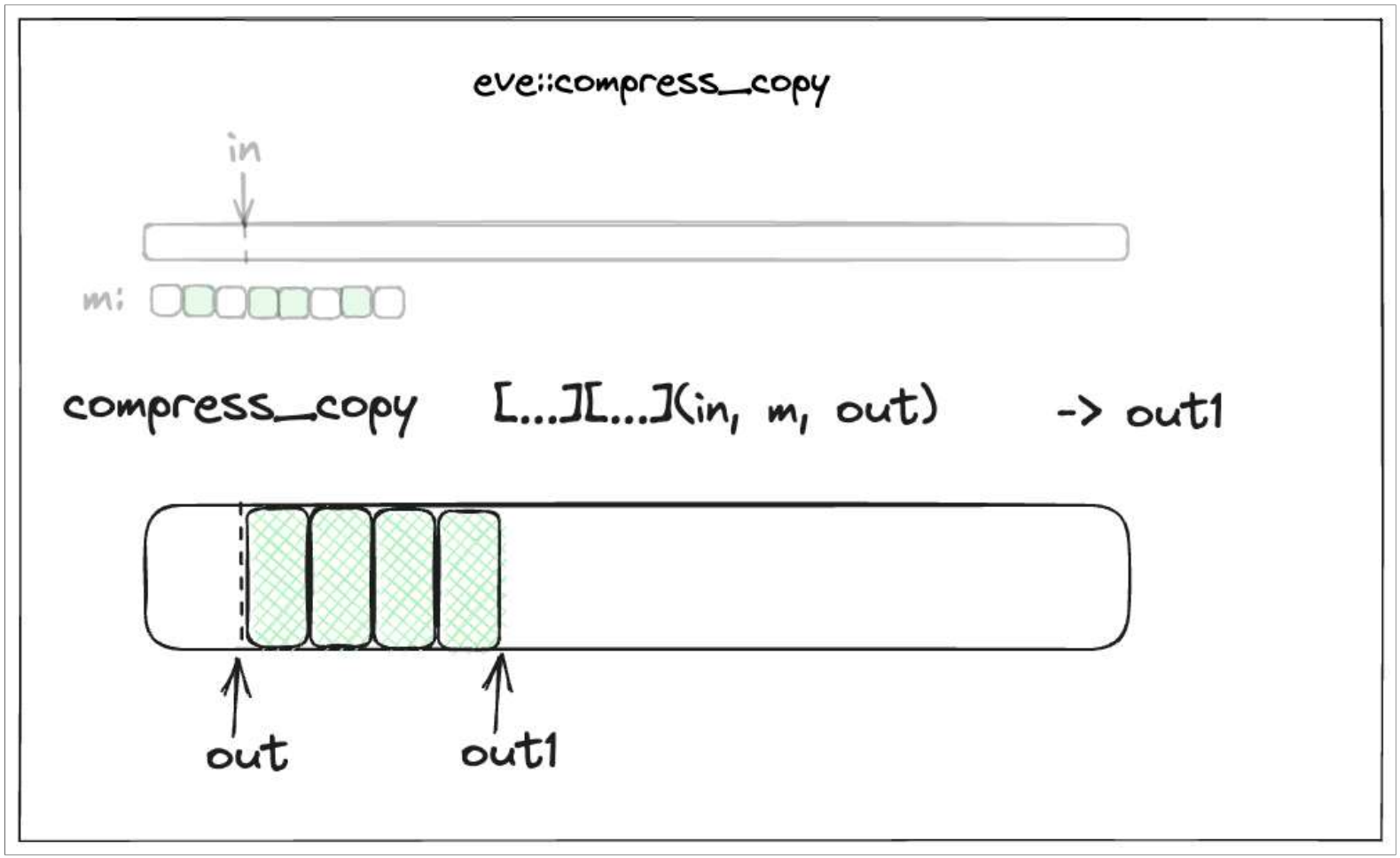


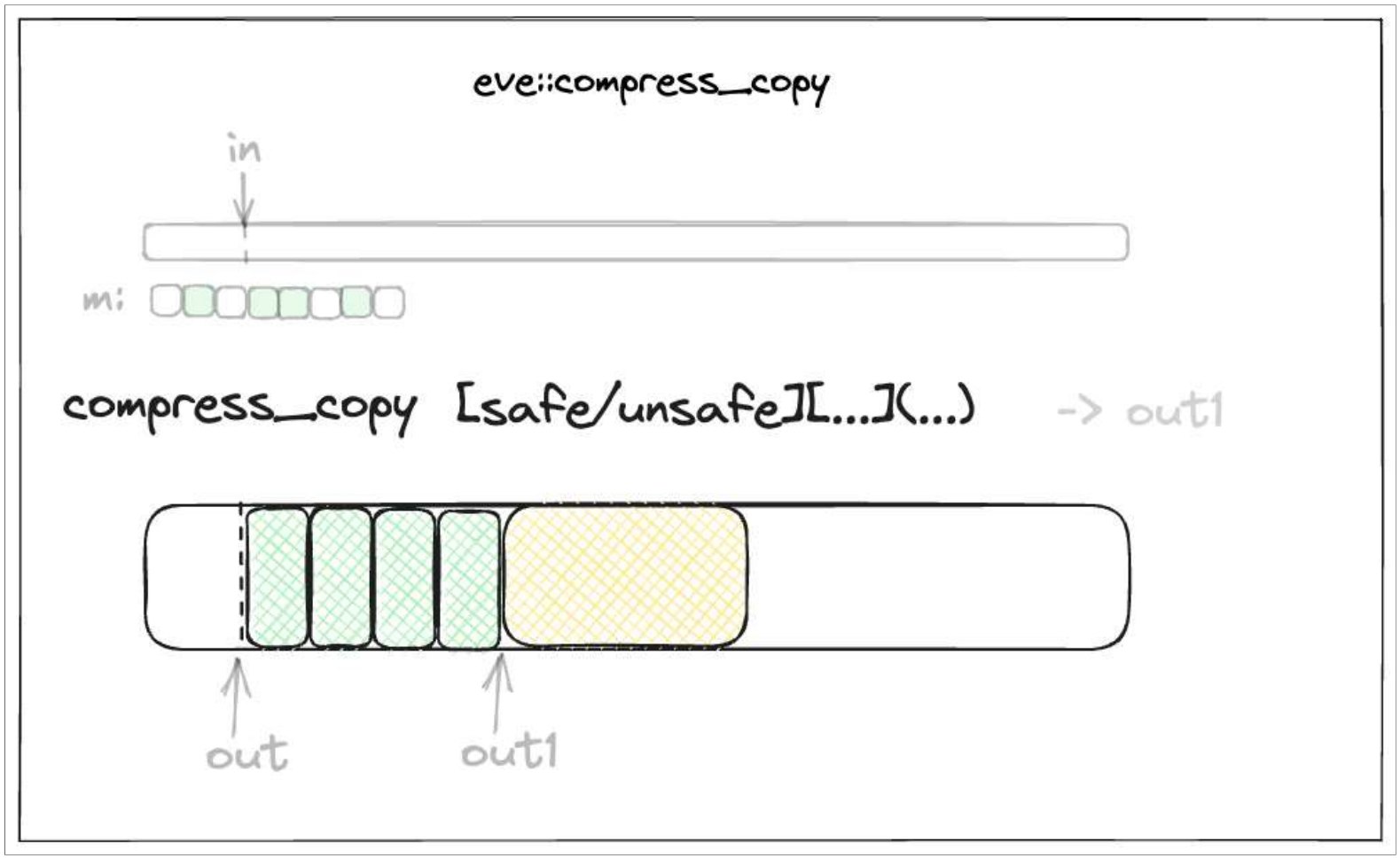

the notion of "compress"

