

CPPNOW 2019

# RISE OF THE STATE MACHINES

Kris Jusiak, Quantlab Financial

[kris@jusiak.net](mailto:kris@jusiak.net) | [@KrisJusiak](https://twitter.com/@KrisJusiak) | [LINKEDIN.COM/IN/KRIS-JUSIAK](https://www.linkedin.com/in/kris-jusiak)

# MOTIVATION

# RISE OF THE...

# STATE MACHINES

# STATE MACHINES

---

# STATE MACHINES

---

- SKYNET: JUST A SOFTWARE GLITCH

# STATE MACHINES

---

- SKYNET: JUST A SOFTWARE GLITCH
  - BECAUSE IT WASN'T IMPLEMENTED USING DECLARATIVE STATE MACHINES?

# OUTLINE

- 
- 
- 
- 
- 
- 
- 
-

# OUTLINE

- Problem

- 

- 



- 

-

# OUTLINE

- Problem
- State Machine
- 



# OUTLINE

- Problem
- State Machine
- Solutions



# OUTLINE

- Problem
- State Machine
- Solutions
  - Naive
  - 
  - 
  - 
  -

# OUTLINE

- Problem
- State Machine
- Solutions
  - Naive
  - STL
  -
- 
-

# OUTLINE

- Problem
- State Machine
- Solutions
  - Naive
  - STL
  - Boost
- 
-

# OUTLINE

- Problem
- State Machine
- Solutions
  - Naive
  - STL
  - Boost
- Benchmarks
-

# OUTLINE

- Problem
  - State Machine
  - Solutions
    - Naive
    - STL
    - Boost
  - Benchmarks
  - Summary
-

# OUTLINE

- Problem
- State Machine
- Solutions
  - Naive
  - STL
  - Boost
- Benchmarks
- Summary

---

darkblue background - something to remember ✓

# PROBLEM

---

---

# PROBLEM

Feature: Connection

---

# PROBLEM

Feature: Connection

---

Scenario: Establish connection

---

# PROBLEM

Feature: Connection

---

Scenario: Establish connection

Given I don't have a connection

---

# PROBLEM

Feature: Connection

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

---

# PROBLEM

Feature: Connection

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

Then I should try to establish the connection

---

# PROBLEM

Feature: Connection

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

Then I should try to establish the connection

When I receive an established acknowledgement

---

# PROBLEM

Feature: Connection

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

Then I should try to establish the connection

When I receive an established acknowledgement

Then I should have been connected

---

# PROBLEM

Feature: Connection

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

Then I should try to establish the connection

When I receive an established acknowledgement

Then I should have been connected

---

Scenario: Disconnect

...

# PROBLEM

*What's the*

- 
- 
- 

*way to satisfy/implement Connection  
requirements?*

# PROBLEM

*What's the*

- most readable
- 
- 

*way to satisfy/implement Connection  
requirements?*

# PROBLEM

*What's the*

- most readable
- most maintainable
- 

*way to satisfy/implement Connection  
requirements?*

# PROBLEM

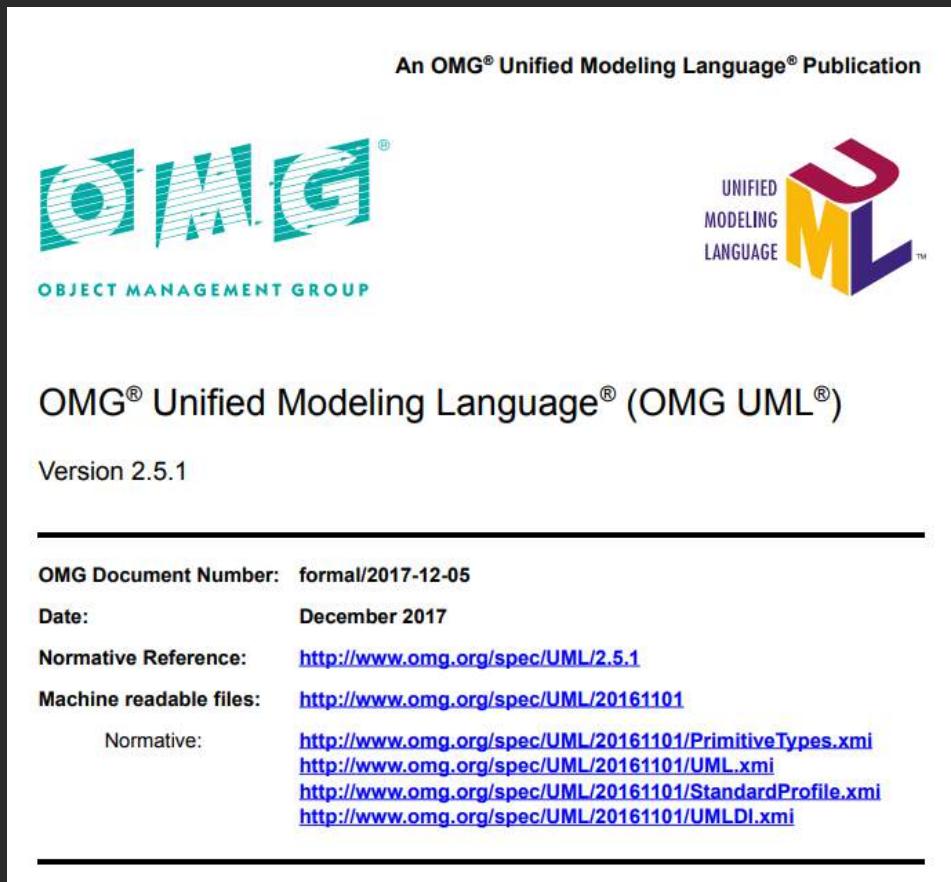
*What's the*

- most readable
- most maintainable
- most efficient

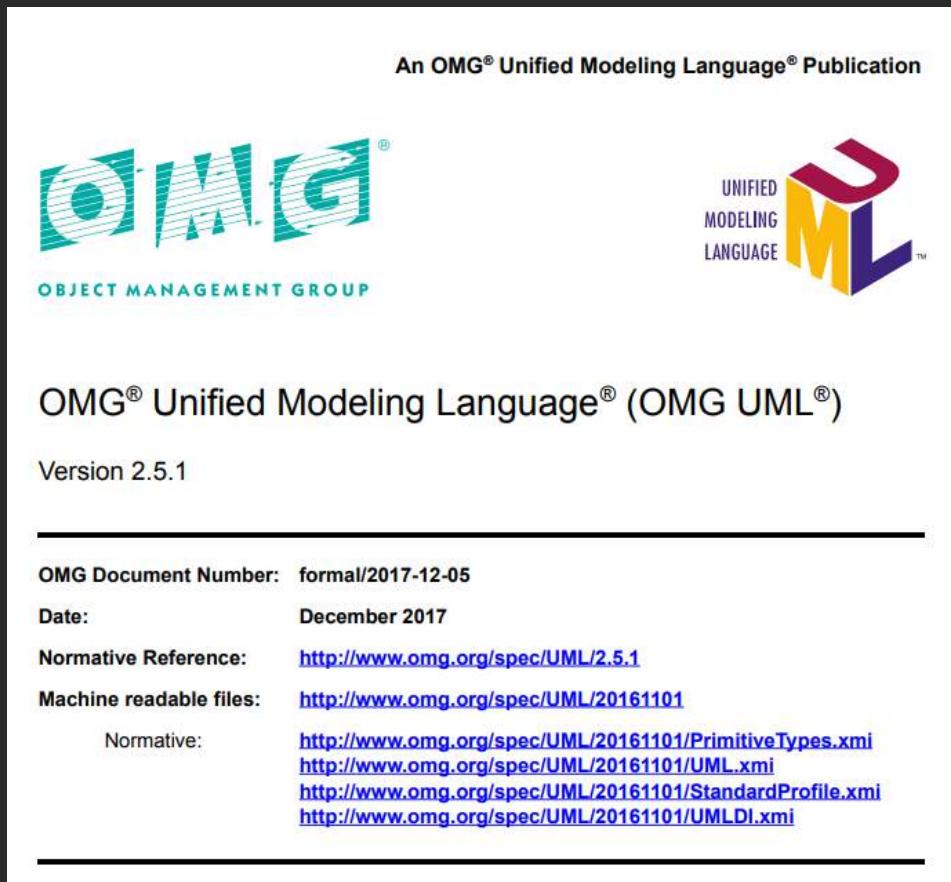
*way to satisfy/implement Connection  
requirements?*

# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5



# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5



[HTTPS://WWW.OMG.ORG/SPEC/UML/2.5.1/PDF](https://www.omg.org/spec/UML/2.5.1/PDF)

# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

Feature: Connection

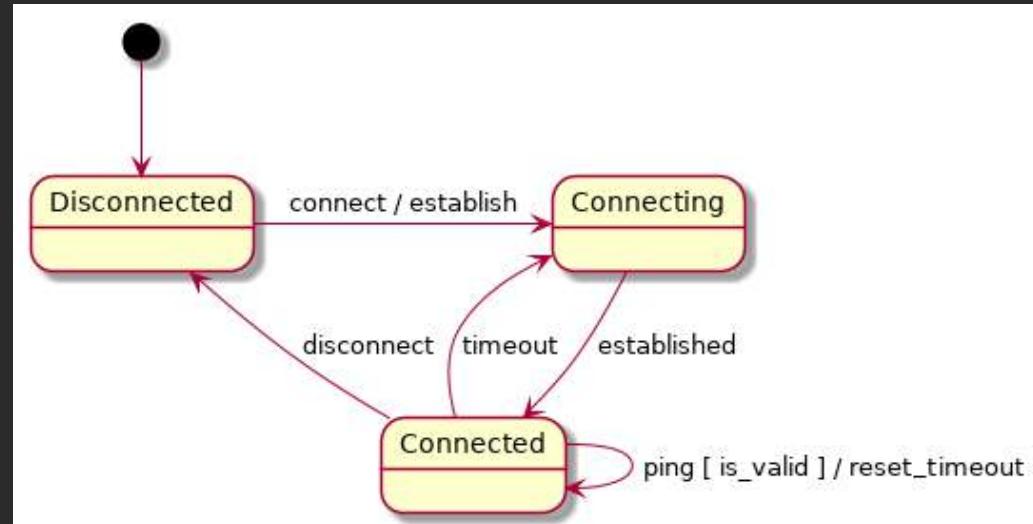
Scenario: Connect

Scenario: Disconnect

Scenario: ...

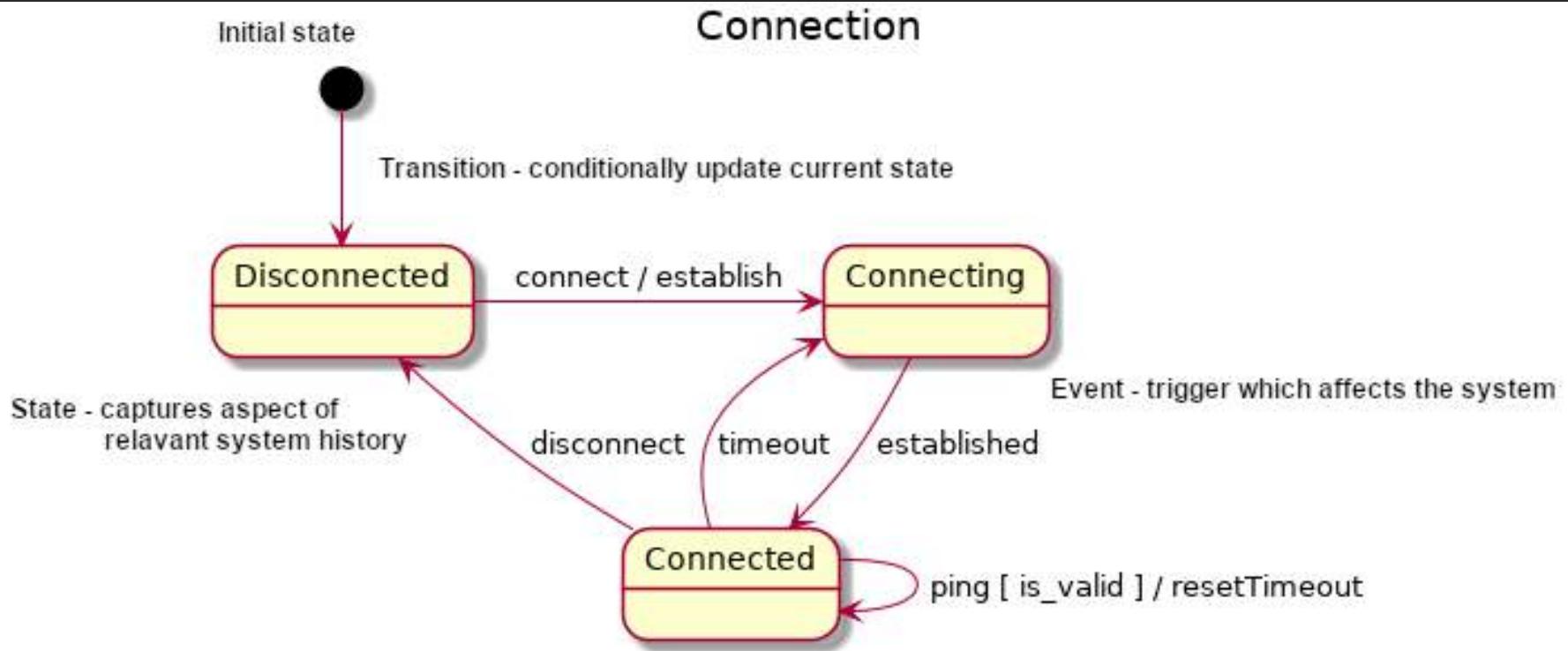
# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

Feature: Connection  
Scenario: Connect  
Scenario: Disconnect  
Scenario: ...

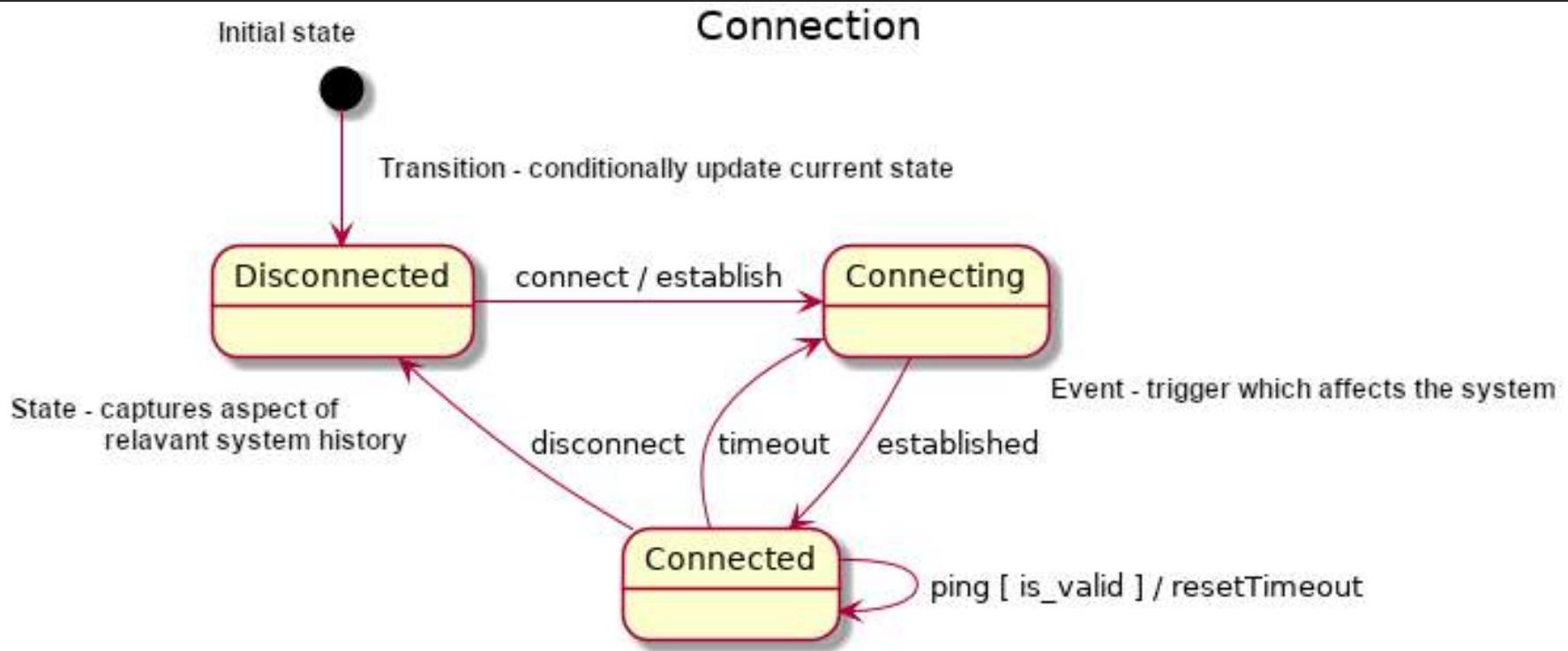


# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

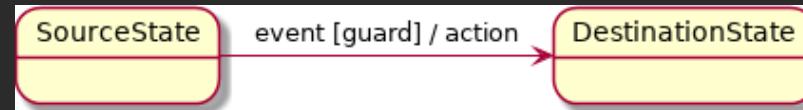
# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5



# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

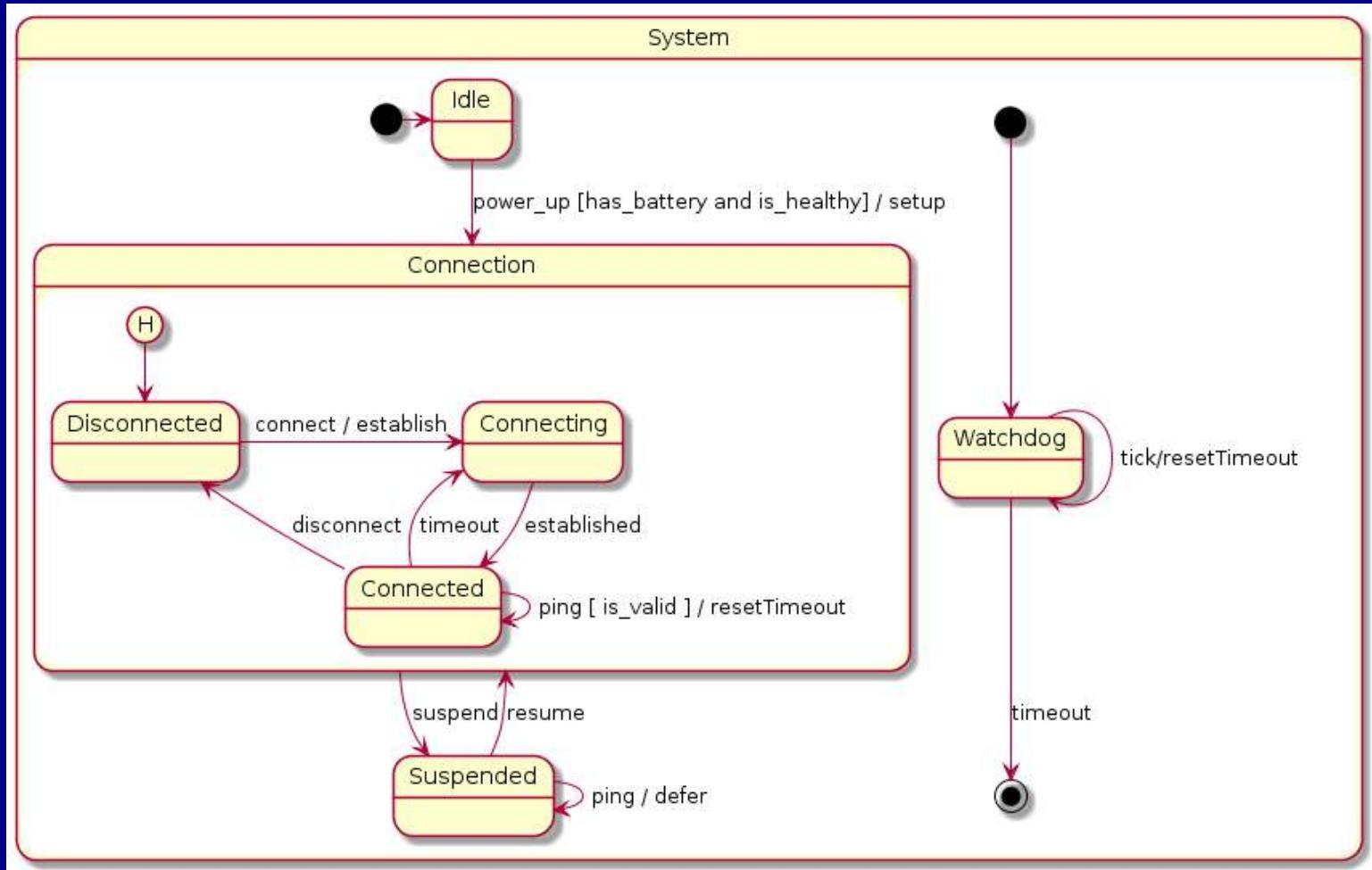


## Transition - UML



# STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS

# STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS



# SOLUTIONS

# SOLUTIONS

Naive	STL	Boost
If/Else (C++98)	std::variant (C++17)	Boost.Statechart (C++98)
Switch/Enum (C++98)	Coroutines (C++20)	Boost.MSM (C++98)
Inheritance / State pattern (C++98)		[Boost].SML (C++14)

# COMMON - IMPLEMENTATION

# COMMON - IMPLEMENTATION

## EVENTS

```
struct connect{};  
struct established{};  
struct ping{};  
struct disconnect{};  
struct timeout{};
```

# COMMON - IMPLEMENTATION

## EVENTS

```
struct connect{};  
struct established{};  
struct ping{};  
struct disconnect{};  
struct timeout{};
```

## GUARDS

```
constexpr auto is_valid = [] (auto const& event) { return true; };
```

# COMMON - IMPLEMENTATION

## EVENTS

```
struct connect{};  
struct established{};  
struct ping{};  
struct disconnect{};  
struct timeout{};
```

## GUARDS

```
constexpr auto is_valid = [] (auto const& event) { return true; };
```

## ACTIONS

```
constexpr auto establish = [] { std::puts("establish"); };  
constexpr auto close = [] { std::puts("close"); };  
constexpr auto reset_timeout = [] { std::puts("reset_timeout"); };
```

# NAIVE

*C++98 features*

# NAIVE - IF/ELSE - IMPLEMENTATION

## NAIVE - IF/ELSE - IMPLEMENTATION

```
class Connection {  
    // Implicit states using booleans  
    bool disconnected = true, connected = false, connecting = false;
```

## NAIVE - IF/ELSE - IMPLEMENTATION

```
class Connection {
    // Implicit states using booleans
    bool disconnected = true, connected = false, connecting = false;

    constexpr void process_event(connect const&)
    {
        if (disconnected) {
            establish();
            connected = disconnected = false; // Just in case reset it all!
            connecting = true; // Set the new state
        }
    }
}
```

## NAIVE - IF/ELSE - IMPLEMENTATION

```
class Connection {
    // Implicit states using booleans
    bool disconnected = true, connected = false, connecting = false;

    constexpr void process_event(connect const&)
    {
        if (disconnected) {
            establish();
            connected = disconnected = false; // Just in case reset it all!
            connecting = true; // Set the new state
        }
    }

    constexpr void process_event(ping const& event)
    {
        if (connected and is_valid(event)) {
            reset_timeout();
        } // Stay in the current state
    }
}
```

# NAIVE - IF/ELSE - FULL EXAMPLE

The screenshot shows the Godbolt compiler explorer interface. On the left, the C++ source code is displayed with line numbers 43 through 65. The code implements a naive if/else logic for processing events. Lines 43-45 handle a timeout reset. Lines 47-54 implement the `process_event` function, which checks if the connection is connected. If connected, it calls `establish()`, sets `connecting` to true, and sets `connected` to false. Lines 56-65 show the `main` function, which creates a `Connection` object and processes various events: connect, established, ping, disconnect, connect, established, and ping.

The right side of the interface shows the generated assembly code. The assembly output is colored by section: `.text` (red), `.L.str` (green), `.L.str.1` (blue), and `.L.str.2` (yellow). The assembly code corresponds to the C++ code, showing the execution flow and the assembly equivalents for each instruction. The assembly code includes `pushq %rax`, `movl $.L.str, %edi`, `callq puts`, `movl $.L.str.1, %edi`, `callq puts`, `movl $.L.str.2, %edi`, `callq puts`, `movl $.L.str, %edi`, `callq puts`, `movl $.L.str.1, %edi`, `callq puts`, `xorl %eax, %eax`, `popq %rcx`, and `retq`. It also includes string literals for "establish", "reset\_timeout", and "connect".

```
43     |     reset_timeout();
44   } 
45 }
46
47 constexpr void process_event(timeout const&)
48 {
49     if (connected) {
50         establish();
51         connecting = true;
52         connected = disconnected = false;
53     }
54 };
55
56 int main()
57 {
58     Connection connection{};
59     connection.process_event(connect{});
60     connection.process_event(established{});
61     connection.process_event(ping{});
62     connection.process_event(disconnect{});
63     connection.process_event(connect{});
64     connection.process_event(established{});
65     connection.process_event(ping{});
}

```

# @main

```
1 main:
2     pushq %rax
3     movl $.L.str, %edi
4     callq puts
5     movl $.L.str.1, %edi
6     callq puts
7     movl $.L.str.2, %edi
8     callq puts
9     movl $.L.str, %edi
10    callq puts
11    movl $.L.str.1, %edi
12    callq puts
13    xorl %eax, %eax
14    popq %rcx
15    retq
16 .L.str:
17     .asciz "establish"
18
19 .L.str.1:
20     .asciz "reset_timeout"
21
```

Output (0/0) clang version 7.0.0 (tags/RELEASE\_700/final) - cached (122766B)

<https://godbolt.org/z/APHwnc>

# NAIVE - IF/ELSE - SUMMARY

## NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)

## NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)
- (+) No heap usage

## NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)
- (+) No heap usage
- (~) Small-ish memory footprint
  - `sizeof(Connection) == 3b`

## NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)
- (+) No heap usage
- (~) Small-ish memory footprint
  - `sizeof(Connection) == 3b`
- (-) Hard to reuse

# NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)
- (+) No heap usage
- (~) Small-ish memory footprint
  - `sizeof(Connection) == 3b`
- (-) Hard to reuse

```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/[a-zA-Z0-9]{2,64}$/', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```





STATE MACHINE MIGHT BE EASILY IDENTIFIED BY  
IMPLICIT STATES



# STATE MACHINE MIGHT BE EASILY IDENTIFIED BY IMPLICIT STATES

```
bool disconnected = true, connected = false, connecting = false; // 🤦
```

# NAIVE - SWITCH/ENUM - IMPLEMENTATION

# NAIVE - SWITCH/ENUM - IMPLEMENTATION

```
class Connection {  
    // Only one state can be active  
    enum class State : std::uint8_t { DISCONNECTED,  
                                      CONNECTING,  
                                      CONNECTED } state = DISCONNECTED;
```

# NAIVE - SWITCH/ENUM - IMPLEMENTATION

```
class Connection {  
    // Only one state can be active  
    enum class State : std::uint8_t { DISCONNECTED,  
                                      CONNECTING,  
                                      CONNECTED } state = DISCONNECTED;  
  
    constexpr void process_event(connect const&) {  
        switch (state) { // Handle current state  
            default: break;  
            case State::DISCONNECTED:  
                establish(); state = State::CONNECTING; break;  
                    // Set the new state  
        }  
    }  
}
```

# NAIVE - SWITCH/ENUM - IMPLEMENTATION

```
class Connection {  
    // Only one state can be active  
    enum class State : std::uint8_t { DISCONNECTED,  
                                      CONNECTING,  
                                      CONNECTED } state = DISCONNECTED;
```

```
constexpr void process_event(connect const&) {  
    switch (state) { // Handle current state  
        default: break;  
        case State::DISCONNECTED:  
            establish(); state = State::CONNECTING; break;  
                // Set the new state  
    }  
}
```

```
constexpr void process_event(ping const& event) {  
    switch (state) {  
        default: break;  
        case State::CONNECTED:  
            if (is_valid(event)) { reset_timeout(); }  
            break; // Stay in the current state  
    }  
}
```

# NAIVE - SWITCH/ENUM - FULL EXAMPLE

The screenshot shows the Godbolt compiler explorer interface. On the left, the C++ source code is displayed:

```
46 }
47
48 constexpr void process_event(disconnect const&)
49     switch(state) {
50         default: break;
51         case State::CONNECTING:
52             case State::CONNECTED: close(); state = Sta
53     }
54 }
55 };
56
57 int main() {
58     Connection connection{};
59     connection.process_event(connect{});
60     connection.process_event(established{});
61     connection.process_event(ping{});
62     connection.process_event(disconnect{});
63     connection.process_event(connect{});
64     connection.process_event(established{});
65     connection.process_event(ping{});
66 }
```

On the right, the assembly output is shown in Intel syntax:

```
4    callq   puts
5    movl   $.L.str.1, %edi
6    callq   puts
7    movl   $.L.str.2, %edi
8    callq   puts
9    movl   $.L.str, %edi
10   callq  puts
11   movl   $.L.str.1, %edi
12   callq  puts
13   xorl   %eax, %eax
14   popq   %rcx
15   retq
16 .L.str:
17     .asciz  "establish"
18
19 .L.str.1:
20     .asciz  "reset_timeout"
21
22 .L.str.2:
```

The assembly output corresponds to the main loop of the C++ code, where each event leads to a call to the `puts` function with a different string argument.

[https://godbolt.org/z/NM\\_-oY](https://godbolt.org/z/NM_-oY)

# NAIVE - SWITCH/ENUM - SUMMARY

*Fusilli code (A spin to make bad code look good)*

## NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)

# *Fusilli code (A spin to make bad code look good)*

## NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)
- (+) Small memory footprint
  - `sizeof(ConnectionString) == 1b`

# *Fusilli code (A spin to make bad code look good)*

## NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)
- (+) Small memory footprint
  - `sizeof(Connection) == 1b`
- (+) No heap usage

# *Fusilli code (A spin to make bad code look good)*

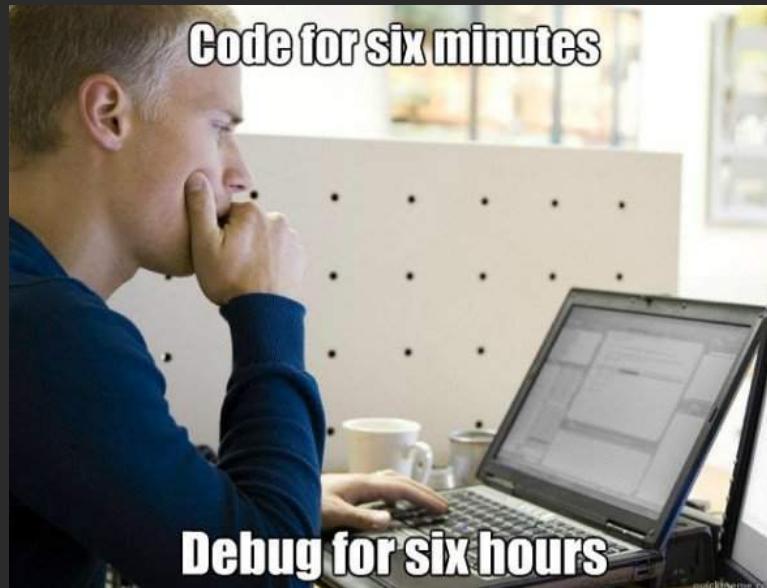
## NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)
- (+) Small memory footprint
  - `sizeof(Connection) == 1b`
- (+) No heap usage
- (-) Hard to reuse

# *Fusilli code (A spin to make bad code look good)*

# NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)
- (+) Small memory footprint
  - `sizeof(Connection) == 1b`
- (+) No heap usage
- (-) Hard to reuse



*Fusilli code (A spin to make bad code look good)*

# **INHERITANCE / STATE PATTERN - IMPLEMENTATION**

# INHERITANCE / STATE PATTERN - IMPLEMENTATION

```
struct State {  
    virtual ~State() noexcept = default;  
    virtual void process_event(connect const&) = 0;  
    virtual void process_event(ping const&) = 0;  
    virtual void process_event(established const&) = 0;  
    virtual void process_event(timeout const&) = 0;  
    virtual void process_event(disconnect const&) = 0;  
};
```

# **INHERITANCE / STATE PATTERN - IMPLEMENTATION**

# INHERITANCE / STATE PATTERN - IMPLEMENTATION

```
struct Disconnected : State {  
    Connection& connection;  
  
    void process_event(connect const&) override final {  
        establish();  
        connection.change_state<Connecting>();  
    }  
};
```

# INHERITANCE / STATE PATTERN - IMPLEMENTATION

```
struct Disconnected : State {  
    Connection& connection;  
  
    void process_event(connect const&) override final {  
        establish();  
        connection.change_state<Connecting>();  
    }  
};
```

```
struct Connected : State {  
    Connection& connection;  
  
    void process_event(ping const& event) override final {  
        if (is_valid(event)) {  
            reset_timeout();  
        }  
    }  
    // ...  
};
```

# INHERITANCE / STATE PATTERN - FULL EXAMPLE

The screenshot shows three windows from the Godbolt Compiler Explorer:

- Editor (Left):** Displays the C++ source code for a state pattern example. It includes a main loop that initializes a connection, processes various events (connect, established, ping, disconnect), and changes its state accordingly.
- Clang++ Assembly Output (Top Right):** Shows the assembly code generated by clang++ for the typeinfo for State. It includes labels for .L.str.1 (reset\_timeout) and .L.str.2 (close).
- G++ Assembly Output (Bottom Right):** Shows the assembly code generated by g++ for the main function. It includes a call to the vtable for Disconnected+16.

```
connection.change_state<Connecting>();  
}  
  
void process_event(disconnect const&) override  
{  
    close();  
    connection.change_state<Disconnected>();  
}  
  
private:  
    Connection& connection;  
};  
  
int main() {  
    Connection connection{};  
    connection.init_state<Disconnected>();  
    connection.process_event(connect{});  
    connection.process_event(established{});  
    connection.process_event(ping{});  
    connection.process_event(disconnect{});  
    connection.process_event(connect{});  
    connection.process_event(established{});  
    connection.process_event(ping{});  
}
```

