



1000X

WORLD TOUR

13 CITIES | 6 DAYS



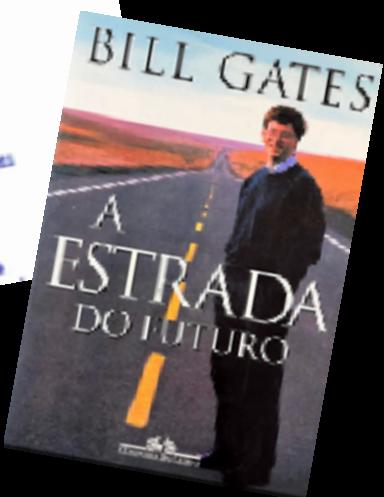


1000X

WORLD TOUR

13 CITIES | 6 DAYS

1000X



1000x

OLAP

Online Analytical Processing

1000x

“A história não se repete, mas rima.”

Mark Twain

1000X

The World is Becoming More Transactional

In the past 10 years, transaction volumes surged 100x-1,000x across cloud, energy, and gaming — and up to 10,000x in real-time payments. Mainframe or not, legacy systems are at a breaking point.

2015

2025

10,000x
Real-Time Payments

1,000x
Energy, Ride-Sharing
100x
In-Game Economies

1000X

The World is Becoming More Transactional

In the past 10 years, transaction volumes surged 100x-1,000x across cloud, energy, and gaming — and up to 10,000x in real-time payments. Mainframe or not, legacy systems are at a breaking point.

2015

2025

10,000x
Real-Time Payments

1,000x
Energy, Ride-Sharing
100x
In-Game Economies

1000X

The World is Becoming More Transactional

In the past 10 years, transaction volumes surged 100x-1,000x across cloud, energy, and gaming — and up to 10,000x in real-time payments. Mainframe or not, legacy systems are at a breaking point.

2015

2025

10,000x
Real-Time Payments

1,000x
Energy, Ride-Sharing
100x
In-Game Economies

1000x

Transações estão em toda parte!



1000x

Transações estão em toda parte!



1000x

OLTP

Online Transaction Processing

1000X

THE FINANCIAL TRANSACTIONS DATABASE ^{1000X} FASTER

To power the next 30 years of Online Transaction Processing.

Install TigerDex

Read the Docs



1000X

THE FINANCIAL TRANSACTIONS DATABASE ^{1000X} FASTER

To power the next 30 years of Online Transaction Processing.

Install TigerDex

Read the Docs



1000x

A filosofia por trás do processo de desenvolvimento do TigerBeetle

Rafael Batiati

Senior Engineer @ TigerBeetle

x.com/rbatiati

github.com/batiati

1000x

10^1 = Systems Programming

1000x

10^1 = Systems Programming



Systems programming, or system programming, is the activity of programming^[1] computer system software. The primary distinguishing characteristic of systems programming when compared to application programming is that application programming aims to produce software which provides services to the user directly (e.g. word processor), whereas systems programming aims to produce software and software platforms which provide services to other software. are performance constrained, or both (e.g. operating systems, computational science applications, game engines, industrial automation, and software as a service applications).^[1]

1000x

10^1 = Systems Programming

*“A way of modeling
software development”*



Andrew Kelley
Criador da linguagem
de programação ZIG

<https://youtu.be/Qncdi-Fg0-I>



1000x

10^1 = Systems Programming

*“A way of modeling
software development”*



Andrew Kelley
Criador da linguagem
de programação ZIG

<https://youtu.be/Qncdi-Fg0-I>



1000x

10^1 = Systems Programming

*“A way of modeling
software development”*



Andrew Kelley
Criador da linguagem
de programação ZIG

<https://youtu.be/Qncdi-Fg0-I>



1000x

10^1 = Systems Programming

“The purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise.”

Edsger W. Dijkstra

1000x

10^1 = Systems Programming

“The purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise.”