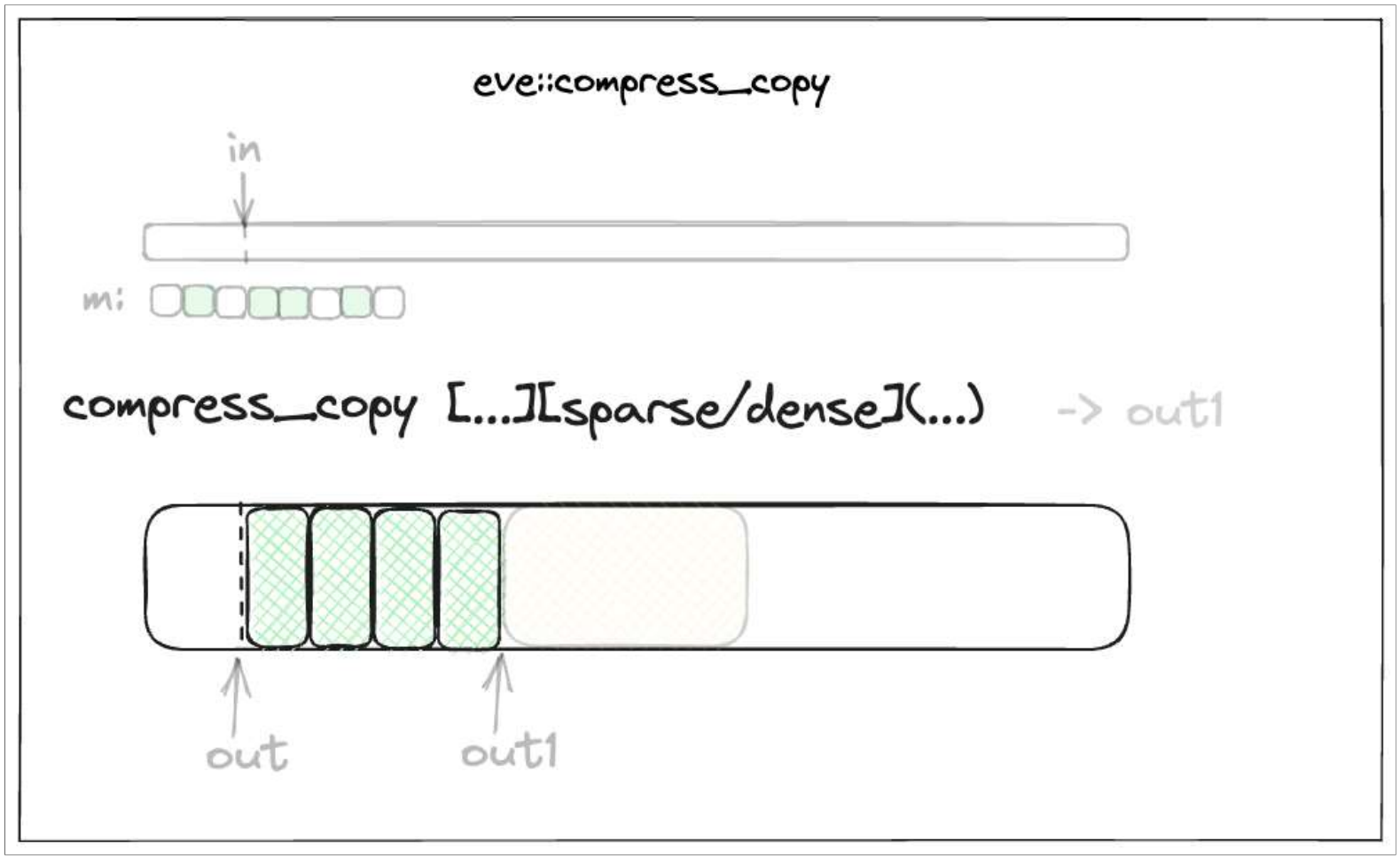

eve::compress_copy

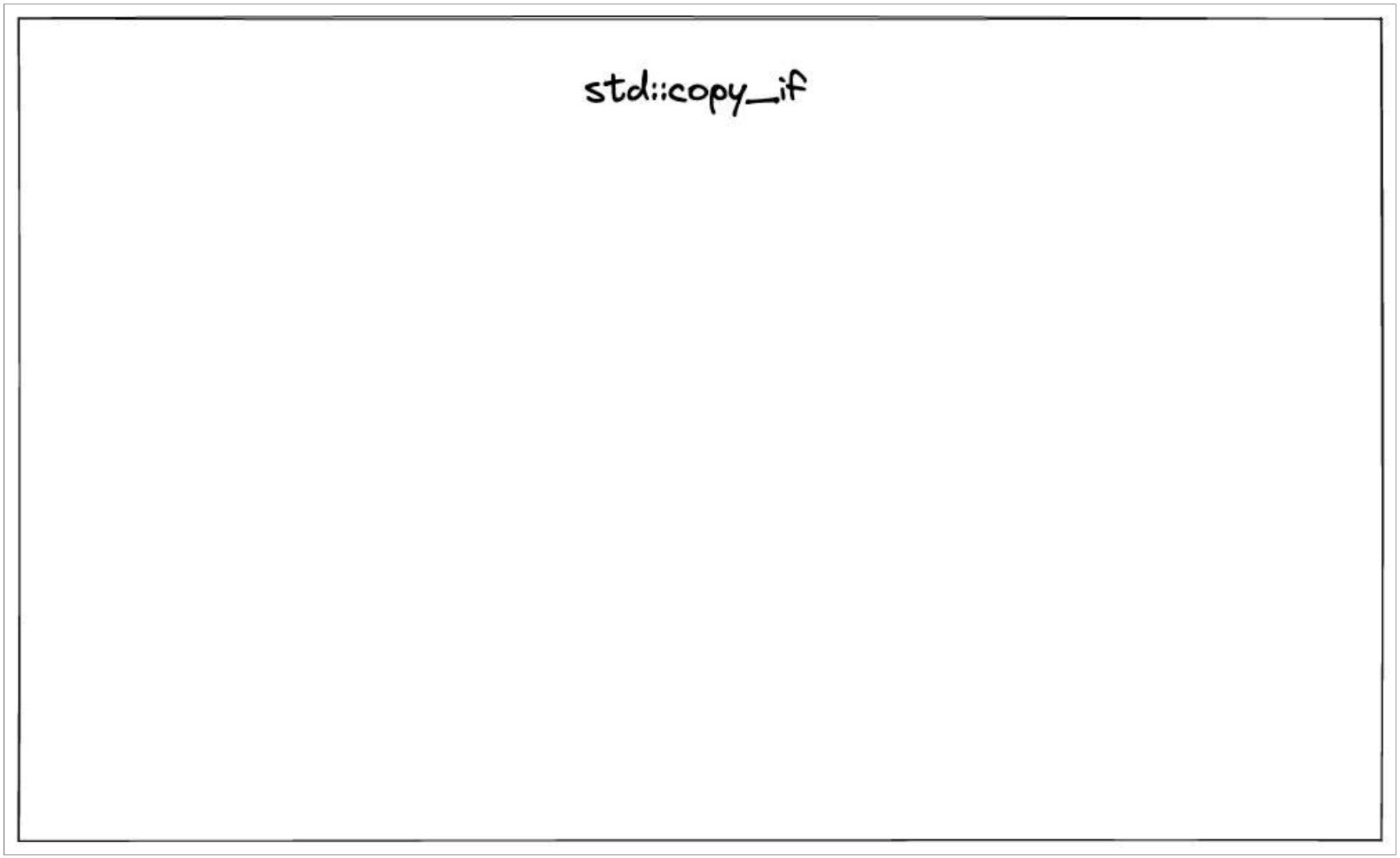

eve::compress_copy
in
m:
out

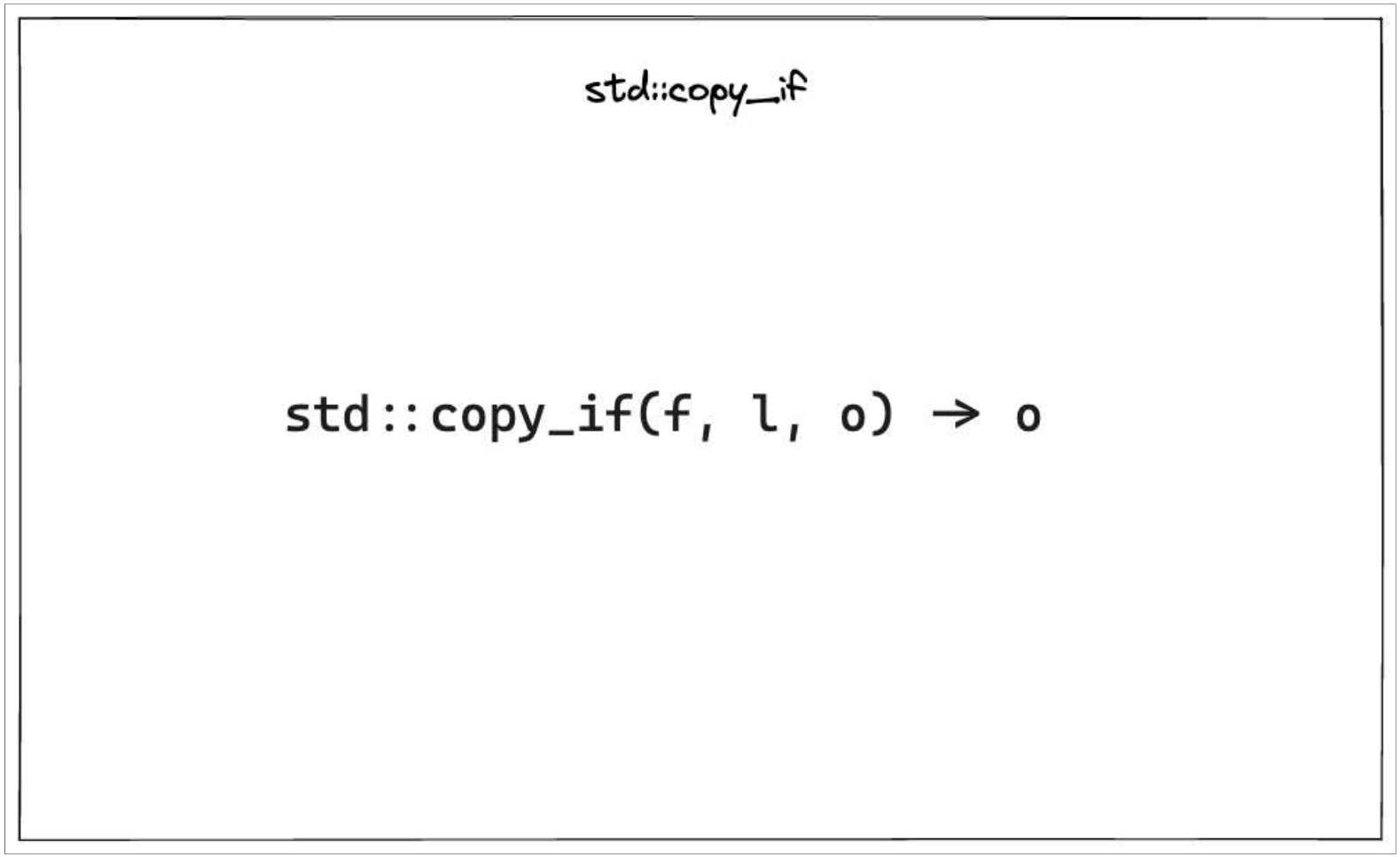

eve::compress_copy
in
m:
compress_copy [...][...](in, m, out)
out

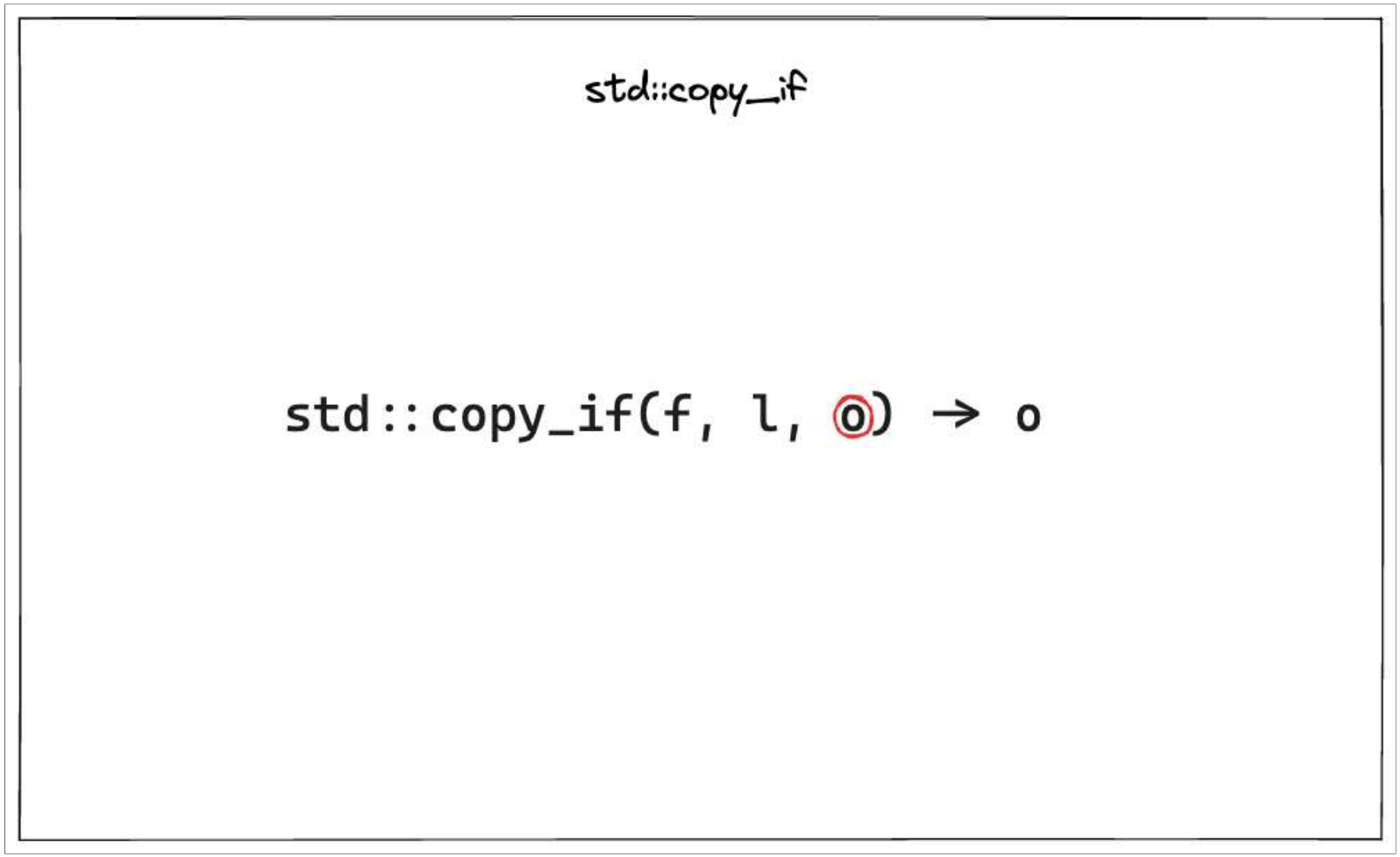

eve::compress_copy
in
m:
compress_copy [...][...](in, m, out) -> out1
out
out1

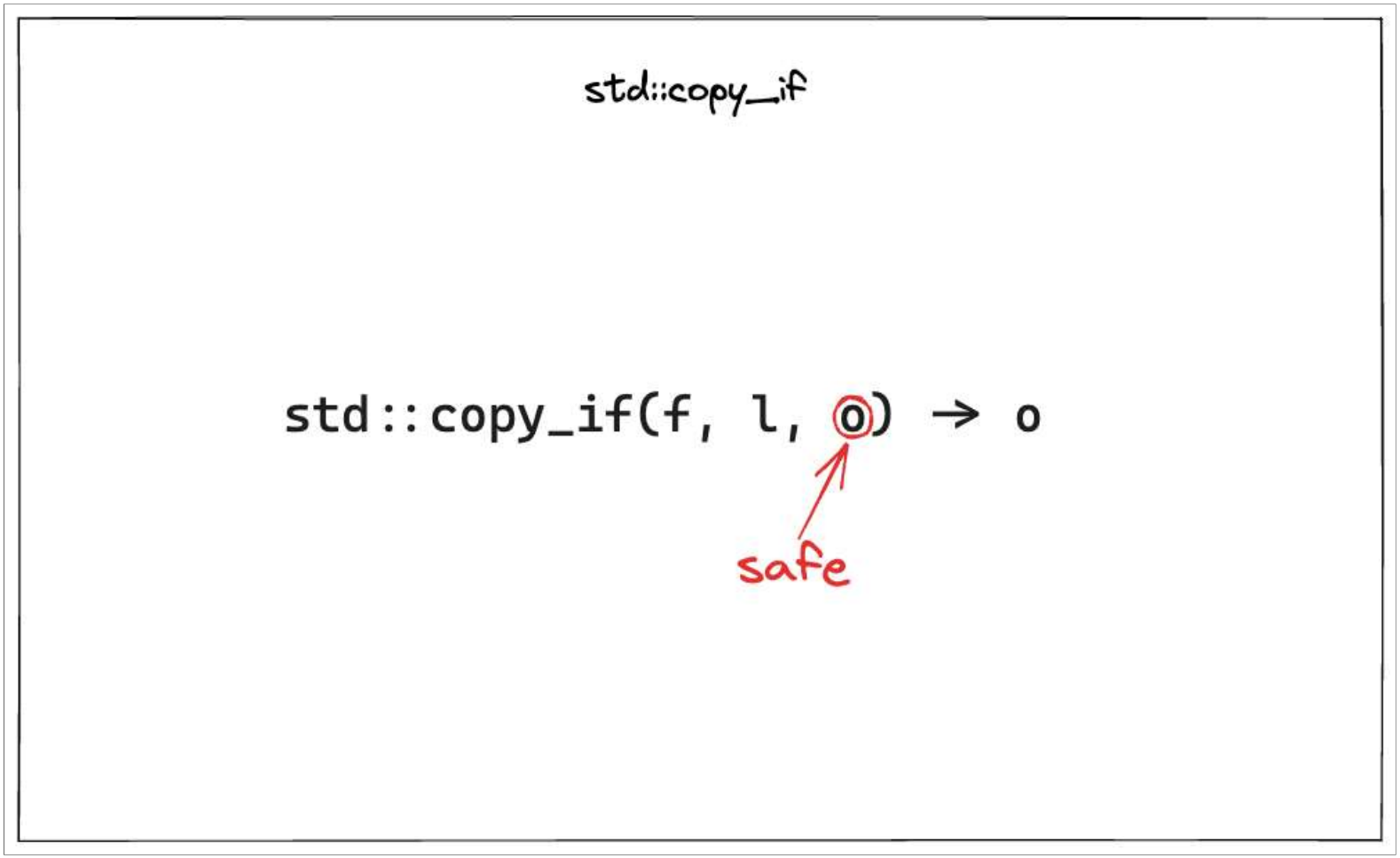

eve::compress_copy
in
m:
compress_copy [safe/unsafe][...](...) -> out1
out
out1

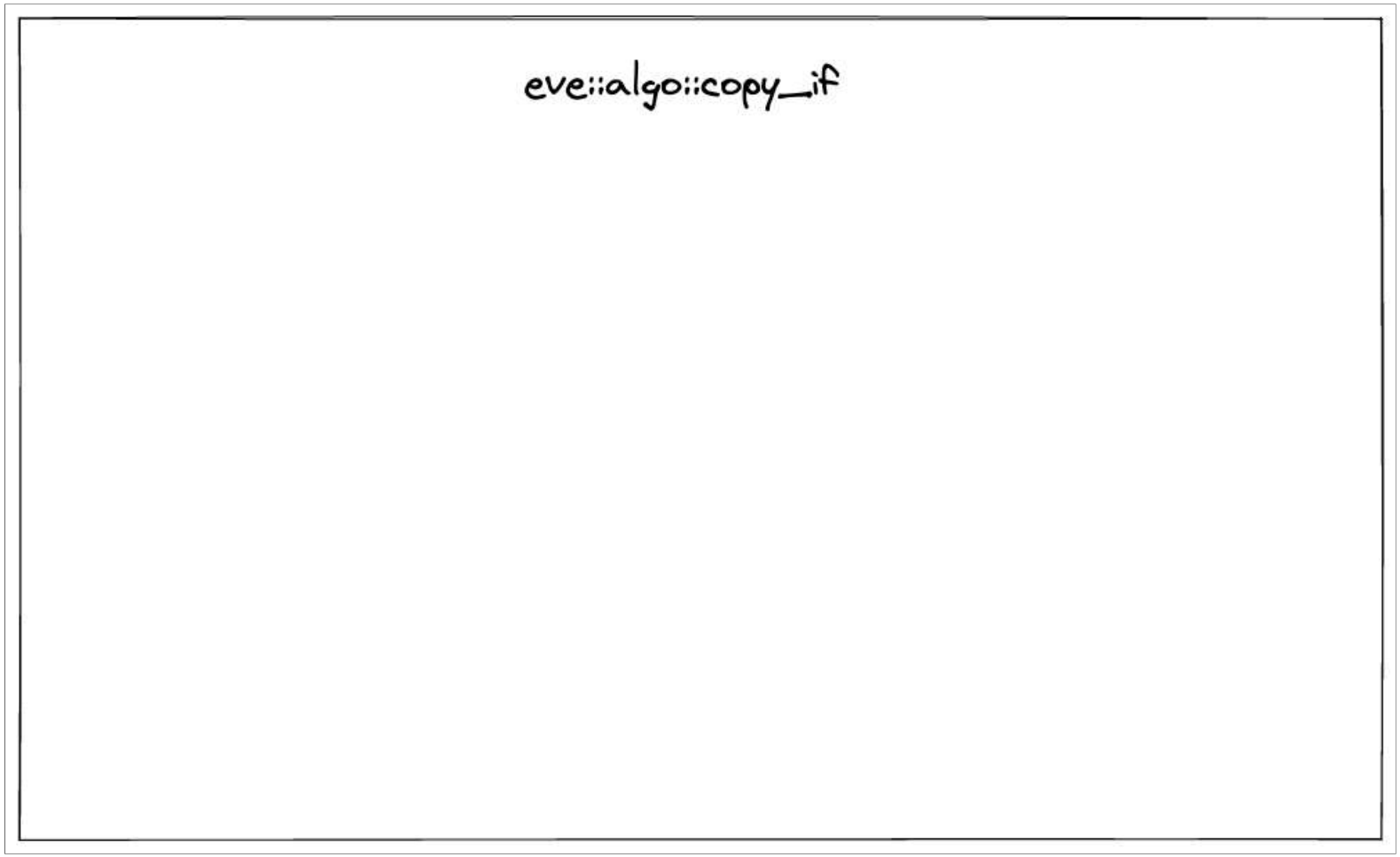

eve::compress_copy
in
m:
