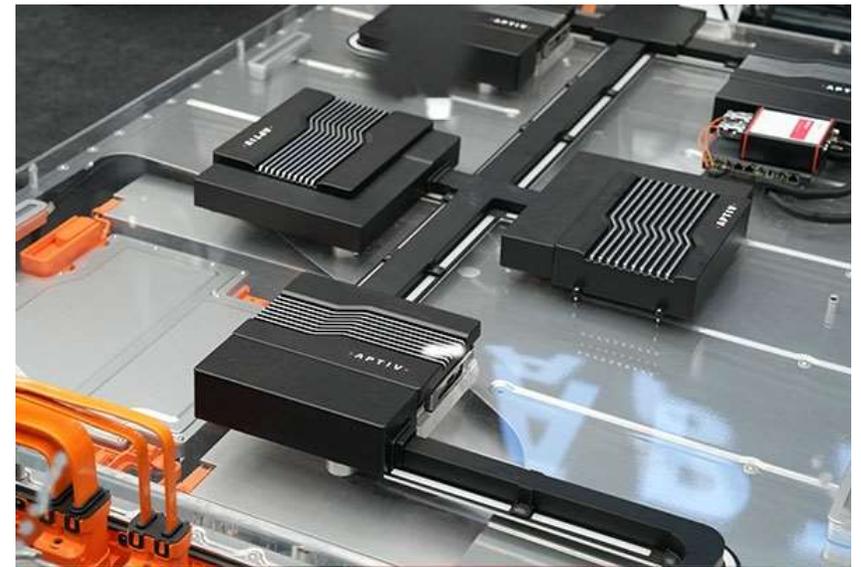


November 12th, 2024

Christof Petig
Staff SW engineer

● COMPONENT MODEL IN SOFTWARE DEFINED VEHICLES



● APTIV ●

Contents