```
.quad    typeinfo for State  
.L.str.1:  
.asciz  "reset_timeout"  
.L.str.2:  
.asciz  "close"  
ret  
main:  
    subq   $40, %rsp  
    movl   $vtable for Disconnected+16, %edi  
    movl   $16, %edi  
    leaq   24(%rsp), %rax  
    vmovq %rcx, %xmm0  
    movq   $0, 24(%rsp)
```

<https://godbolt.org/z/dui-ar>

# INHERITANCE / STATE PATTERN - SUMMARY

## INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)

## INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)
- (~) High-ish memory footprint

## INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)
- (~) High-ish memory footprint
- (-) Heap usage / dynamic allocations

## INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)
- (~) High-ish memory footprint
- (-) Heap usage / dynamic allocations
- (-) Not inlined/devirtualized (even with final)

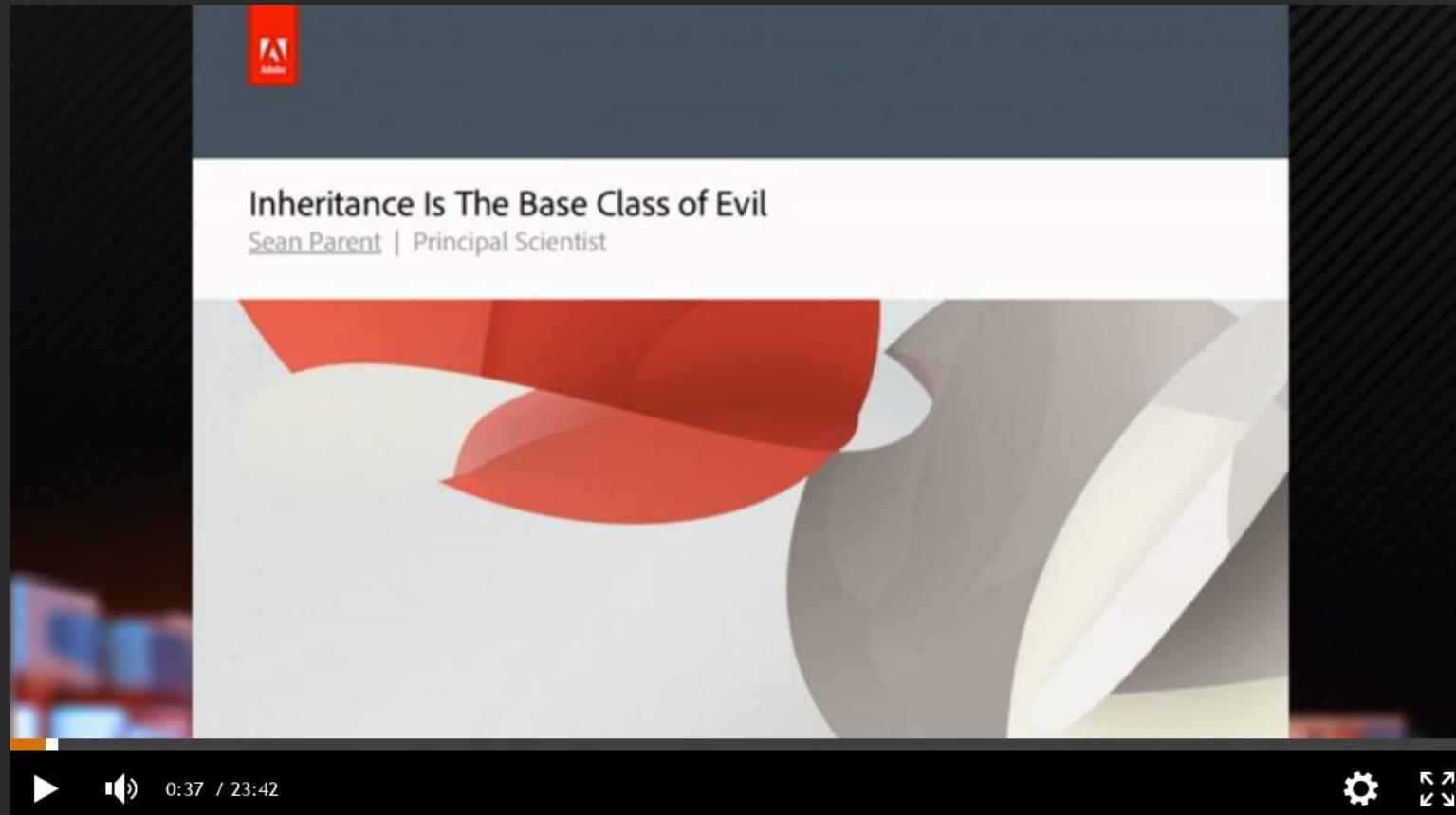
# INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)
- (~) High-ish memory footprint
- (-) Heap usage / dynamic allocations
- (-) Not inlined/devirtualized (even with final)



# INHERITANCE IS THE BASE CLASS OF EVIL, SEAN PARENT

# INHERITANCE IS THE BASE CLASS OF EVIL, SEAN PARENT



# STL

*C++17 / C++20 Standard Template  
Library*

# **STD::VARIANT - IMPLEMENTATION**

# STD::VARIANT - IMPLEMENTATION

```
class Connection {  
    struct Disconnected { }; // May have additional data  
    struct Connecting { };  
    struct Connected { };  
  
    // Only one active state  
    std::variant<Disconnected, Connecting, Connected> state  
    = Disconnected{};
```

# STD::VARIANT - IMPLEMENTATION

```
class Connection {  
    struct Disconnected { }; // May have additional data  
    struct Connecting { };  
    struct Connected { };  
  
    // Only one active state  
    std::variant<Disconnected, Connecting, Connected> state  
    = Disconnected{};
```

```
constexpr void process_event(connect const&) {  
    std::visit(overload{ // Choose one of the following...  
        [&] (Disconnected) { establish(); state = Connecting{}; },  
        [] (auto) { } // No changes...  
    , state);  
}
```

# STD::VARIANT - IMPLEMENTATION

```
class Connection {  
    struct Disconnected { }; // May have additional data  
    struct Connecting { };  
    struct Connected { };  
  
    // Only one active state  
    std::variant<Disconnected, Connecting, Connected> state  
    = Disconnected{};
```

```
constexpr void process_event(connect const&) {  
    std::visit(overload{ // Choose one of the following...  
        [&] (Disconnected) { establish(); state = Connecting{}; },  
        [] (auto) { } // No changes...  
    , state);  
}
```

```
void process_event(ping const& event) {  
    if (std::get_if<Connected>(&state) and is_valid(event)) {  
        reset_timeout();  
    } // Stay in the current state  
}
```

# STD::VARIANT - FULL EXAMPLE

The screenshot shows the Godbolt Compiler Explorer interface with three panes:

- Left Pane (Editor):** Displays the C++ source code for a `main` function. The code initializes a `Connection` object and processes various events (connect, established, ping, disconnect) multiple times.
- Middle Pane (Assembly 1):** Shows the assembly output for the `main` function using `clang++` version 7.0.0. The assembly includes standard library calls like `puts` and `abort`, along with calls to `std::__detail::__variant::__gen_vtable<void, ...>`.
- Right Pane (Assembly 2):** Shows the assembly output for the same code using `g++` version 8.2.0. This assembly is significantly shorter, primarily consisting of `movl`, `call`, and `jmp` instructions, reflecting the different optimization and standard library implementation between the two compilers.

<https://godbolt.org/z/oY2FBI>

# **STD::VARIANT - SUMMARY**

## STD::VARIANT - SUMMARY

- (+) Small/efficient memory footprint

- ```
Disconnect { };
Connecting { connection_id id{}; };
Connected   { connection_id id{};
              time last_ping{}; };
```

## STD::VARIANT - SUMMARY

- (+) Small/efficient memory footprint

- ```
Disconnect { };
Connecting { connection_id id{}; };
Connected { connection_id id{};
            time last_ping{}; };
```

- (+) Integrates well with std::expected/static exceptions

- ```
return Error{ "timeout" };
```

## STD::VARIANT - SUMMARY

- (+) Small/efficient memory footprint

- ```
Disconnect { };
Connecting { connection_id id{}; };
Connected { connection_id id{};
            time last_ping{}; };
```

- (+) Integrates well with std::expected/static exceptions

- ```
return Error{ "timeout" };
```

- (~) Inlined (clang only)

## STD::VARIANT - SUMMARY

- (+) Small/efficient memory footprint
  - Disconnect { } ;  
Connecting { connection\_id id{}; } ;  
Connected { connection\_id id{};  
time last\_ping{}; } ;
- (+) Integrates well with std::expected/static exceptions
  - return Error{ "timeout" } ;
- (~) Inlined (clang only)
- (-) Hard to reuse (Similar to switch/enum)

# COROUTINES (C++20)

---

# COROUTINES (C++20)

*Resumable Functions / Functions on steroids*

# COROUTINES (C++20)

*Resumable Functions / Functions on steroids*

```
void process_event(const TEvent& event) {  
    this->event = event;  
    coroutine.resume();  
}
```

# COROUTINES (C++20)

*Resumable Functions / Functions on steroids*

```
/**  
 * @param event - a common event type  
 * @coroutine    - coroutine_handle (implementation detail)  
 */
```

```
void process_event(const TEvent& event) {  
    this->event = event;  
    coroutine.resume();  
}
```

# COROUTINES / LOOP - IMPLEMENTATION

# COROUTINES / LOOP - IMPLEMENTATION

```
auto Connection = [](auto& in) {
    for (;;) { // Wait for an event... -> Disconnected
        if (auto [event, data] = co_await in; event == connect) {
            establish();
        } // Otherwise go back to co_await and suspend...
    } };
}
```

# COROUTINES / LOOP - IMPLEMENTATION

```
auto Connection = [](auto& in) {
    for (;;) { // Wait for an event... -> Disconnected
        if (auto [event, data] = co_await in; event == connect) {
            establish();
        }
    } end:;
} // Otherwise go back to co_await and suspend...
};};
```

# COROUTINES / LOOP - IMPLEMENTATION

```
auto Connection = [](auto& in) {
    for (;;) { // Wait for an event... -> Disconnected
        if (auto [event, data] = co_await in; event == connect) {
            establish();
        }
        for (;;) { // -> Connecting
            if (auto [event, data] = co_await in; event == established) {
                for (;;) { // -> Connected
                    switch (auto [event, data] = co_await in; event) {
                        case ping:
                            if (is_valid(data)) { reset_timeout(); continue; }
                        case timeout: establish(); break;
                        case disconnect: close(); goto end;
                    }
                }
            }
        }
    } end:;
} // Otherwise go back to co_await and suspend...
};};
```

# COROUTINES / LOOP - FULL EXAMPLE

The screenshot shows a debugger interface with two panes. The left pane displays the C++ source code, and the right pane shows the generated assembly code.

**Source Code (Left Pane):**

```
97     case ping{}: if (is_valid(data))
98     case timeout{}: establish(); break;
99     case disconnect{}: close(); goto end;
100    }
101   }
102 }
103 }
104 end:;
105 }
106 }
107 }
108 }
109
110 int main() {
111     Connection connection{};
112     connection.process_event(connect{});
113     connection.process_event(established{});
114     connection.process_event(ping{});
115     connection.process_event(disconnect{});
116     connection.process_event(connect{});
117     connection.process_event(established{});
118     connection.process_event(ping{});
119 }
```

**Assembly Output (Right Pane):**

```
113     mov    rax, qword ptr [r14 + 24]
114     jmp    .LBB1_21
115 .LBB1_19:
116     mov    rax, qword ptr [r14 + 24]
117     mov    ecx, dword ptr [rax]
118     test   ecx, ecx
119     jne    .LBB1_7
120     mov    byte ptr [r14 + 17], 0
121 .LBB1_21:
122     mov    qword ptr [rax + 8], r14
123     add    rsp, 8
124     pop    rbx
125     pop    r14
126     ret
127 Connection::connection() [clone .cleanup]:  #
128     ret
129 .L.str:
130     .asciz  "establish"
131
132 .L.str.1:
133     .asciz  "reset_timeout"
134
135 .L.str.2:
```

Output (0/0) clang version 7.0.0 (tags/RELEASE\_700/final) - 596ms (448674B)

<https://godbolt.org/z/P3zaNt>

# COROUTINES / LOOP - SUMMARY

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)
- (~) Requires heap (heap elision / devirtualization / no control)

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)
- (~) Requires heap (heap elision / devirtualization / no control)
- (~) Implicit states (position in the function)

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)
- (~) Requires heap (heap elision / devirtualization / no control)
- (~) Implicit states (position in the function)
- (-) Events require a common type

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)
- (~) Requires heap (heap elision / devirtualization / no control)
- (~) Implicit states (position in the function)
- (-) Events require a common type
- (-) Weird usage of infinite loops

# COROUTINES / GOTO - IMPLEMENTATION

# COROUTINES / GOTO - IMPLEMENTATION

```
auto Connection = [](auto& in) {
    for (;;) {          // State is represented by
        disconnected: // a position in the function
        if (auto [event, data] = co_await in; event == connect) {
            establish();
        }
    }
};
```

# COROUTINES / GOTO - IMPLEMENTATION

```
auto Connection = [](auto& in) {
    for (;;) {          // State is represented by
        disconnected: // a position in the function
        if (auto [event, data] = co_await in; event == connect) {
            establish();
        }
        connecting:
        if (auto [event, data] = co_await in; event == established) {
            ...
        }
    }
};
```

# COROUTINES / GOTO - IMPLEMENTATION

```
auto Connection = [](auto& in) {
    for (;;) { // State is represented by
        disconnected: // a position in the function
        if (auto [event, data] = co_await in; event == connect) {
            establish();
        }
        connecting:
        if (auto [event, data] = co_await in; event == established) {
            connected:
            switch (auto [event, data] = co_await in; event) {
                case ping:
                    if (is_valid(data)) { reset_timeout(); goto connected; }
                case timeout: establish();
                    goto connecting; // Set the new state
                case disconnect:
                    close(); goto disconnected;
            }
        }
    }
};
```

# COROUTINES / GOTO - FULL EXAMPLE

The screenshot shows the Godbolt compiler explorer interface. On the left, the C++ source code is displayed with line numbers from 95 to 117. The code implements a coroutine-based event loop. Lines 95-107 show the main loop body, which processes events and handles connection states. Lines 108-117 show the `main()` function which initializes a connection and processes various events.

On the right, the assembly output is shown for the `main()` function. The assembly is generated by Clang 7.0.0. The assembly code includes labels for event types like `.L.str`, `.L.str.1`, and `.L.str.2`, and handles cases for `ping`, `reset_timeout`, and `close`.

```
95     connected:
96     switch (auto [event, data] = co_await s
97         case ping{}: if (is_valid(data)) reset_
98         case timeout{}: establish(); goto connect
99         case disconnect{}: close(); goto disconnect
100    }
101   }
102 }
103 }
104 }
105 ;
106 };
107
108 int main() {
109 Connection connection{};
110 connection.process_event(connect());
111 connection.process_event(established());
112 connection.process_event(ping());
113 connection.process_event(disconnect());
114 connection.process_event(connect());
115 connection.process_event(established());
116 connection.process_event(ping());
117 }
```

```
105     mov    rax, qword ptr [rax + 16]
106     mov    ecx, dword ptr [rax]
107     test   ecx, ecx
108     jne   .LBB1_3
109     mov    byte ptr [rbx + 17], 1
110     jmp   .LBB1_19
111 .LBB1_5:
112     mov    byte ptr [rbx + 17], 0
113 .LBB1_19:
114     mov    qword ptr [rax + 8], rbx
115     add    rsp, 8
116     pop    rbx
117     pop    r14
118     ret
119 Connection::connection() [clone .cleanup]: # @Connection
120     ret
121 .L.str:
122     .asciz "establish"
123
124 .L.str.1:
125     .asciz "reset_timeout"
126
127 .L.str.2:
128     .asciz "close"
```

<https://godbolt.org/z/BjuHL9>

# COROUTINES / GOTO - SUMMARY

## COROUTINES / GOTO - SUMMARY

- (+) No infinite loops

## COROUTINES / GOTO - SUMMARY

- (+) No infinite loops
- (~) Explicit states

## COROUTINES / GOTO - SUMMARY

- (+) No infinite loops
- (~) Explicit states
- (-) GOTO!

# COROUTINES / GOTO - SUMMARY

- (+) No infinite loops
- (~) Explicit states
- (-) GOTO!



# COROUTINES / FUNCTIONS / VARIANT

# COROUTINES / FUNCTIONS / VARIANT

```
/**  
 * State -> function  
 */  
auto disconnected();  
auto connecting();  
auto connected();
```

# COROUTINES / FUNCTIONS / VARIANT

```
/**  
 * State -> function  
 */  
auto disconnected();  
auto connecting();  
auto connected();
```

```
/**  
 * Event -> std::variant  
 */  
std::variant<connect, established, ping, timeout, disconnect> event{};
```

# COROUTINES / FUNCTIONS / VARIANT

```
/**  
 * State -> function  
 */  
auto disconnected();  
auto connecting();  
auto connected();
```

```
/**  
 * Event -> std::variant  
 */  
std::variant<connect, established, ping, timeout, disconnect> event{};
```

*Interestingly, so far, states have been  
first-class citizens, not events*

# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

```
auto disconnected() {  
    for (;;) { // Wait for the connect event...  
    }  
}
```

# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

```
auto disconnected() {
    for (;;) { // Wait for the connect event...
        if (auto const event = co_await in; std::get_if<connect>(&event)) {
            establish(); co_return connecting(); // Set the new state
        }
    }
}
```

# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

```
auto disconnected() {
    for (;;) { // Wait for the connect event...
        if (auto const event = co_await in; std::get_if<connect>(&event)) {
            establish(); co_return connecting(); // Set the new state
        }
    }
}

auto connected() {
    for (;;) { // Wait for the ping event...
        auto const event = co_await in; // `in` - a member variable
    }
}
```

# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

```
auto disconnected() {
    for (;;) { // Wait for the connect event...
        if (auto const event = co_await in; std::get_if<connect>(&event)) {
            establish(); co_return connecting(); // Set the new state
        }
    }
}

auto connected() {
    for (;;) { // Wait for the ping event...
        auto const event = co_await in; // `in` - a member variable
        if (std::get_if<ping>(&event) and is_valid(std::get<ping>(event))) {
            reset_timeout();
        } else if (std::get_if<timeout>(&event)) {
            establish(); co_return connecting();
        } else if (std::get_if<disconnect>(&event)) {
            close(); co_return disconnected();
        }
    }
}
```

# COROUTINES / FUNCTIONS / VARIANT - FULL EXAMPLE

The screenshot shows the Godbolt Compiler Explorer interface. On the left, the C++ source code is displayed with line numbers from 100 to 122. The code implements a coroutine-based connection manager. On the right, the generated assembly code is shown, produced by clang version 7.0.0. The assembly code includes labels for different event types and their corresponding handlers.