compress_copy [...][sparse/dense](...)
-> out1
out
out1

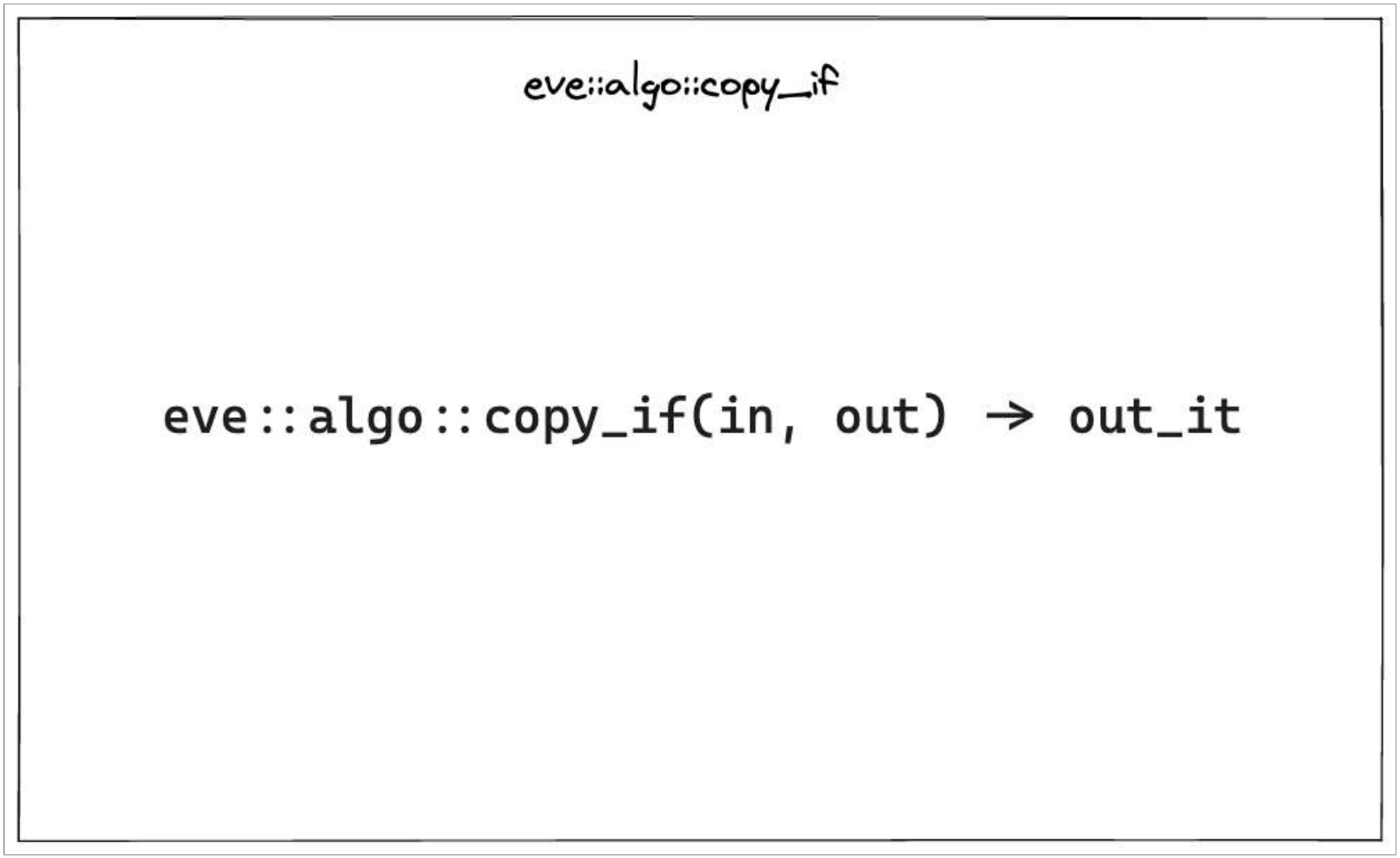

std::copy_if

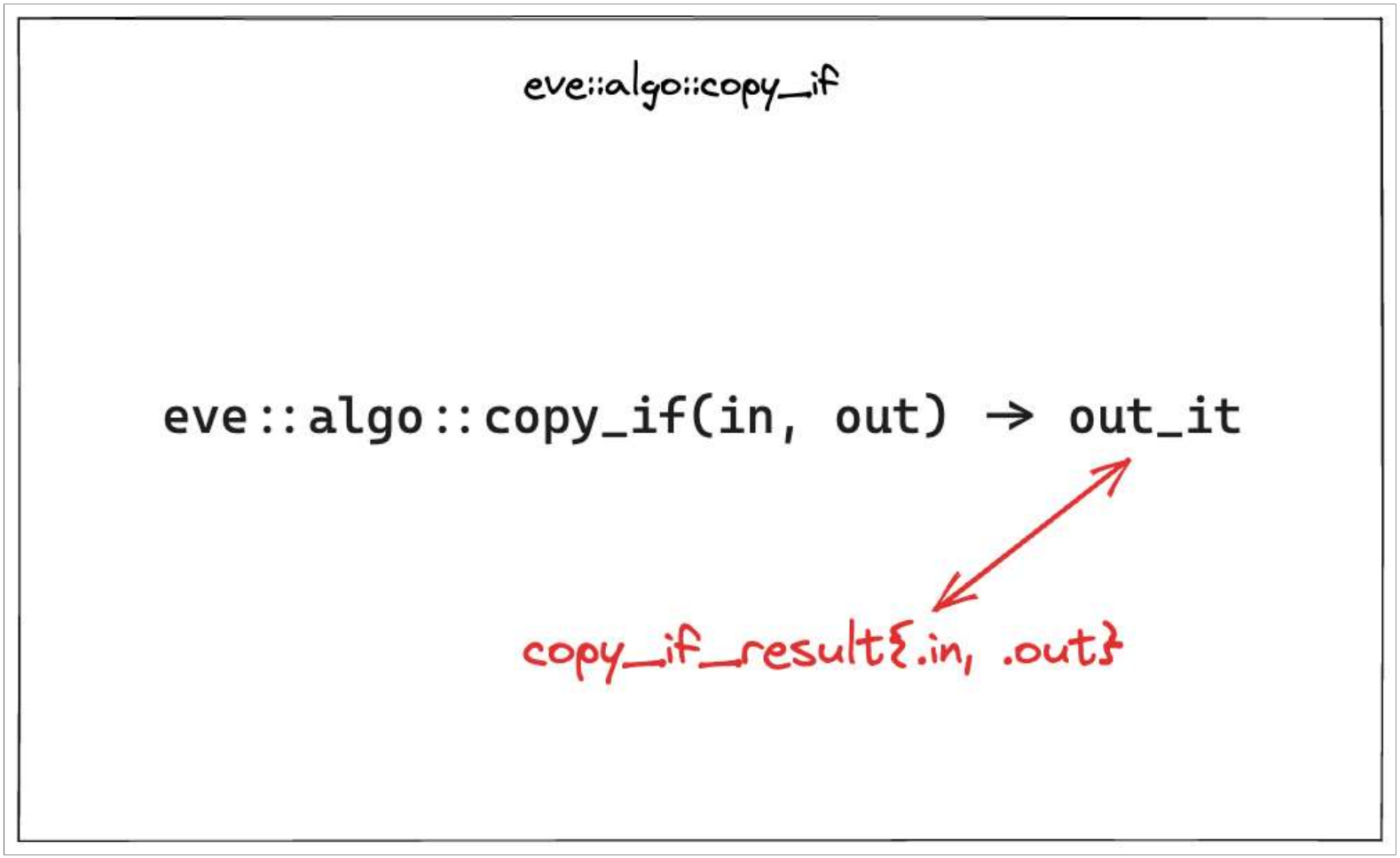

std::copy_if
std::copy_if(f, l, o) → o


std::copy_if
std::copy_if(f, l, o) → o


std::copy_if
std::copy_if(f, l, o) → o
safe

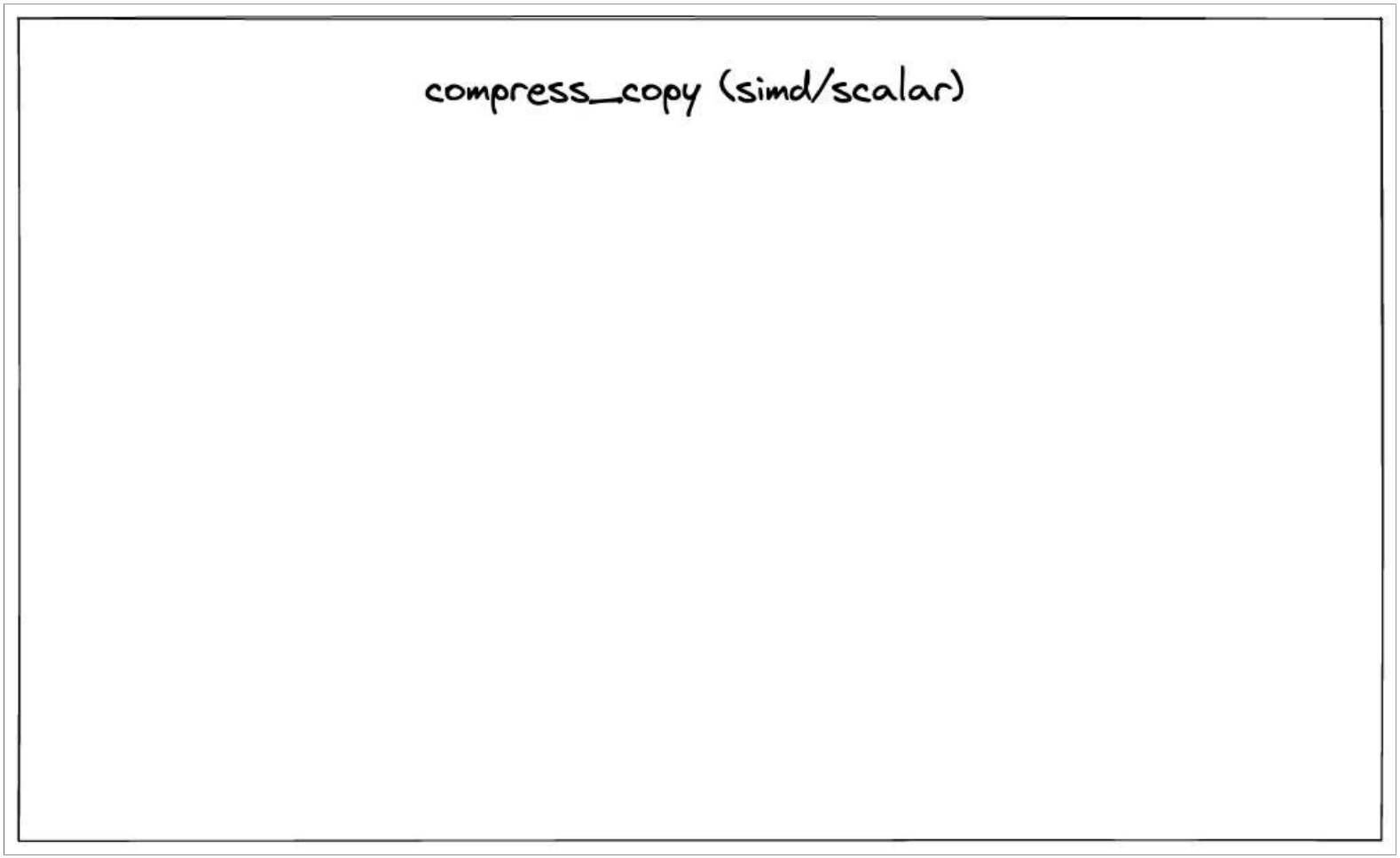

eve::algo::copy_if

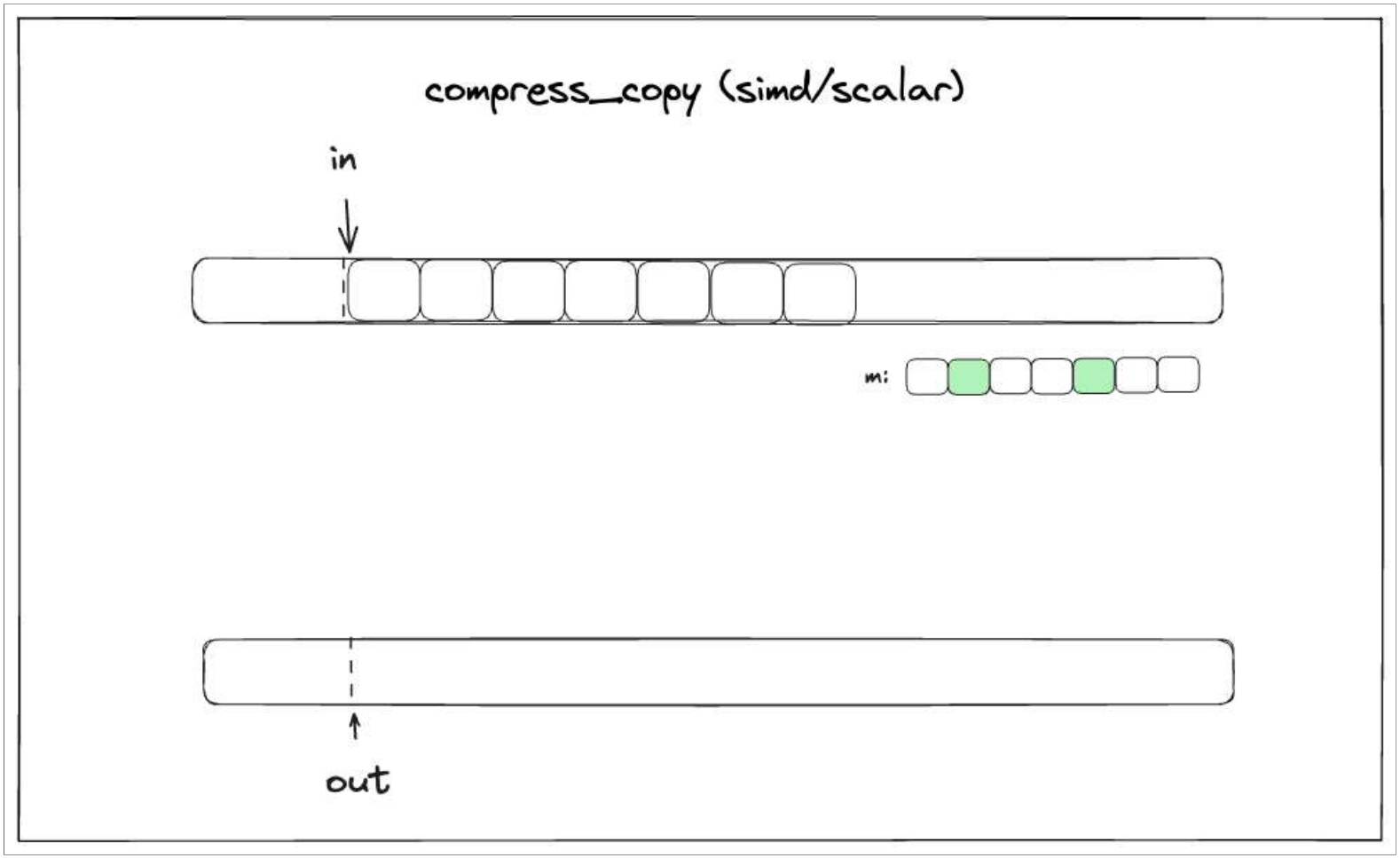

eve::algo::copy_if
eve::algo::copy_if(in, out) → out_it

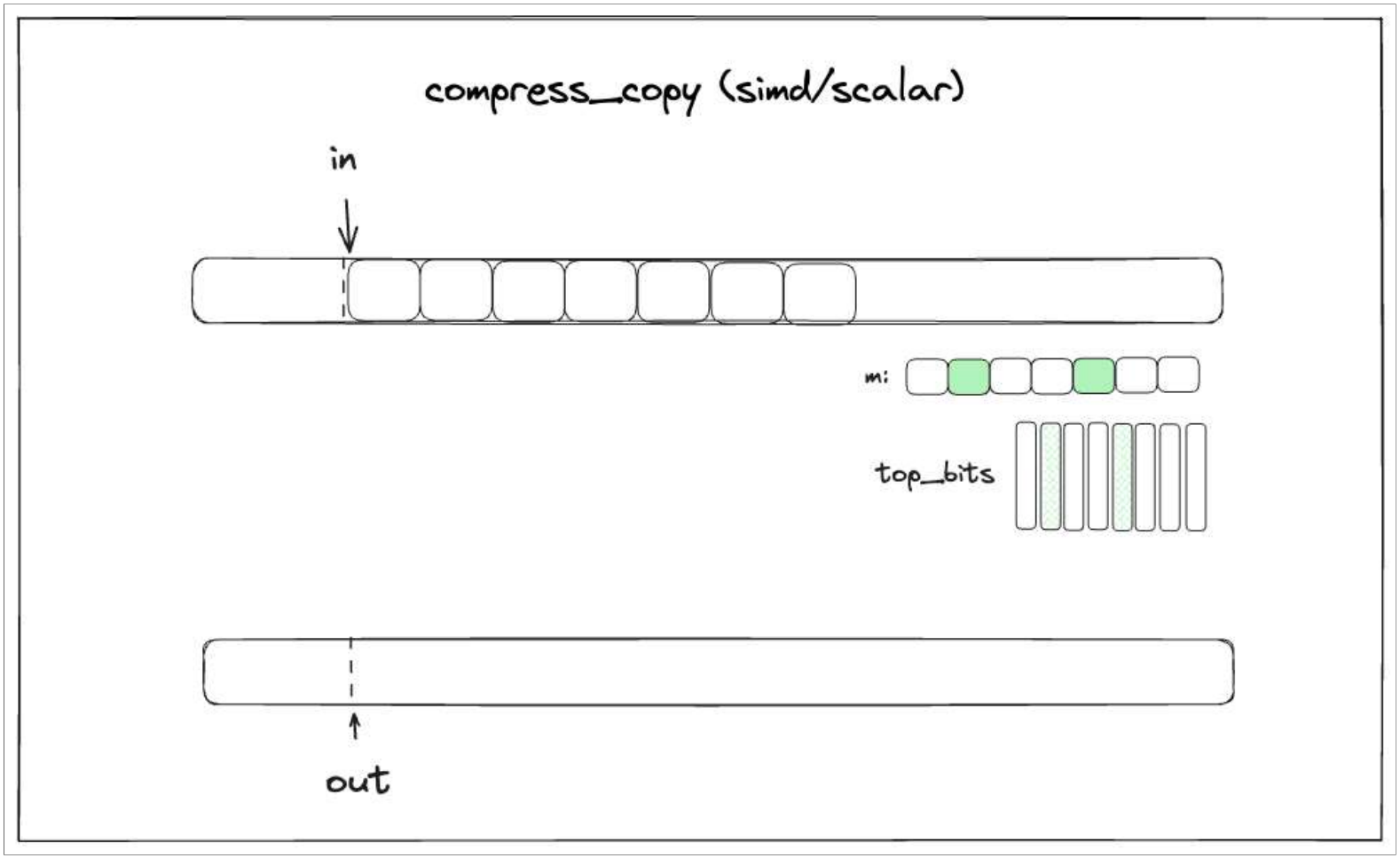

eve::algo::copy_if
eve::algo::copy_if(in, out) → out_it
copy_if_result{.in, .out}

