Proving Performance

nikolai vazquez

~~Proving~~ Performance

nikolai vazquez

Vibe Checking

~~Proving~~ Performance

nikolai vazquez

nikolai
vazquez

Performance

Direct Issues

nikolai
vazquez

Performance

Direct Issues

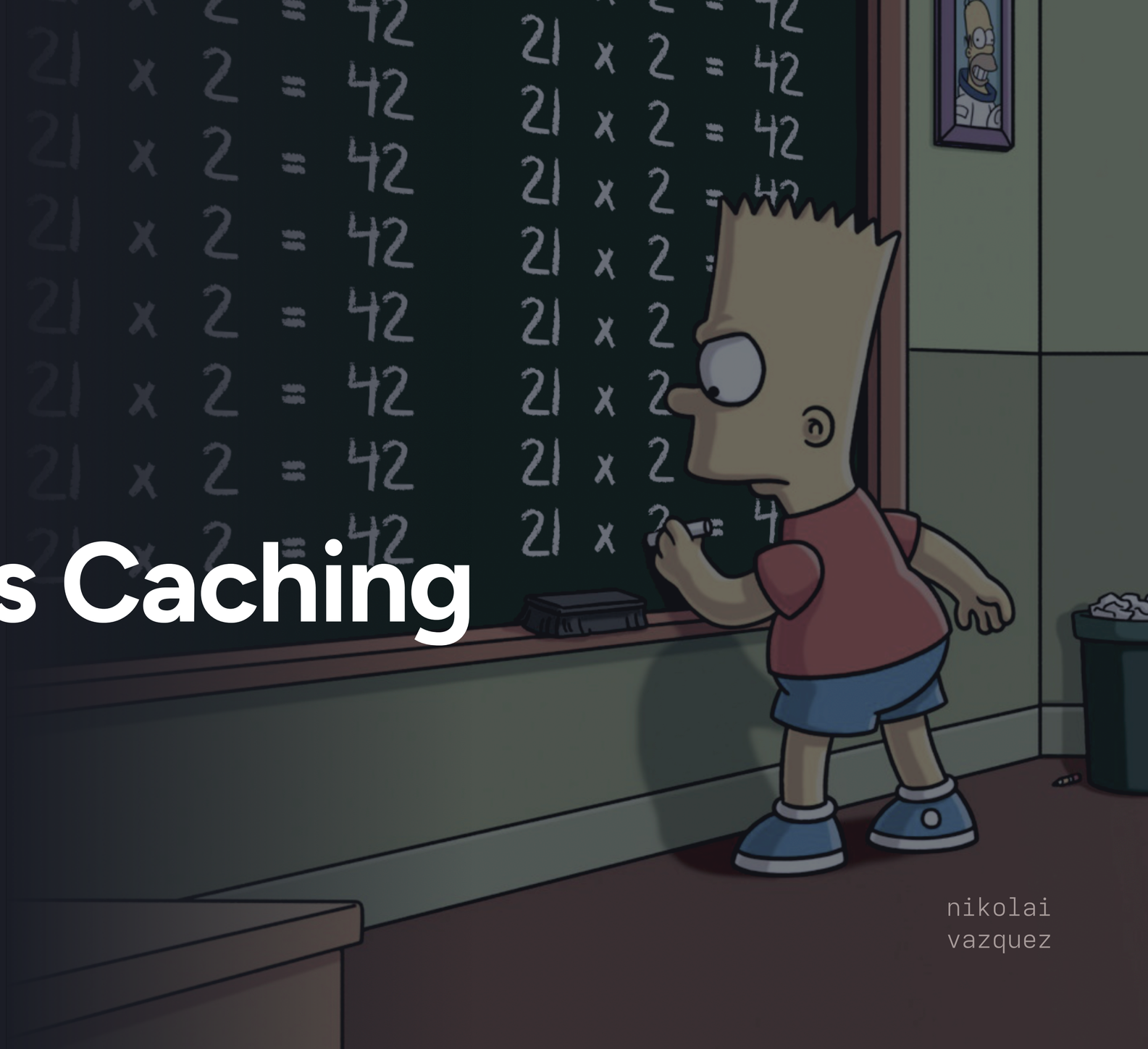
Inefficient Algorithms

nikolai
vazquez

Performance

Direct Issues

Repetition vs Caching



nikolai
vazquez

Performance

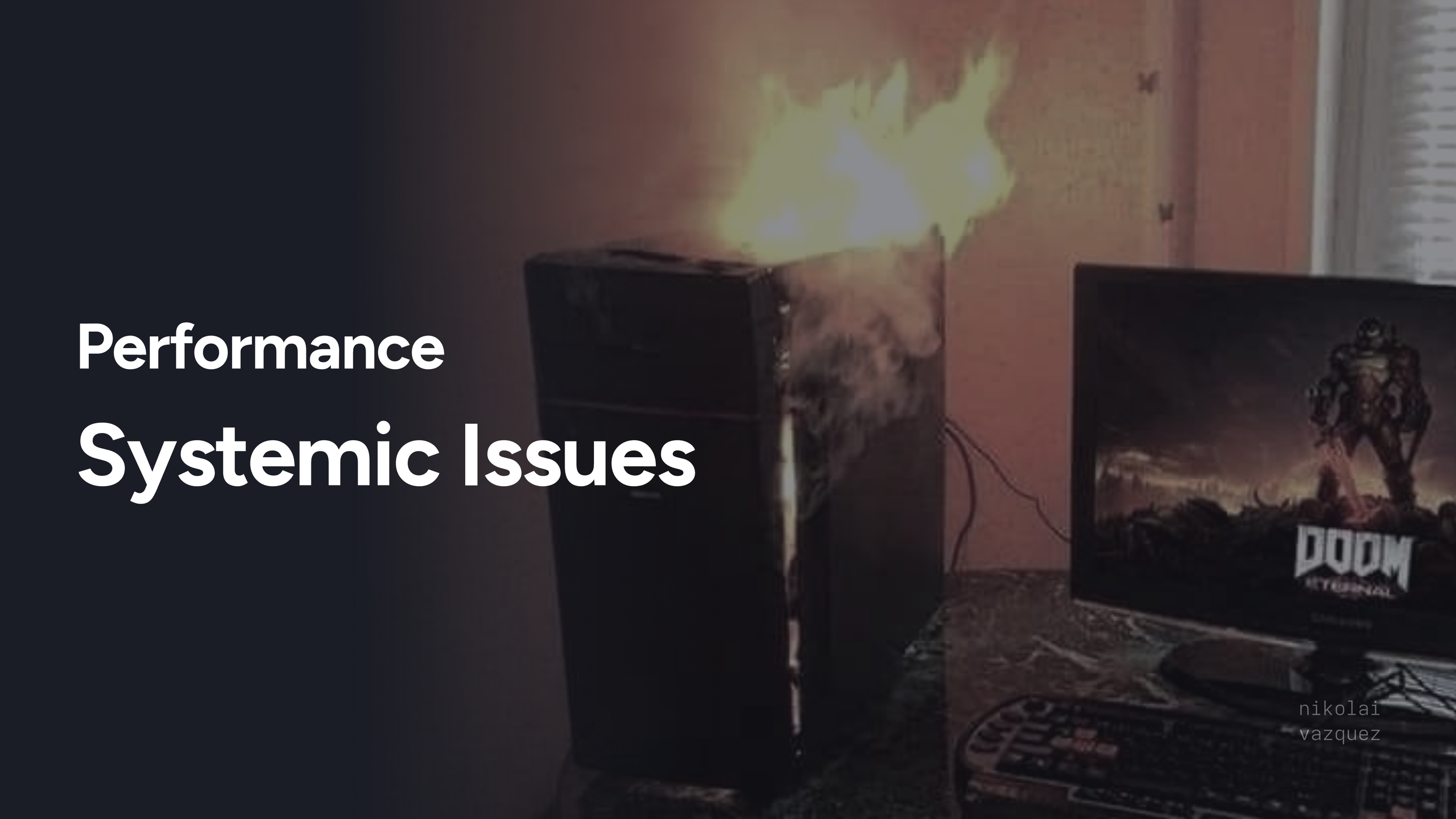
Direct Issues

Slower OS APIs

nikolai
vazquez

Performance

Systemic Issues

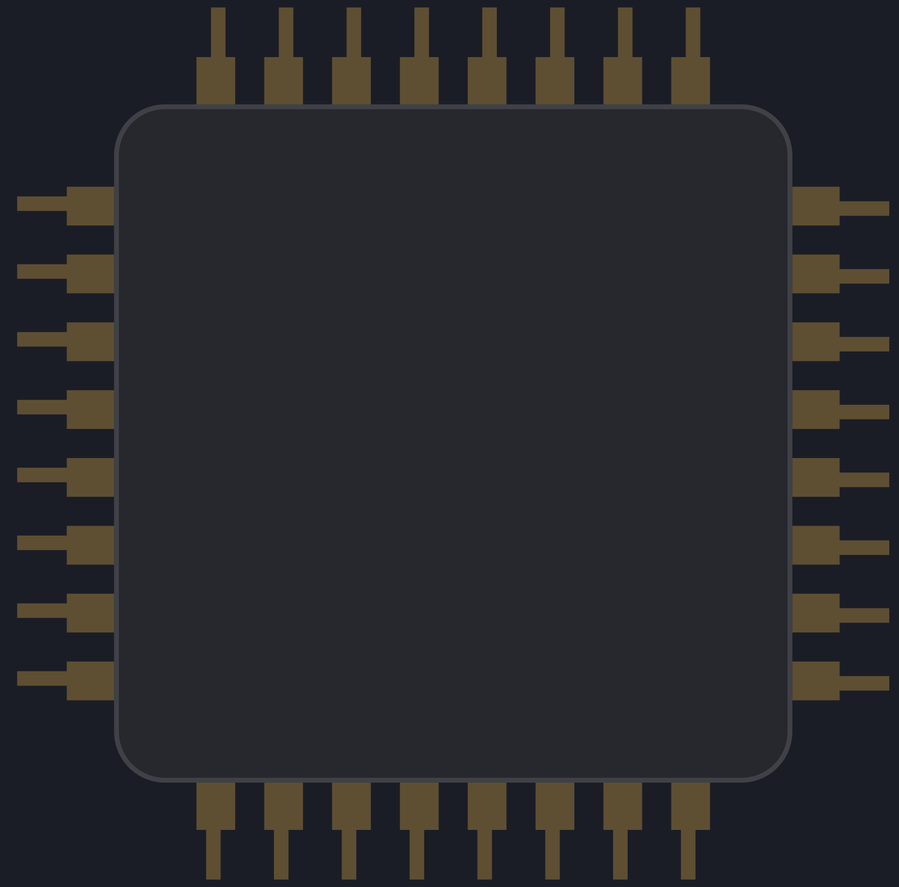


nikolai
vazquez

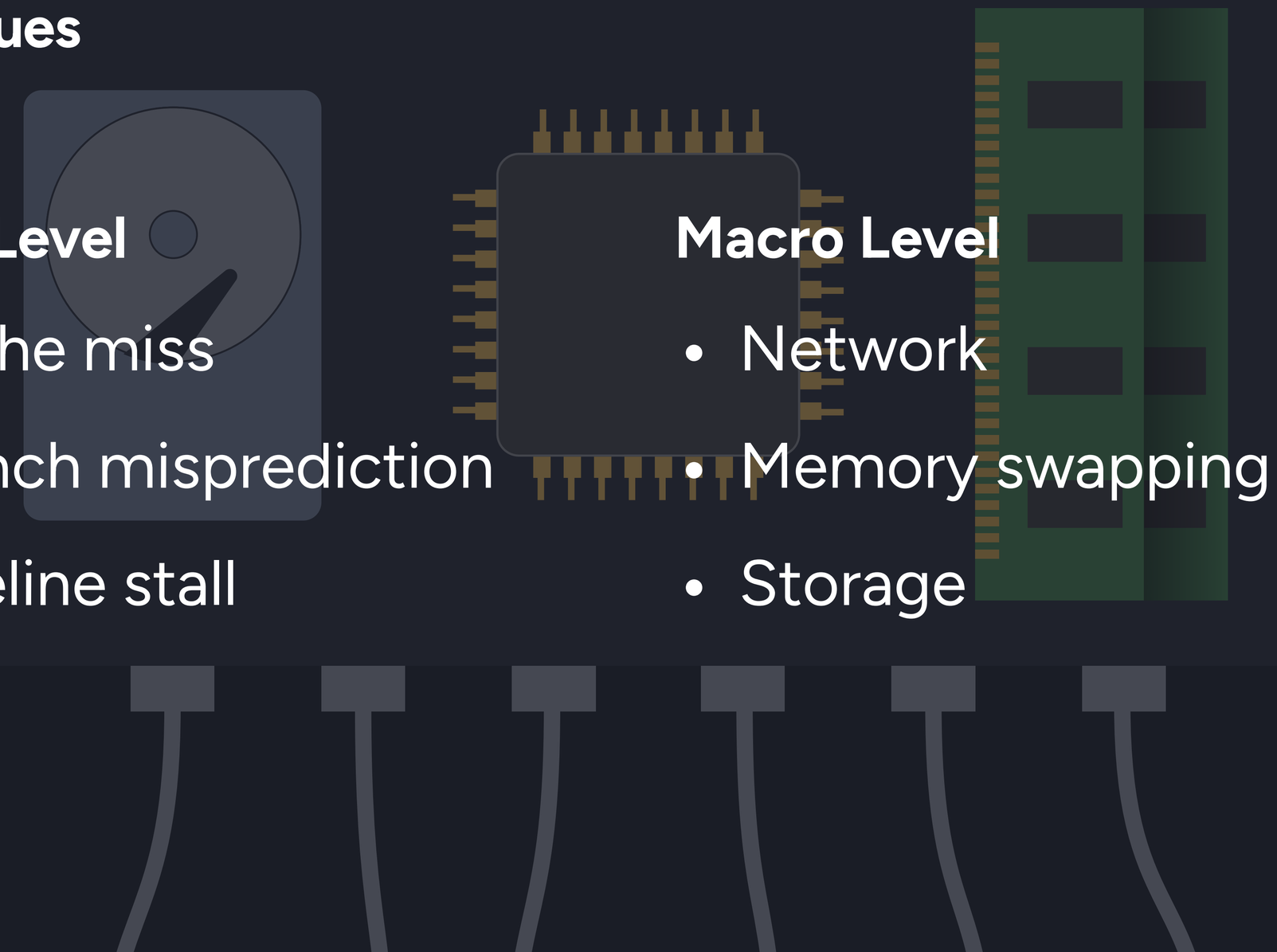
Systemic Issues

Micro Level

- Cache miss
- Branch misprediction
- Pipeline stall



Systemic Issues



Micro Level

- Cache miss
- Branch misprediction
- Pipeline stall

Macro Level

- Network
- Memory swapping
- Storage

Why Rust?