Edsger W. Dijkstra

1000x

10^1 = Systems Programming

The Power of Zero Dependencies

1000x

10^1 = Systems Programming

The Power of Zero Dependencies

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

“Go slow to go fast”



1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

TIGER STYLE



100

$10^2 =$

Tempo =

```
1 in execute_expire_pending_transfers(self: *StateMachine, timestamp: u64) {
2     assert(timestamp > self.commit_timestamp);
3     assert(self.scan_lookup_results.items.len > 0);
4
5     // ...
6
7     assert(result_count < result_max);
8     assert(self.scan_lookup_buffer_index > 0);
9     assert(self.scan_lookup_buffer_index == result_count + 0izeOf(Transfer));
10
11    for (transfers_pending, & p, index) {
12        assert(p.flags.pending);
13        assert(p.timeout > 0);
14
15        const event_timestamp = timestamp - transfers_pending.len + index + 1;
16        assertTimestampRange.valid(event_timestamp));
17        assert(self.commit_timestamp < event_timestamp);
18
19        const expires_at = p.timestamp + p.timeout.us();
20        assert(expires_at <= event_timestamp);
21
22        const dr_account = self.get_account(p.debit_account_id);
23        assert(dr_account.debits_pending > p.amount);
24
25        const cr_account = self.get_account(p.credit_account_id);
26        assert(cr_account.credits_pending > p.amount);
27
28        // Pending transfers can expire in closed accounts.
29        assert(dr_account.flags.closed || dr_account.flags.closed);
30        assert(cr_account.flags.closed || cr_account.flags.closed);
31
32        // ...
33
34        const transfer_pending = self.get_transfer_pending(p.timestamp);
35        assert(p.timestamp == transfer_pending.timestamp);
36        assert(transfer_pending.state == .pending);
37        self.transfer_update_pending_status(&transfer_pending, .expired);
38    }
39
40    // ...
41 }
```

:ado

+ Testes + Incidentes

100

10² =

Tempo =

```
1 fn execute_expire_pending_transfers(self: *StateMachine, timestamp: u64) usize {
2     assert(timestamp > self.commit_timestamp);
3     assert(self.scan_lookup_results.items.len > 0);
4
5     // ...
6
7     assert(result_count < result_max);
8     assert(self.scan_lookup_buffer_index > 0);
9     assert(self.scan_lookup_buffer_index == result_count * @sizeOf(Transfer));
10
11    for (transfers_pending, 0..) |*p, index| {
12        assert(p.flags.pending);
13        assert(p.timeout > 0);
14
15        const event_timestamp = timestamp - transfers_pending.len + index + 1;
16        assert(TimestampRange.valid(event_timestamp));
17        assert(self.commit_timestamp < event_timestamp);
18
19        const expires_at = p.timestamp + p.timeout_ns();
20        assert(expires_at <= event_timestamp);
21
22        const dr_account = self.get_account(p.debit_account_id);
23        account!(dr_account.debit_balance >= p.amount);
```

precondições

100

10² =

Tempo =

```
1 fn execute_expire_pending_transfers(self: *StateMachine, timestamp: u64) usize {
2     assert(timestamp > self.commit_timestamp);
3     assert(self.scan_lookup_results.items.len > 0);
4
5     // ....
6
7     assert(result_count < result_max);
8     assert(self.scan_lookup_buffer_index > 0);
9     assert(self.scan_lookup_buffer_index == result_count * @sizeOf(Transfer));
10
11    for (transfers_pending, 0..) |*p, index| {
12        assert(p.flags.pending);
13        assert(p.timeout > 0);
14
15        const event_timestamp = timestamp - transfers_pending.len + index + 1;
16        assert(TimestampRange.valid(event_timestamp));
17        assert(self.commit_timestamp < event_timestamp);
18
19        const expires_at = p.timestamp + p.timeout_ns();
20        assert(expires_at <= event_timestamp);
21
22        const dr_account = self.get_account(p.debit_account_id);
23        assert(dr_account != null);
24    }
25}
```