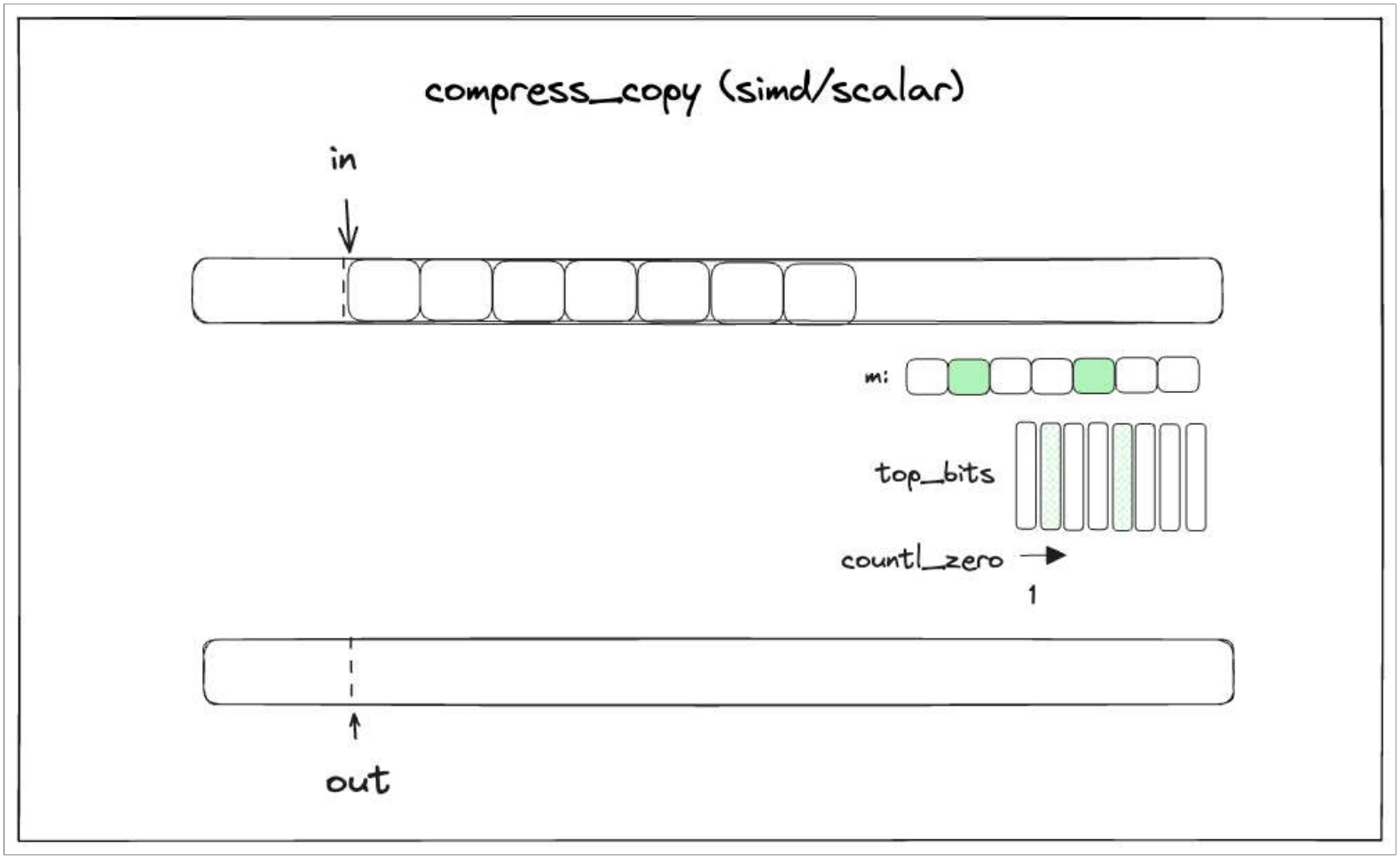
COMPRESS_COPY (PURE SIMD)

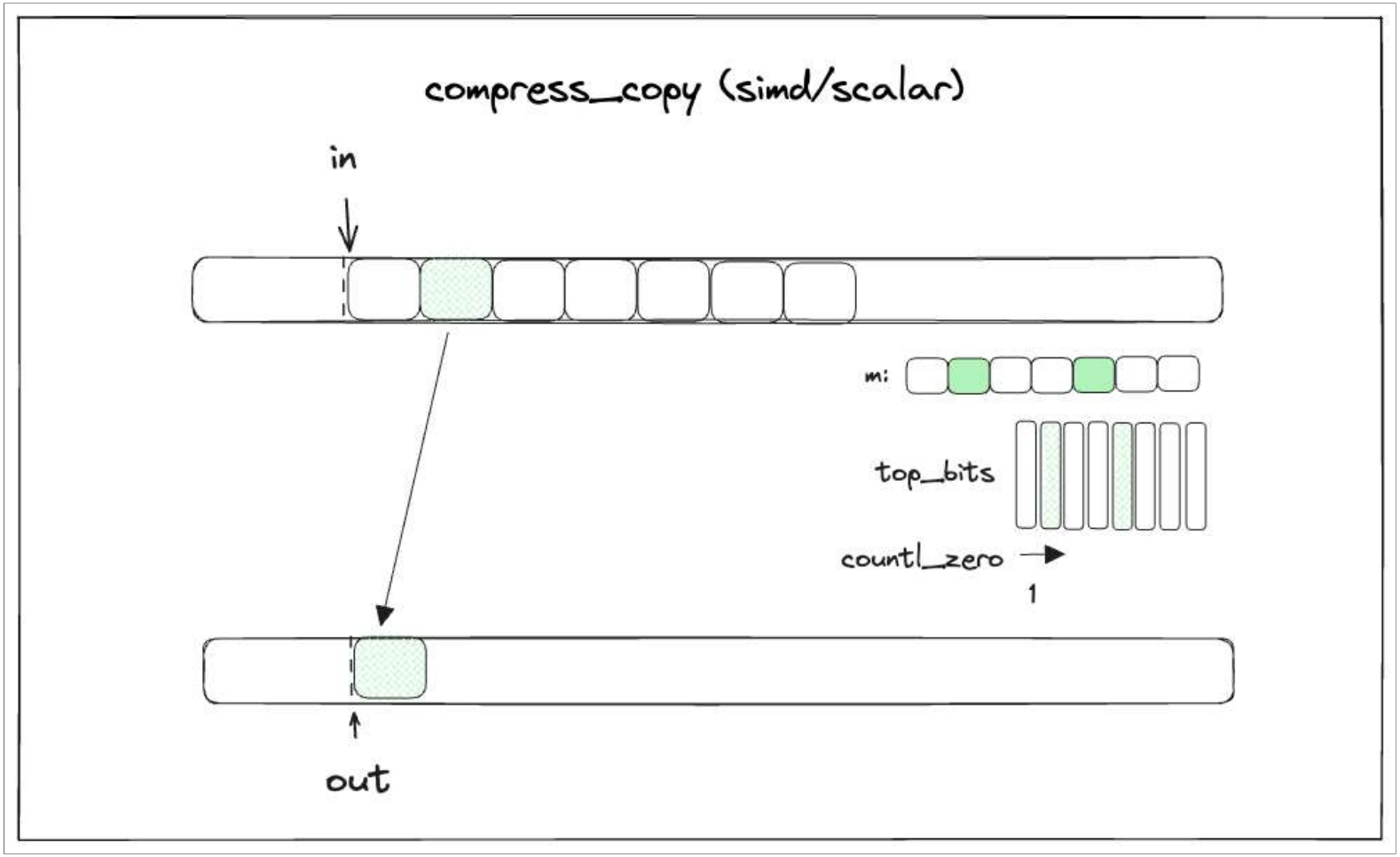
- compress instructions (avx512, sve)
- tiny lookup tables (@aqrit)
- bmi2 (Peter Cordes)
- switch + shuffle (@Z boson)

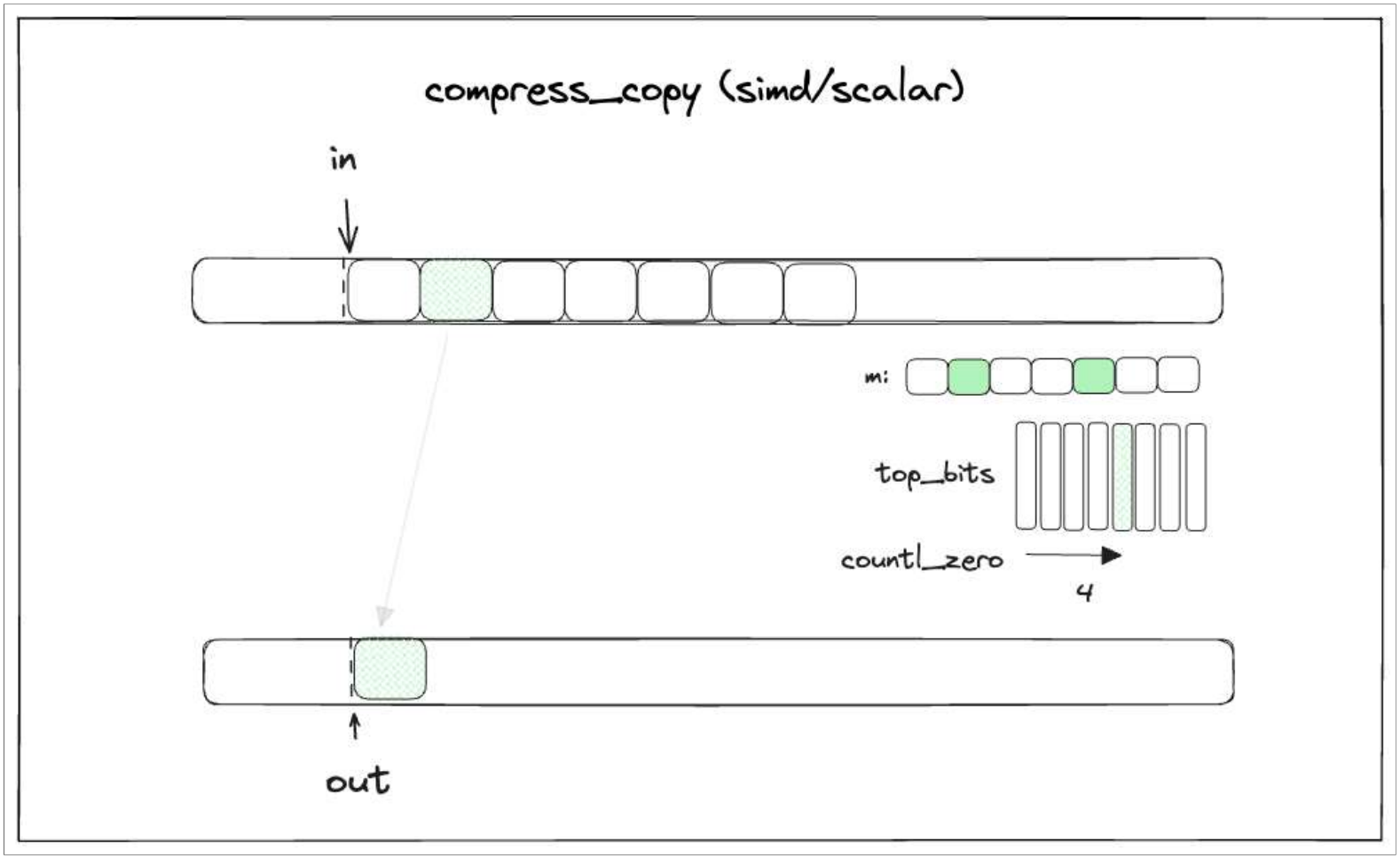

compress_copy (simd/scalar)

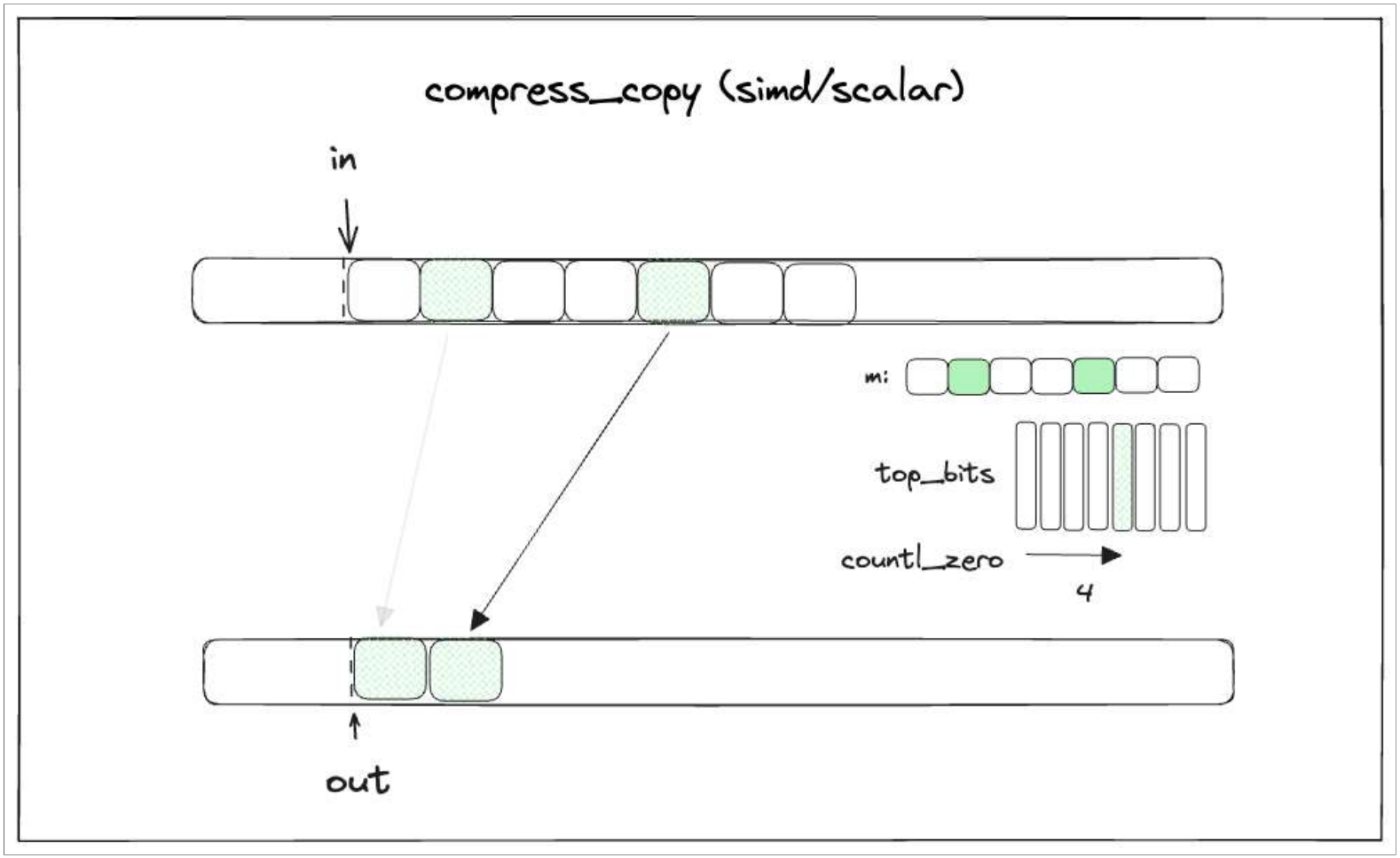

compress_copy (simd/scalar)
in
m:
out


compress_copy (simd/scalar)
in
m:
top_bits
out


compress_copy (simd/scalar)
in
m:
top_bits
countl_zero
1
out


compress_copy (simd/scalar)
in
m:
top_bits
countl_zero
1
out

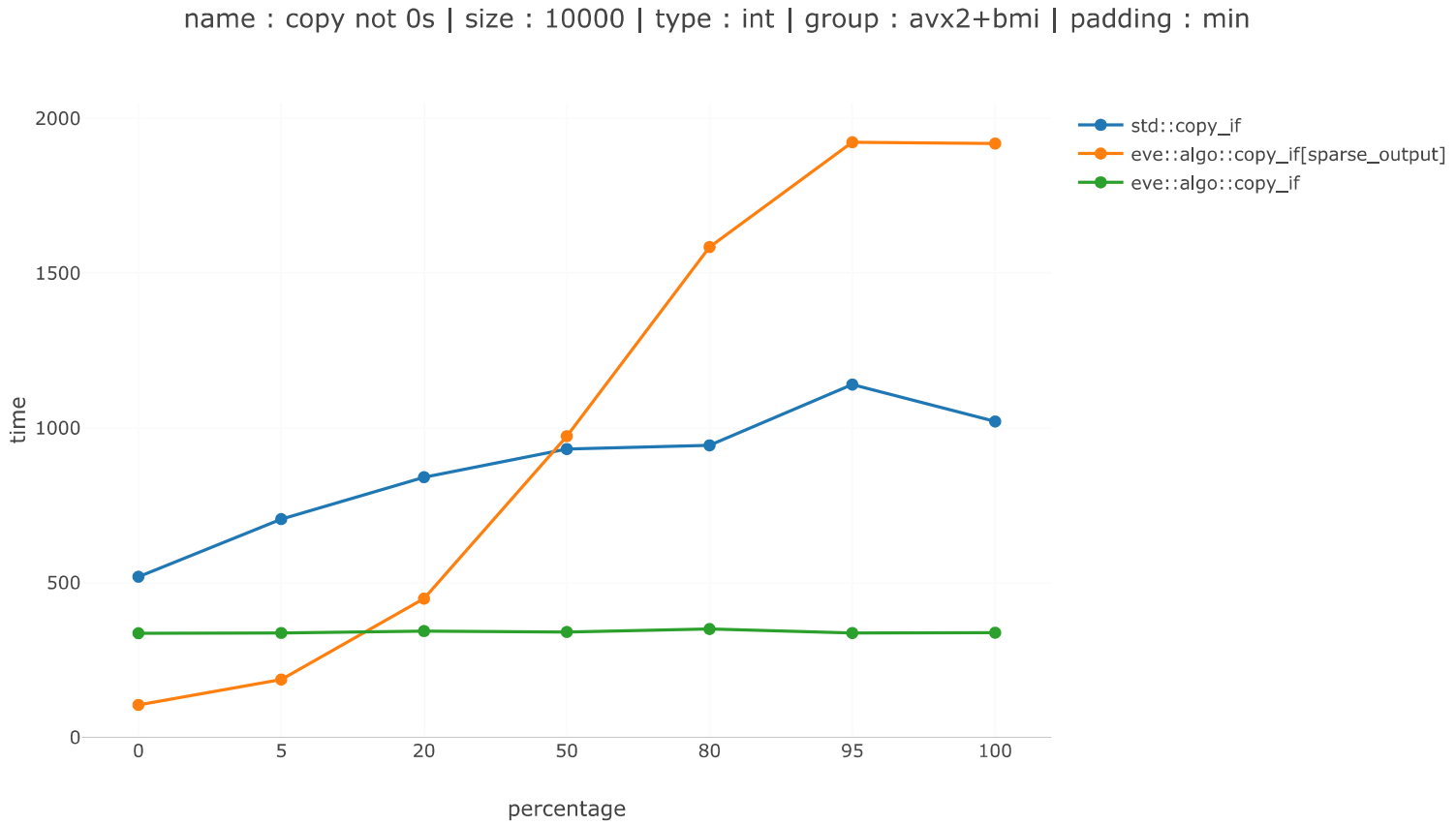

compress_copy (simd/scalar)
in
m:
top_bits
countl_zero
4
out


compress_copy (simd/scalar)
in
m:
top_bits
countl_zero
4
out

COMPRESS_COPY (SIMD/SCALAR)

- Peter Cordes, Ilya Albrecht

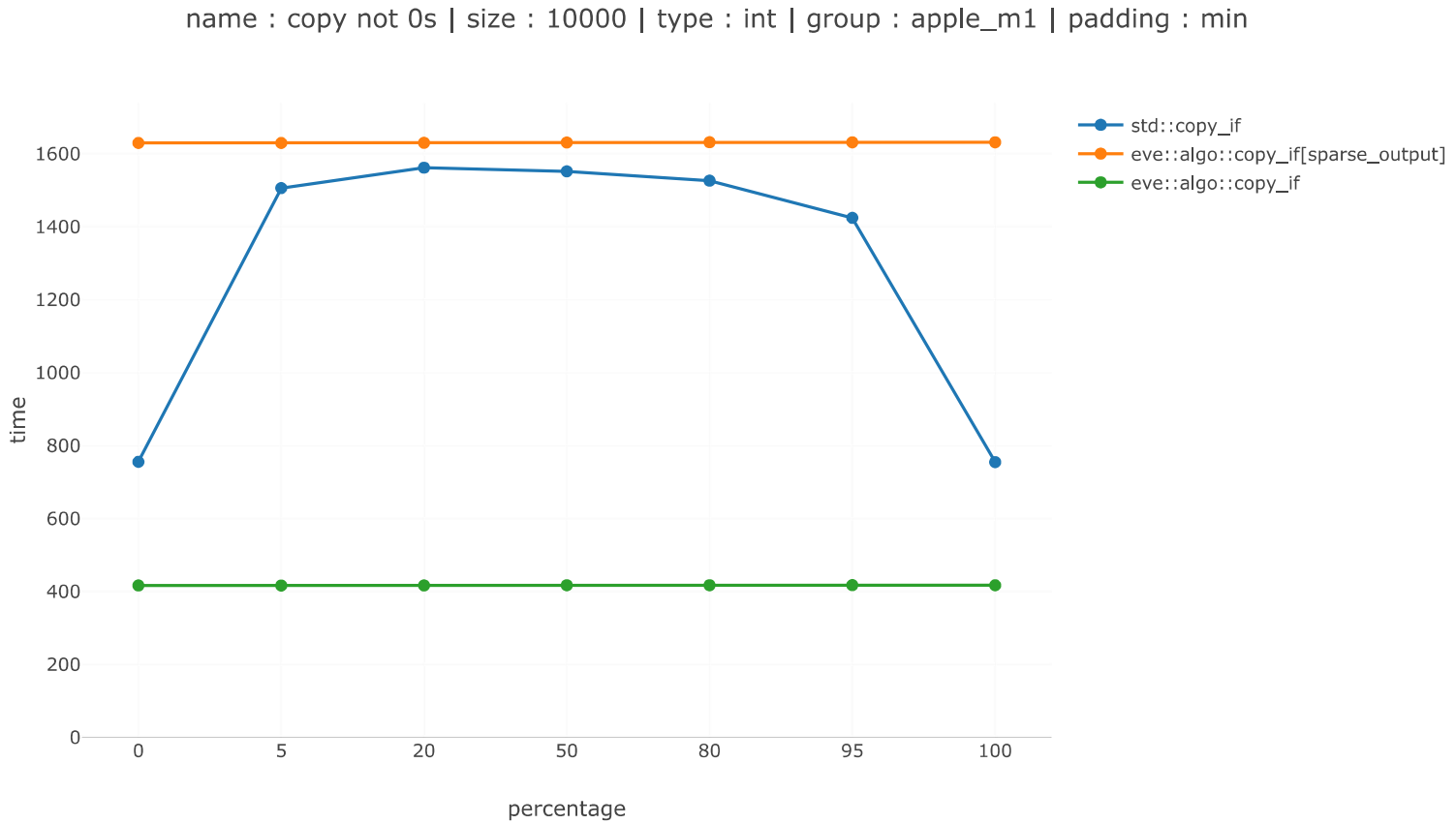
COPY_IF, INT


name : copy not 0s | size : 10000 | type : int | group : avx2+bmi | padding : min
2000
1500
1000
500
0
time
0
5
20
50
80
95
100
percentage
std::copy_if
eve::algo::copy_if[sparse_output]
eve::algo::copy_if