```
100     if (std::get_if<ping>(&event) and is_valid())
101         reset_timeout();
102     } else if (std::get_if<timeout>(&event)) {
103         establish();
104         co_return connecting();
105     } else if (std::get_if<disconnect>(&event))
106         close();
107     co_return disconnected();
108 }
109 }
110 }
111 };
112
113 int main() {
114     Connection connection{};
115     connection.process_event(connect{});
116     connection.process_event(established{});
117     connection.process_event(ping{});
118     connection.process_event(disconnect{});
119     connection.process_event(connect{});
120     connection.process_event(established{});
121     connection.process_event(ping{});
122 }
```

|     | 11010 | .a.out | LXO: | .text                      | //                     | ls+      | Intel | Demangle | Libraries | + Add new... |
|-----|-------|--------|------|----------------------------|------------------------|----------|-------|----------|-----------|--------------|
| 377 |       |        |      | addq                       | \$8,                   | %rsp     |       |          |           |              |
| 378 |       |        |      | popq                       |                        | %rbx     |       |          |           |              |
| 379 |       |        |      | popq                       |                        | %r14     |       |          |           |              |
| 380 |       |        |      | retq                       |                        |          |       |          |           |              |
| 381 |       |        |      | Connection::disconnected() | [clone .destroy]: # @  |          |       |          |           |              |
| 382 |       |        |      | testq                      | %rdi, %rdi             |          |       |          |           |              |
| 383 |       |        |      | je                         |                        | .LBB12_1 |       |          |           |              |
| 384 |       |        |      | jmp                        | operator delete(void*) |          |       |          |           |              |
| 385 |       |        |      | .LBB12_1:                  |                        |          |       |          |           |              |
| 386 |       |        |      | retq                       |                        |          |       |          |           |              |
| 387 |       |        |      | Connection::disconnected() | [clone .cleanup]: # @  |          |       |          |           |              |
| 388 |       |        |      | retq                       |                        |          |       |          |           |              |
| 389 |       |        |      | .L.str:                    |                        |          |       |          |           |              |
| 390 |       |        |      | .asciz                     | "establish"            |          |       |          |           |              |
| 391 |       |        |      |                            |                        |          |       |          |           |              |
| 392 |       |        |      | .L.str.1:                  |                        |          |       |          |           |              |
| 393 |       |        |      | .asciz                     | "reset_timeout"        |          |       |          |           |              |
| 394 |       |        |      |                            |                        |          |       |          |           |              |
| 395 |       |        |      | .L.str.2:                  |                        |          |       |          |           |              |
| 396 |       |        |      | .asciz                     | "close"                |          |       |          |           |              |

<https://godbolt.org/z/tCiWKh>

# COROUTINES / FUNCTIONS / VARIANT - SUMMARY

## COROUTINES / FUNCTIONS / VARIANT - SUMMARY

- (+) Easier to add/follow new states/behaviour

## COROUTINES / FUNCTIONS / VARIANT - SUMMARY

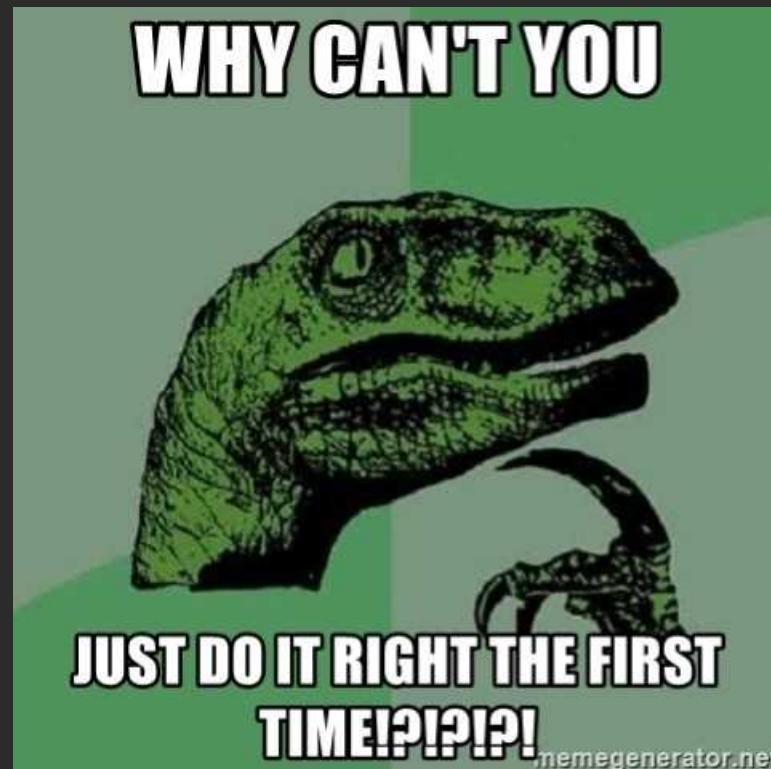
- (+) Easier to add/follow new states/behaviour
- (+) Type safe events

## COROUTINES / FUNCTIONS / VARIANT - SUMMARY

- (+) Easier to add/follow new states/behaviour
- (+) Type safe events
- (-) Dynamic allocations

## COROUTINES / FUNCTIONS / VARIANT - SUMMARY

- (+) Easier to add/follow new states/behaviour
- (+) Type safe events
- (-) Dynamic allocations



# BOOST

# BOOST

| Library  | Boost.Statechart | Boost.MSM   | [Boost].SML |
|----------|------------------|-------------|-------------|
| Standard | C++98/03         | C++98/03    | C++14       |
| Version  | 1.68             | 1.68        | 1.1.0       |
| License  | Boost 1.0        | Boost 1.0   | Boost 1.0   |
| Linkage  | header only      | header only | header only |
| UML      | 1.5              | 2.0         | 2.5         |

# BOOST

| Library  | Boost.Statechart | Boost.MSM   | [Boost].SML |
|----------|------------------|-------------|-------------|
| Standard | C++98/03         | C++98/03    | C++14       |
| Version  | 1.68             | 1.68        | 1.1.0       |
| License  | Boost 1.0        | Boost 1.0   | Boost 1.0   |
| Linkage  | header only      | header only | header only |
| UML      | 1.5              | 2.0         | 2.5         |

Disclaimer: [Boost].SML is not an official Boost library

# BOOST.STATECHART - IMPLEMENTATION

# BOOST.STATECHART - IMPLEMENTATION

## EVENTS

```
struct connect      : sc::event<connect> {};  
struct ping        : sc::event<ping> {};  
struct established : sc::event<established> {};  
struct timeout     : sc::event<timeout> {};  
struct disconnect  : sc::event<disconnect> {};
```

# BOOST.STATECHART - IMPLEMENTATION

## EVENTS

```
struct connect      : sc::event<connect> {};
struct ping        : sc::event<ping> {};
struct established : sc::event<established> {};
struct timeout     : sc::event<timeout> {};
struct disconnect  : sc::event<disconnect> {};
```

## ACTIONS/GUARDS

```
struct Connection : sc::state_machine<Connection, Disconnected> {
    template<class TEvent>
    void establish(TEvent const&) { std::puts("establish"); }
    void reset_timeout(ping const&) { std::puts("reset_timeout"); }
    void close(disconnect const&) { std::puts("close"); }
    bool is_valid(ping const&) { return true; }
};
```

# BOOST.STATECHART - IMPLEMENTATION

# BOOST.STATECHART - IMPLEMENTATION

```
struct Disconnected : sc::simple_state<Disconnected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<connect, Connecting,  
                      Connection, &Connection::establish>>;  
};
```

# BOOST.STATECHART - IMPLEMENTATION

```
struct Disconnected : sc::simple_state<Disconnected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<connect, Connecting,  
                      Connection, &Connection::establish>>;  
};
```

```
struct Connected : sc::simple_state<Connected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<timeout, Connecting,  
                      Connection, &Connection::establish>,  
        sc::transition<disconnect, Disconnected,  
                      Connection, &Connection::close>,  
        sc::custom_reaction<ping>  
    >;
```

# BOOST.STATECHART - IMPLEMENTATION

```
struct Disconnected : sc::simple_state<Disconnected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<connect, Connecting,  
                      Connection, &Connection::establish>;  
};
```

```
struct Connected : sc::simple_state<Connected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<timeout, Connecting,  
                      Connection, &Connection::establish>,  
        sc::transition<disconnect, Disconnected,  
                      Connection, &Connection::close>,  
        sc::custom_reaction<ping>  
    >;
```

```
sc::result react(ping const& event) {  
    if (context<Connection>().is_valid(event)) {  
        context<Connection>().reset_timeout(event);  
    }  
    return discard_event();  
}
```

# BOOST.STATECHART - FULL EXAMPLE

The screenshot shows the Godbolt compiler explorer interface with two compiler windows side-by-side.

**Left Compiler Window (clang++):**

```
40     sc::transition<disconnect, Disconnected, Conn>
41     sc::custom_reaction<ping>
42   >;
43
44   sc::result react(ping const & event) {
45     if (context<Connection>().is_valid(event)) {
46       context<Connection>().reset_timeout(event);
47     }
48     return discard_event();
49   }
50 };
51
52 int main() {
53   Connection connection{};
54   connection.initiate();
55   connection.process_event(connect());
56   connection.process_event(established());
57   connection.process_event(ping());
58   connection.process_event(disconnect());
59   connection.process_event(connect());
60   connection.process_event(established());
61   connection.process_event(ping());
62 }
```

**Right Compiler Window (clang++):**

| Line | Assembly                               | Description                    |
|------|----------------------------------------|--------------------------------|
| 43   | movl \$1, %esi                         | Initial state assignment       |
| 44   | movq %r14, %rdi                        | Parameter passing              |
| 45   | vzeroupper                             | Processor state cleanup        |
| 46   | callq boost::statechart::state_machine | Function call to state machine |
| 47   | movq \$0, 200(%rsp)                    | Return value assignment        |
| 48   | movl \$48, %edi                        | Temporary variable assignment  |
| 49   | callq operator new(unsigned long)      | Memory allocation              |
| 50   | movq %rax, %rbx                        | Temporary variable assignment  |

**Bottom Compiler Window (g++):**

| Line | Assembly                               | Description                 |
|------|----------------------------------------|-----------------------------|
| 2854 | .quad typeinfo for Connected           | Typeinfo for Connected      |
| 2855 | .quad boost::statechart::simple_state< | Simple state registration   |
| 2856 | .quad Connected::~Connected() [complet | Connected state destruction |
| 2857 | .quad Connected::~Connected() [deletin | Connected state destruction |
| 2858 | .quad boost::statechart::simple_state< | Simple state registration   |
| 2859 | .quad boost::statechart::detail::leaf_ | Leaf state registration     |
| 2860 | .quad boost::statechart::simple_state< | Simple state registration   |

<https://godbolt.org/z/NN8UyH>

# **BOOST.STATECHART - SUMMARY**

## **BOOST.STATECHART - SUMMARY**

- (+) UML-1.5 features

## **BOOST.STATECHART - SUMMARY**

- (+) UML-1.5 features
- (~) Learning curve (Similar to State Pattern)

## **BOOST.STATECHART - SUMMARY**

- (+) UML-1.5 features
- (~) Learning curve (Similar to State Pattern)
- (-) Dynamic allocations

## **BOOST.STATECHART - SUMMARY**

- (+) UML-1.5 features
- (~) Learning curve (Similar to State Pattern)
- (-) Dynamic allocations
- (-) Dynamic dispatch

## **BOOST.STATECHART - SUMMARY**

- (+) UML-1.5 features
- (~) Learning curve (Similar to State Pattern)
- (-) Dynamic allocations
- (-) Dynamic dispatch
- (-) High memory footprint

# **BOOST.MSM / EUML - IMPLEMENTATION**

# **BOOST.MSM / EUML - IMPLEMENTATION**

## **EVENTS**

```
BOOST_MSM_EUML_EVENT(connect)
BOOST_MSM_EUML_EVENT(ping)
BOOST_MSM_EUML_EVENT(established)
BOOST_MSM_EUML_EVENT(timeout)
BOOST_MSM_EUML_EVENT(disconnect)
```

# **BOOST.MSM / EUML - IMPLEMENTATION**

## **EVENTS**

```
BOOST_MSM_EUML_EVENT(connect)
BOOST_MSM_EUML_EVENT(ping)
BOOST_MSM_EUML_EVENT(established)
BOOST_MSM_EUML_EVENT(timeout)
BOOST_MSM_EUML_EVENT(disconnect)
```

## **STATES**

```
BOOST_MSM_EUML_STATE((), Disconnected)
BOOST_MSM_EUML_STATE((), Connecting)
BOOST_MSM_EUML_STATE((), Connected)
```

# **BOOST.MSM / EUML - IMPLEMENTATION**

# BOOST.MSM / EUML - IMPLEMENTATION

## ACTIONS

```
BOOST MSM EUML ACTION(establish) {  
    template <class FSM, class EVT, class SourceState, class TargetState>  
    void operator()(EVT const&, FSM &, SourceState &, TargetState &) {  
        std::puts("establish");  
    }  
};
```

# **BOOST.MSM / EUML - IMPLEMENTATION**

## **ACTIONS**

```
BOOST_MSM_EUML_ACTION(establish) {  
    template <class FSM, class EVT, class SourceState, class TargetState>  
    void operator()(EVT const&, FSM &, SourceState &, TargetState &) {  
        std::puts("establish");  
    }  
};
```

## **GUARDS**

```
BOOST_MSM_EUML_ACTION(is_valid) {  
    template <class FSM, class EVT, class SourceState, class TargetState>  
    auto operator()(EVT const&, FSM&, SourceState&, TargetState&) {  
        return true;  
    }  
};
```

# **BOOST.MSM / EUML - IMPLEMENTATION**

## BOOST.MSM / EUML - IMPLEMENTATION

```
BOOST_MSM_EUML_TRANSITION_TABLE( (
    Connecting == Disconnected + connect / establish,
    Connected == Connecting + established,
                Connected + ping [ is_valid ] / reset_timeout,
    Connecting == Connected + timeout / establish,
    Disconnected == Connected + disconnect / close
),
transition_table)
```

## BOOST.MSM / EUML - IMPLEMENTATION

```
BOOST_MSM_EUML_TRANSITION_TABLE( (
    Connecting == Disconnected + connect / establish,
    Connected == Connecting + established,
                Connected + ping [ is_valid ] / reset_timeout,
    Connecting == Connected + timeout / establish,
    Disconnected == Connected + disconnect / close
),
transition_table)
```

```
BOOST_MSM_EUML_DECLARE_STATE_MACHINE
((transition_table, init_ << Disconnected), ConnectionImpl)
```

## BOOST.MSM / EUML - IMPLEMENTATION

```
BOOST_MSM_EUML_TRANSITION_TABLE( (
    Connecting == Disconnected + connect / establish,
    Connected == Connecting + established,
                Connected + ping [ is_valid ] / reset_timeout,
    Connecting == Connected + timeout / establish,
    Disconnected == Connected + disconnect / close
),
transition_table)
```

```
BOOST_MSM_EUML_DECLARE_STATE_MACHINE
((transition_table, init_ << Disconnected), ConnectionImpl)
```

```
using Connection = msm::back::state_machine<ConnectionImpl>;
```

# BOOST.MSM / EUML - FULL EXAMPLE

The screenshot shows two compiler windows side-by-side, illustrating the assembly output for a Boost.MSM/EUML example.

**Left Window (clang++ Output):**

```
BOOST_MSM_EUML_DECLARE_STATE_MACHINE((  
    transition_table,  
    init_ << Disconnected,  
    no_action,  
    no_action,  
    attributes_ << no_attributes_,  
    configure_ << no_exception << no_msg_queue,  
    Log_No_Transition  
, ConnectionImpl)  
  