limits

100

10² =

Tempo =

```
11  for (transfers_pending, 0..) |*p, index| {
12    assert(p.flags.pending);
13    assert(p.timeout > 0);
14
15    const event_timestamp = timestamp - transfers_pending.len + index + 1;
16    assert(TimestampRange.valid(event_timestamp));
17    assert(self.commit_timestamp < event_timestamp);
18
19    const expires_at = p.timestamp + p.timeout_ns();
20    assert(expires_at ≤ event_timestamp);
21
22    const dr_account = self.get_account(p.debit_account_id);
23    assert(dr_account.debits_pending ≥ p.amount);
24
25    const cr_account = self.get_account(p.credit_account_id);
26    assert(cr_account.credits_pending ≥ p.amount);
27
28    // Pending transfers can expire in closed accounts.
29    assert(dr_account.flags.closed || !dr_account.flags.closed);
30    assert(cr_account.flags.closed || !cr_account.flags.closed);
31
32    // ....
33
```

regras de negócio

100

10² =

Tempo =

```
11  for (transfers_pending, 0..) |*p, index| {
12      assert(p.flags.pending);
13      assert(p.timeout > 0);
14
15      const event_timestamp = timestamp - transfers_pending.len + index + 1;
16      assert(TimestampRange.valid(event_timestamp));
17      assert(self.commit_timestamp < event_timestamp);
18
19      const expires_at = p.timestamp + p.timeout_ns();
20      assert(expires_at ≤ event_timestamp);
21
22      const dr_account = self.get_account(p.debit_account_id);
23      assert(dr_account.debits_pending ≥ p.amount);
24
25      const cr_account = self.get_account(p.credit_account_id);
26      assert(cr_account.credits_pending ≥ p.amount);
27
28      // Pending transfers can expire in closed accounts.
29      assert(dr_account.flags.closed || !dr_account.flags.closed);
30      assert(cr_account.flags.closed || !cr_account.flags.closed);
31
32      // ....
33
```

no-op!

100

10² =

Tempo =

```
20     assert(expires_at <= event_timestamp);
21
22     const dr_account = self.get_account(p.debit_account_id);
23     assert(dr_account.debits_pending >= p.amount);
24
25     const cr_account = self.get_account(p.credit_account_id);
26     assert(cr_account.credits_pending >= p.amount);
27
28     // Pending transfers can expire in closed accounts.
29     assert(dr_account.flags.closed || !dr_account.flags.closed);
30     assert(cr_account.flags.closed || !cr_account.flags.closed);
31
32     // ...
33
34     const transfer_pending = self.get_transfer_pending(p.timestamp);
35     assert(p.timestamp == transfer_pending.timestamp);
36     assert(transfer_pending.status == .pending);
37     self.transfer_update_pending_status(&transfer_pending, .expired);
38 }
39
40 // ...
41 }
```

ação

1000x

$10^2 = \text{Esfoga}$

Tempo = Design +

```
1 fn execute_size_pending_transfers(self: *StateMachine, timestamp: u64) usize {
2     assert(timestamp > self.commit_timestamp);
3     assert(self.scan_loops == 0);
4     assert(self.scan_loops == 0);
5     // ...
6
7     assert(result_count == 0);
8     assert(self.scan_loops == 0);
9     assert(self.scan_loops == 0);
10    for transfers_pending in self.transfers_pending {
11        assert(transfers_pending.timestamp != 0);
12        assert(transfers_pending.timestamp > self.commit_timestamp);
13        const event_timestamp = transfers_pending.timestamp;
14        assert(event_timestamp == timestamp);
15        assert(TimestampRange::Valid == event_timestamp);
16        assert(self.commit_timestamp == 0);
17        assert(self.commit_timestamp == 0);
18        assert(transfers_pending.status == TransferPendingStatus::Pending);
19        assert(transfers_pending.timestamp == timestamp);
20        assert(transfers_pending.timestamp == timestamp);
21        const dr_account = transfers_pending.dr_account;
22        assert(dr_account.id == 0);
23        assert(dr_account.id == 0);
24        const cr_account = transfers_pending.cr_account;
25        assert(cr_account.id == 0);
26        assert(cr_account.id == 0);
27        // Pending transfer
28        assert(cr_account.fee == 0);
29        assert(cr_account.fee == 0);
30        assert(cr_account.fee == 0);
31        assert(cr_account.fee == 0);
32        // ...
33
34        const transfer_pending = self.get_transfer_pending(p.timestamp);
35        assert(p.timestamp == transfer_pending.timestamp);
36        assert(transfer_pending.status == .pending);
37        self.transfer_update_pending_status(&transfer_pending, .expired);
38    }
39    // ...
40 }
41 }
```

+ Incidentes
tudo outra vez!!!