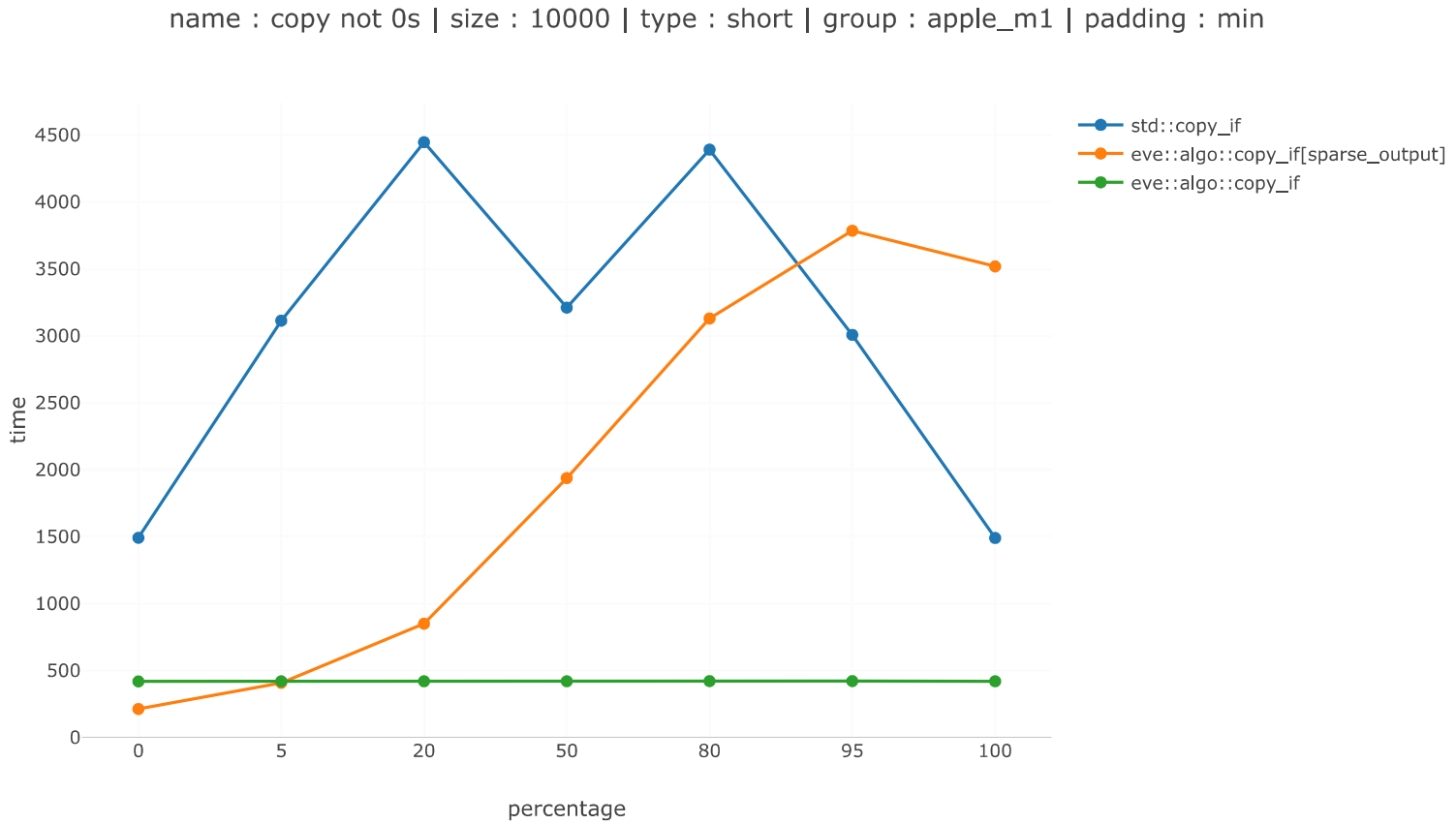
COPY_IF, SHORT


name : copy not 0s | size : 10000 | type : short | group : avx2+bmi | padding : min
time
10k
8k
6k
4k
2k
0
0
5
20
50
80
95
100
percentage
std::copy_if
eve::algo::copy_if[sparse_output]
eve::algo::copy_if

COPY_IF, INT, M1


name : copy not 0s | size : 10000 | type : int | group : apple_m1 | padding : min
time
0
200
400
600
800
1000
1200
1400
1600
0
5
20
50
80
95
100
percentage
std::copy_if
eve::algo::copy_if[sparse_output]
eve::algo::copy_if

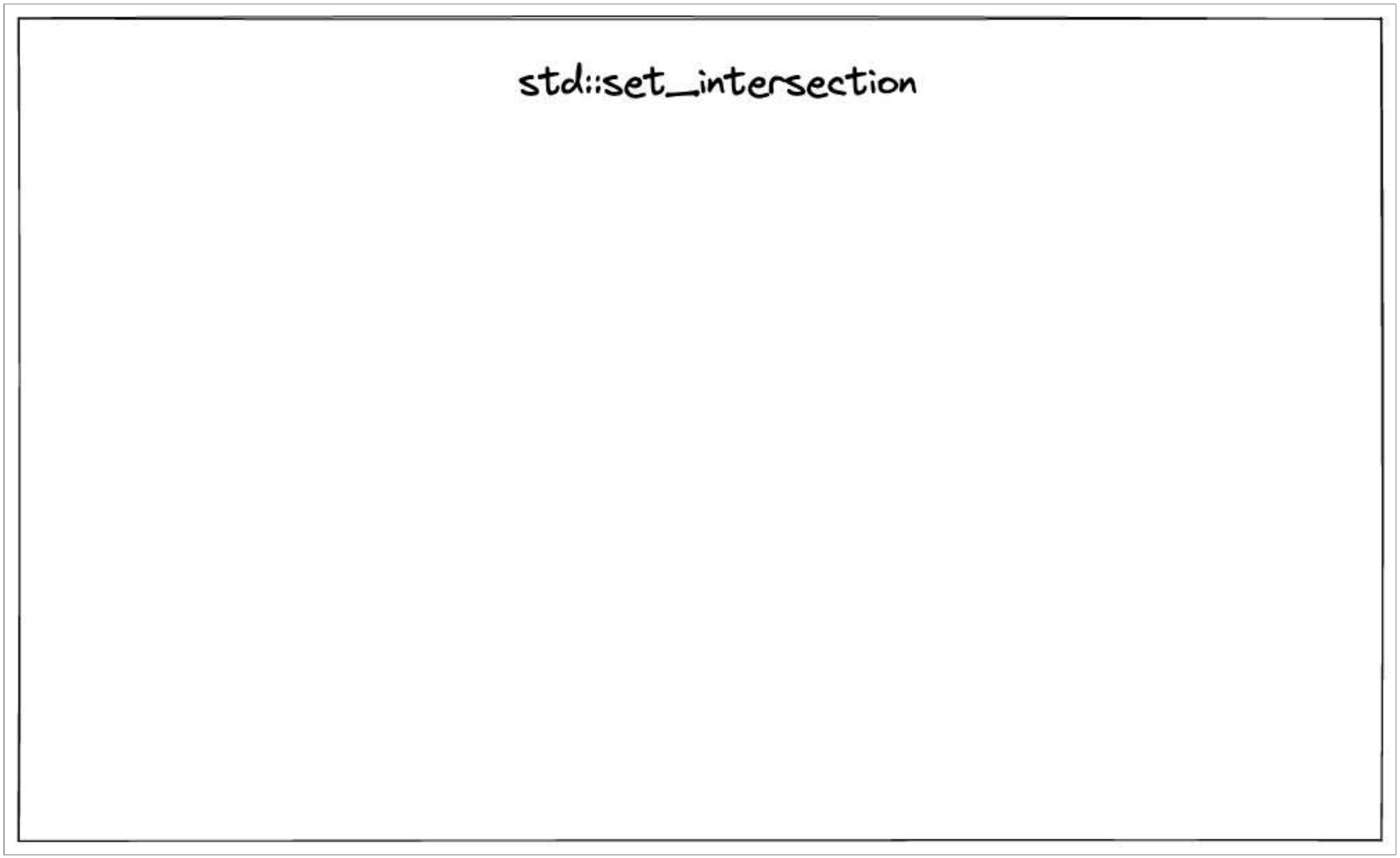
COPY_IF, SHORT, M1

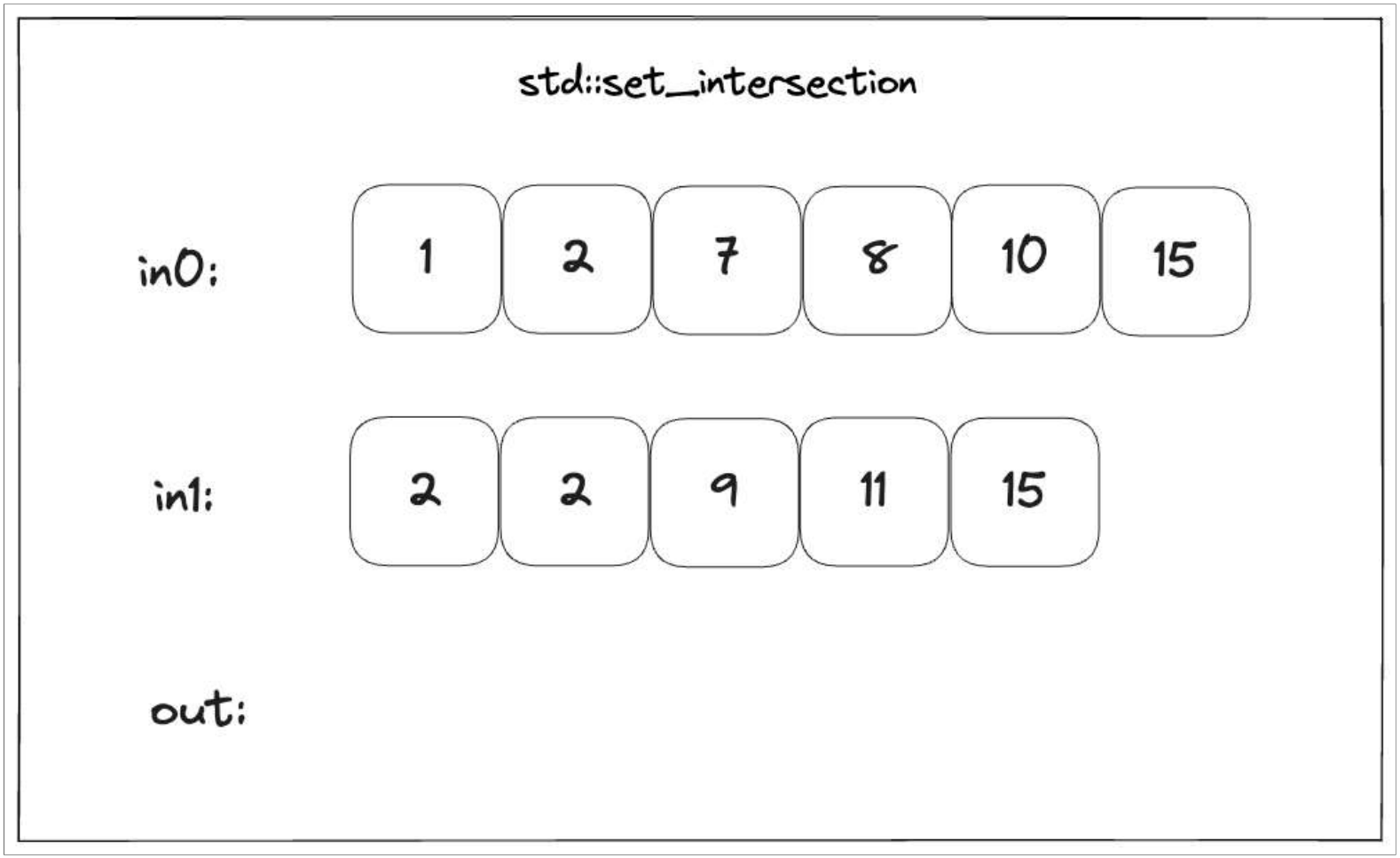

name : copy not 0s | size : 10000 | type : short | group : apple_m1 | padding : min
time
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
5
20
50
80
95
100
percentage
std::copy_if
eve::algo::copy_if[sparse_output]
eve::algo::copy_if

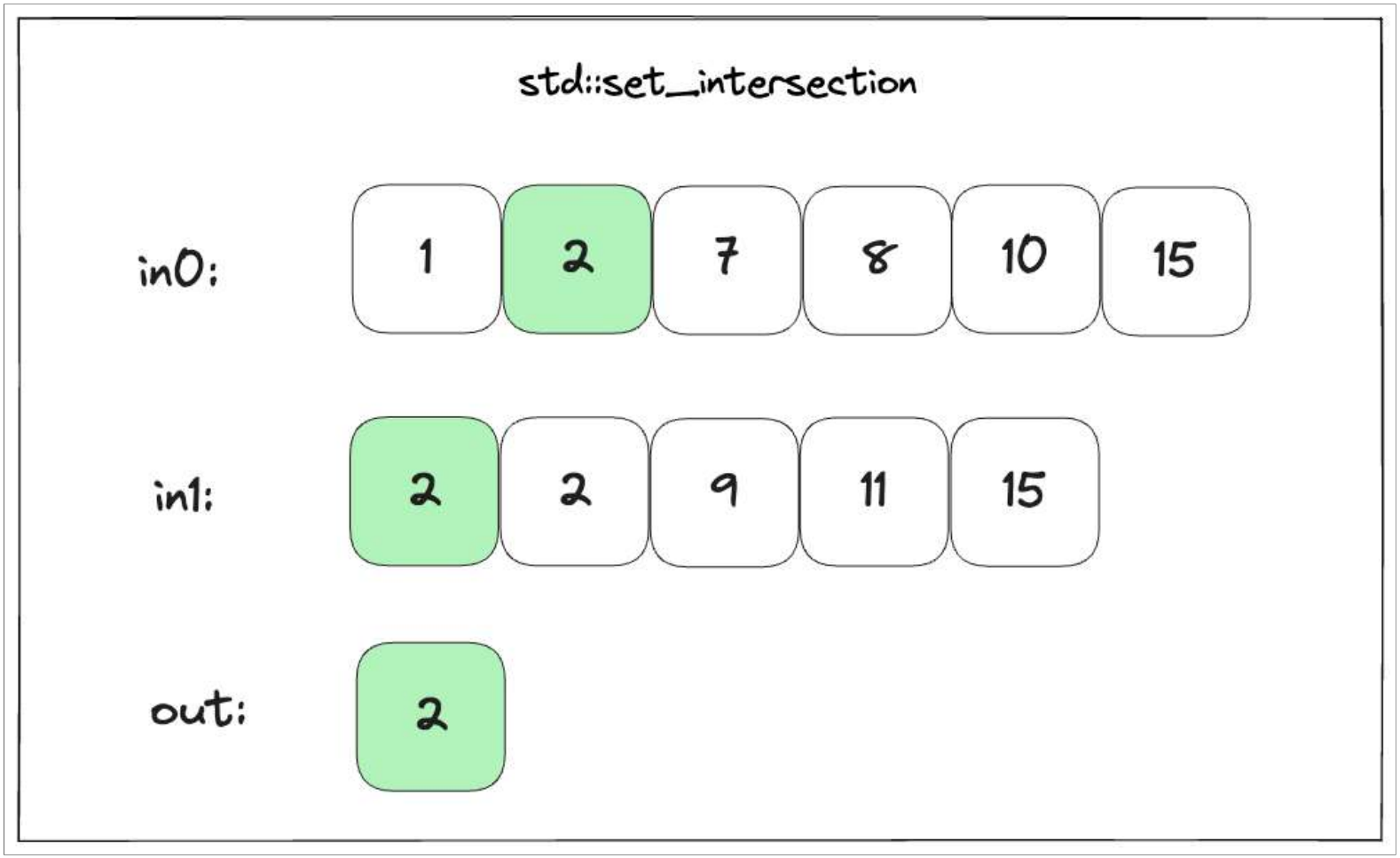
PROPOSAL P2664R3

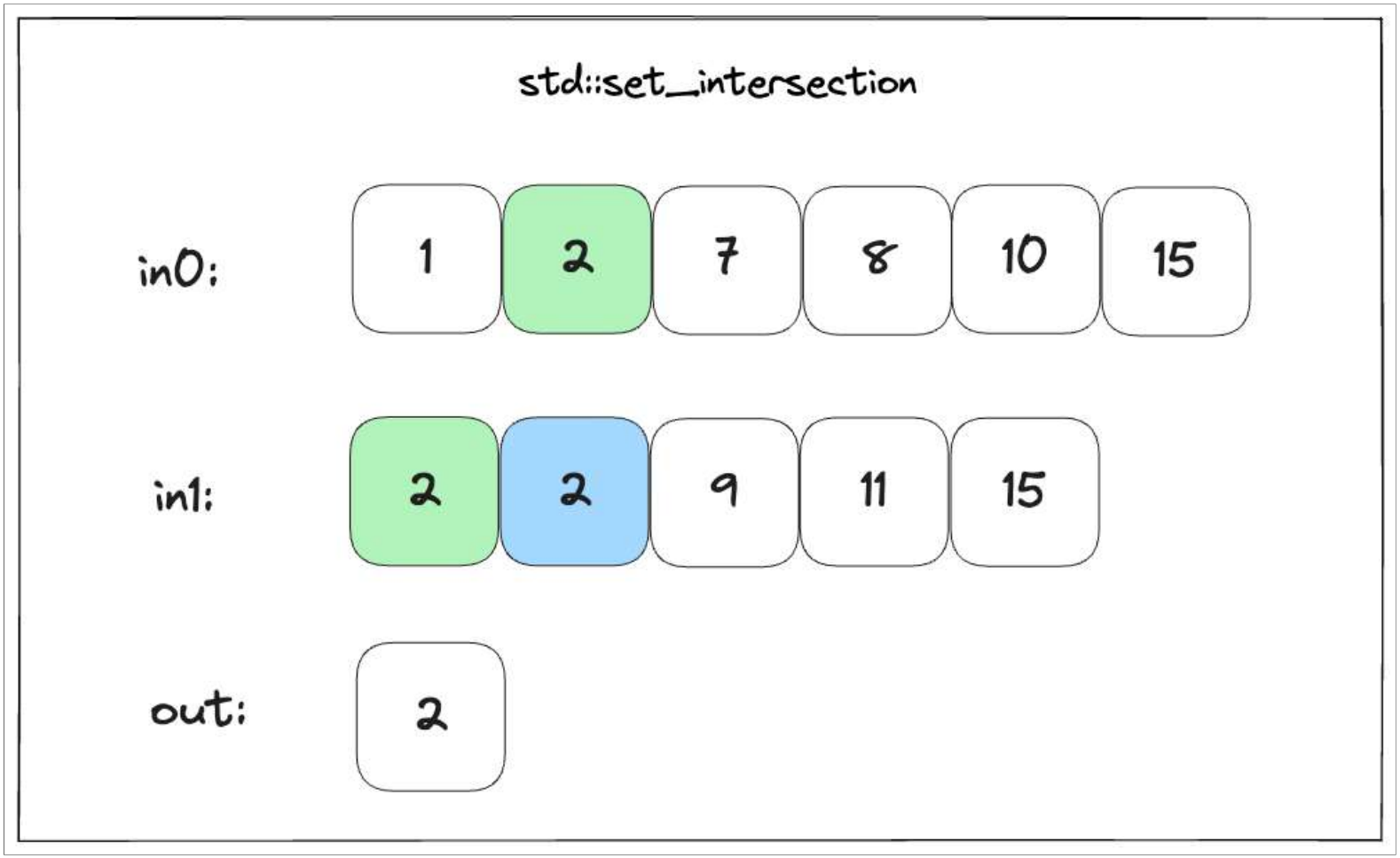
```
template<typename T, typename Abi>
constexpr simd<T, Abi>
compress(const simd<T, Abi>& value,
         const simd_mask<T, MAbi>& mask);
```