using Connection = msm::back::state_machine<Conne  
  
int main() {  
    Connection connection{};  
    connection.start();  
    connection.process_event(connect);  
    connection.process_event(established);  
    connection.process_event(ping);  
    connection.process_event(disconnect);  
    connection.process_event(connect);  
    connection.process_event(established);  
    connection.process_event(ping);  
}
```

**Right Window (clang++ Output):**

| Line  | Assembly                                | Description                              |
|-------|-----------------------------------------|------------------------------------------|
| 11010 | movl \$-_ZL11established, %ecx          | Move address of established state to ECX |
| 21    | movq %rbx, %rdi                         | Move RBP to RDI                          |
| 22    | callq *boost::msm::back::dispatch_table | Call dispatch table for ConnectionImpl   |
| 23    | movslq 4(%rsp), %rdx                    | Move value at RSP+4 to RDX               |
| 24    | xorl %esi, %esi                         | XOR ESI with itself (clear ESI)          |
| 25    | movl \$_ZL11established, %ecx           | Move address of established state to ECX |
| 26    | movq %rbx, %rdi                         | Move RBP to RDI                          |
| 27    | callq *boost::msm::back::dispatch_table | Call dispatch table for ConnectionImpl   |
| 28    |                                         |                                          |

**Bottom Window (g++ Output):**

| Line | Assembly                                        | Description                    |
|------|-------------------------------------------------|--------------------------------|
| 191  | .zero 32                                        | Zero-filled 32 bytes           |
| 192  | boost::msm::back::dispatch_table<boost::msm::ba | Start of dispatch table        |
| 193  | .zero 32                                        | Zero-filled 32 bytes           |
| 194  | .LC3:                                           | Label LC3                      |
| 195  | .quad boost::msm::back::HandledEnum bo          | First entry in dispatch table  |
| 196  | .LC4:                                           | Label LC4                      |
| 197  | .quad boost::msm::back::HandledEnum bo          | Second entry in dispatch table |

<https://godbolt.org/z/nVTV0J>

# **BOOST.MSM / EUML - SUMMARY**

## **BOOST.MSM / EUML - SUMMARY**

- (+) Declarative/Expressive (UML transition)

## **BOOST.MSM / EUML - SUMMARY**

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch O(1) - jump table

## **BOOST.MSM / EUML - SUMMARY**

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch O(1) - jump table
- (+) UML-2.0 features

## **BOOST.MSM / EUML - SUMMARY**

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch O(1) - jump table
- (+) UML-2.0 features
- (+) Small memory footprint

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch O(1) - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve

## **BOOST.MSM / EUML - SUMMARY**

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch O(1) - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve
- (~) Domain Specific Language (DSL) based

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch O(1) - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve
- (~) Domain Specific Language (DSL) based
- (-) Macro based

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch O(1) - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve
- (~) Domain Specific Language (DSL) based
- (-) Macro based
- (-) Slow compilation times
  - Timeouts in the Compiler-Explorer

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch O(1) - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve
- (~) Domain Specific Language (DSL) based
- (-) Macro based
- (-) Slow compilation times
  - Timeouts in the Compiler-Explorer
- (-) Error messages

# [BOOST].SML - OVERVIEW

## [BOOST].SML - OVERVIEW

- Single header / `sml.hpp` / 2k LOC

## [BOOST].SML - OVERVIEW

- Single header / `sml.hpp` / 2k LOC
- Neither Boost nor STL is required

## [BOOST].SML - OVERVIEW

- Single header / `sml.hpp` / 2k LOC
- Neither Boost nor STL is required
- No 'virtual's (-fno-rtti)

## [BOOST].SML - OVERVIEW

- Single header / `sml.hpp` / 2k LOC
- Neither Boost nor STL is required
- No 'virtual's (`-fno-rtti`)
- No exceptions required (`-fno-exceptions`)

## [BOOST].SML - OVERVIEW

- Single header / `sml.hpp` / 2k LOC
- Neither Boost nor STL is required
- No 'virtual's (`-fno-rtti`)
- No exceptions required (`-fno-exceptions`)
- Supported compilers (C++14)

## [BOOST].SML - OVERVIEW

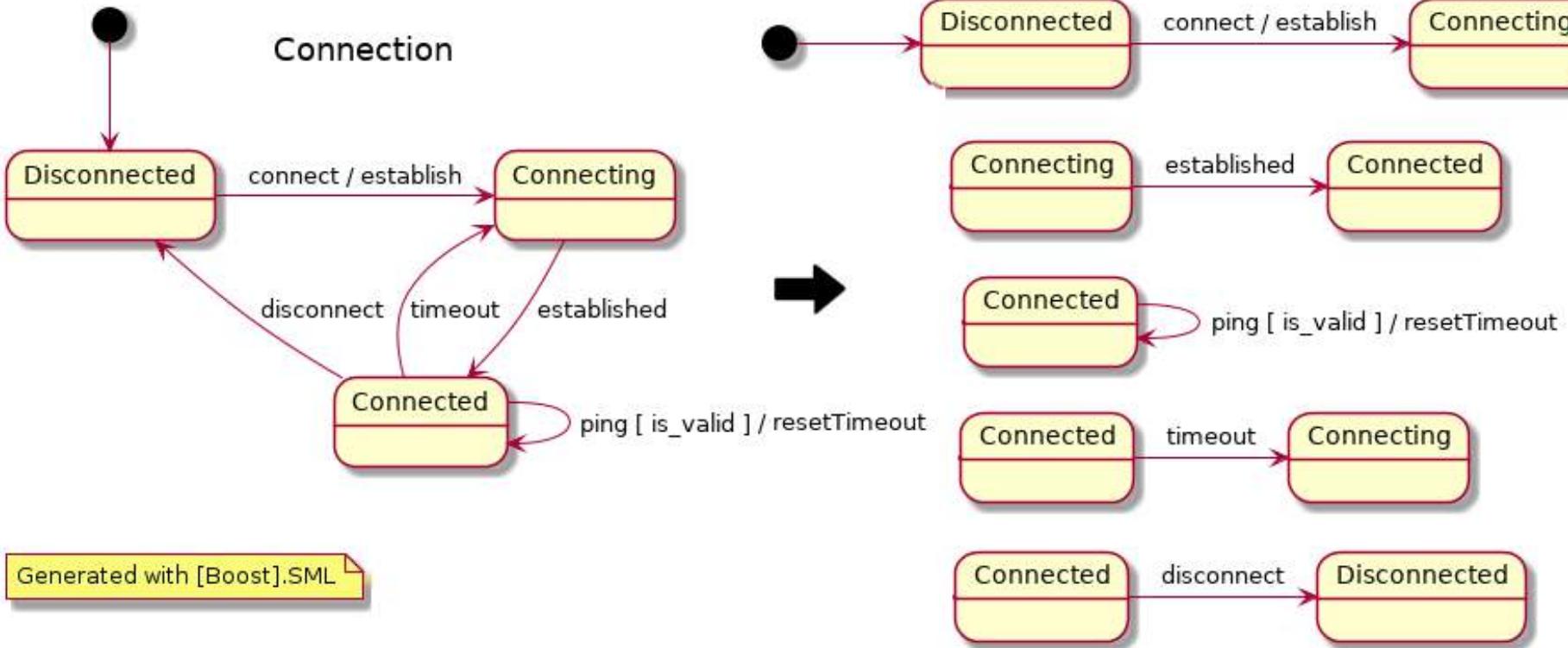
- Single header / `sml.hpp` / 2k LOC
- Neither Boost nor STL is required
- No 'virtual's (`-fno-rtti`)
- No exceptions required (`-fno-exceptions`)
- Supported compilers (C++14)
- `Clang-3.4+`, `XCode-6.1+`, `GCC-5.2+`, `MSVC-2015+`

# [BOOST].SML - IMPLEMENTATION

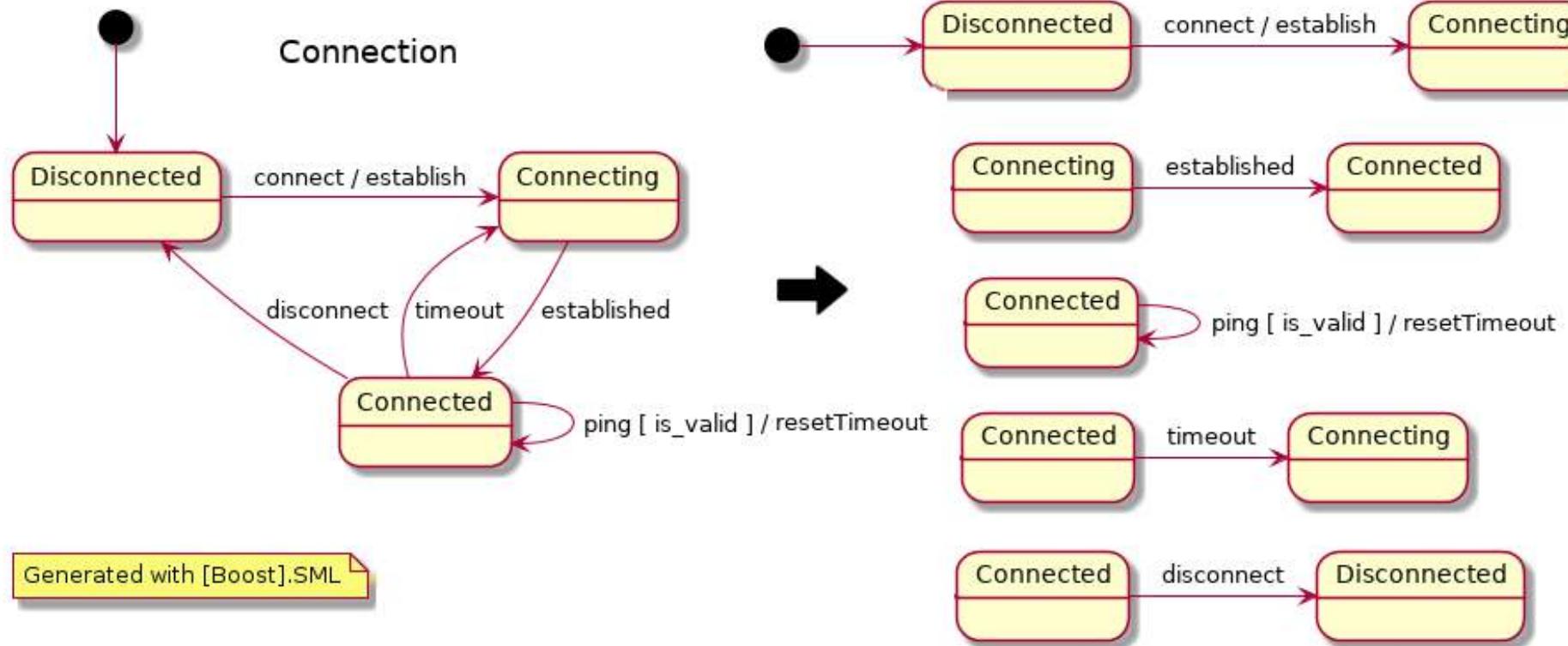
## [BOOST].SML - IMPLEMENTATION

```
sml::sm connection = [] {
    using namespace sml;
    return transition_table{
        * "Disconnected"_s + event<connect> / establish = "Connecting"_s,
        "Connecting"_s + event<established>             = "Connected"_s,
        "Connected"_s + event<ping> [ is_valid ] / reset_timeout,
        "Connected"_s + event<timeout> / establish = "Connecting"_s,
        "Connected"_s + event<disconnect> / close     = "Disconnected"_s
    };
};
```

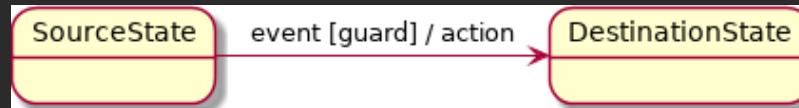
# [BOOST].SML - DOMAIN SPECIFIC LANGUAGE (DSL)



# [BOOST].SML - DOMAIN SPECIFIC LANGUAGE (DSL)



## Transition - UML



# [BOOST].SML - DSL

---

# [BOOST].SML - DSL

\* -> Disconnected : connect / establish

-> Connecting

# [BOOST].SML - DSL

\* -> Disconnected : connect / establish                          -> Connecting

Connecting      : established                          -> Connected

# [BOOST].SML - DSL

\* -> Disconnected : connect / establish                                          -> Connecting

Connecting      : established                                                  -> Connected

Connected      : ping [ is\_valid ] / reset\_timeout

# [BOOST].SML - DSL

\* -> Disconnected : connect / establish                          -> Connecting

Connecting      : established                          -> Connected

Connected      : ping [ is\_valid ] / reset\_timeout

Connected      : timeout                          -> Connecting

# [BOOST].SML - DSL

|                   |                                     |                 |
|-------------------|-------------------------------------|-----------------|
| * -> Disconnected | : connect / establish               | -> Connecting   |
| Connecting        | : established                       | -> Connected    |
| Connected         | : ping [ is_valid ] / reset_timeout |                 |
| Connected         | : timeout                           | -> Connecting   |
| Connected         | : disconnect                        | -> Disconnected |

# [BOOST].SML - DSL

```
* -> Disconnected : connect / establish          -> Connecting
```

```
Connecting     : established                      -> Connected
```

```
Connected      : ping [ is_valid ] / reset_timeout
```

```
Connected      : timeout                         -> Connecting
```

```
Connected      : disconnect                     -> Disconnected
```

---

```
sml::sm connection = [] {
    using namespace sml;
    return transition_table{
        * "Disconnected"_s + event<connect> / establish = "Connecting"_s,
        "Connecting"_s + event<established>           = "Connected"_s,
        "Connected"_s + event<ping> [ is_valid ] / reset_timeout,
        "Connected"_s + event<timeout> / establish = "Connecting"_s,
        "Connected"_s + event<disconnect> / close   = "Disconnected"_s
    };
};
```



# THE POWER OF DECLARATIVE DESIGN

---



# THE POWER OF DECLARATIVE DESIGN

---

Express What, Not how!

# [BOOST].SML - DIAGRAMS GENERATION / CONNECTION

# [BOOST].SML - DIAGRAMS GENERATION / CONNECTION

```
@startuml connection.png
```

```
title Connection
```

```
[*]           --> Disconnected
Disconnected --> Connecting    : connect / establish
Connecting   --> Connected     : established
Connected    --> Connected     : ping [ is_valid ] / resetTimeout
Connected    --> Connecting    : timeout
Connected    --> Disconnected   : disconnect
```

```
@enduml
```

# [BOOST].SML - DIAGRAMS GENERATION / CONNECTION

```
@startuml connection.png
title Connection

[*]           --> Disconnected
Disconnected   --> Connecting      : connect / establish
Connecting    --> Connected       : established
Connected     --> Connected       : ping [ is_valid ] / resetTimeout
Connected     --> Connecting      : timeout
Connected     --> Disconnected    : disconnect

@enduml
```

<http://plantuml.com/state-diagram>

# [BOOST].SML - DIAGRAMS GENERATION / CONNECTION

# [BOOST].SML - DIAGRAMS GENERATION / CONNECTION

```
template <class TSM>
void plant_uml() {
```

```
}
```

# [BOOST].SML - DIAGRAMS GENERATION / CONNECTION

```
template <class TSM>
void plant_uml() {

    if (T::initial) {
        std::cout << "[*] -> " << src_state << '\n';
    }

}
```

# [BOOST].SML - DIAGRAMS GENERATION / CONNECTION

```
template <class TSM>
void plant_uml() {

    if (T::initial) {
        std::cout << "[*] -> " << src_state << '\n';
    }

    std::cout << name<typename TSM::src_state> << " -> "
        << name<typename TSM::dst_state> << " : "
        << name<typename TSM::event> << "["
        << name<typename TSM::guard> << "] / "
        << name<typename TSM::action>;
}

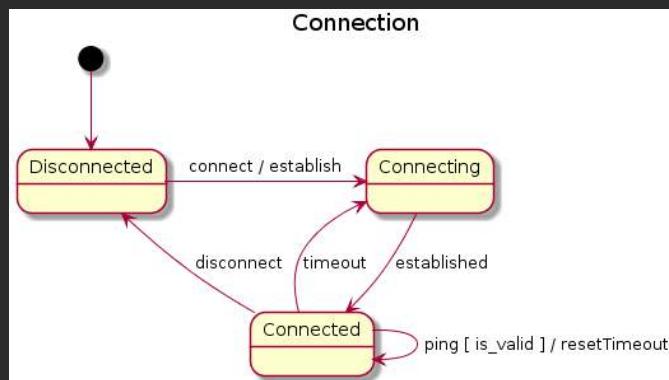
}
```

# [BOOST].SML - DIAGRAMS GENERATION / CONNECTION

```
template <class TSM>
void plant_uml() {
    if (T::initial) {
        std::cout << "[*] -> " << src_state << '\n';
    }
}
```

```
std::cout << name<typename TSM::src_state> << " -> "
    << name<typename TSM::dst_state> << " : "
    << name<typename TSM::event> << "["
    << name<typename TSM::guard> << "] / "
    << name<typename TSM::action>;
```

```
}
```



# [BOOST].SML - PERFORMANCE TUNING

# [BOOST].SML - PERFORMANCE TUNING



# [BOOST].SML - PERFORMANCE TUNING



*Don't pay for what you don't use!*

## DISPATCH POLICY

| Name             | Policy     | Default              |
|------------------|------------|----------------------|
| Jump Table       | jump_table | gcc < 8.0            |
| Nested Switch    | switch     | gcc >= 8.0           |
| If/Else          | branch     | clang                |
| Fold expressions | fold_expr  | gcc/clang with C++17 |

## [BOOST].SML / DISPATCH POLICY - IF/ELSE

## [BOOST].SML / DISPATCH POLICY - IF/ELSE

```
template <class TMappings, // back-end -> generated transitions
          auto N,
          class TState,
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {
}
```

## [BOOST].SML / DISPATCH POLICY - IF/ELSE

```
template <class TMappings, // back-end -> generated transitions
          auto N,
          class TState,
          class... TStates,
          class TEvet>
constexpr auto dispatch(state_t &current_state, TEvet const &event) {

    if constexpr(sizeof...(TStates...) > 0) {
        return current_state == N
            ? TMappings<TState>::execute(event)
            : dispatch<TMappings, N + 1, TStates...>(current_state, event);
    }
}
```

# [BOOST].SML / DISPATCH POLICY - IF/ELSE

The screenshot shows the Godbolt Compiler Explorer interface comparing the assembly output of two compilers for the same C++ code.

**Code:**

```
20     return make_transition_table(
21         * "Disconnected"_s + event<connect> / established,
22         "Connecting"_s + event<established>
23         "Connected"_s + event<ping> [ is_valid ],
24         "Connected"_s + event<timeout> / established,
25         "Connected"_s + event<disconnect> / closed
26     );
27 }
28 ;
29
30 int main() {
31     sml::sm<Connection,
32         sml::dispatch<sml::back::policies::branch_stmt>
33     connection.process_event(connect{});
34     connection.process_event(established{});
35     connection.process_event(ping{});
36     connection.process_event(disconnect{});
37     connection.process_event(connect{});
38     connection.process_event(established{});
39     connection.process_event(ping{});
40 }
```

**Clang++ Assembly Output:**

```
1 main:
2     pushq %rax
3     movl $.L.str, %edi
4     callq puts
5     movl $.L.str.1, %edi
6     callq puts
7     movl $.L.str.2, %edi
```

**G++ Assembly Output:**

```
15    mov    edi, OFFSET FLAT:.LC0
16    call   puts
17    mov    edi, OFFSET FLAT:.LC1
18    call   puts
19    xor    eax, eax
20    add    rsp, 8
21    ret
```

<https://godbolt.org/z/Dsn1PF>

# [BOOST].SML / DISPATCH POLICY - SWITCH

## [BOOST].SML / DISPATCH POLICY - SWITCH

```
template <class TMappings, // back-end -> generated transitions
          auto N = 0,
          class TState,
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {
}
```

## [BOOST].SML / DISPATCH POLICY - SWITCH

```
template <class TMappings, // back-end -> generated transitions
          auto N = 0,
          class TState,
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

    if constexpr(sizeof...(TStates...) > 0) {
        switch (current_state) {
            default: return dispatch<TMappings, N + 1, TStates...>(
                        current_state, event);
            case N:   return TMappings<TState>::execute(event);
        }
    }
}
```

# [BOOST].SML / DISPATCH POLICY - SWITCH

The screenshot shows the Godbolt Compiler Explorer interface with three panes:

- Left Pane (C++ Code):** Displays the source code for a `Connection` struct and its `main` function. The `main` function uses `sml::sm` and `sml::dispatch` to process various events like `connect`, `established`, `ping`, and `disconnect`.
- Middle Pane (clang++ Assembly):** Shows the assembly output for the `main` function generated by clang++. The assembly includes instructions for pushing arguments, calling `puts`, and moving strings from memory.
- Right Pane (g++ Assembly):** Shows the assembly output for the `main` function generated by g++. This assembly is more complex, involving `call` instructions to external labels `.LC0` and `.LC1`, and `xor`, `add`, and `ret` instructions.

<https://godbolt.org/z/AKmGiY>

# [BOOST].SML / DISPATCH POLICY - JUMP TABLE

## [BOOST].SML / DISPATCH POLICY - JUMP TABLE

```
template <class TMappings, // back-end -> generated transitions
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {
}
```

## [BOOST].SML / DISPATCH POLICY - JUMP TABLE

```
template <class TMappings, // back-end -> generated transitions
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

    using dispatch_table_t = bool (*)(TEvent const&);

    constexpr static dispatch_table_t dispatch_table[] = {
        &TMappings<TStates>::template execute<TEvent>...
    };

    return dispatch_table[current_state](event);
}
```

# [BOOST].SML / DISPATCH POLICY - JUMP TABLE

The screenshot shows the Godbolt Compiler Explorer interface with two compiler panes and a code editor.

**Code Editor:**

```
17 struct Connection {
18     auto operator()() const {
19         using namespace sml;
20         return make_transition_table(
21             * "Disconnected"_s + event<connect> / establish
22             "Connecting"_s + event<established>
23             "Connected"_s + event<ping> [ is_valid
24             "Connected"_s + event<timeout> / establish
25             "Connected"_s + event<disconnect> / close
26         );
27     }
28
29
30     int main() {
31         sml::sm<Connection,
32             sml::dispatch<sml::back::policies::jump_table>
33         connection.process_event(connect{});
34         connection.process_event(established{});
35         connection.process_event(ping{});
36         connection.process_event(disconnect{});
37         connection.process_event(connect{});
38         connection.process_event(established{});
39         connection.process_event(ping{});
40     }
41 }
```

**Compiler 1 (Clang 7.0.0):** Shows assembly for the jump table logic. The assembly code is highlighted in red, indicating it is generated by the compiler for the dispatch logic.