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

**+
Revisão**



1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

**+
Revisão**



1000x

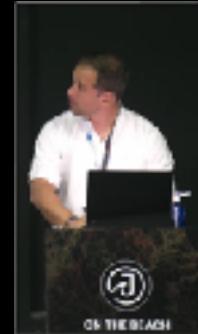
$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes



Federico Lorenzi

<https://youtu.be/P9nLS2reUOo>

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

Deterministic **S**imulation **T**esting

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

Deterministic **S**imulation **T**esting

100

$10^2 = [$

Tempo = [



```
fn sum_two_numbers(a: int, b: int) int {
    return a + b;
}

fn test_sum_two_numbers() void {
    const sum = sum_two_numbers(1, 2);
    expect_equals(sum, 3);
}
```

Deterministic Simulation Testing

100

$10^2 = [$

Tempo = [



```
fn sum_two_numbers(a: int, b: int) int {
    return a + b;
}

fn test_sum_two_numbers() void {
    const sum = sum_two_numbers(1, 2);
    expect_equals(sum, 3);
}
```

Deterministic Simulation Testing
comportamento esperado

100

$10^2 = [$

Tempo = [

```
fn sum_two_numbers(a: int, b: int) int {
    const sum = a + b;
    assert(sum >= a);
    assert(sum >= b);

    return sum;
}

fn fuzz_sum_two_numbers() void {
    for (0..10_000) {
        const a = random_int();
        const b = random_int();
        const sum = sum_two_numbers(a, b);
        expect_equals(sum, a + b);
    }
}
```

100

$10^2 = [$

Tempo = [

```
fn sum_two_numbers(a: int, b: int) int {
    const sum = a + b;
    assert(sum >= a);
    assert(sum >= b);

    return sum;
}

fn fuzz_sum_two_numbers() void {
    for (0..10_000) {
        const a = random_int();
        const b = random_int();
        const sum = sum_two_numbers(a, b);
        expect_equals(sum, a + b);
    }
}
```