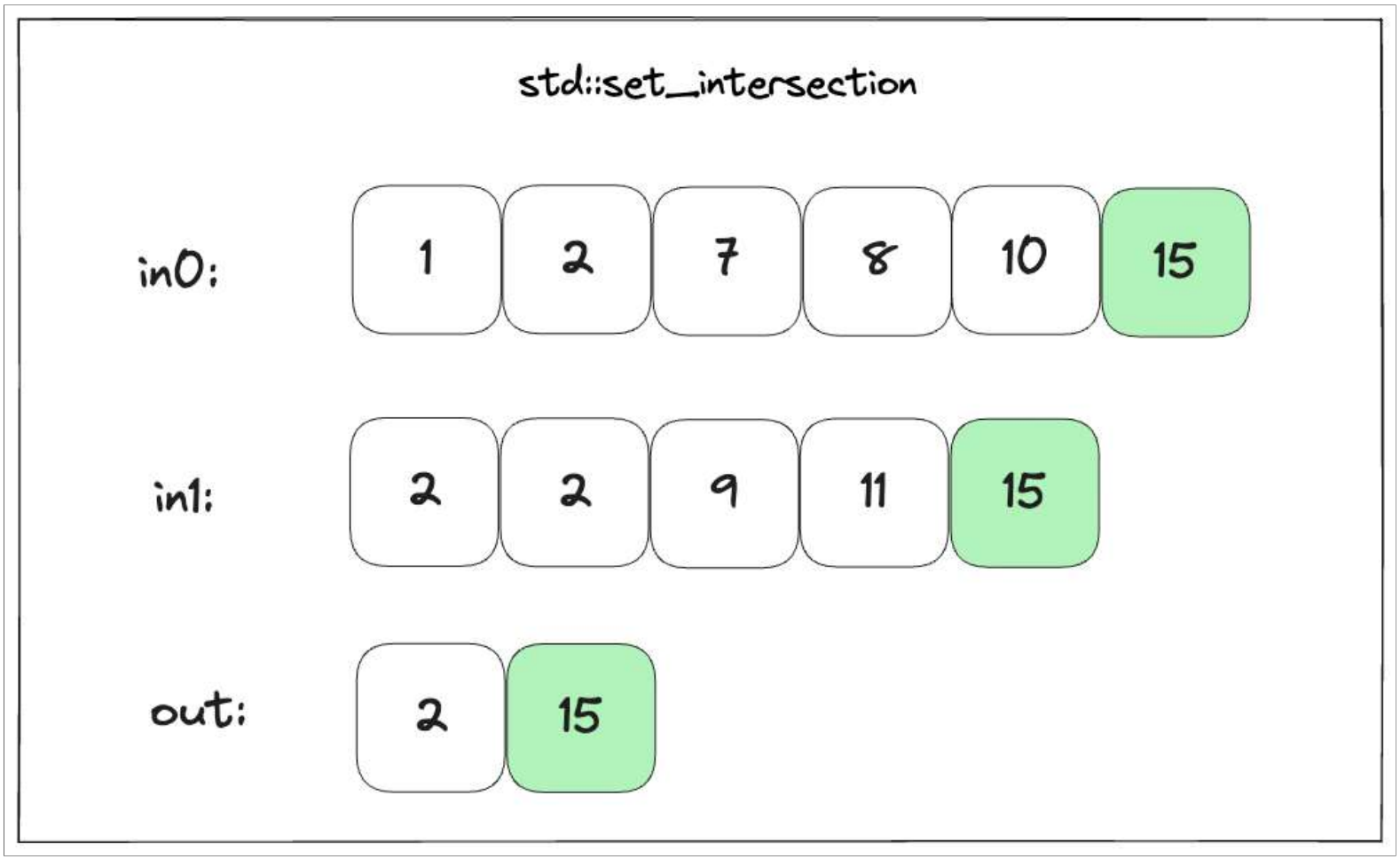
SET_INTERSECTION

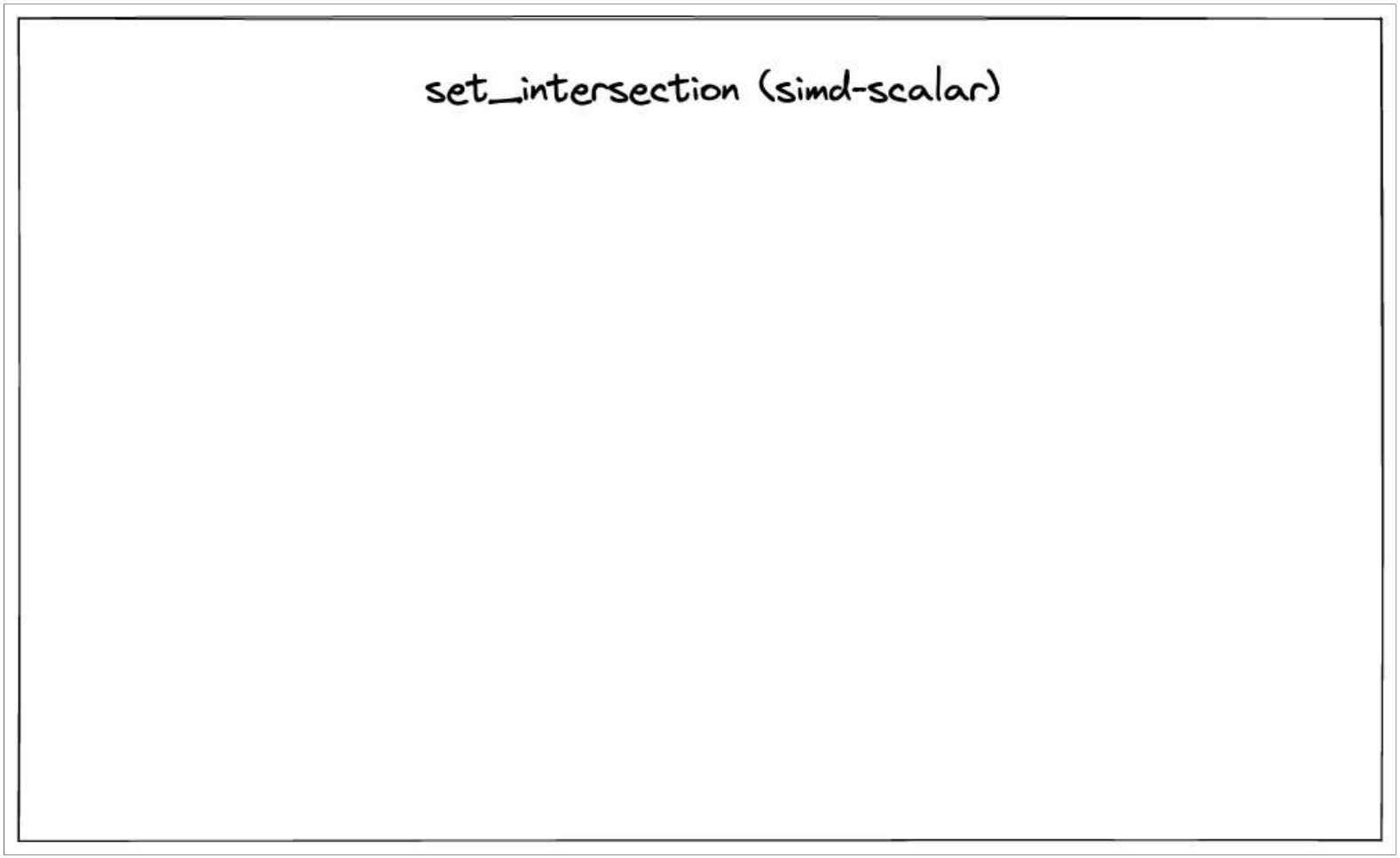

std::set_intersection

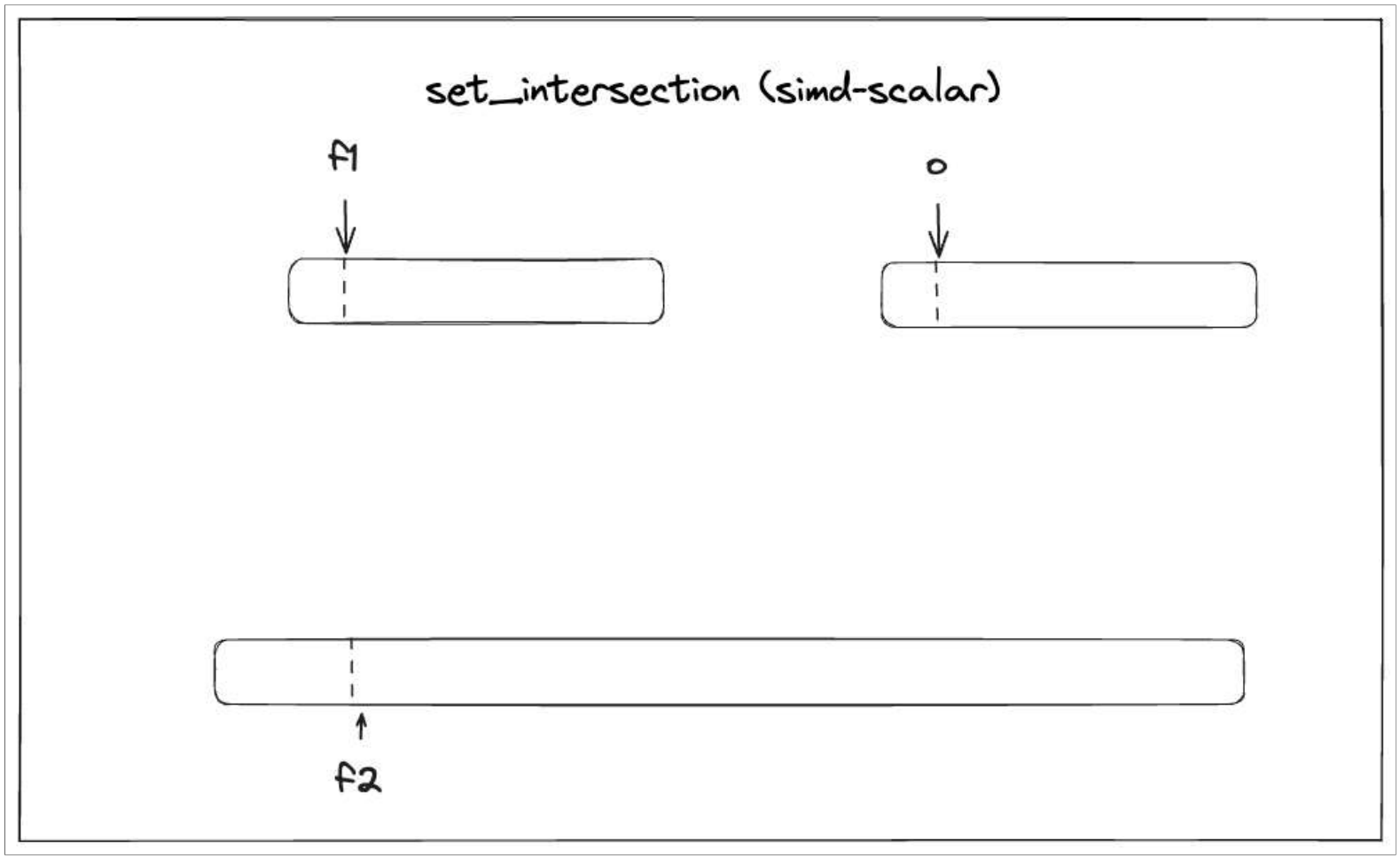

std::set_intersection
in0:
1
2
7
8
10
15
in1:
2
2
9
11
15
out:

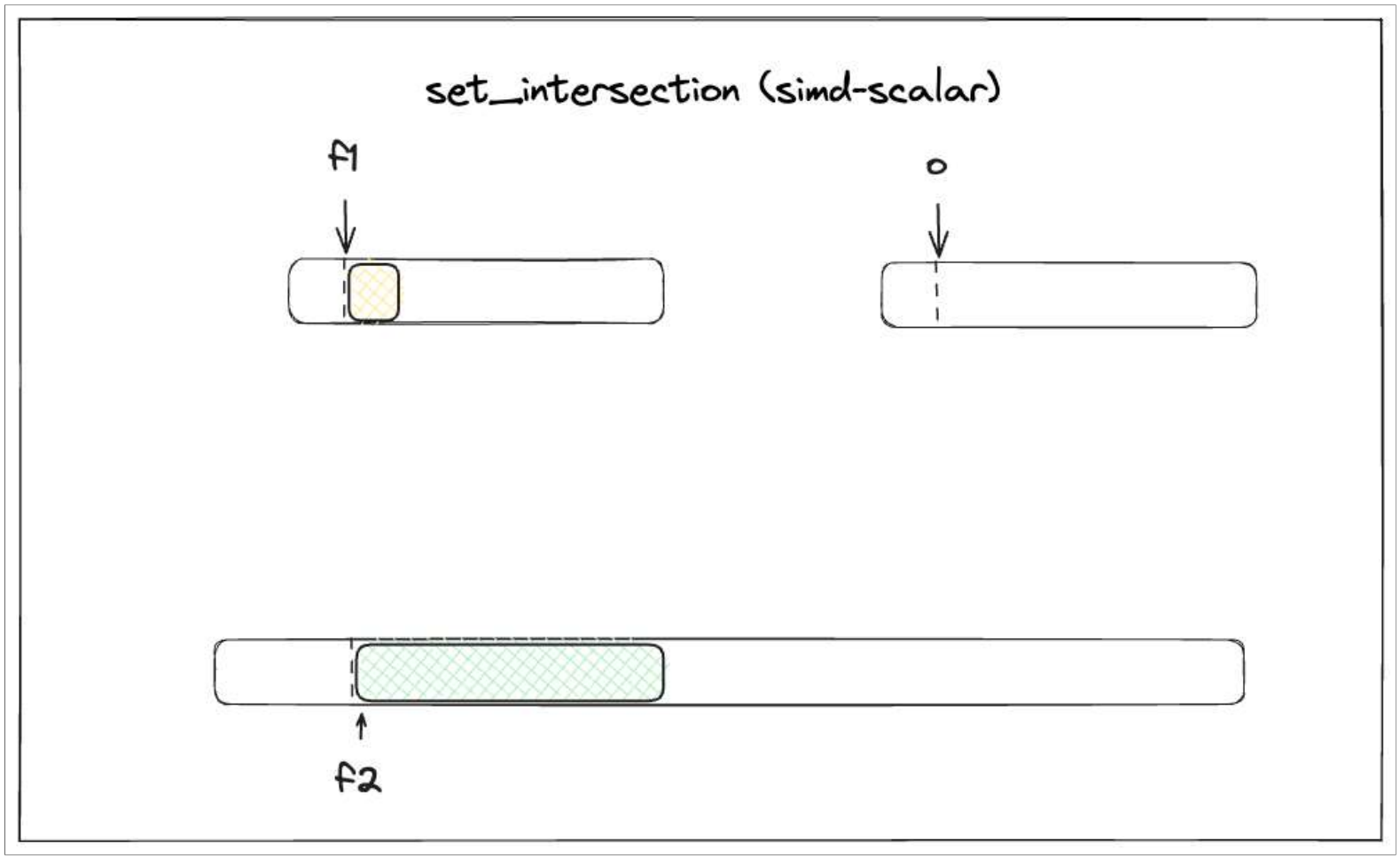

std::set_intersection
in0:
1
2
7
8
10
15
in1:
2
2
9
11
15
out:
2

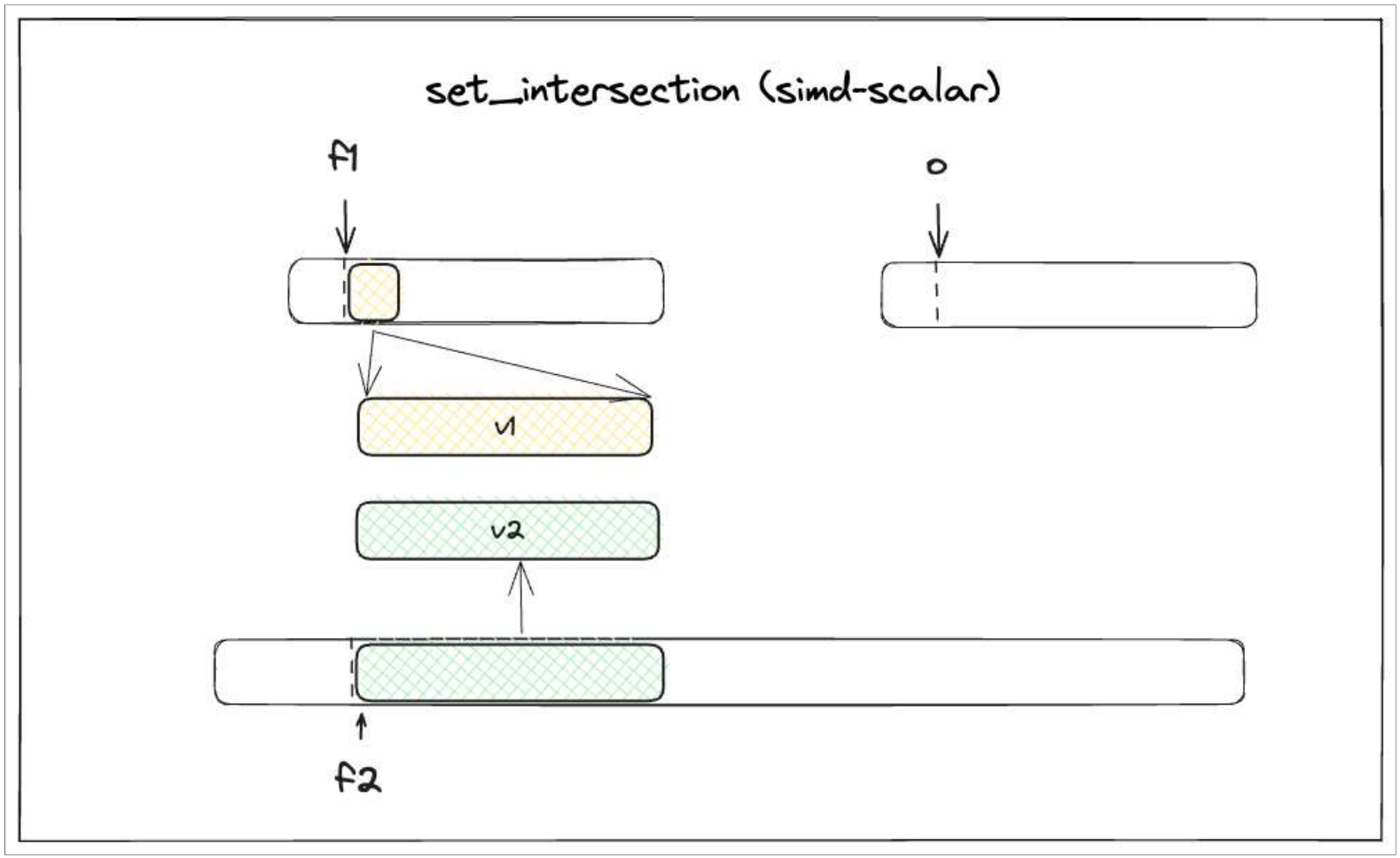

std::set_intersection
in0:
1
2
7
8
10
15
in1:
2
2
9
11
15
out:
2

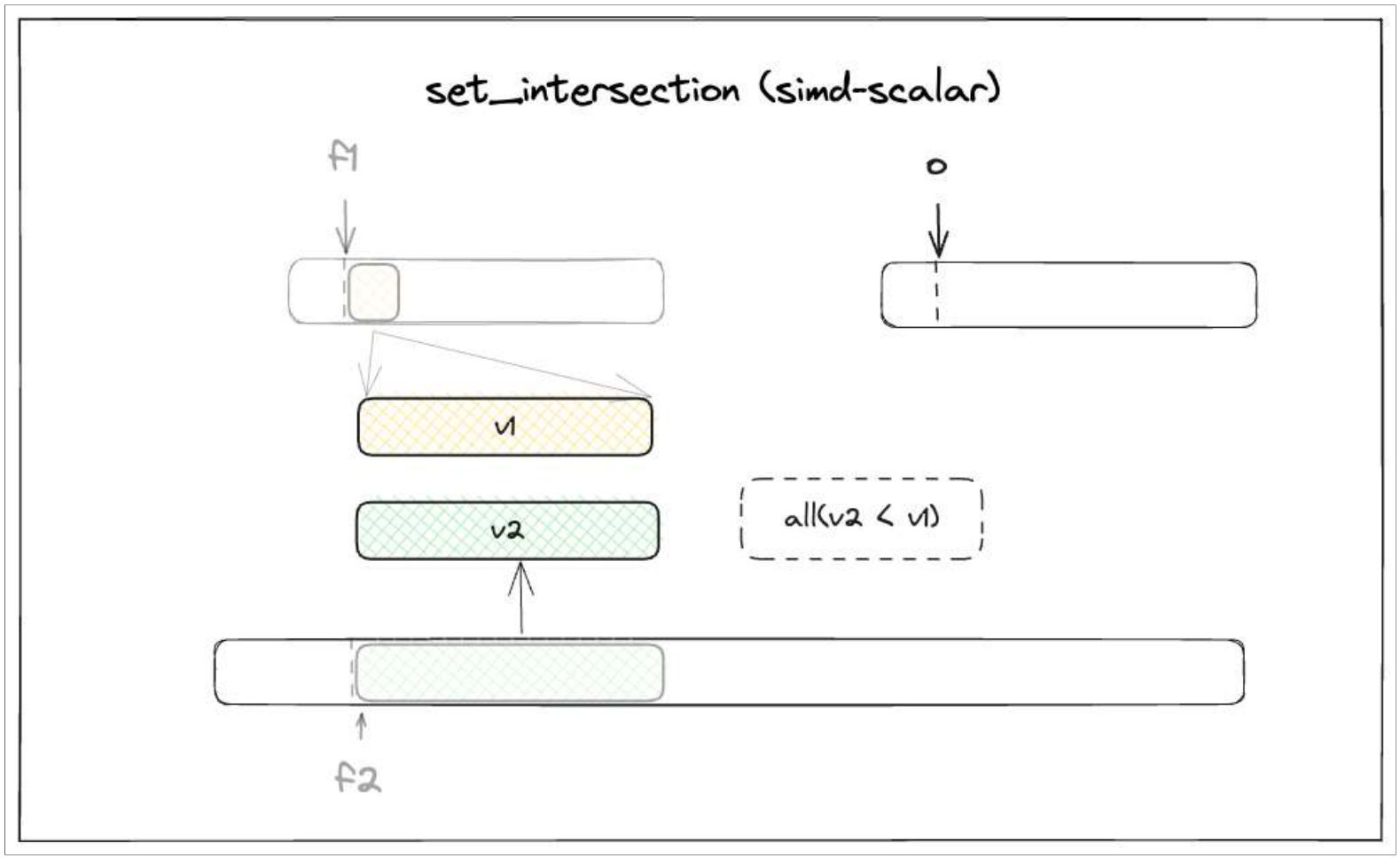

std::set_intersection
in0:
1
2
7
8
10
15
in1:
2
2
9
11
15
out:
2
15

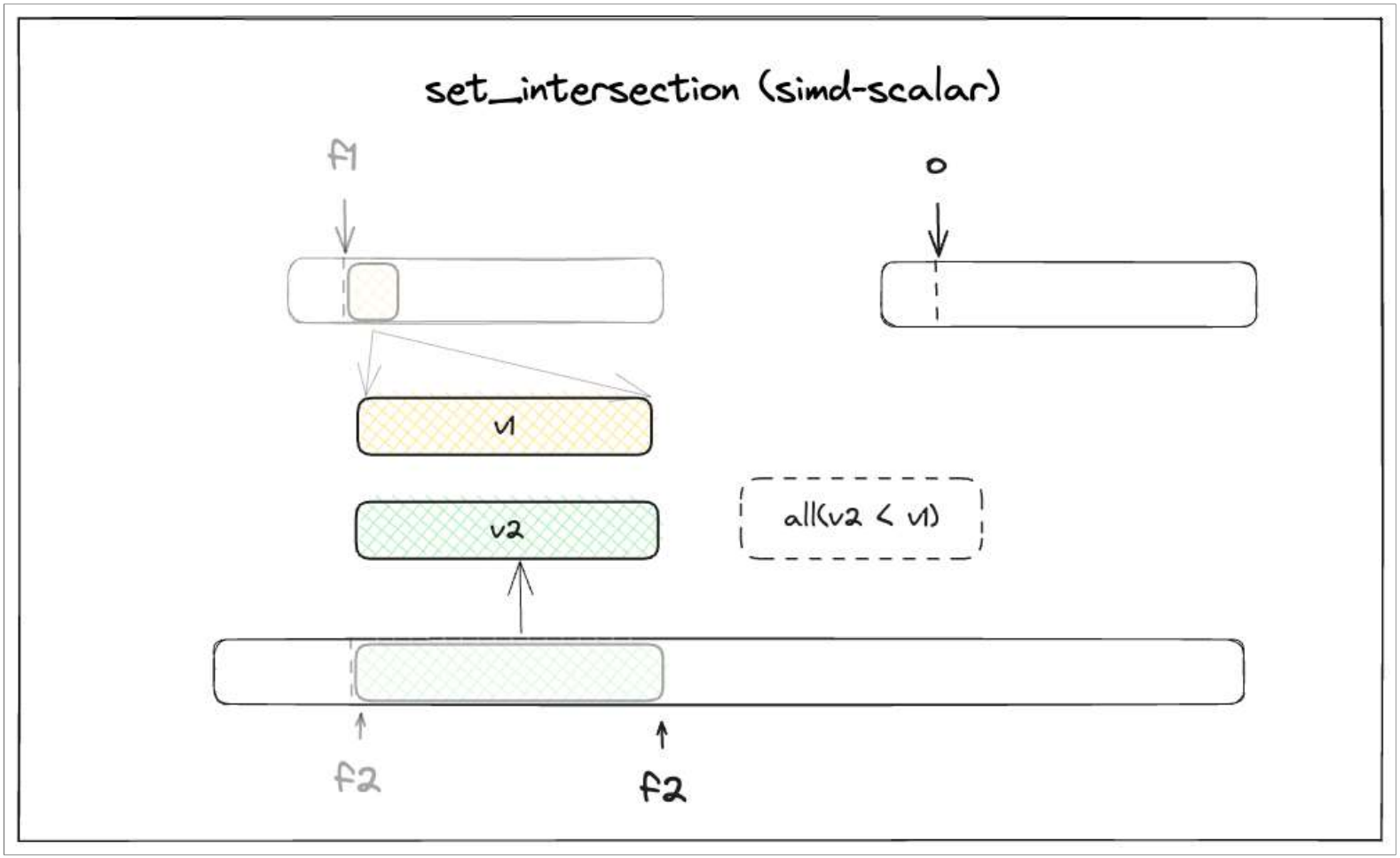

set_intersection (simd-scalar)