nikolai
vazquez

Why Rust?

Culture of Performance

nikolai
vazquez

Measuring Performance

nikolai
vazquez

Measuring Performance

```
Vec::from_iter(1..=100);
```

nikolai
vazquez

Measuring Performance

```
let start = Instant::now();

Vec::from_iter(1..=100);

let time = start.elapsed();
```

Measuring Performance

0 nanoseconds

```
let start = Instant::now();

Vec::from_iter(1..=100);

let time = start.elapsed();
```

nikolai
vazquez

0 nanoseconds?

```
let start = Instant::now();

Vec::from_iter(1..=100);

let time = start.elapsed();
```

nikolai
vazquez

Measuring Performance

```
let start = Instant::now();

black_box(Vec::from_iter(1..=100));

let time = start.elapsed();
```

nikolai
vazquez

Measuring Performance

500 nanoseconds

```
let start = Instant::now();
black_box(Vec::from_iter(1..=100));
let time = start.elapsed();
```

nikolai
vazquez

500 nanoseconds?

nikolai
vazquez

Measuring Performance

1200
700
500 nanoseconds?
400
200

nikolai
vazquez

Divan

nikolaivazquez.com/blog/divan

github.com/nvzqz/divan

docs.rs/divan

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn my_benchmark() {
    ...
}
```

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn my_benchmark() {
    ...
}
```

```rust
#[test]
fn my_test() {
    ...
}
```

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() {
    black_box(Vec::from_iter(1..=100));
}
```

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() {
    black_box(Vec::from_iter(1..=100));
}
```

```
benches       fastest   | slowest   | median    | mean      | samples | iters
└─ alloc_vec  72.53 ns  | 221.6 ns  | 73.18 ns  | 74.65 ns  | 100     | 6400
```

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() {
    black_box(Vec::from_iter(1..=100));
}
```

```
benches          fastest  | slowest  | median   | mean     | samples | iters
└─ alloc_vec     72.53 ns | 221.6 ns | 73.18 ns | 74.65 ns | 100     | 6400
```

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() {
    black_box(Vec::from_iter(1..=100));
}
```

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec	72.53 ns	221.6 ns	73.18 ns	74.65 ns	100	6400

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() {
    black_box(Vec::from_iter(1..=100));
}
```

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec	72.53 ns	221.6 ns	73.18 ns	74.65 ns	100	6400

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() {
    black_box(Vec::from_iter(1..=100));
}
```

benches	fastest	slowest	median	mean	samples	iters
└─ alloc_vec	72.53 ns	221.6 ns	73.18 ns	74.65 ns	100	6400