inputs
randômicos

100

$10^2 = [$

Tempo = [

```
fn sum_two_numbers(a: int, b: int) int {
    const sum = a + b;
    assert(sum >= a);
    assert(sum >= b);

    return sum;
}

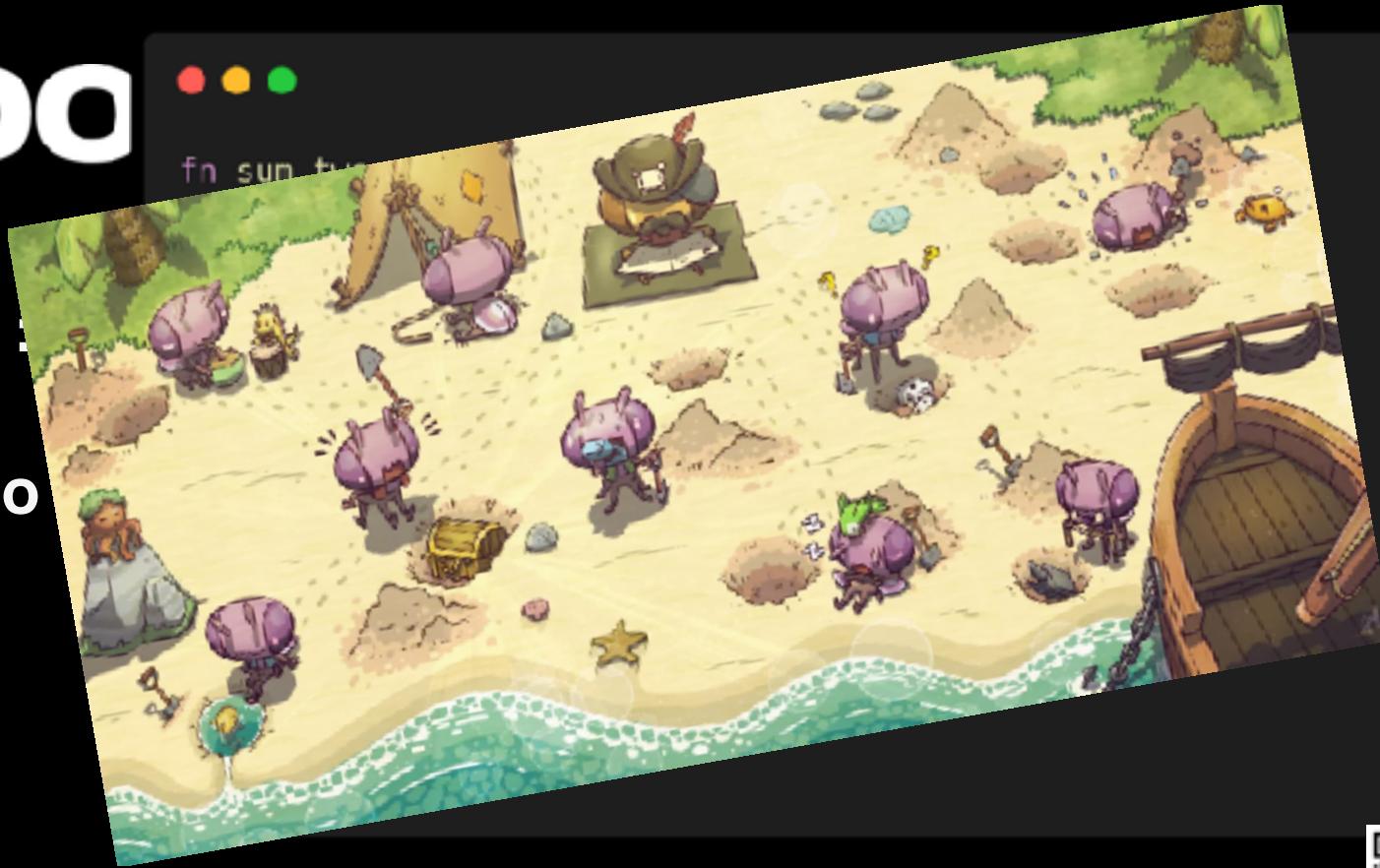
fn fuzz_sum_two_numbers() void {
    for (0..10_000) {
        const a = random_int();
        const b = random_int();
        const sum = sum_two_numbers(a, b);
        expect_equals(sum, a + b);
    }
}
```

mímica da
regra de negócio

100

10²

Tempo



100

$10^2 = [$

Tempo = [

```
fn sum_two_numbers(a: int, b: int) int {
    const sum = a + b;
    assert(sum >= a);
    assert(sum >= b);

    return sum;
}
```

```
fn fuzz_sum_two_numbers() void {
    for (0..10_000) {
        const a = random_int();
        const b = random_int();
        const sum = sum_two_numbers(a, b);
        expect_equals(sum, a + b);
    }
}
```

overflow!

100

$10^2 = [$

Tempo = [

```
fn sum_two_numbers(a: int, b: int) int {
    const sum = a + b;
    assert(sum >= a);
    assert(sum >= b);

    return sum;
}
```

```
fn fuzz_sum_two_numbers() void {
    for (0..10_000) {
        const a = random_int();
        const b = random_int();
        const sum = sum_two_numbers(a, b);
        expect_equals(sum, a + b);
    }
}
```