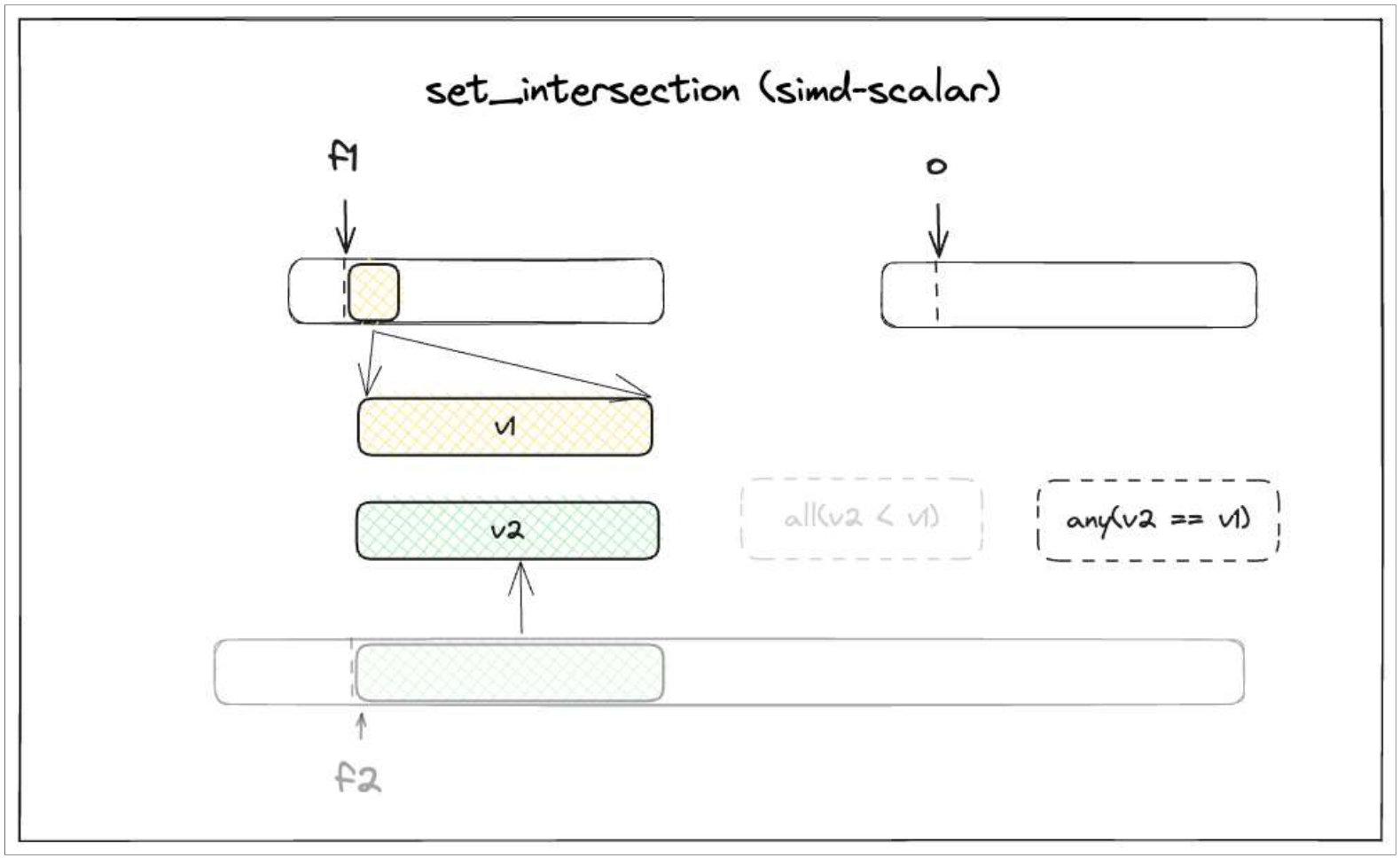

set_intersection (simd-scalar)
f1
o
f2

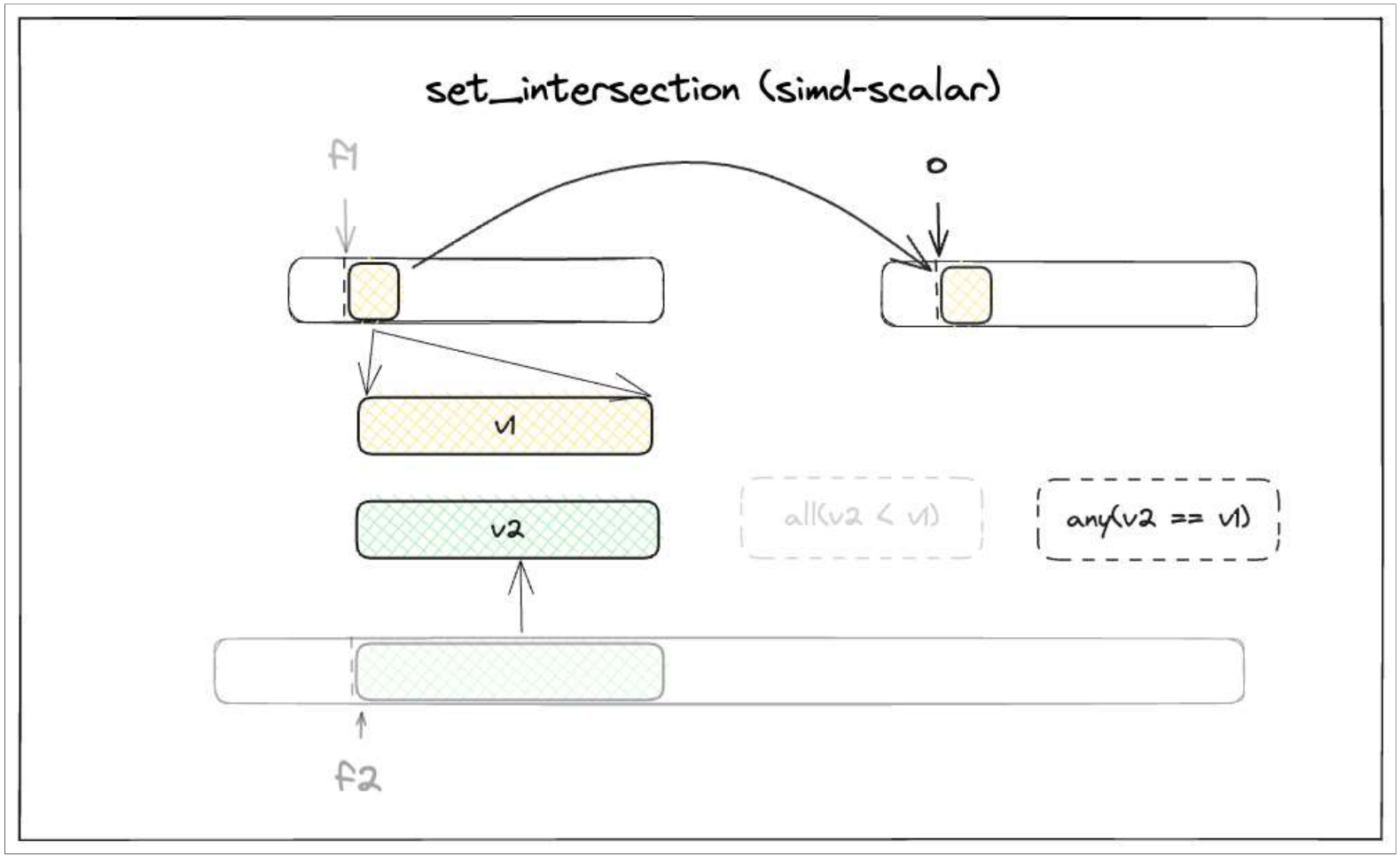

set_intersection (simd-scalar)
f1
0
f2

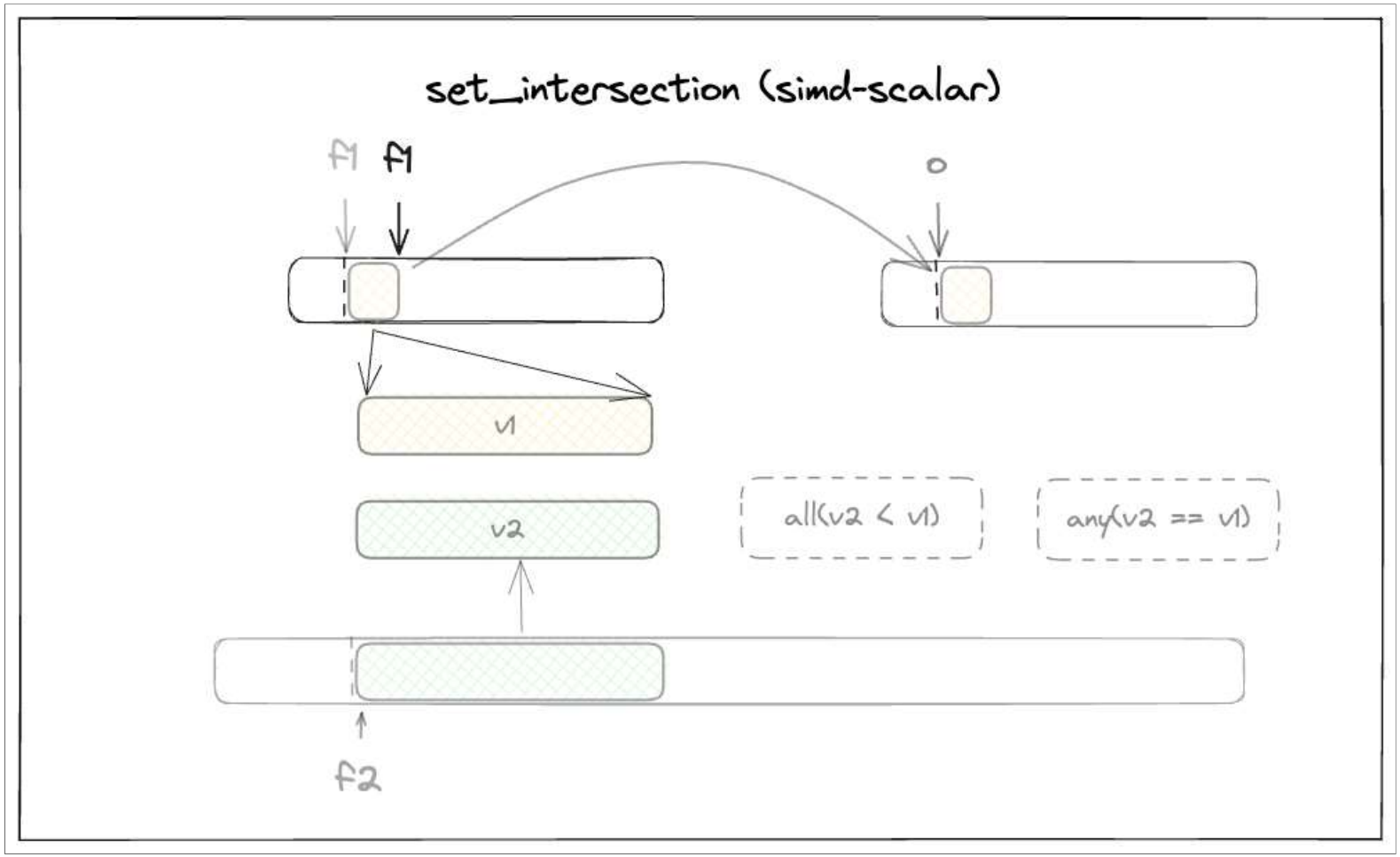

set_intersection (simd-scalar)
f1
0
v1
v2
f2


set_intersection (simd-scalar)
f1
0
v1
v2
all(v2 < v1)
f2


set_intersection (simd-scalar)
f1
0
v1
v2
all(v2 < v1)
f2
f2

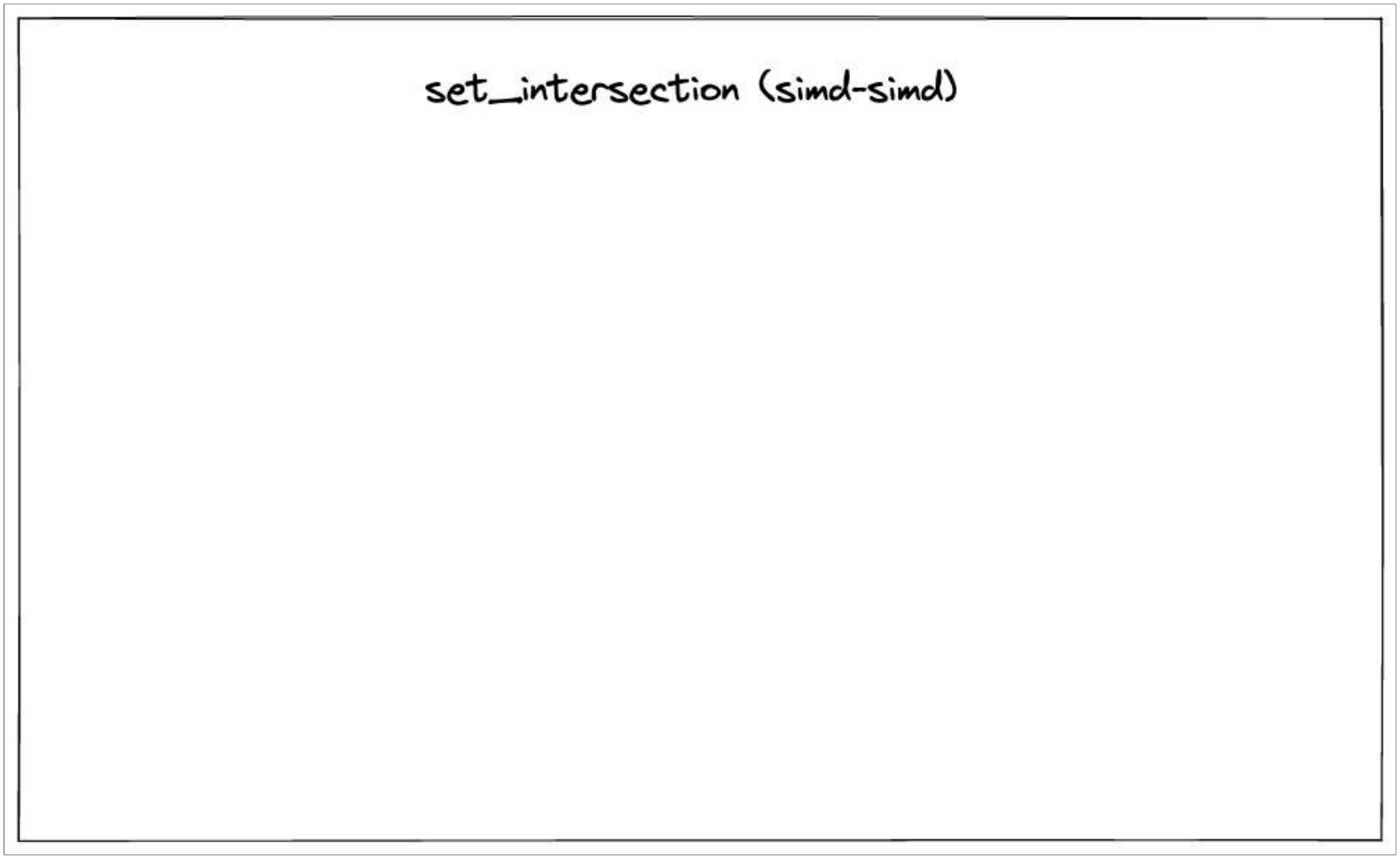

set_intersection (simd-scalar)
f1
0
v1
v2
all(v2 < v1)
any(v2 == v1)
f2

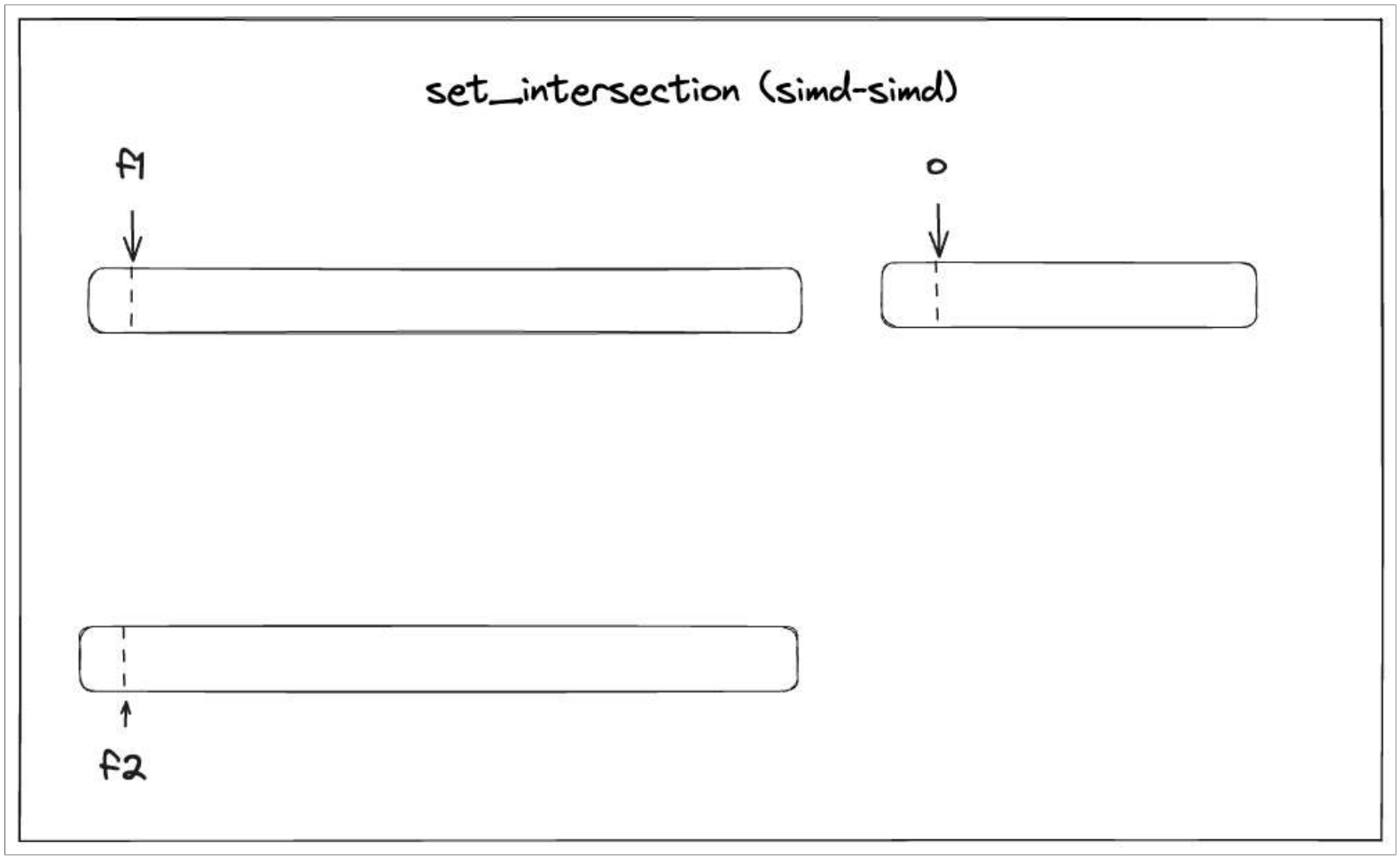

set_intersection (simd-scalar)
f1
o
v1
v2
all(v2 < v1)
any(v2 == v1)
f2

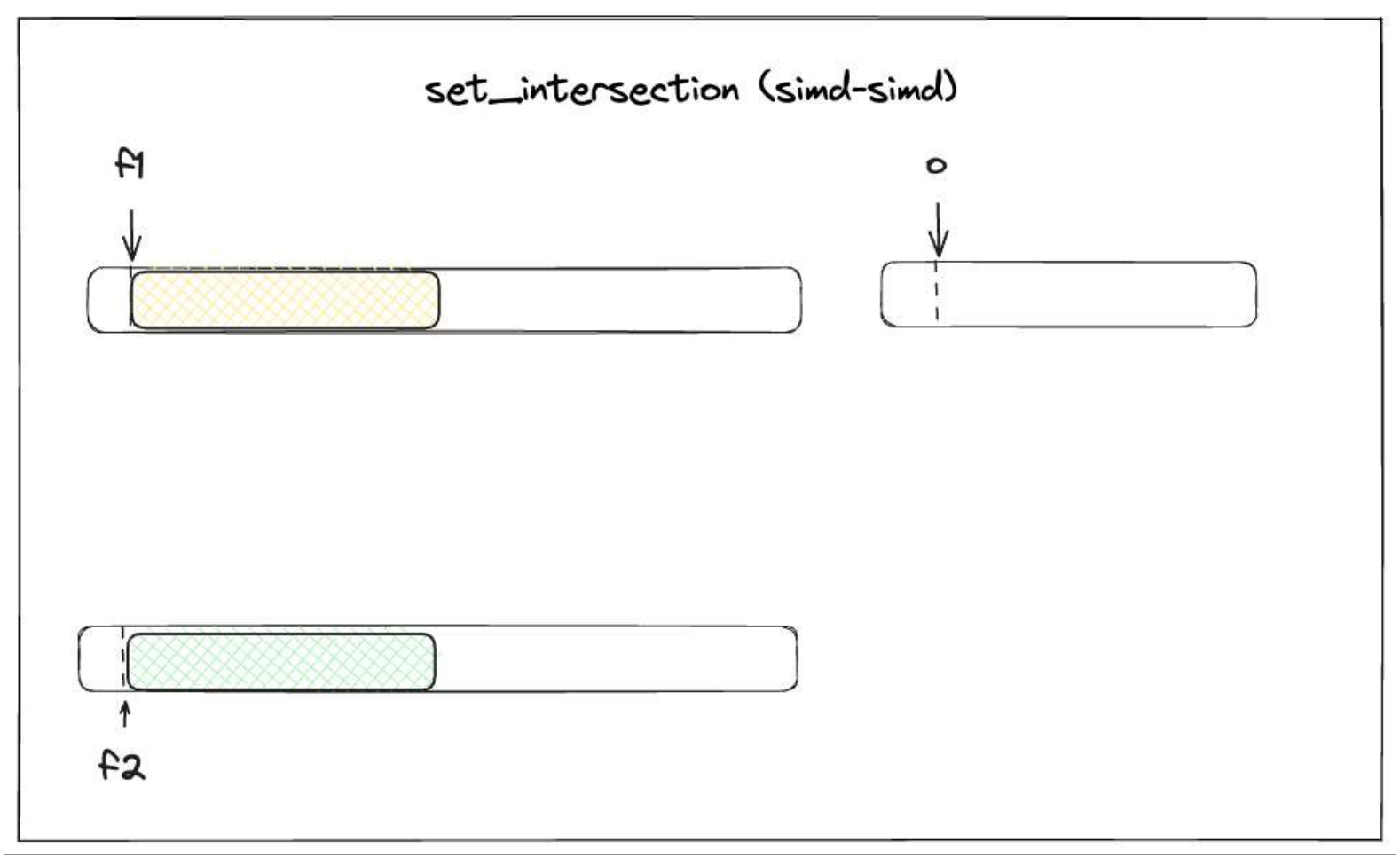

set_intersection (simd-scalar)
f1
f1
o
v1
v2
all(v2 < v1)
any(v2 == v1)
f2

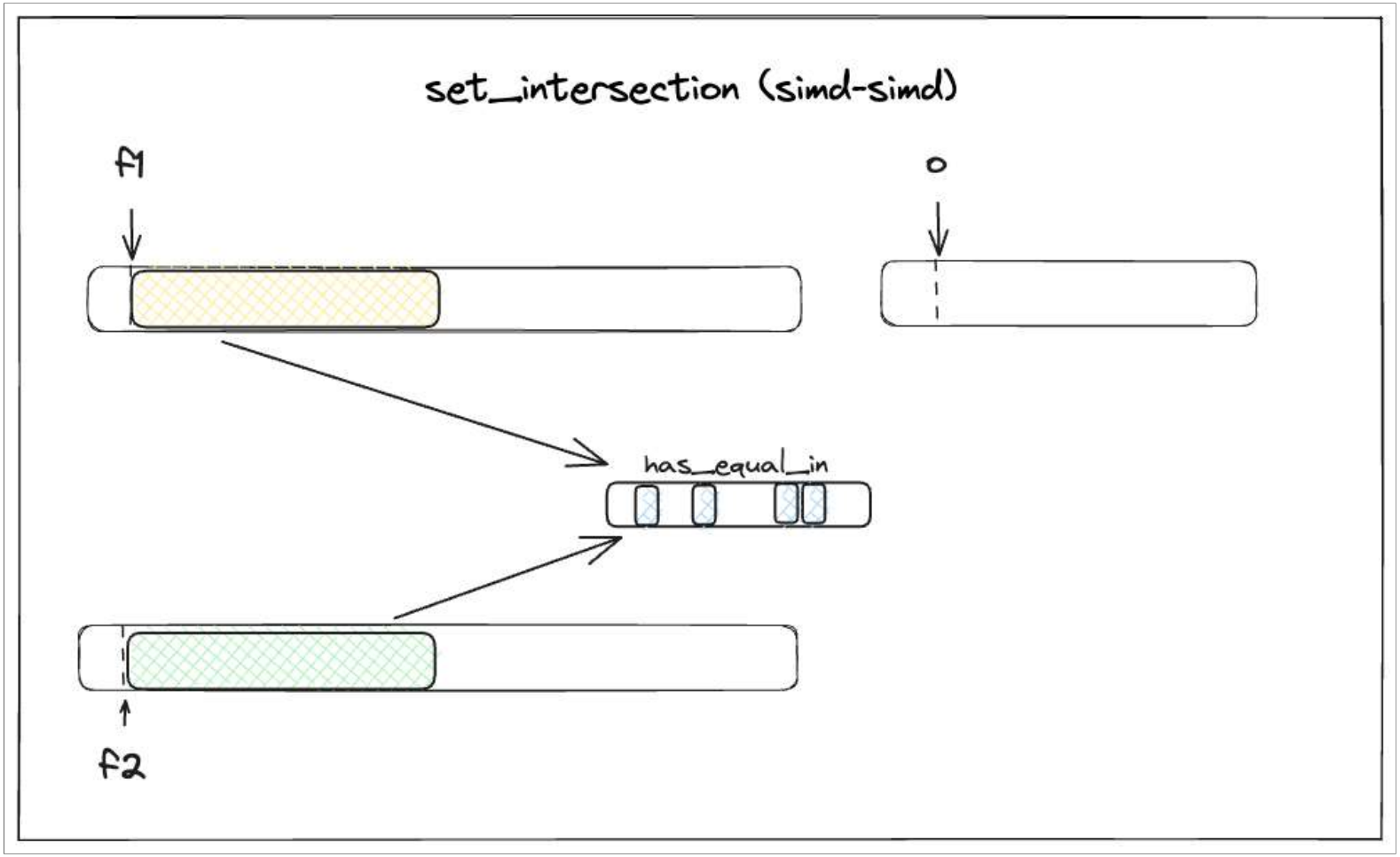
SET INTERSECTION (SIMD/SIMD)

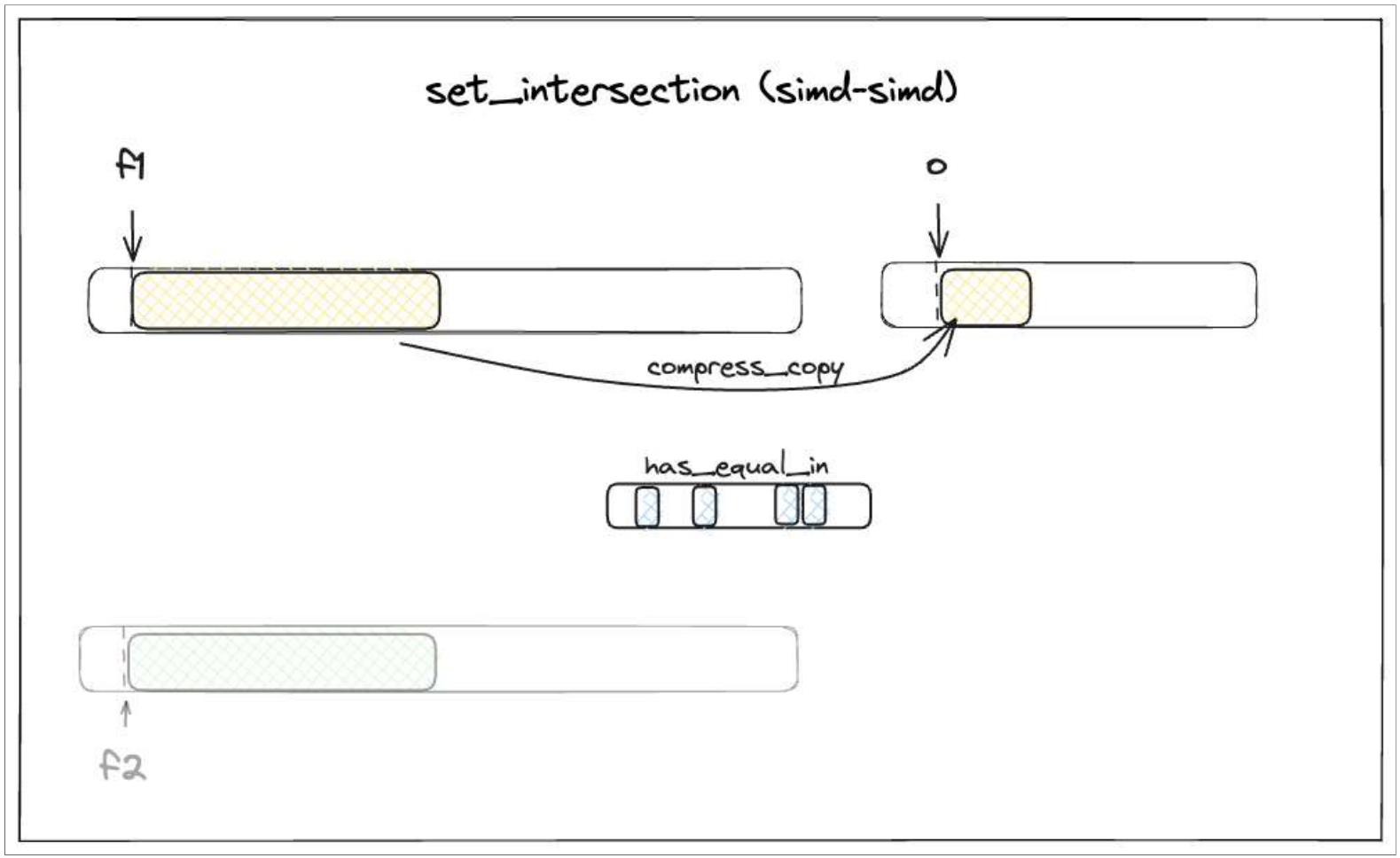
- **Faster-Than-Native Alternatives for x86 VP2INTERSECT Instructions**
- Guillermo Diez-Canas

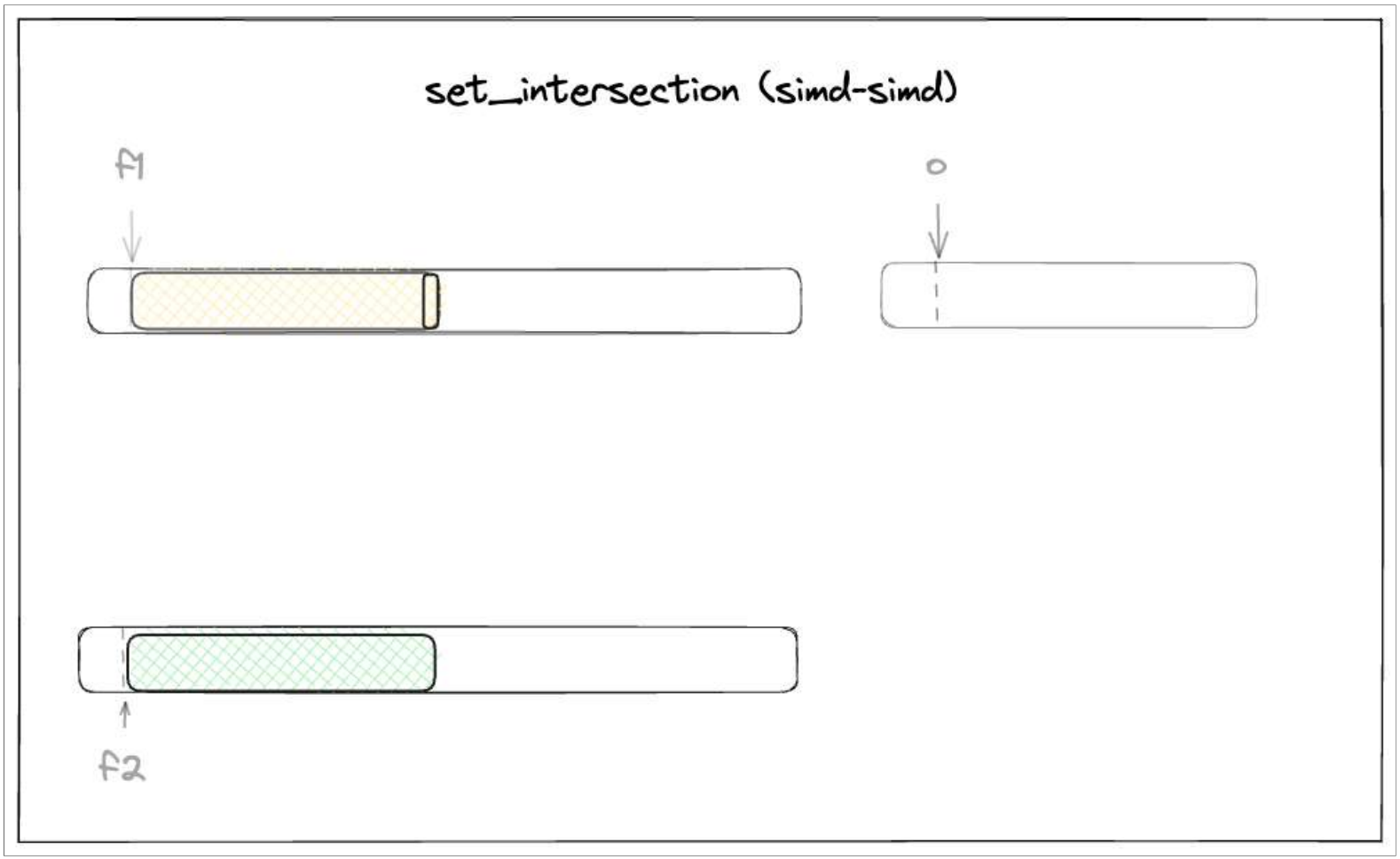

set_intersection (simd-simd)