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() -> Vec<i32> {
    black_box(Vec::from_iter(1..=100))
}
```

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() -> Vec<i32> {
    Vec::from_iter(1..=100)
}
```

nikolai
vazquez

Divan

Simple API

```rust
#[divan::bench]
fn alloc_vec() -> Vec<i32> {
    Vec::from_iter(1..=100)
}
```

benches	fastest	slowest	median	mean	samples	iters
└─ alloc_vec	28.32 ns	88.87 ns	28.98 ns	29.69 ns	100	6400

nikolai
vazquez

Divan

Case Parameters

```rust
#[divan::bench]
fn alloc_vec() -> Vec<i32> {
    Vec::from_iter(1..=100)
}
```

nikolai
vazquez

Divan

Case Parameters

```rust
#[divan::bench(args = [1, 5, 10, 1000, 10000000])]
fn alloc_vec(n: i32) -> Vec<i32> {
    Vec::from_iter(1..=n)
}
```

nikolai
vazquez

Divan

Case Parameters

```rust
#[divan::bench(args = [1, 5, 10, 1000, 10000000])]
fn alloc_vec(n: i32) -> Vec<i32> {
    Vec::from_iter(1..=n)
}
```

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec						
├ 1	38.43 ns	40.38 ns	39.08 ns	38.97 ns	100	12800
├ 5	42.01 ns	45.92 ns	42.66 ns	42.78 ns	100	12800
├ 10	37.13 ns	49.5 ns	45.92 ns	44.41 ns	100	12800
├ 1000	194.3 ns	475.6 ns	212.5 ns	213.4 ns	100	1600
└ 10000000	2.829 ms	8.297 ms	3.073 ms	3.184 ms	100	100

Divan

Case Parameters

```rust
#[divan::bench(args = [1, 5, 10, 1000, 10000000])]
fn alloc_vec(n: i32) -> Vec<i32> {
    Vec::from_iter(1..=n)
}
```

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec						
├ 1	38.43 ns	40.38 ns	39.08 ns	38.97 ns	100	12800
├ 5	42.01 ns	45.92 ns	42.66 ns	42.78 ns	100	12800
├ 10	37.13 ns	49.5 ns	45.92 ns	44.41 ns	100	12800
├ 1000	194.3 ns	475.6 ns	212.5 ns	213.4 ns	100	1600
└ 10000000	2.829 ms	8.297 ms	3.073 ms	3.184 ms	100	100

Divan

Case Parameters

```rust
#[divan::bench(args = [1, 5, 10, 1000, 10000000])]
fn alloc_vec(n: i32) -> Vec<i32> {
    Vec::from_iter(1..=n)
}
```

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec						
├ 1	38.43 ns	40.38 ns	39.08 ns	38.97 ns	100	12800
├ 5	42.01 ns	45.92 ns	42.66 ns	42.78 ns	100	12800
├ 10	37.13 ns	49.5 ns	45.92 ns	44.41 ns	100	12800
├ 1000	194.3 ns	475.6 ns	212.5 ns	213.4 ns	100	1600
└ 10000000	2.829 ms	8.297 ms	3.073 ms	3.184 ms	100	100

Divan

Generic Type Parameters

```rust
#[divan::bench(
    types = [Vec<i32>, SmallVec<[i32; 10]>],
    args  = [1, 5, 10, 1000, 10000000],
)]
fn alloc_vec<T>(n: i32) -> T
where
    T: FromIterator<i32>,
{
    T::from_iter(1..=n)
}
```

nikolai
vazquez

Divan

Generic Type Parameters

```rust
#[divan::bench(
    types = [Vec<i32>, SmallVec<[i32; 10]>],
    args  = [1, 5, 10, 1000, 10000000],
)]
fn alloc_vec<T>(n: i32) -> T
where
    T: FromIterator<i32>,
{
    T::from_iter(1..=n)
}
```

nikolai
vazquez

Divan

Generic Type Parameters

```rust
#[divan::bench(
    types = [Vec<i32>, SmallVec<[i32; 10]>],
    args  = [1, 5, 10, 1000, 10000000],
)]
fn alloc_vec<T>(n: i32) -> T
where
    T: FromIterator<i32>,
{
    T::from_iter(1..=n)
}
```