1. **Wasm and the Software Defined Vehicle**

The component model

Canonical lifting and lowering

2. **Native binaries**

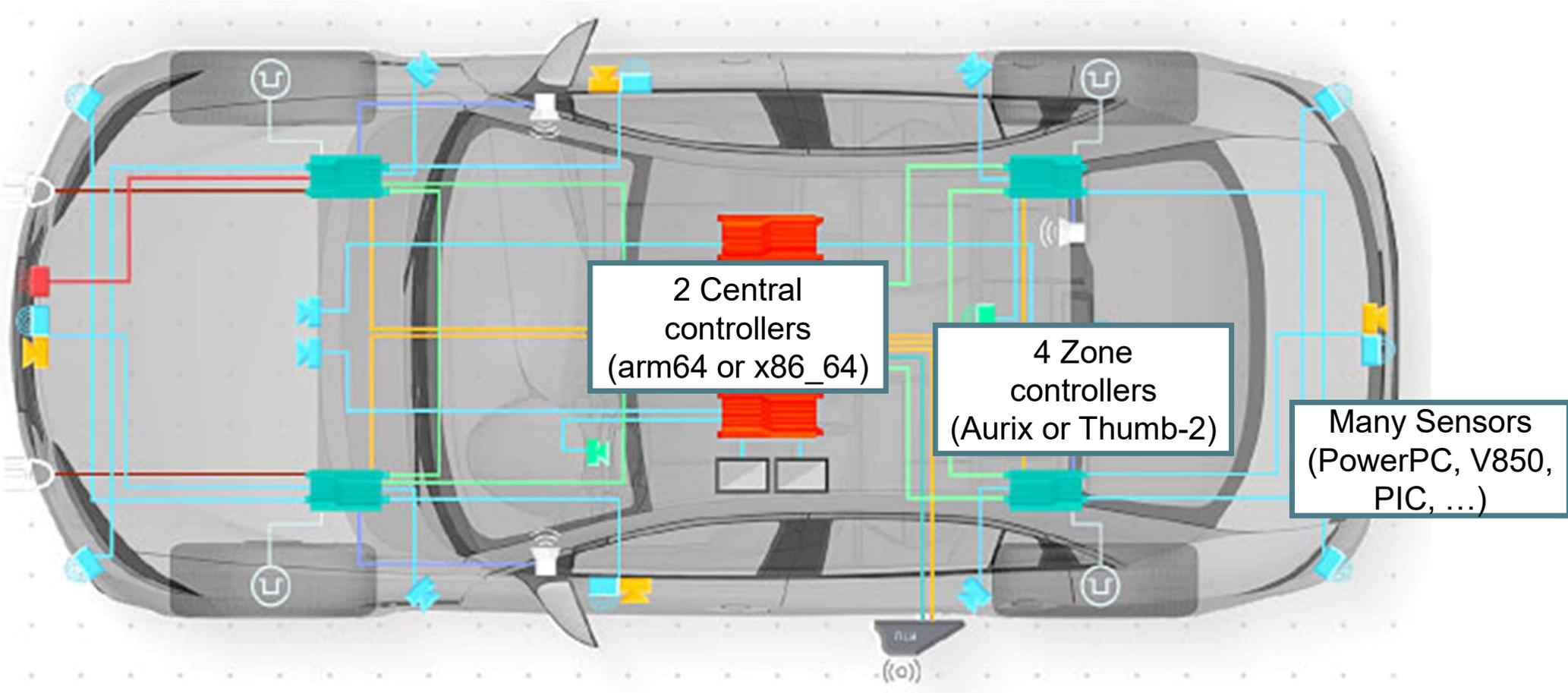