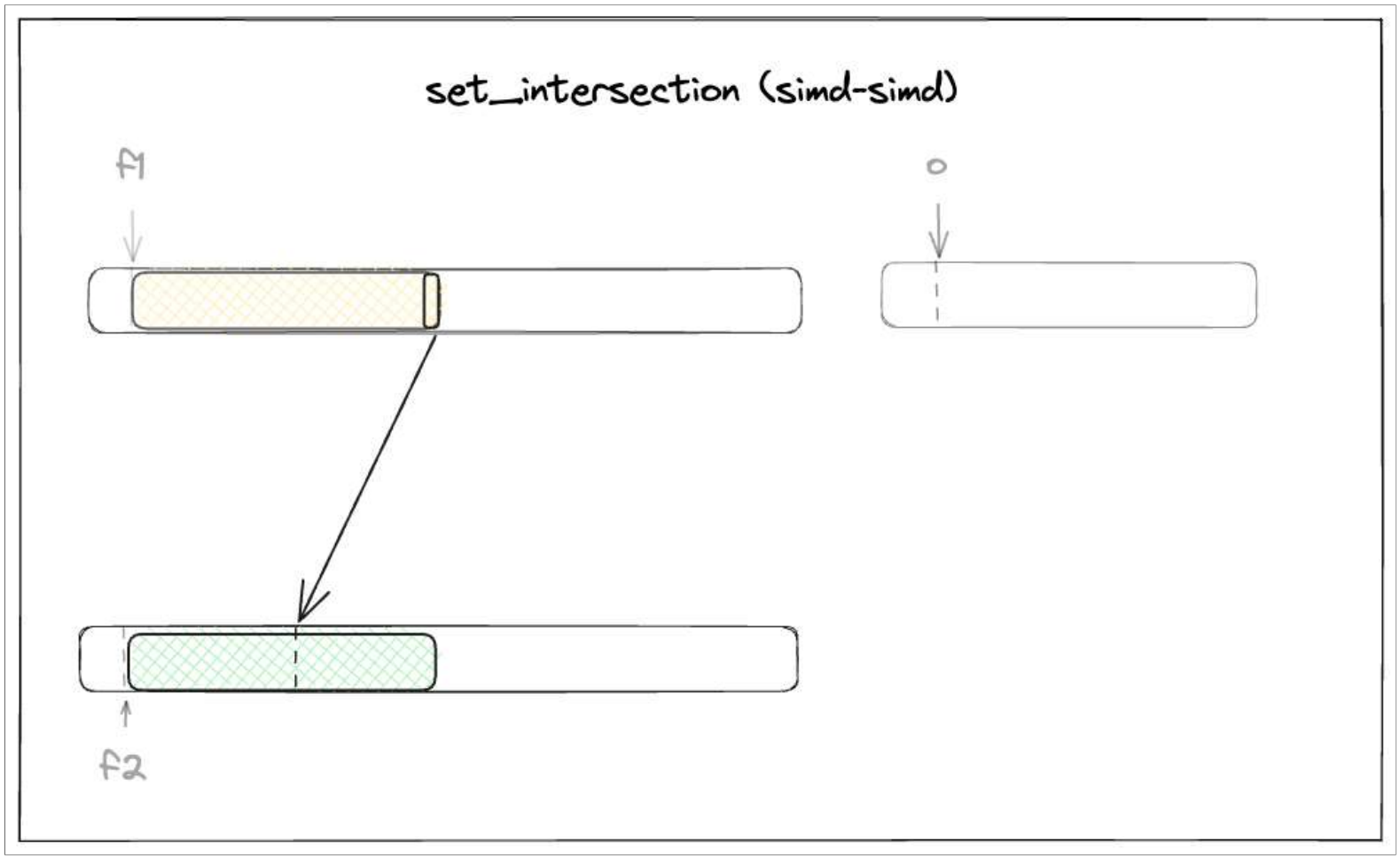

set_intersection (simd-simd)
f1
o
f2

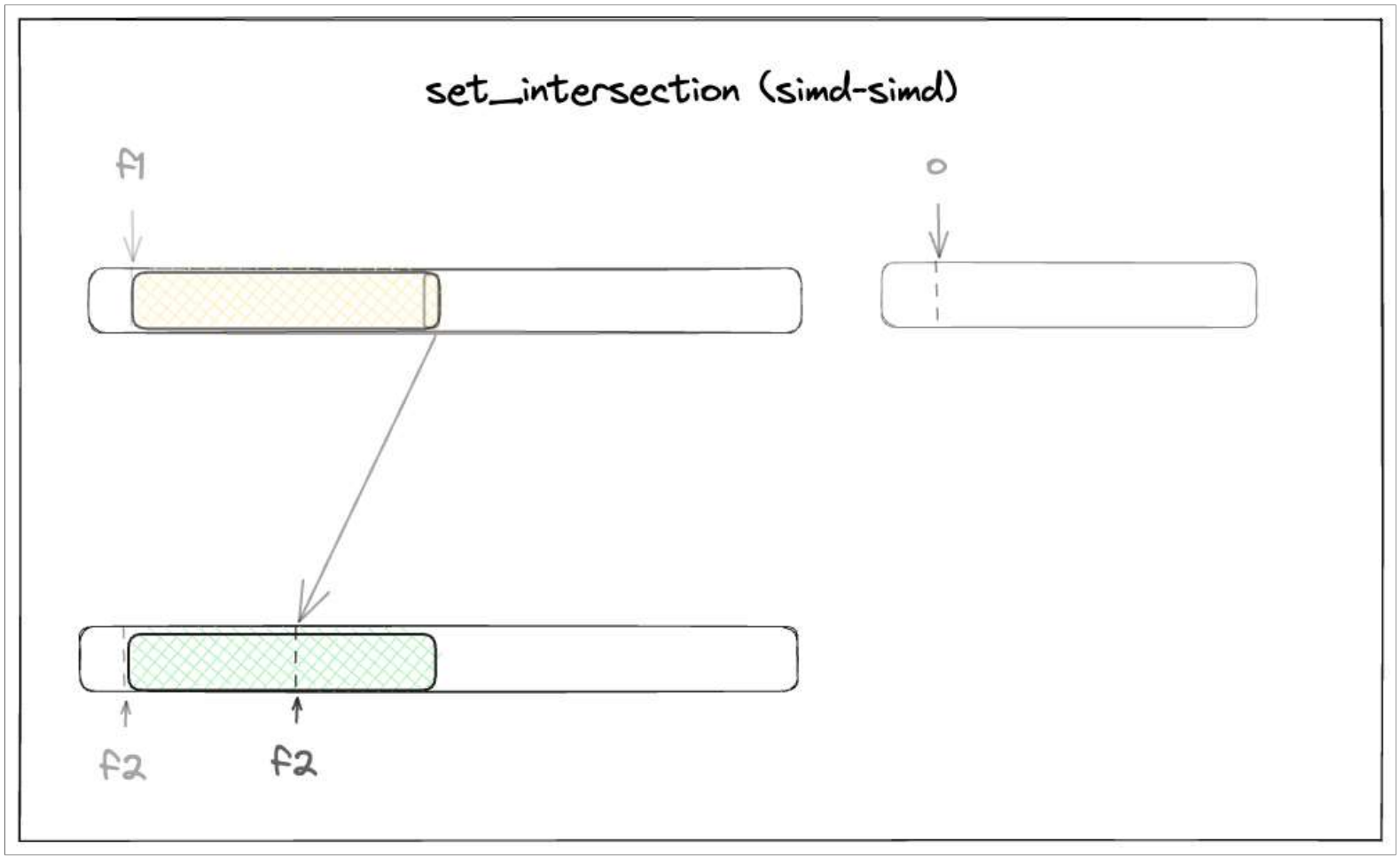

set_intersection (simd-simd)
f1
o
f2

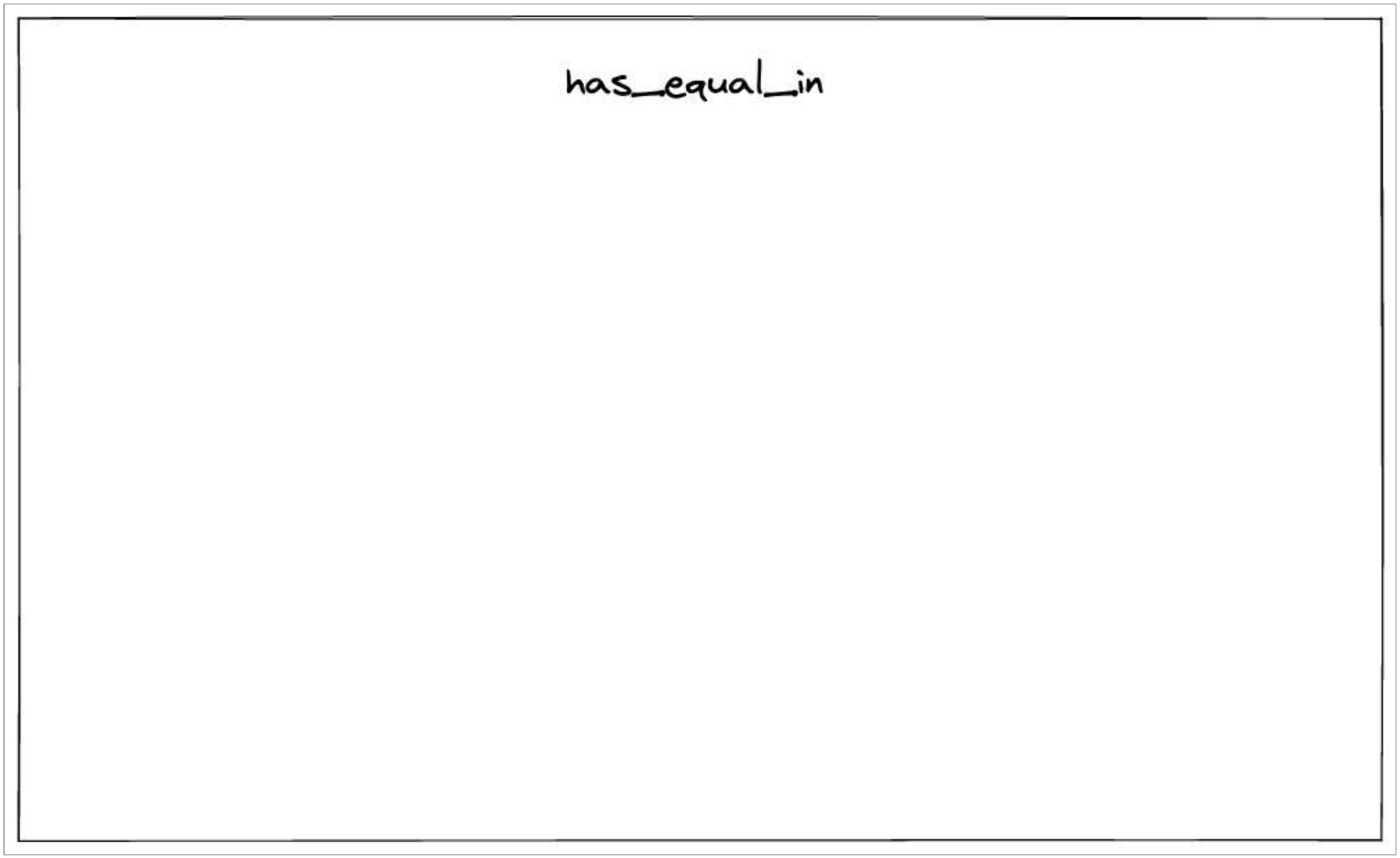

set_intersection (simd-simd)
f1
o
has_equal_in
f2

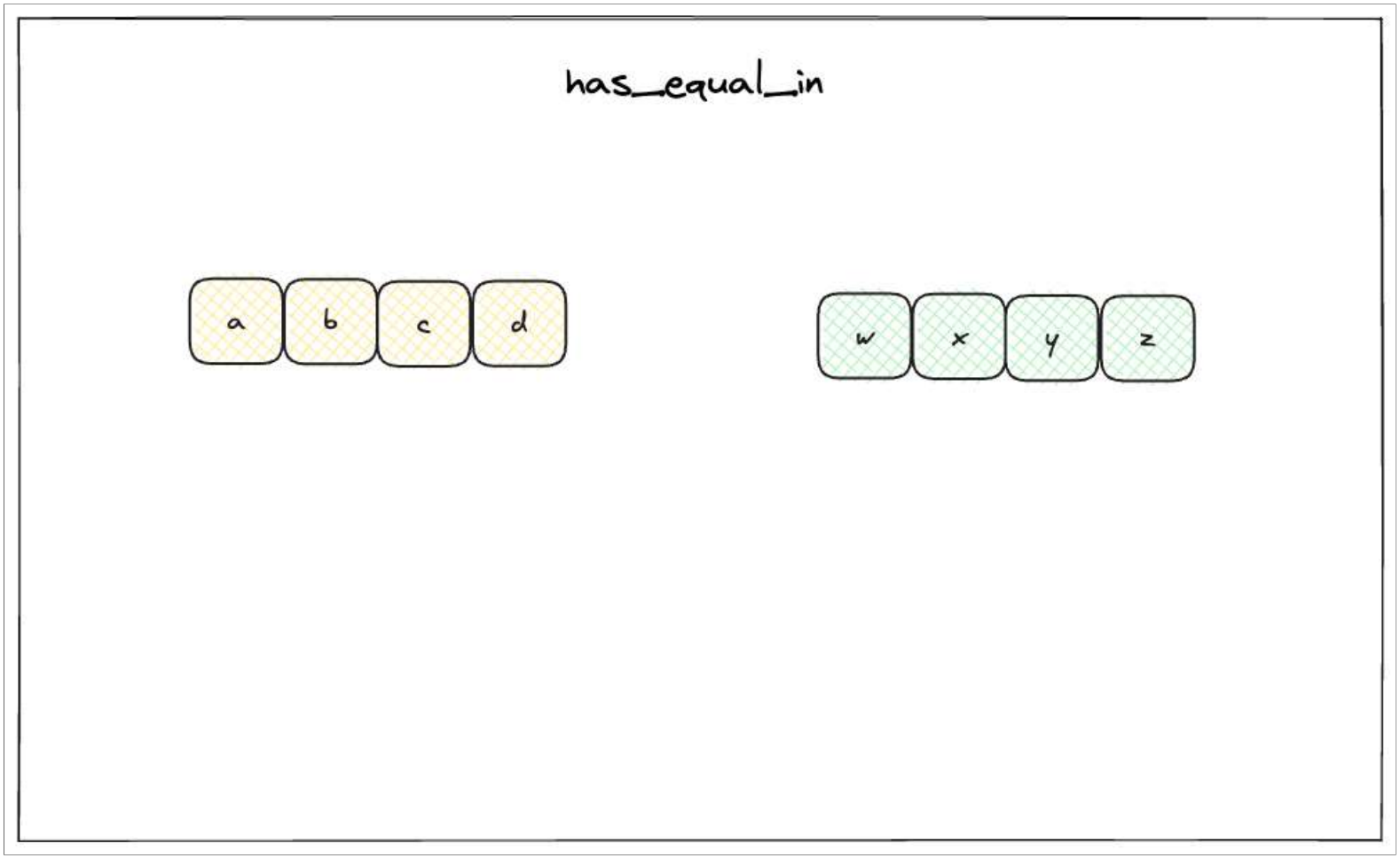

set_intersection (simd-simd)
f1
o
compress_copy
has_equal_in
f2

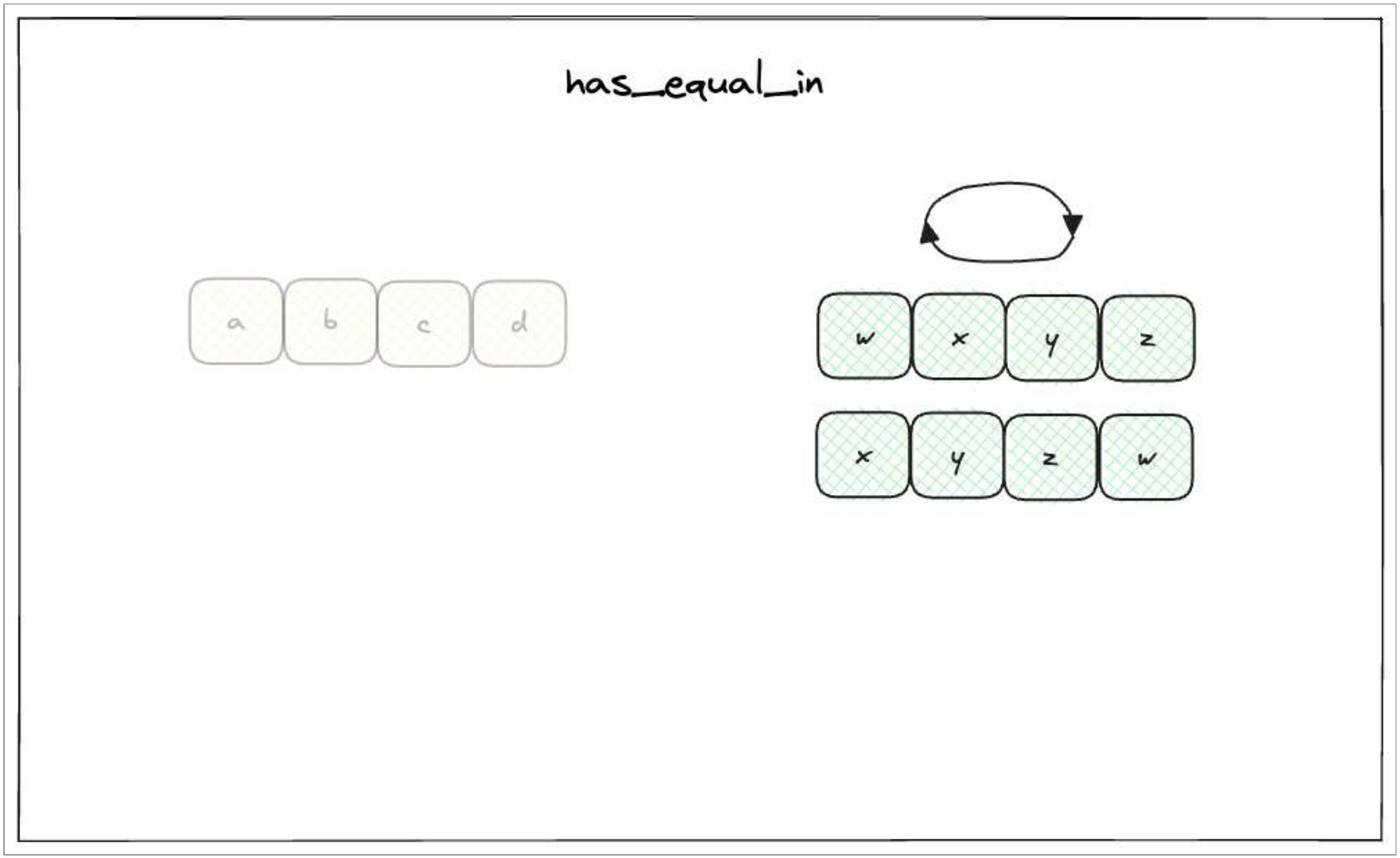

set_intersection (simd-simd)
f1
o
f2

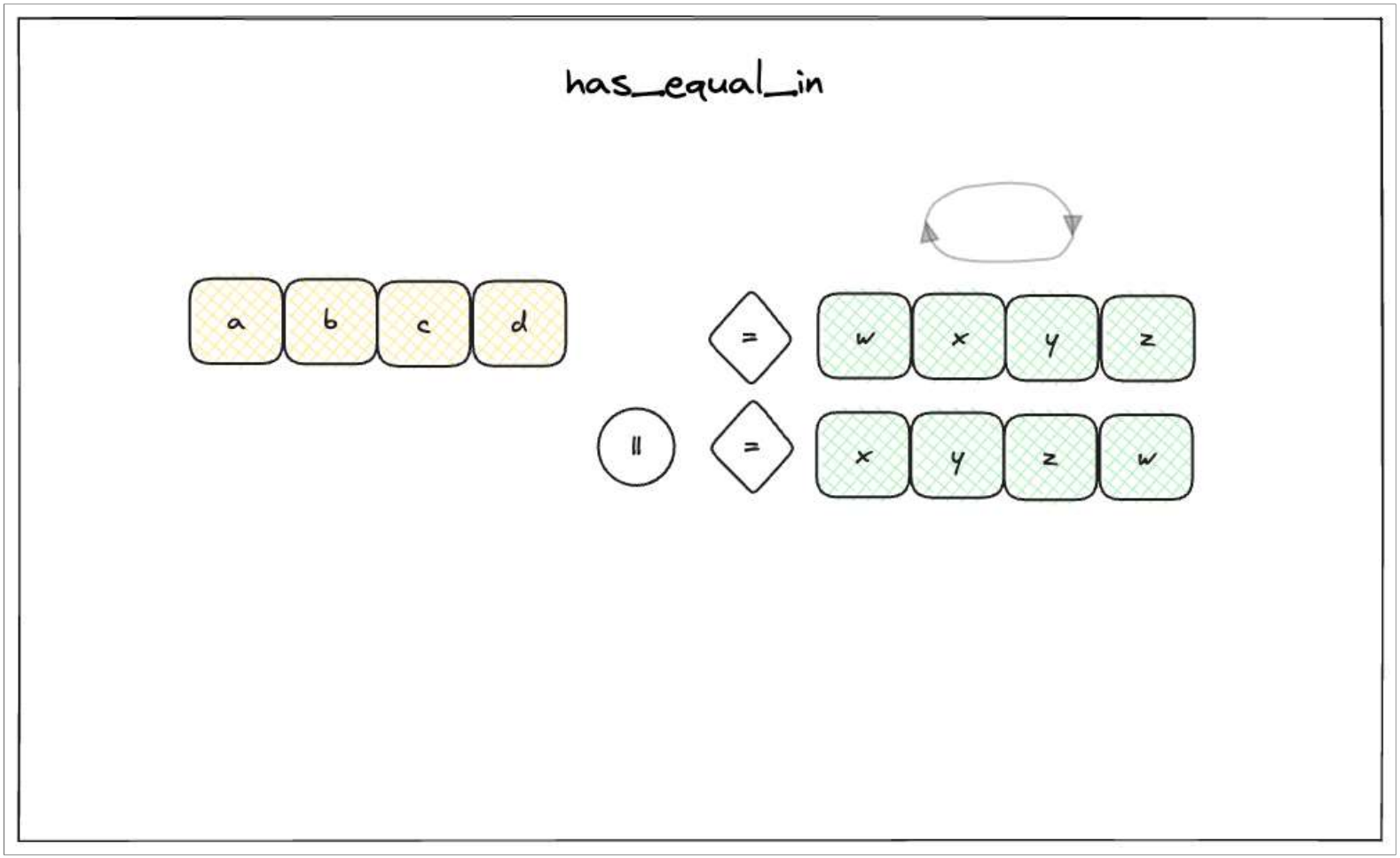

set_intersection (simd-simd)
f1
o
f2

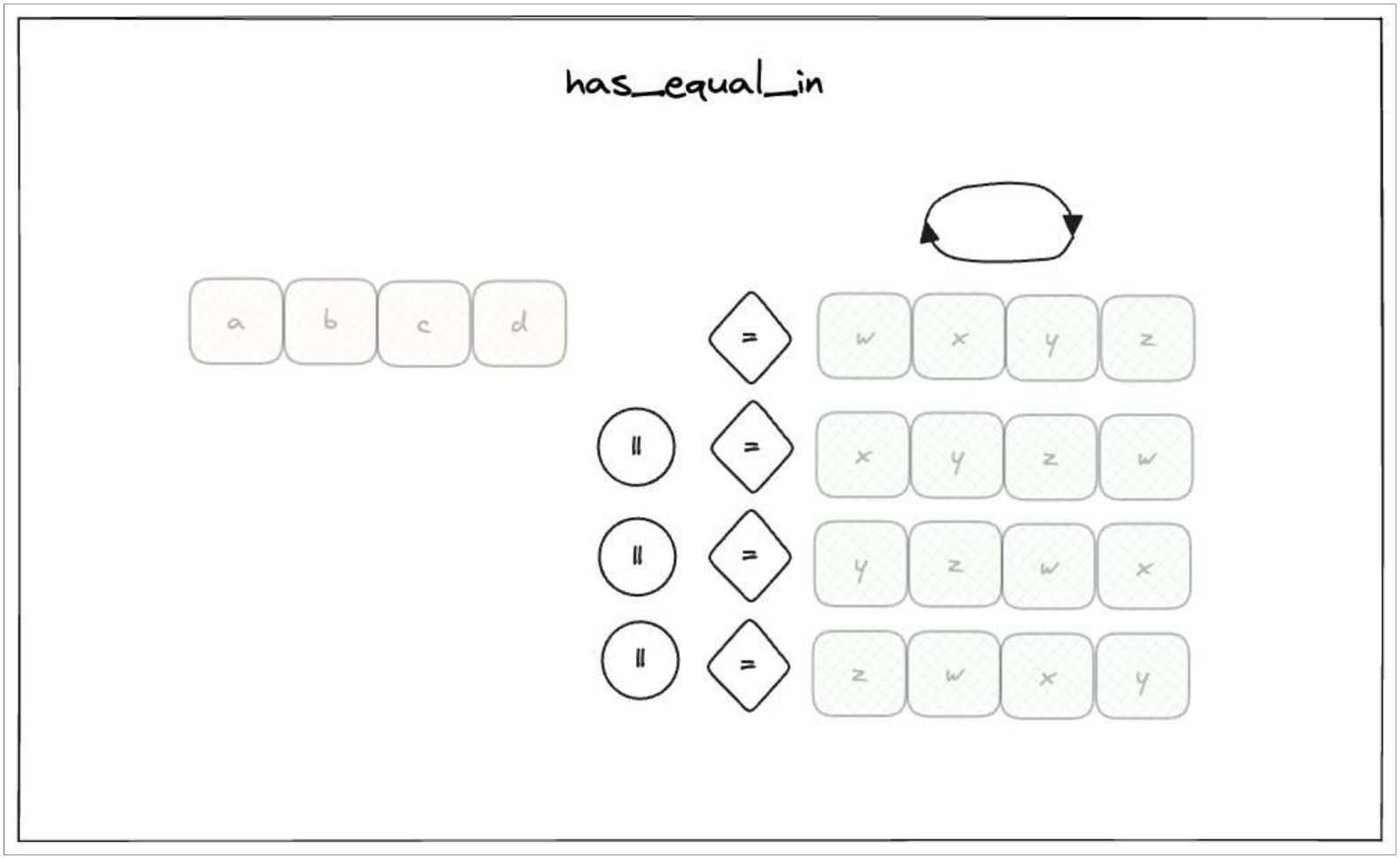

set_intersection (simd-simd)
f1
o
f2
f2


has_equal_in

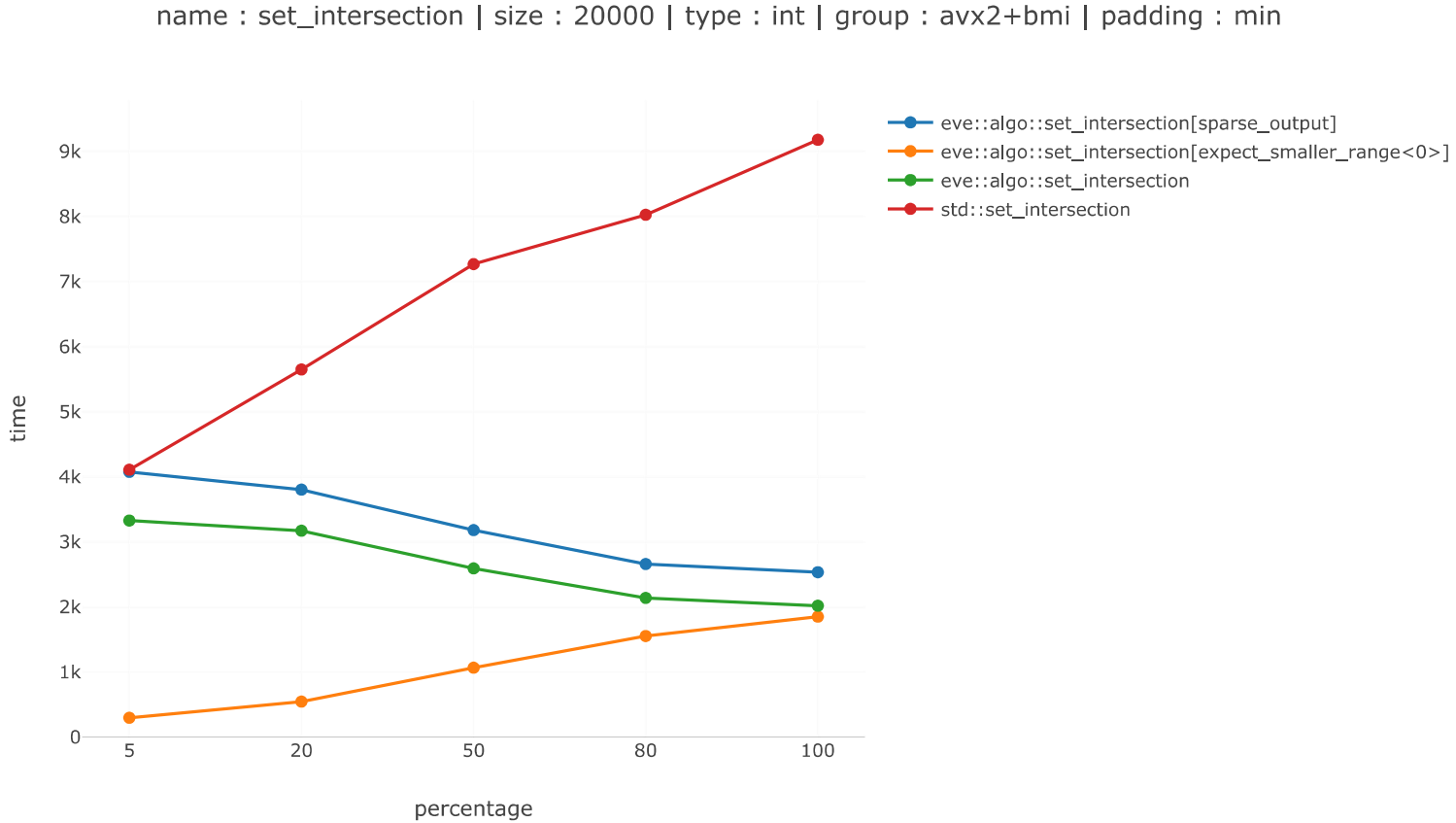

has_equal_in
a
b
c
d
w
x
y
z


has_equal_in
a
b
c
d
w
x
y
z
x
y
z
w

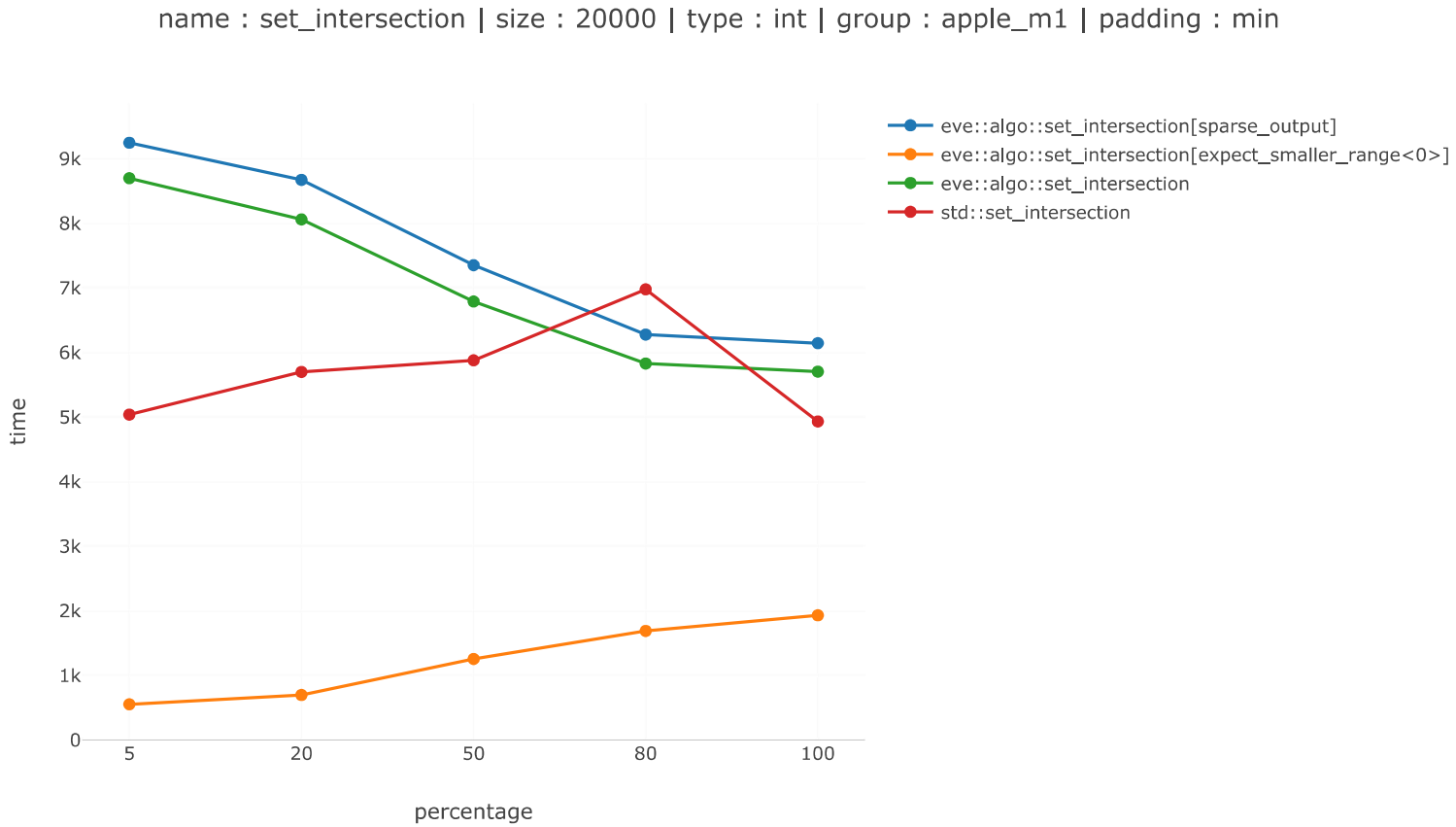

has_equal_in
a
b
c
d
=
w
x
y
z
||
=
x
y
z
w


has_equal_in
a
b
c
d
=
w
x
y
z
||
=
x
y
z
w
||
=
y
z
w
x
||
=
z
w
x
y

SET_INTERSECTION, INT


name : set_intersection | size : 20000 | type : int | group : avx2+bmi | padding : min
time
9k
8k
7k
6k
5k
4k
3k
2k
1k
0
5
20
50
80
100
percentage
eve::algo::set_intersection[sparse_output]
eve::algo::set_intersection[expect_smaller_range<0>]
eve::algo::set_intersection
std::set_intersection

SET_INTERSECTION, INT


name : set_intersection | size : 20000 | type : int | group : apple_m1 | padding : min
eve::algo::set_intersection[sparse_output]
eve::algo::set_intersection[expect_smaller_range<0>]
eve::algo::set_intersection
std::set_intersection
time
9k
8k
7k
6k
5k
4k
3k
2k
1k
0
5
20
50
80
100
percentage

THANKS

- Joel Falcou, Jean-Thierry Lapresté
- Amazing people mentioned

LINKS

- eve: **github.com/jfalcou/eve**
- **tinyurl.com/dycppcon2023**
- **My First Simd**
- **SIMD in C++20: EVE of a new Era**
- Advanced Simd Algorithms In Pictures, Meeting C++
- CppLang Slack, #simd

Speaker notes