invariante

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

Deterministic **S**imulation **T**esting

100

10² = |

Tempo = [

```
> ./zig/zig build fuzz -- lsm_scan
info(fuzz): Fuzz seed = 2372797884104813212
info(lsm_scan_fuzz): commits = 986
info(lsm_scan_fuzz): query_specs[0]: (index_82 OR index_07 OR index_88 OR index_10) ascending
info(lsm_scan_fuzz): query_specs[1]: (index_81 AND index_11 AND (index_07 OR index_89 OR index_85 OR index_86) AND index_88 AND (index_08 OR (index_84 AND index_18) OR index_18)) descending
info(lsm_scan_fuzz): query_specs[2]: (((index_86 OR index_89) AND index_84 AND index_82 AND index_88) OR (index_03 AND index_85) OR ((index_11 AND index_13) OR index_61 OR index_10))) ascending
info(lsm_scan_fuzz): query_specs[3]: (index_12 AND index_84 AND index_86 AND index_11 AND index_82) descending
info(lsm_scan_fuzz): query_specs[4]: ((index_08 OR index_01) AND index_03 AND (index_89 OR index_05 OR index_12 OR index_84 OR index_86) AND index_82) ascending
info(lsm_scan_fuzz): query_specs[5]: (index_88 OR index_13 OR index_82 OR index_87 OR index_10) descending
info(lsm_scan_fuzz): query_specs[6]: (((index_10 OR (index_11 AND index_05 AND index_88) OR index_84 OR index_09) AND (index_13 OR index_03) AND index_87 AND (index_12 AND index_86))) descending
info(lsm_scan_fuzz): query_specs[7]: (index_87 AND (index_89 OR (index_05 AND index_08 AND index_13) OR index_02 OR index_01) AND index_12) descending
info(lsm_scan_fuzz): Passed!
info(fuzz): done in 30.456s
```

Deterministic **S**imulation **T**esting

100

10² = |

Tempo = [

```
> ./zig/zig build fuzz -- lsm_scan
Info(fuzz): Fuzz seed = 2372797884104813212
Info(lsm_scan_fuzz): commits = 986
Info(lsm_scan_fuzz): query_specs[0]: (index_82 OR index_07 OR index_88 OR index_10) ascending
Info(lsm_scan_fuzz): query_specs[1]: (index_81 AND index_11 AND (index_07 OR index_89 OR index_85 OR index_86) AND index_88 AND (index_08 OR (index_84 AND index_18) OR index_10)) descending
Info(lsm_scan_fuzz): query_specs[2]: (((index_86 OR index_89) AND index_84 AND index_82 AND index_88) OR (index_03 AND index_85) OR ((index_11 AND index_13) OR index_61 OR index_10)) ascending
Info(lsm_scan_fuzz): query_specs[3]: (index_12 AND index_84 AND index_86 AND index_11 AND index_82) descending
Info(lsm_scan_fuzz): query_specs[4]: ((index_08 OR index_81) AND index_03 AND (index_89 OR index_05 OR index_12 OR index_84 OR index_86) AND index_82) ascending
Info(lsm_scan_fuzz): query_specs[5]: (index_88 OR index_13 OR index_82 OR index_87 OR index_10) descending
Info(lsm_scan_fuzz): query_specs[6]: (((index_10 OR (index_11 AND index_05 AND index_88) OR index_84 OR index_09) AND (index_13 OR index_03) AND index_87 AND (index_12 AND index_86))) descending
Info(lsm_scan_fuzz): query_specs[7]: (index_87 AND (index_89 OR (index_05 AND index_08 AND index_13) OR index_02 OR index_81) AND index_12) descending
Info(lsm_scan_fuzz): Passed!
Info(fuzz): done in 30.456s
```