```
39    .quad   bool boost::sml::v1_0::back::transitions<boost::sml::sm<Connection, sml::dispatch<sml::back::policies::jump_table>>::process_event<connect>()
40    .quad   bool boost::sml::v1_0::back::transitions<boost::sml::sm<Connection, sml::dispatch<sml::back::policies::jump_table>>::process_event<established>()
41    .quad   bool boost::sml::v1_0::back::transitions<boost::sml::sm<Connection, sml::dispatch<sml::back::policies::jump_table>>::process_event<ping>()
42    .quad   bool boost::sml::v1_0::back::transitions<boost::sml::sm<Connection, sml::dispatch<sml::back::policies::jump_table>>::process_event<timeout>()
43    .quad   bool boost::sml::v1_0::back::transitions<boost::sml::sm<Connection, sml::dispatch<sml::back::policies::jump_table>>::process_event<disconnect>()
```

**Compiler 2 (GCC 8.2.0):** Shows assembly for the dispatch logic. The assembly code is highlighted in red, indicating it is generated by the compiler for the dispatch logic.

```
112    .quad   bool boost::sml::v1_0::back::policies::jump_table::process_event<connect>()
113    .quad   bool boost::sml::v1_0::back::policies::jump_table::process_event<established>()
114    .quad   bool boost::sml::v1_0::back::policies::jump_table::process_event<ping>()
115    .quad   _ZN5boost3sml6v1_0_4back11trans<boost::sml::sm<Connection, sml::dispatch<sml::back::policies::jump_table>>::process_event<timeout>()
116    .quad   bool boost::sml::v1_0::back::policies::jump_table::process_event<disconnect>()
117    .quad   bool boost::sml::v1_0::back::policies::jump_table::process_event<connect>()
118    .quad   bool boost::sml::v1_0::back::policies::jump_table::process_event<established>()
```

<https://godbolt.org/z/lvjjiX9>

# [BOOST].SML / DISPATCH POLICY - FOLD EXPRESSIONS (C++17)

## [BOOST].SML / DISPATCH POLICY - FOLD EXPRESSIONS (C++17)

```
template <class TMappings, // back-end -> generated transitions
          auto... Ns,
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {
}
```

## [BOOST].SML / DISPATCH POLICY - FOLD EXPRESSIONS (C++17)

```
template <class TMappings, // back-end -> generated transitions
          auto... Ns,
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {
    static_assert(sizeof...(TStates) == sizeof...(Ns));
    return ((current_state == Ns
             ? TMappings<TStates>::execute(event)
             : false
            ) or ...);
}
```

# [BOOST].SML / DISPATCH POLICY - FOLD EXPRESSIONS (C++17)

The screenshot shows the Godbolt Compiler Explorer interface with two compiler panes. The left pane displays C++ code, and the right pane shows the generated assembly code for both clang++ and g++.

**C++ Code:**

```
17 struct Connection {
18     auto operator()() const {
19         using namespace sml;
20         return make_transition_table(
21             * "Disconnected"_s + event<connect> / estab]
22             "Connecting"_s + event<established>
23             "Connected"_s + event<ping> [ is_valid
24             "Connected"_s + event<timeout> / estab]
25             "Connected"_s + event<disconnect> / cld
26         );
27     }
28 ;
29
30 int main() {
31     sml::sm<Connection,
32         sml::dispatch<sml::back::policies::fold_expr>>
33     connection.process_event(connect{});
34     connection.process_event(established{});
35     connection.process_event(ping{});
36     connection.process_event(disconnect{});
37     connection.process_event(connect{});
38     connection.process_event(established{});
39     connection.process_event(ping{});
}
```

**clang++ Assembly Output:**

```
1 main:
2     pushq %rax
3     movl $.L.str, %edi
4     callq puts
5     movl $.L.str.1, %edi
6     callq puts
7     movl $.L.str.2, %edi
8     callq puts
```

**g++ Assembly Output:**

```
14     call puts
15     mov edi, OFFSET FLAT:.LC0
16     call puts
17     mov edi, OFFSET FLAT:.LC1
18     call puts
19     xor eax, eax
20     add rsp, 8
21     ret
```

[https://godbolt.org/z/V\\_b7nM](https://godbolt.org/z/V_b7nM)

 **POLICIES ALLOW TO TWEAK THE PERFORMANCE**

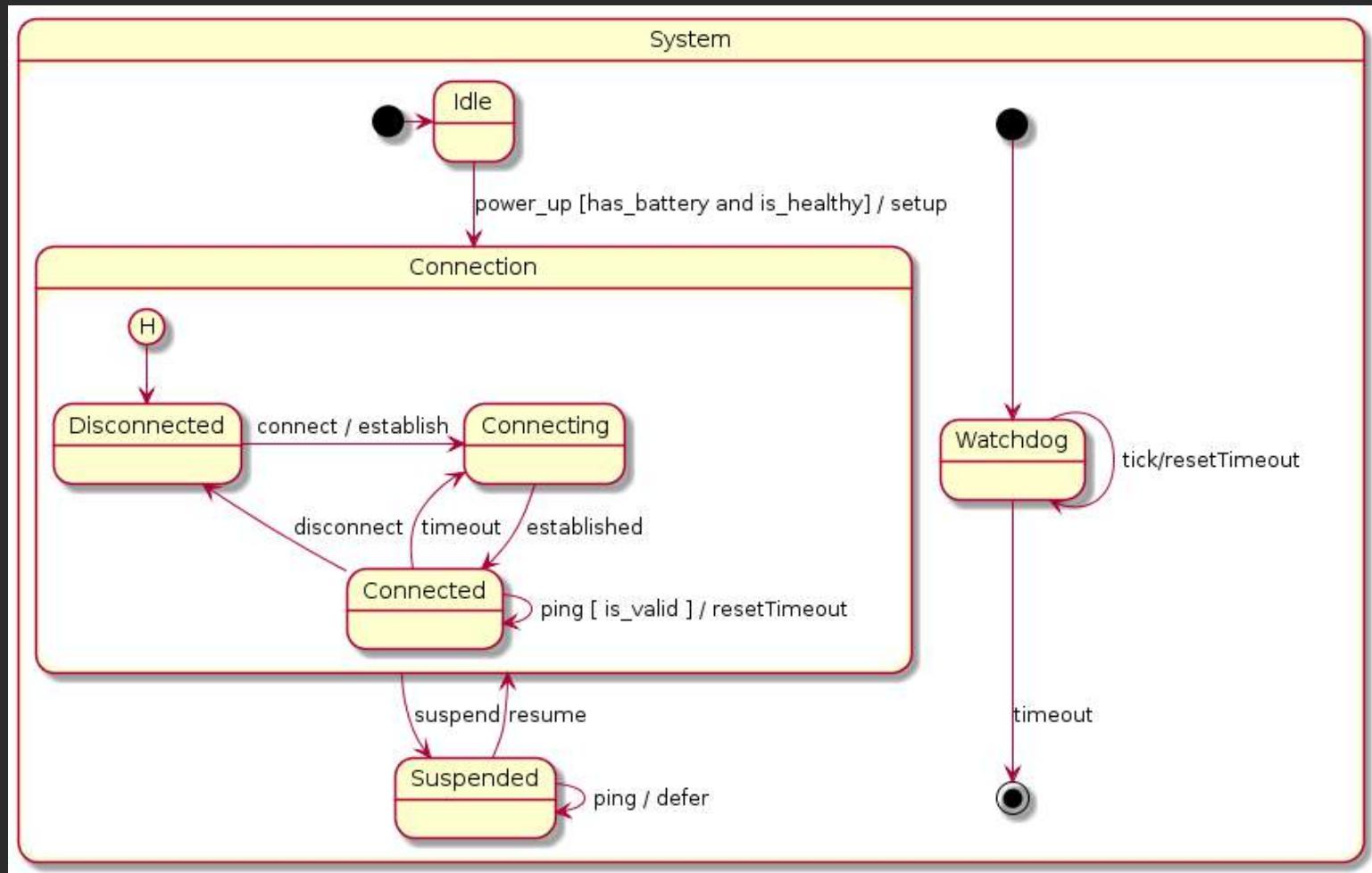


# POLICIES ALLOW TO TWEAK THE PERFORMANCE

*Always measure!*

# STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS

# STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS



# [BOOST].SML - STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS

# [BOOST]SML - STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS

```
class System {
```

# [BOOST].SML - STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS

```
class System {

    struct Connection {
        auto operator()() const {
            using namespace sml;
            return make_transition_table(
                "Disconnected"_s(H) + event<connect> / establish = "Connecting"_s,
                "Connecting"_s + event<established> = "Connected"_s,
                "Connected"_s + event<ping> [ is_valid ] / reset_timeout,
                "Connected"_s + event<timeout> / establish = "Connecting"_s,
                "Connected"_s + event<disconnect> / close = "Disconnected"_s
            );
        }
    };
}
```

# [BOOST].SML - STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS

# [BOOST].SML - STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS

```
public:  
auto operator()() const {  
    using namespace sml;  
    return make_transition_table(  
        * "Idle"_s           + event<power_up>  
            [ has_battery and is_healthy ] / setup  
                = state<Connection>,  
        state<Connection> + event<suspend>           = "Suspended"_s,  
        "Suspended"_s   + event<resume>             = state<Connection>,  
        "Suspended"_s   + event<ping> / defer,  
        // ----- //  
        * "Watchdog"_s    + event<tick> / reset_timeout,  
        "Watchdog"_s    + event<timeout>             = X  
    );  
}  
};
```

# [BOOST].SML - STATE MACHINES ARE MORE THAN JUST SIMPLE TRANSITIONS

```
public:  
auto operator()() const {  
    using namespace sml;  
    return make_transition_table(  
        * "Idle"_s           + event<power_up>  
            [ has_battery and is_healthy ] / setup  
                = state<Connection>,  
        state<Connection> + event<suspend>           = "Suspended"_s,  
        "Suspended"_s    + event<resume>           = state<Connection>,  
        "Suspended"_s    + event<ping> / defer,  
        // ----- //  
        * "Watchdog"_s     + event<tick> / reset_timeout,  
        "Watchdog"_s     + event<timeout>           = X  
    );  
}  
};
```

<https://wandbox.org/permlink/82JwdSBHmLv7koU9>

# [BOOST].SML - SUMMARY

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch O(1)

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch O(1)
- (+) Fast compilation times

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch O(1)
- (+) Fast compilation times
- (+) UML-2.5 features

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch O(1)
- (+) Fast compilation times
- (+) UML-2.5 features
- (+) Minimal memory footprint
  - `sizeof(Connection) == 1b`

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch O(1)
- (+) Fast compilation times
- (+) UML-2.5 features
- (+) Minimal memory footprint
  - `sizeof(Connection) == 1b`
- (~) Learning curve

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch O(1)
- (+) Fast compilation times
- (+) UML-2.5 features
- (+) Minimal memory footprint
  - `sizeof(Connection) == 1b`
- (~) Learning curve
- (~) DSL based

# BENCHMARKS

# ENVIRONMENT / SETUP

# ENVIRONMENT / SETUP

```
const auto action/guard = [] {  
    asm volatile("") : : : "memory");  
};
```

# ENVIRONMENT / SETUP

```
const auto action/guard = [] {
    asm volatile("") : : : "memory");
};
```

```
int main() {
    constexpr auto size = 1'000'000;
    std::array events = rand_events<size>();

    Connection connection{};
    for (auto i = 0; i < size; ++i) {
        process_event(events[i]);
    }
}
```

# ENVIRONMENT / SETUP

```
const auto action/guard = [] {
    asm volatile("") : : : "memory");
};
```

```
int main() {
    constexpr auto size = 1'000'000;
    std::array events = rand_events<size>();

    Connection connection{};
    for (auto i = 0; i < size; ++i) {
        process_event(events[i]);
    }
}
```