nikolai
vazquez

Divan

Generic Type Parameters

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec						
├ SmallVec<[i32; 10]>						
│ ├ 1	3.634 ns	5.954 ns	3.756 ns	3.803 ns	100	102400
│ ├ 5	5.507 ns	7.703 ns	5.669 ns	5.711 ns	100	102400
│ ├ 10	8.721 ns	10.26 ns	8.965 ns	9.119 ns	100	51200
│ ├ 1000	671.5 ns	760 ns	687.1 ns	698 ns	100	800
│ └ 10000000	7.479 ms	9.06 ms	7.621 ms	7.738 ms	100	100
└ Vec<i32>						
├ 1	12.62 ns	14.5 ns	13.19 ns	13.34 ns	100	51200
├ 5	13.76 ns	15.23 ns	13.92 ns	13.94 ns	100	51200
├ 10	15.88 ns	27.11 ns	16.2 ns	16.66 ns	100	51200
├ 1000	80.4 ns	226.2 ns	83.03 ns	85.55 ns	100	3200
└ 10000000	2.824 ms	4.051 ms	3.254 ms	3.27 ms	100	100

Divan

Generic Type Parameters

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec						
├ SmallVec<[i32; 10]>						
│ ├ 1	3.634 ns	5.954 ns	3.756 ns	3.803 ns	100	102400
│ ├ 5	5.507 ns	7.703 ns	5.669 ns	5.711 ns	100	102400
│ ├ 10	8.721 ns	10.26 ns	8.965 ns	9.119 ns	100	51200
│ ├ 1000	671.5 ns	760 ns	687.1 ns	698 ns	100	800
│ └ 10000000	7.479 ms	9.06 ms	7.621 ms	7.738 ms	100	100
└ Vec<i32>						
├ 1	12.62 ns	14.5 ns	13.19 ns	13.34 ns	100	51200
├ 5	13.76 ns	15.23 ns	13.92 ns	13.94 ns	100	51200
├ 10	15.88 ns	27.11 ns	16.2 ns	16.66 ns	100	51200
├ 1000	80.4 ns	226.2 ns	83.03 ns	85.55 ns	100	3200
└ 10000000	2.824 ms	4.051 ms	3.254 ms	3.27 ms	100	100

Divan

Generic Type Parameters

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec						
├ SmallVec<[i32; 10]>						
│ ├ 1	3.634 ns	5.954 ns	3.756 ns	3.803 ns	100	102400
│ ├ 5	5.507 ns	7.703 ns	5.669 ns	5.711 ns	100	102400
│ ├ 10	8.721 ns	10.26 ns	8.965 ns	9.119 ns	100	51200
│ ├ 1000	671.5 ns	760 ns	687.1 ns	698 ns	100	800
│ └ 10000000	7.479 ms	9.06 ms	7.621 ms	7.738 ms	100	100
└ Vec<i32>						
├ 1	12.62 ns	14.5 ns	13.19 ns	13.34 ns	100	51200
├ 5	13.76 ns	15.23 ns	13.92 ns	13.94 ns	100	51200
├ 10	15.88 ns	27.11 ns	16.2 ns	16.66 ns	100	51200
├ 1000	80.4 ns	226.2 ns	83.03 ns	85.55 ns	100	3200
└ 10000000	2.824 ms	4.051 ms	3.254 ms	3.27 ms	100	100

Divan

Measuring Memory

nikolai
vazquez

Divan

Measuring Memory

```rust
#[global_allocator]
static ALLOC: AllocProfiler = AllocProfiler::system();
```

nikolai
vazquez

Divan

Measuring Memory

```
#[global_allocator]
static ALLOC: AllocProfiler<MiMalloc> = AllocProfiler::new(MiMalloc);
```

nikolai
vazquez

Divan

Measuring Memory

benches	fastest	slowest	median	mean	samples	iters
alloc_vec						
SmallVec<[i32; 10]>						
1	3.638 ns	6.079 ns	3.68 ns	3.725 ns	100	102400
5	5.51 ns	7.138 ns	5.592 ns	5.638 ns	100	102400
10	8.887 ns	9.945 ns	8.969 ns	9.033 ns	100	51200
1000	671.5 ns	749.6 ns	697.5 ns	694.2 ns	100	800
	alloc:					
	1	1	1	1		
	4.096 KB	4.096 KB	4.096 KB	4.096 KB		
10000000	7.54 ms	10.99 ms	7.651 ms	7.826 ms	100	100
	alloc:					
	1	1	1	1		
	67.1 MB	67.1 MB	67.1 MB	67.1 MB		
Vec<i32>						
1	12.87 ns	60.72 ns	13.44 ns	14.15 ns	100	51200
	alloc:					
	1	1	1	1		
	4 B	4 B	4 B	4 B		
5	13.85 ns	17.35 ns	14.25 ns	14.27 ns	100	51200
	alloc:					
	1	1	1	1		
	20 B	20 B	20 B	20 B		
10	14.82 ns	17.75 ns	15.15 ns	15.26 ns	100	25600
	alloc:					
	1	1	1	1		
	40 B	40 B	40 B	40 B		
1000	80.37 ns	314.8 ns	85.62 ns	89.61 ns	100	1600
	alloc:					
	1	1	1	1		
	4 KB	4 KB	4 KB	4 KB		
10000000	2.853 ms	3.849 ms	3.304 ms	3.301 ms	100	100
	alloc:					
	1	1	1	1		
	40 MB	40 MB	40 MB	40 MB		