Deterministic **S**imulation **T**esting

condições válidas
aleatórias

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

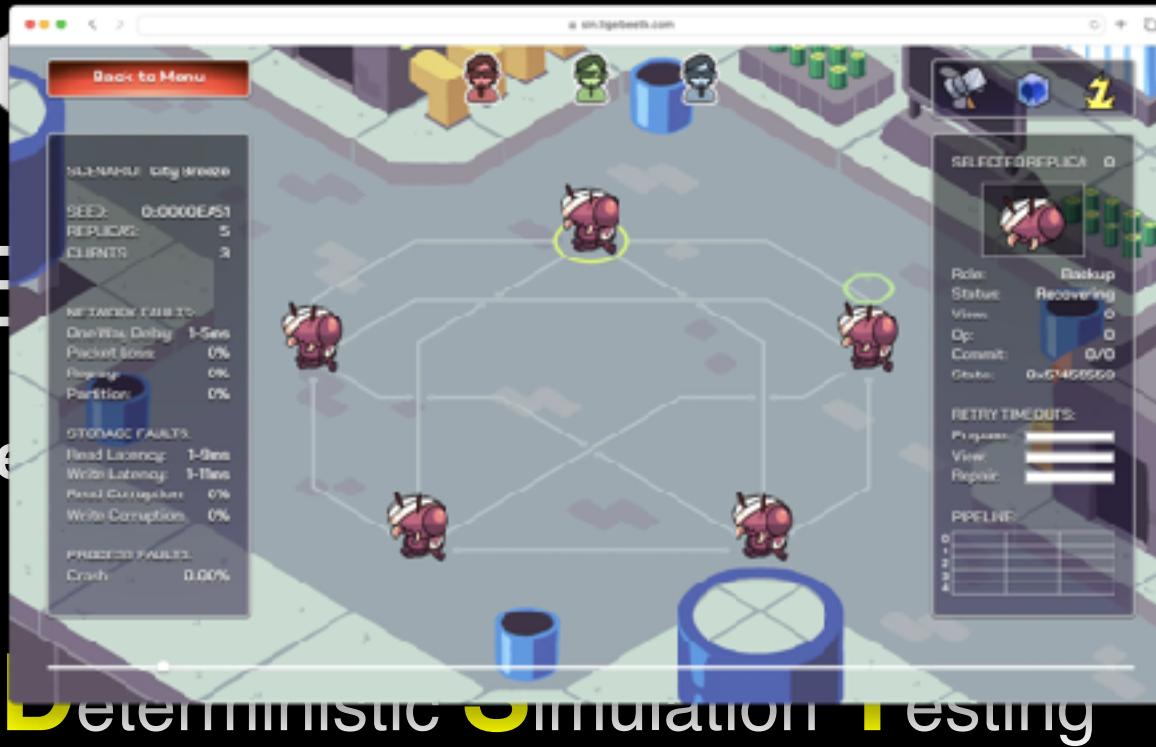
Deterministic **S**imulation **T**esting

1000

$$10^2 = E$$

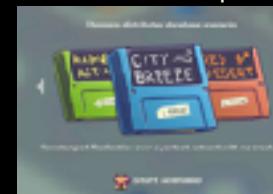
Tempo = De

cidentes



Deterministic Simulation Testing

<https://sim.tigerbeetle.com>



1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

Deterministic **S**imulation **T**esting

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

Deterministic **S**imulation **T**esting

1000x

$10^2 = \text{Esforço} \text{ vs Resultado}$

Tempo = Design + Desenvolvimento + Testes + Incidentes

1000x

$10^2 =$

Tempo =



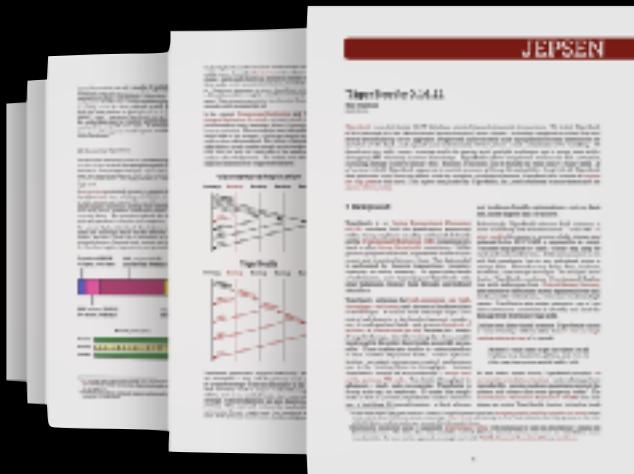
Incidentes



1000x

10^2 = Esforço vs Resultado

Tempo = Design + Desenvolvimento + Testes + **Incidentes**



1000x

10²

Tempo



s + Incidentes



1000X

$10^2 =$

Tempo = D

Incidentes



1000x

$10^2 =$



Tempo :

```
fn assert(ok: bool) void {
    if (!ok) xxxxxxxx
}
```

1000x

$10^3 = \text{Build Trust and Have Fun}$

“Cultivar confiança e se divertir”



1000x

$10^3 = \text{Build Trust and Have Fun}$

“Cultivar confiança e se divertir”



1000x

$10^3 = \text{Build Trust and Have Fun}$

“Cultivar confiança e se divertir”



1000x

Muito obrigado!

- > Perguntas?
- > Comentários!