```
$CXX -std=c++2a      # clang-8.0.0/gcc-8.3
     -stdlib=libc++   # clang-8.0.0
     -fcoroutines-ts # clang-8.0.0
     -O3 -march=native -flto -fno-exceptions -DNDEBUG
     -I boost_1_70_0
```

# PERF TOOLS

# PERF TOOLS

- Linux Perf

# PERF TOOLS

- Linux Perf
- Callgrind

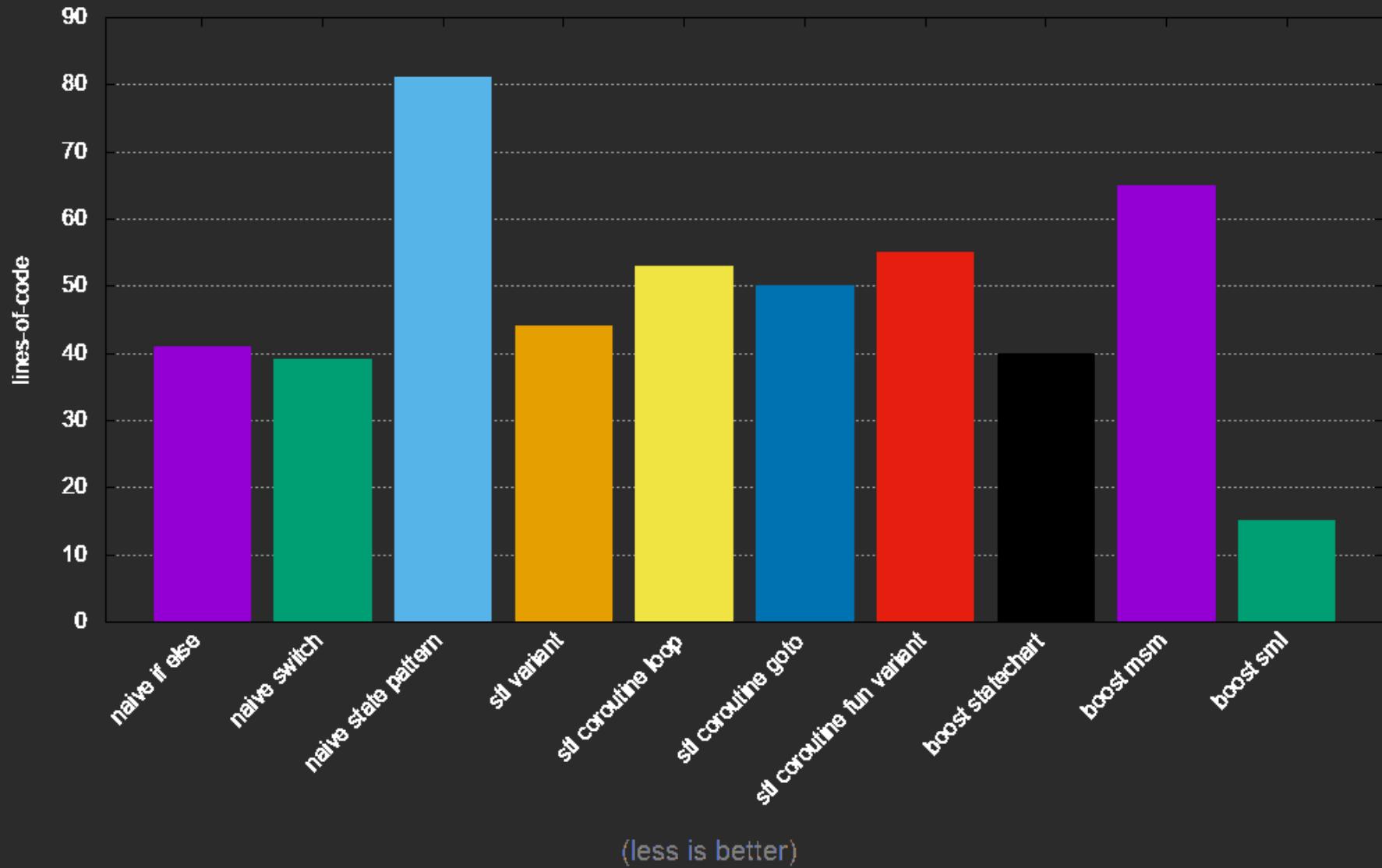
# PERF TOOLS

- Linux Perf
- Callgrind
- Google Benchmark

# PERF TOOLS

- Linux Perf
- Callgrind
- Google Benchmark
- Godbolt

# BENCHMARKS - LINES OF CODE (LOC)





# LINES OF CODE (LOC)

## LINES OF CODE (LOC)

- Express What, Not how!

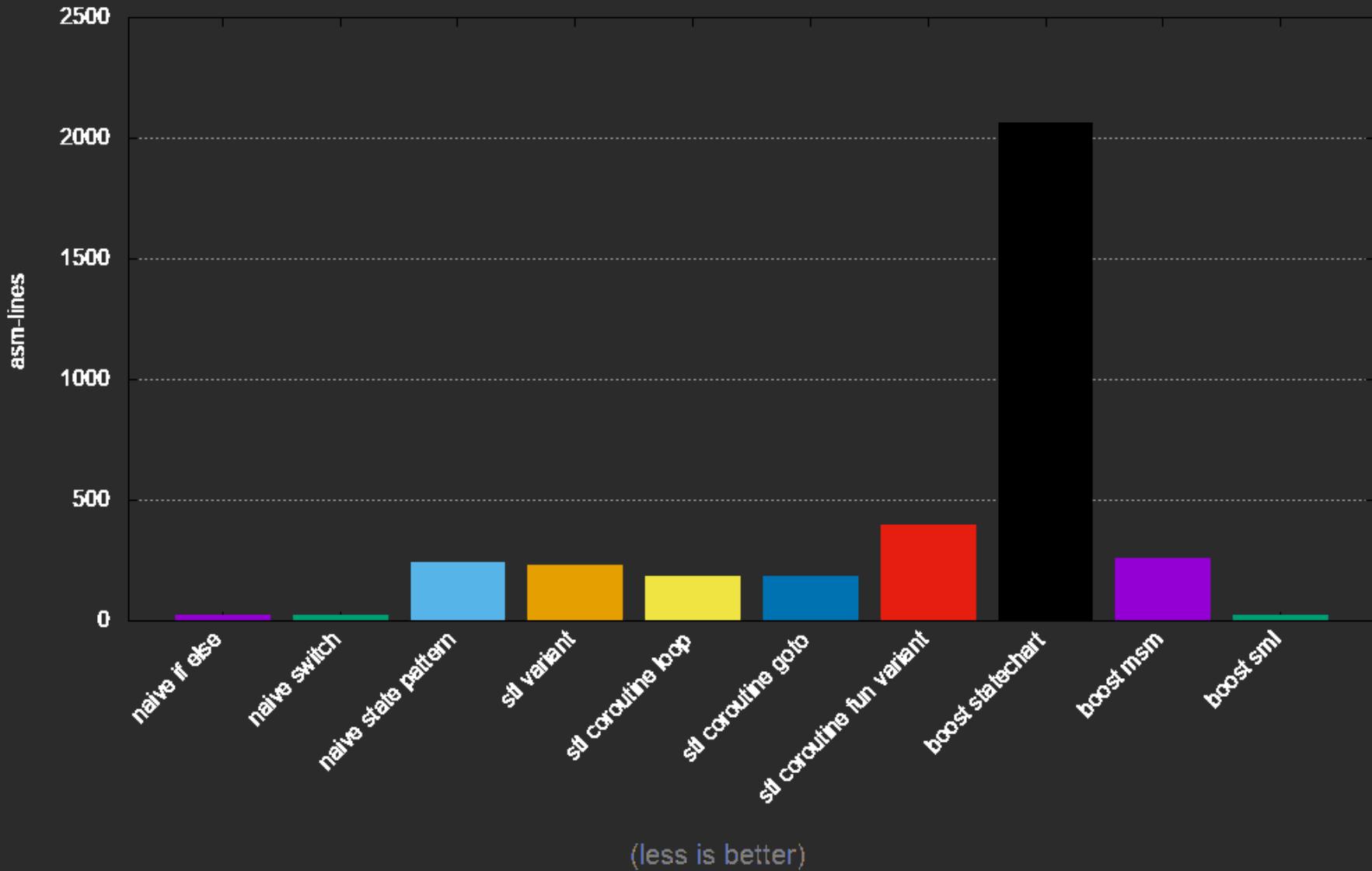
## LINES OF CODE (LOC)

- Express What, Not how!
- Declarative design

## LINES OF CODE (LOC)

- Express What, Not how!
- Declarative design
- Zero overhead abstractions

# BENCHMARKS - ASM LINES





# ASM LINES

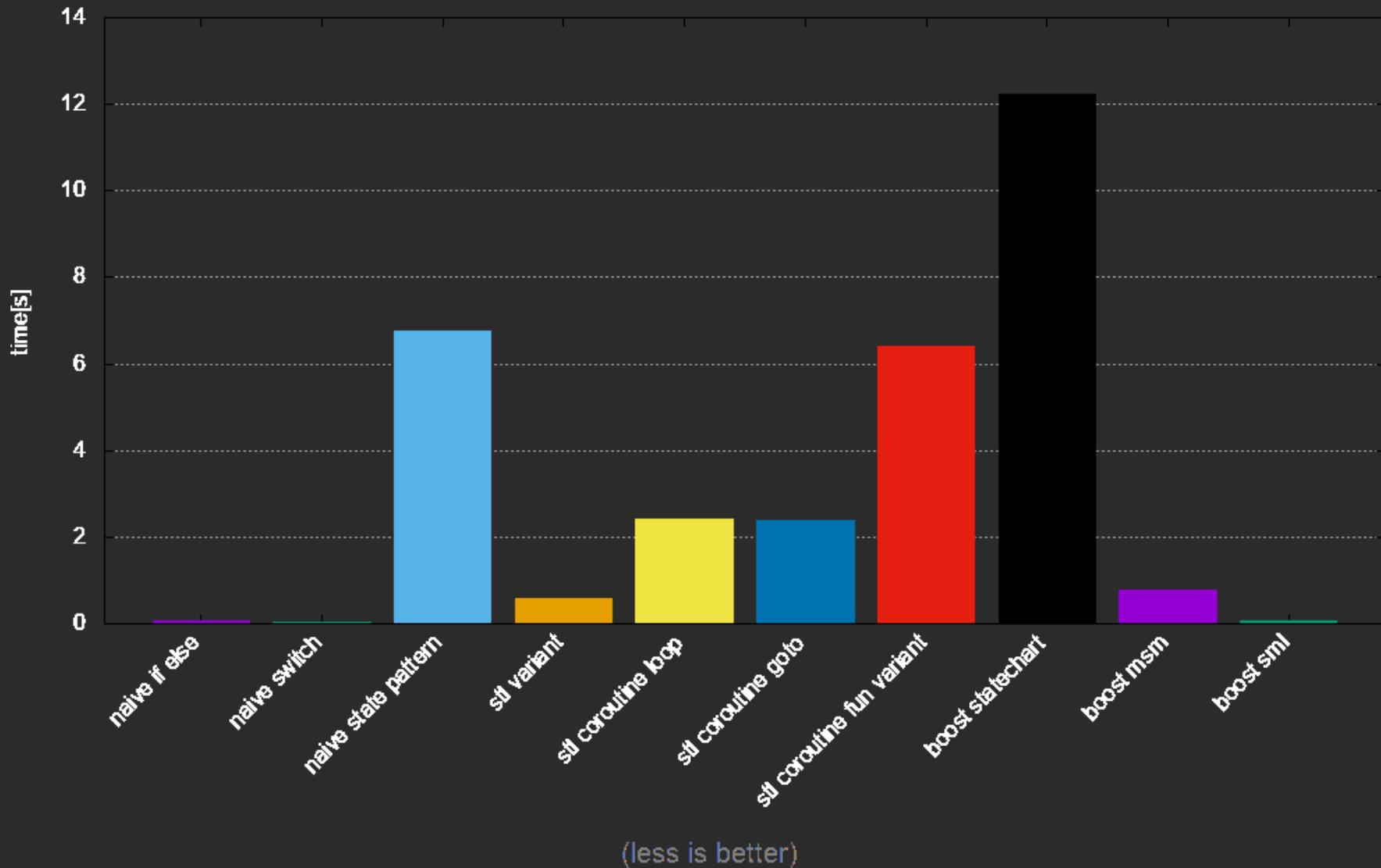
## ASM LINES

- Less Assembly is a good sign 

## ASM LINES

- Less Assembly is a good sign 
- Not all ASM instructions have the same execution time (ex. vectorization)

# BENCHMARKS - RUN-TIME PERFORMANCE - TIME / RELEASE





# RUN-TIME PERFORMANCE

## RUN-TIME PERFORMANCE

- Inlining is usually a good sign 

## RUN-TIME PERFORMANCE

- Inlining is usually a good sign 
- Code bloat might be bad (Inlining the cold path) 

## RUN-TIME PERFORMANCE

- Inlining is usually a good sign 
- Code bloat might be bad (inlining the cold path) 
- Likely/Unlikely (C++20), Profile-guided optimization can improve performance (**Bolt**)

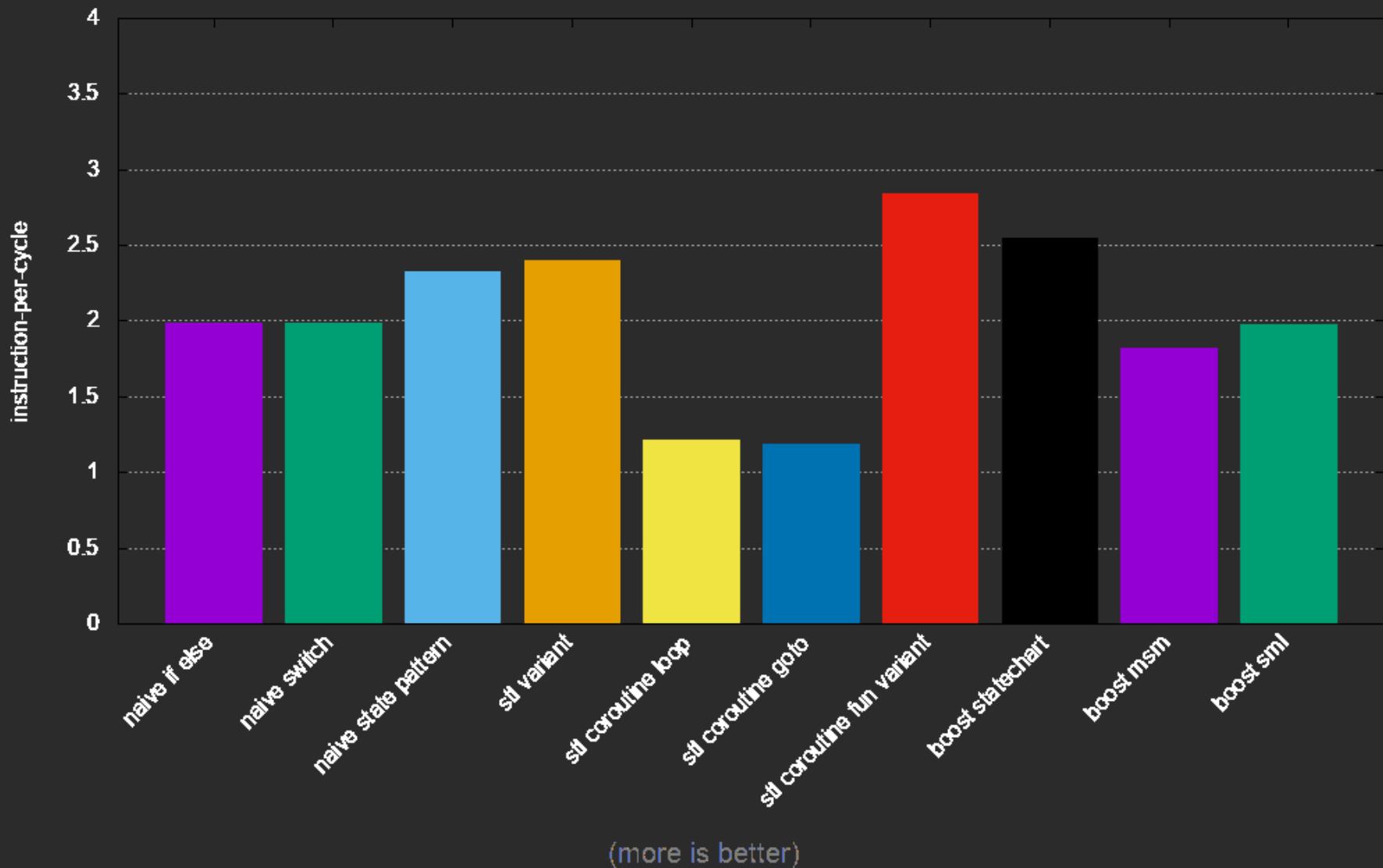
## RUN-TIME PERFORMANCE

- Inlining is usually a good sign 
- Code bloat might be bad (inlining the cold path) 
- Likely/Unlikely (C++20), Profile-guided optimization can improve performance (**Bolt**)
- Depends on the architecture

## RUN-TIME PERFORMANCE

- Inlining is usually a good sign 
- Code bloat might be bad (inlining the cold path) 
- Likely/Unlikely (C++20), Profile-guided optimization can improve performance (**Bolt**)
- Depends on the architecture
- Optimizations: -O3 -march=-flto -fno-exceptions -DNDEBUG (no debug builds)

# BENCHMARKS - RUN-TIME PERFORMANCE - INSTRUCTIONS PER CYCLE / RELEASE





# INSTRUCTIONS PER CYCLE (IPC)

## INSTRUCTIONS PER CYCLE (IPC)

- An average number of instructions executed for each clock cycle

## INSTRUCTIONS PER CYCLE (IPC)

- An average number of instructions executed for each clock cycle
- The higher the IPC the better (~2.0 is a good sign)

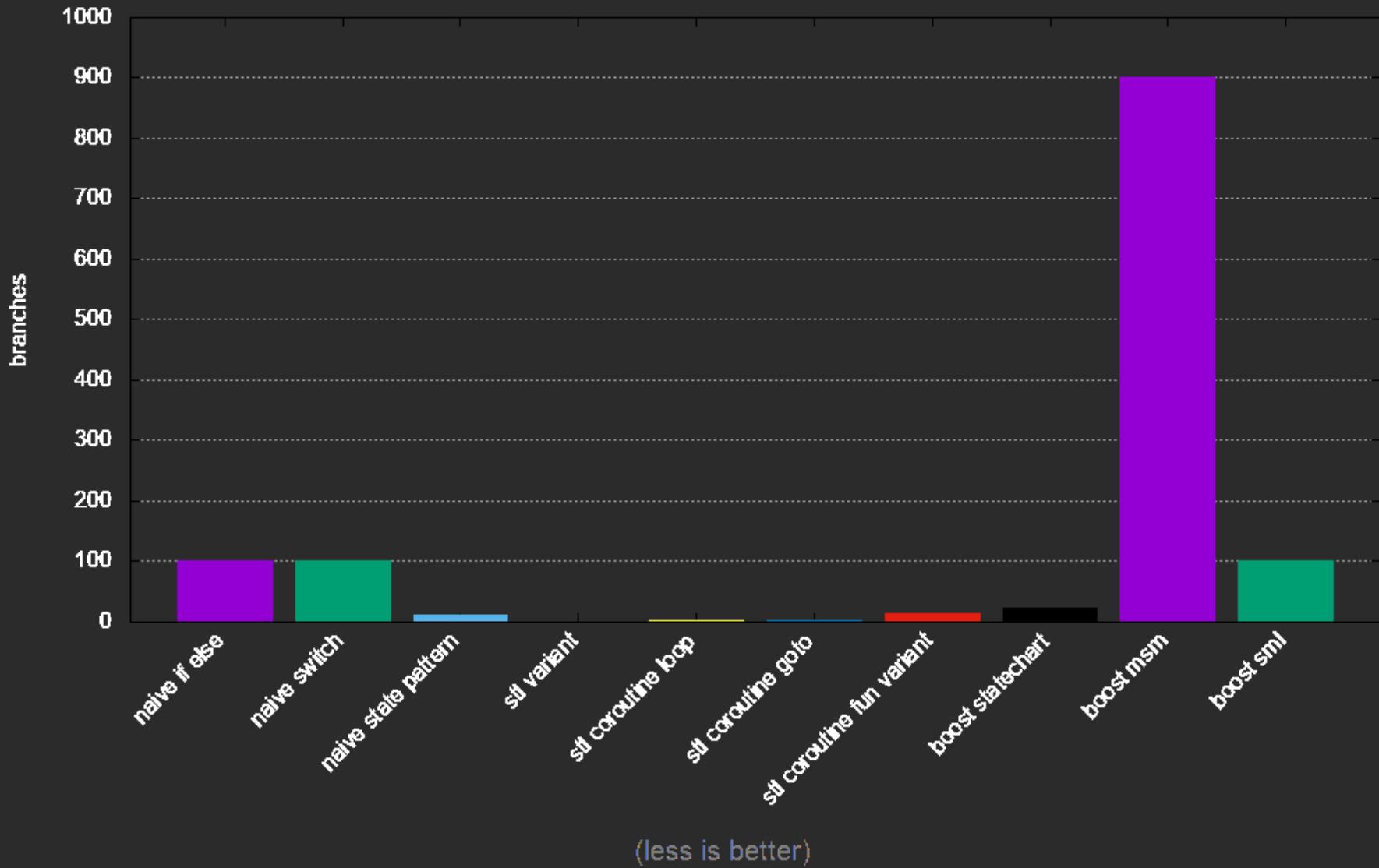


## INSTRUCTIONS PER CYCLE (IPC)

- An average number of instructions executed for each clock cycle
- The higher the IPC the better (~2.0 is a good sign)  

- Depends on the architecture

# BENCHMARKS - RUN-TIME PERFORMANCE - BRANCHES / RELEASE





# **BRANCHES**

# BRANCHES

- Is better to avoid too many branches 

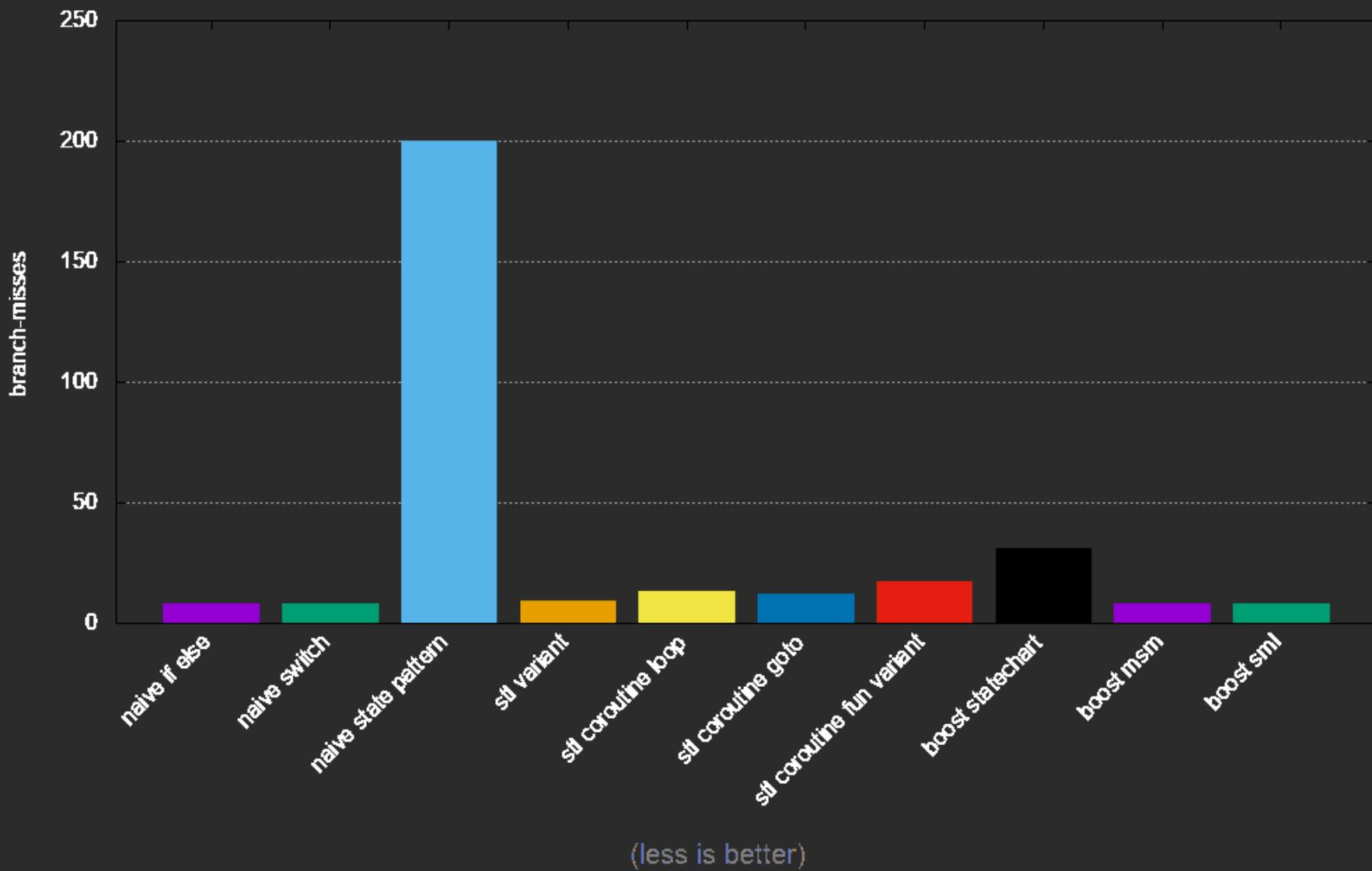
# BRANCHES

- Is better to avoid too many branches 
- Modern branch predictors are really good with learning patterns 

# BRANCHES

- Is better to avoid too many branches 