nikolai
vazquez

benches	fastest	slowest	median	mean	samples	iters
└ alloc_vec						
├ SmallVec<[i32; 10]>						
├ 1	3.638 ns	6.079 ns	3.68 ns	3.725 ns	100	102400
├ 5	5.51 ns	7.138 ns	5.592 ns	5.638 ns	100	102400
├ 10	8.887 ns	9.945 ns	8.969 ns	9.033 ns	100	51200
├ 1000	671.5 ns	749.6 ns	697.5 ns	694.2 ns	100	800
	alloc:					
	1	1	1	1		
	4.096 KB	4.096 KB	4.096 KB	4.096 KB		
└ 10000000	7.54 ms	10.99 ms	7.651 ms	7.826 ms	100	100
	alloc:					
	1	1	1	1		
	67.1 MB	67.1 MB	67.1 MB	67.1 MB		
└ Vec<i32>						
├ 1	12.87 ns	60.72 ns	13.44 ns	14.15 ns	100	51200
	alloc:					
	1	1	1	1		
	4 B	4 B	4 B	4 B		

Divan

Counting Throughput

nikolai
vazquez

Divan

Counting Throughput

```rust
#[divan::bench(
    types = [Vec<i32>, SmallVec<[i32; 10]>],
    args  = [1, 5, 10, 1000, 10000000],
)]
fn my_benchmark<T>(bencher: Bencher, n: i32)
where
    T: FromIterator<i32>,
{
    bencher
        .counter(BytesCount::of_many::<i32>(n as usize))
        .bench(|| T::from_iter(1..=n));
}
```

nikolai
vazquez

Divan

Counting Throughput

benches	fastest	slowest	median	mean	samples	iters
my_benchmark						
SmallVec<[i32; 10]>						
1	3.927 ns	6.368 ns	4.009 ns	4.049 ns	100	102400
	1.018 GB/s	628.1 MB/s	997.7 MB/s	987.8 MB/s		
5	5.799 ns	6.531 ns	5.88 ns	5.994 ns	100	102400
	3.448 GB/s	3.062 GB/s	3.401 GB/s	3.336 GB/s		
10	10.15 ns	14.22 ns	10.31 ns	10.34 ns	100	51200
	3.939 GB/s	2.812 GB/s	3.877 GB/s	3.866 GB/s		
1000	796.5 ns	843.4 ns	817.4 ns	817.4 ns	100	800
	5.021 GB/s	4.742 GB/s	4.893 GB/s	4.893 GB/s		
10000000	7.516 ms	11.73 ms	7.703 ms	7.879 ms	100	100
	5.321 GB/s	3.409 GB/s	5.192 GB/s	5.076 GB/s		
Vec<i32>						
1	12.59 ns	32.85 ns	13.08 ns	13.31 ns	100	51200
	317.6 MB/s	121.7 MB/s	305.7 MB/s	300.3 MB/s		
5	13.81 ns	30.09 ns	13.89 ns	14.2 ns	100	51200
	1.447 GB/s	664.6 MB/s	1.439 GB/s	1.407 GB/s		
10	15.44 ns	22.6 ns	15.85 ns	16.56 ns	100	51200
	2.59 GB/s	1.769 GB/s	2.523 GB/s	2.414 GB/s		
1000	80.34 ns	260 ns	85.59 ns	88.16 ns	100	1600
	49.78 GB/s	15.37 GB/s	46.73 GB/s	45.36 GB/s		
10000000	2.813 ms	3.803 ms	3.168 ms	3.205 ms	100	100
	14.21 GB/s	10.51 GB/s	12.62 GB/s	12.47 GB/s		

nikolai
vazquez

Divan

Features

- Simple API
- Visually compact output
- Parameters as benchmark cases
- Compare generic functions across types
- Counting throughput
- Measuring memory allocations
- Check multi-thread contention