- Modern branch predictors are really good with learning patterns 
- Branches may lead into branch misses 

# BENCHMARKS - RUN-TIME PERFORMANCE - BRANCH MISSES / RELEASE





# **BRANCH MISSES**

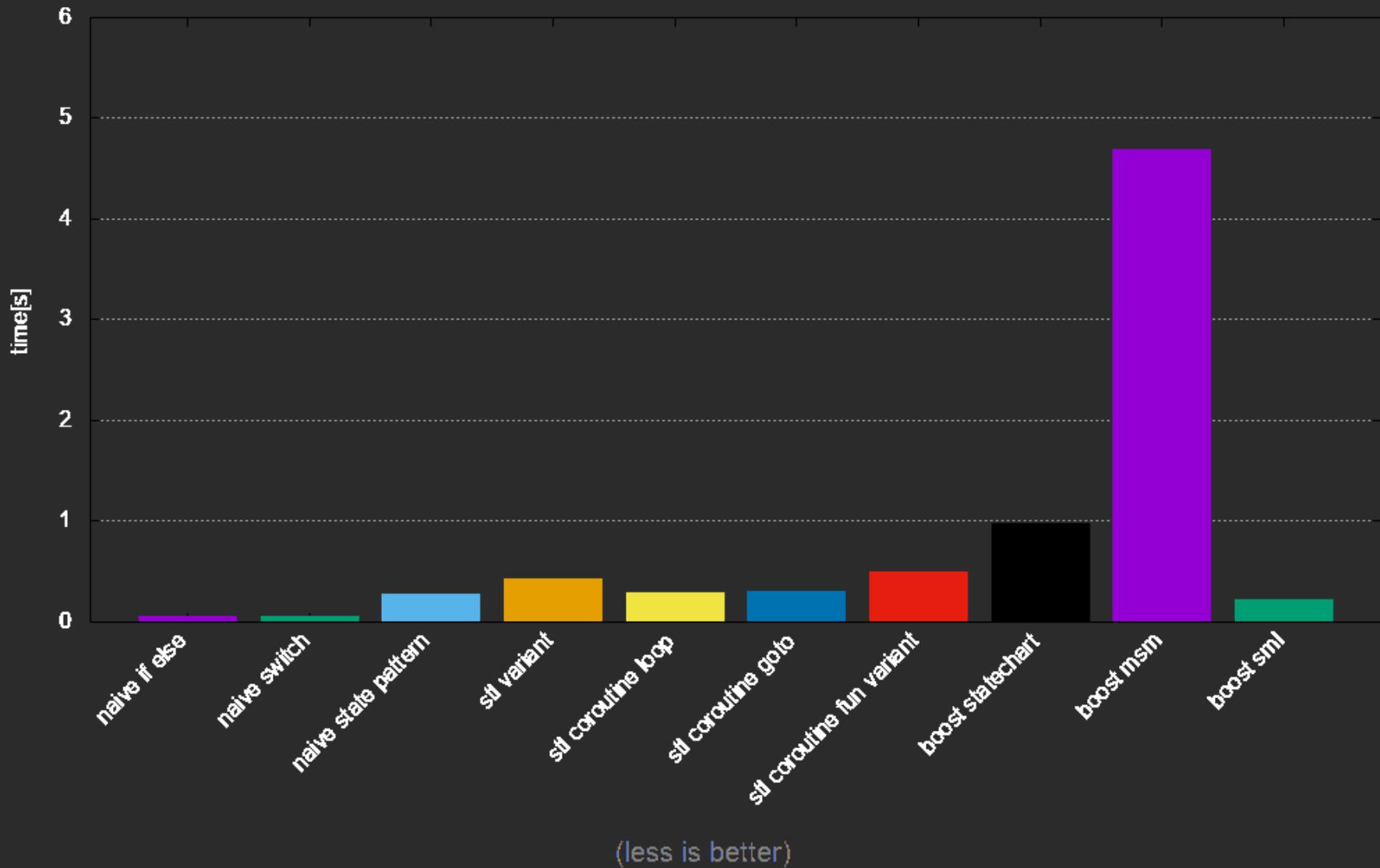
## BRANCH MISSES

- Branch misprediction affects performance 

## BRANCH MISSES

- Branch misprediction affects performance 
- Cache misses have negative performance impact 

# BENCHMARKS - COMPILE TIME / DEBUG





# COMPILATION TIME / DEBUG

## COMPILATION TIME / DEBUG

- Template Meta Programming (TMP) is slow to compile before C++11 

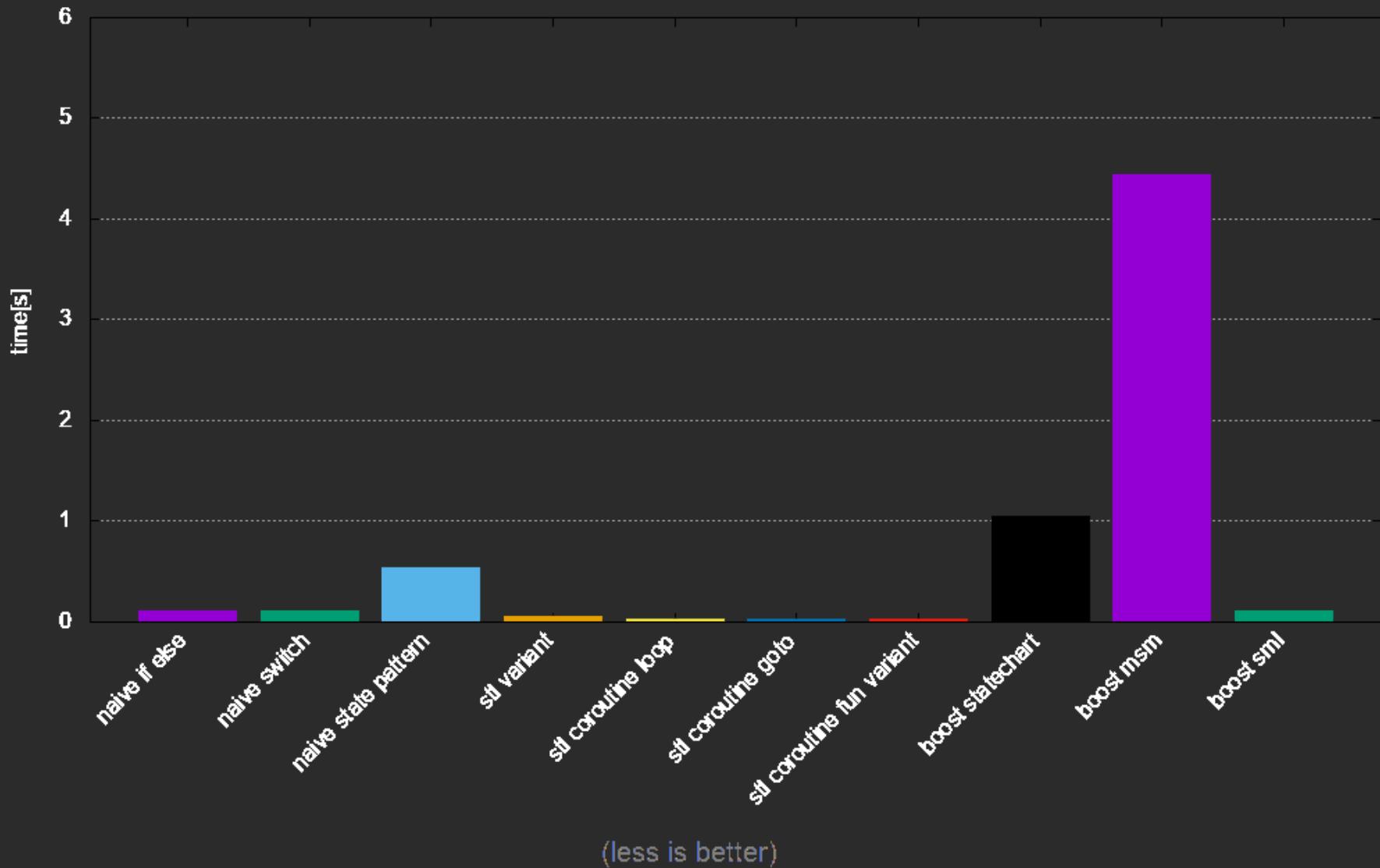
## COMPILATION TIME / DEBUG

- Template Meta Programming (TMP) is slow to compile before C++11 
- TMP may compile really fast since  $\geq$  C++11 

## COMPILATION TIME / DEBUG

- Template Meta Programming (TMP) is slow to compile before C++11 
- TMP may compile really fast since  $\geq$  C++11 
- Creating long debug symbols may slow down compilation times 

# BENCHMARKS - COMPILED TIME / RELEASE





# COMPILATION TIME / RELEASE

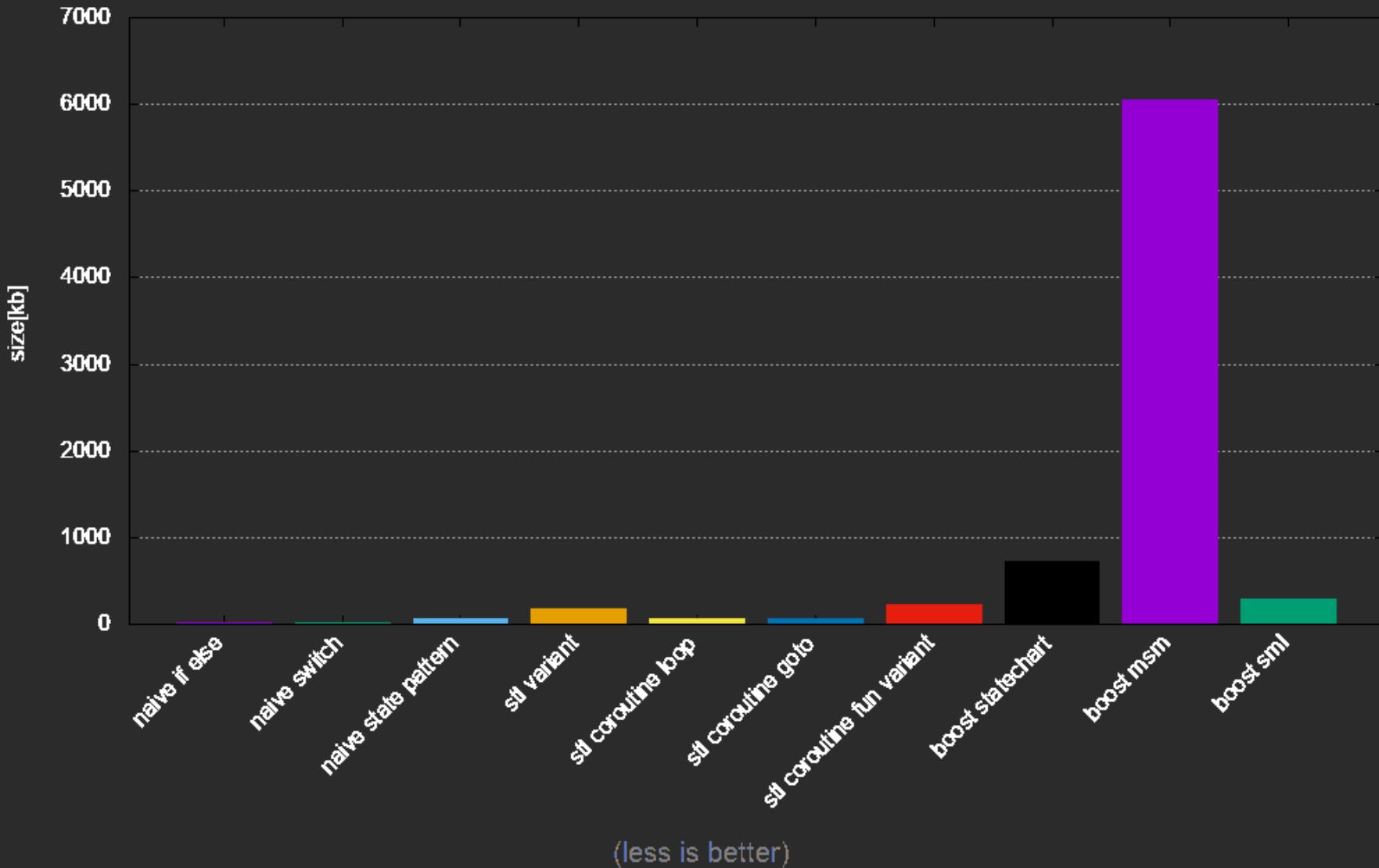
## COMPILATION TIME / RELEASE

- C++98 < TMP >>>> C++11

## COMPILATION TIME / RELEASE

- C++98 < TMP >>>> C++11
- Turning on optimizations may speed up compilation times 

# BENCHMARKS - EXECUTABLE SIZE / DEBUG



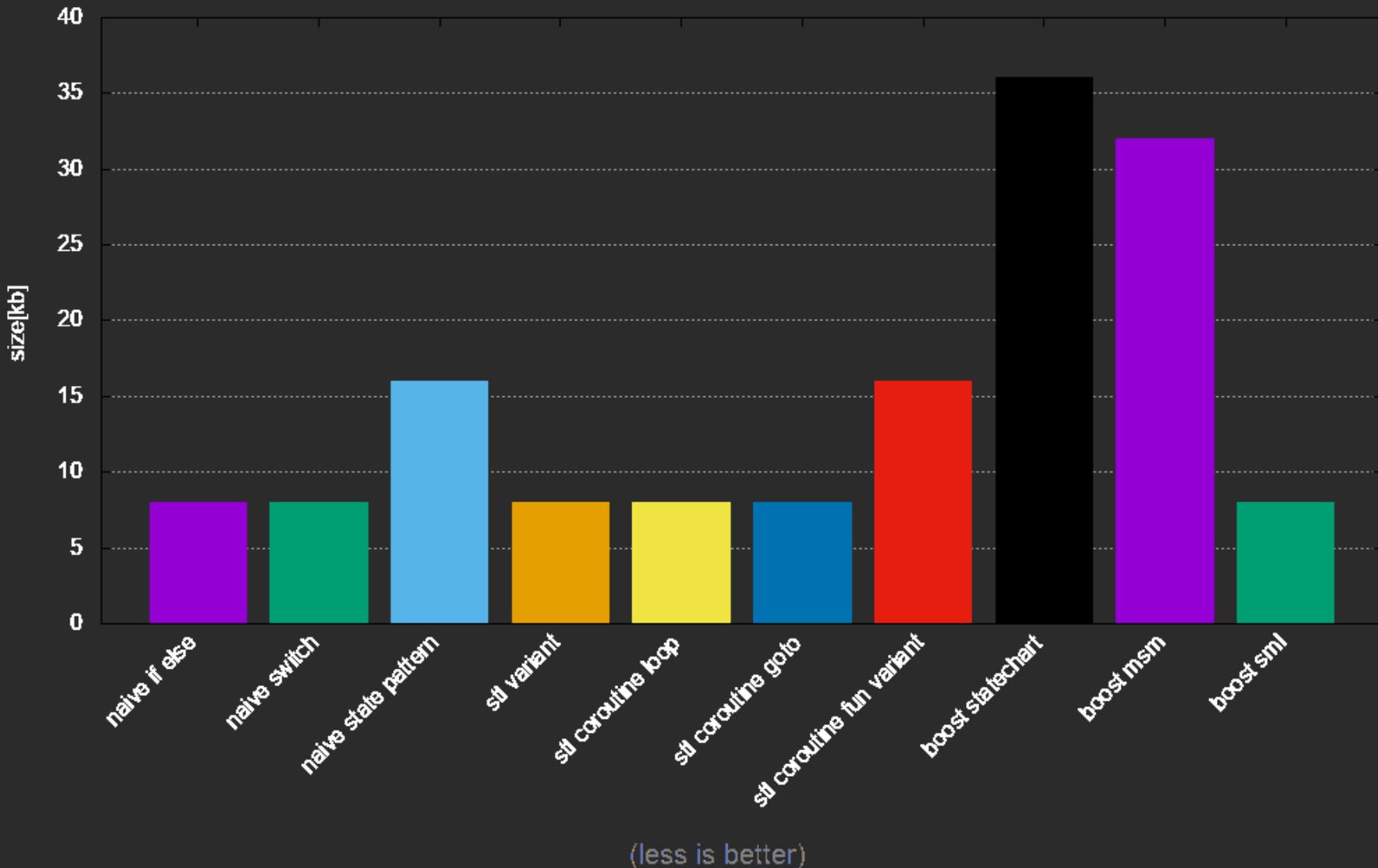


# EXECUTABLE SIZE / DEBUG

## EXECUTABLE SIZE / DEBUG

- Debug symbols for templated code might be huge (GB's) 

# BENCHMARKS - EXECUTABLE SIZE / RELEASE





# EXECUTABLE SIZE / RELEASE

## EXECUTABLE SIZE / RELEASE

- Usually the smaller size the better the performance 

## EXECUTABLE SIZE / RELEASE

- Usually the smaller size the better the performance 
- Code layout may have impact on the performance 

**ALWAYS MEASURE IN THE PRODUCTION 'LIKE'  
ENVIRONMENT!**

# ALWAYS MEASURE IN THE PRODUCTION 'LIKE' ENVIRONMENT!

- There is no silver bullet - keeping all options open via policies is the best bet 

# SUMMARY

# SUMMARY

- State machines are expressive way to represent the application flow

# SUMMARY

- State machines are expressive way to represent the application flow
- There are different ways of implementing state machines in C++ with different trade-offs

# SUMMARY

- State machines are expressive way to represent the application flow
- There are different ways of implementing state machines in C++ with different trade-offs
- State Machines are more than just simple transitions **UML-2.5**

# SUMMARY

- State machines are expressive way to represent the application flow
- There are different ways of implementing state machines in C++ with different trade-offs
- State Machines are more than just simple transitions **UML-2.5**
- Leveraging Zero-cost libraries can boost the design and/or performance

# LET'S EMBRACE ZERO-COST STATE MACHINE LIBRARIES!

---

Benchmarks

<https://github.com/boost-experimental/sml/tree/master/benchmark/connection>

---

Slides

<http://boost-experimental.github.io/sml/cppnow-2019>

-

KRIS@JUSIAK.NET | @KRISJUSIAK | LINKEDIN.COM/IN/KRIS-JUSIAK