nikolai
vazquez

Divan

Motivations

nikolai
vazquez

Other Tools

nikolai
vazquez

Other Tools

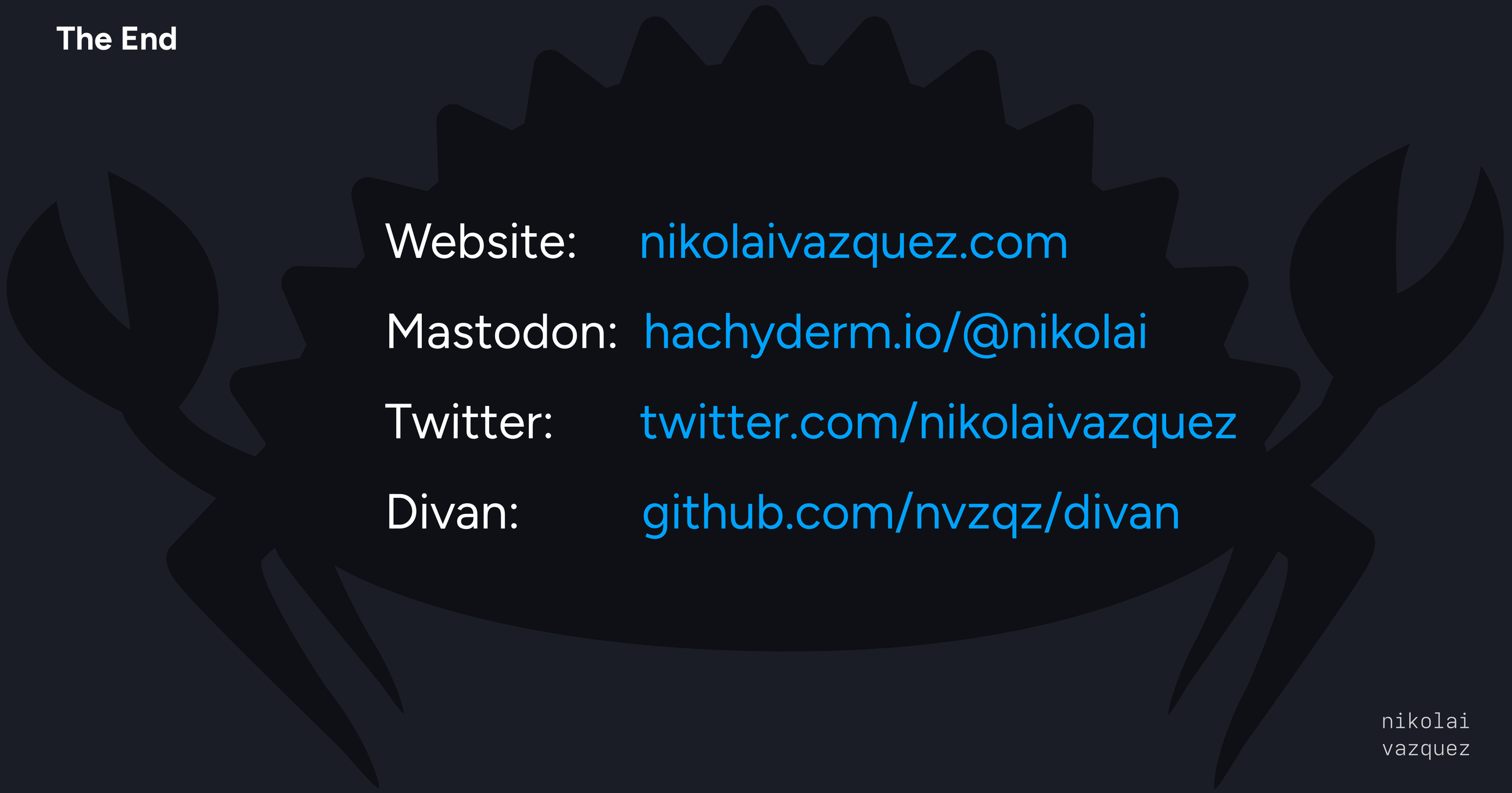
- Criterion
- Tango
- Flamegraphs
- DHAT

Closing Thoughts

nikolai
vazquez


The End

The End



Website: nikolaivazquez.com

Mastodon: hachyderm.io/@nikolai

Twitter: twitter.com/nikolaivazquez

Divan: github.com/nvzqz/divan

nikolai
vazquez