Optimize for Shared everything

Symmetric ABI

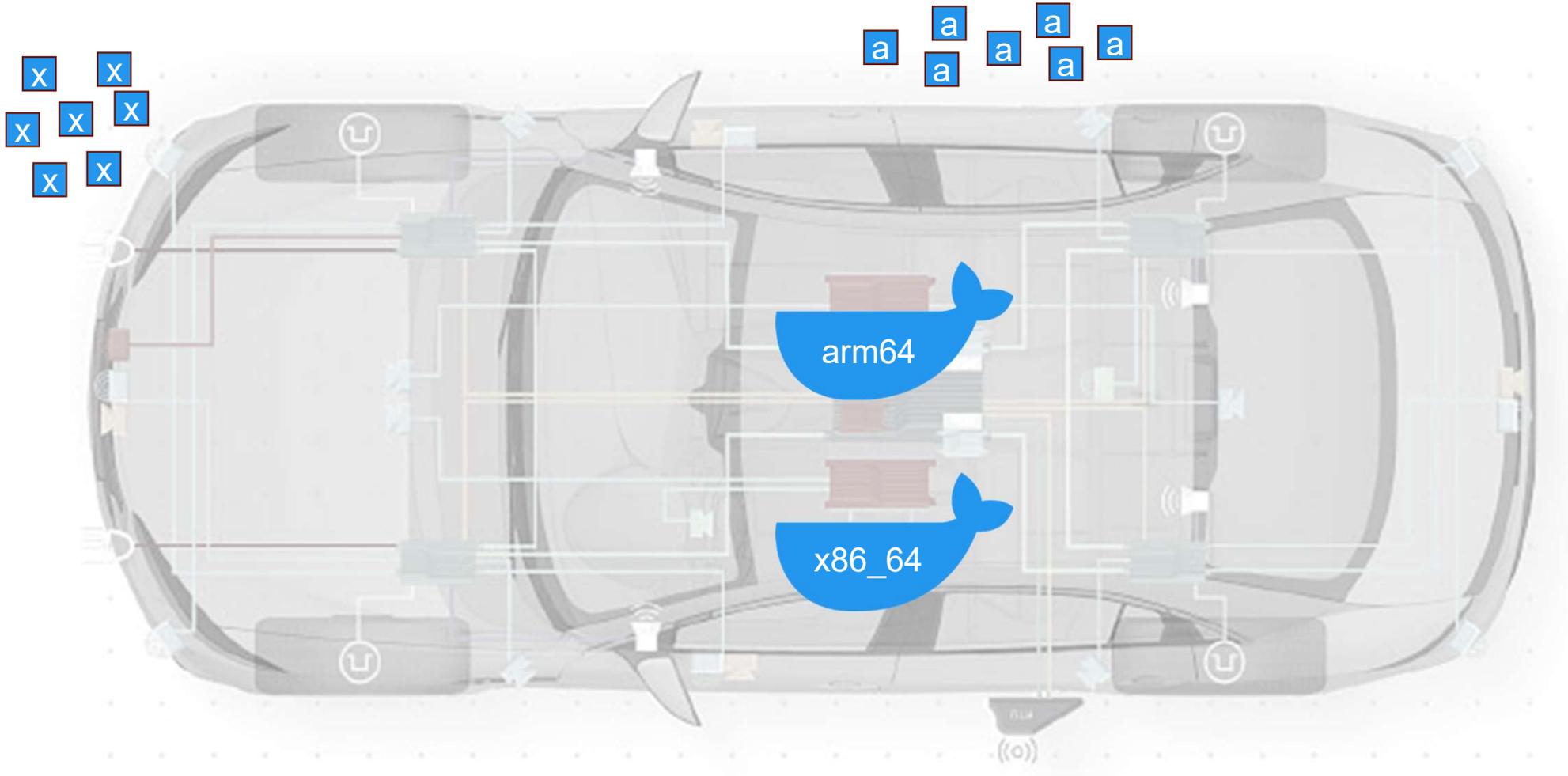
3. **Outlook**

Wasm helps SDV

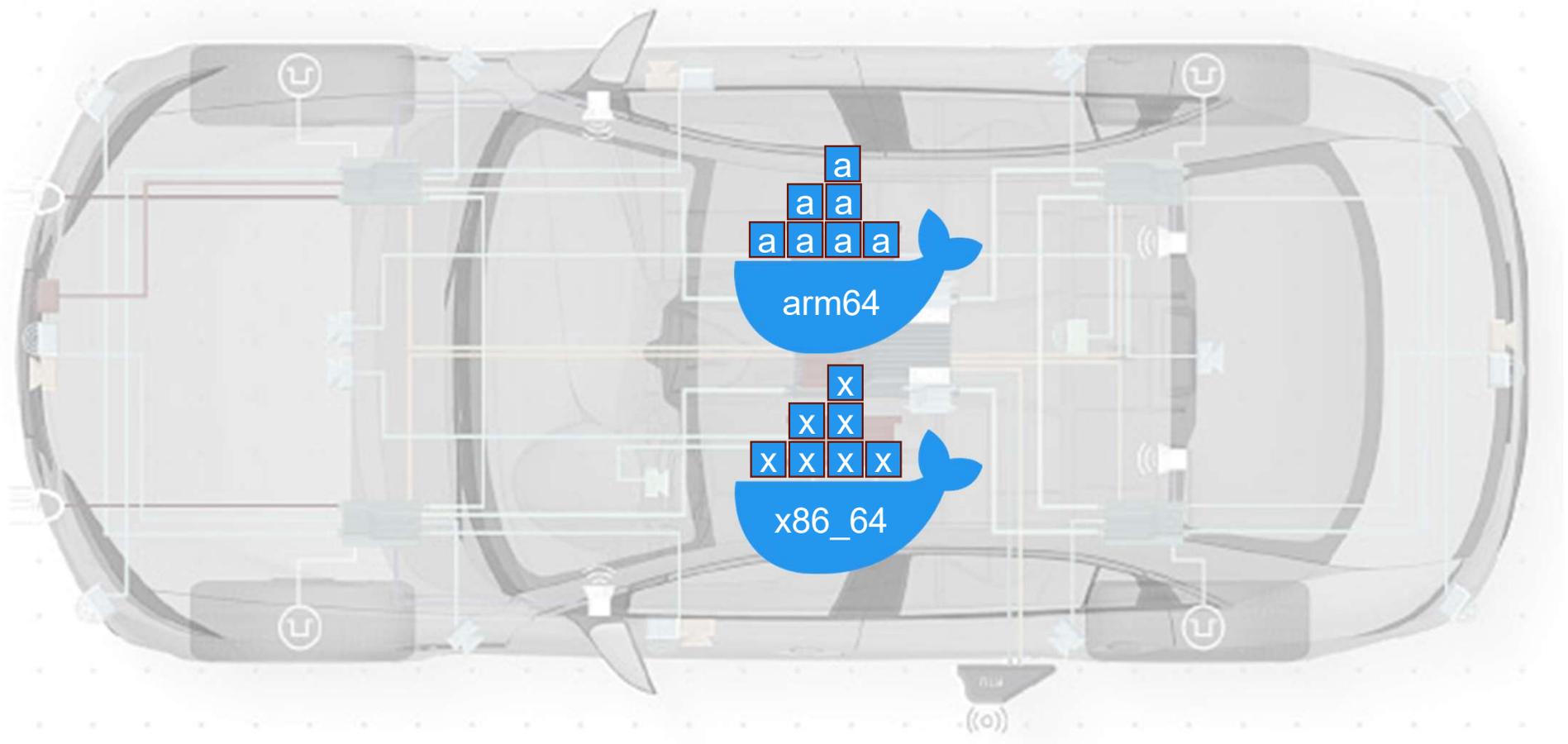
Zonal car architecture



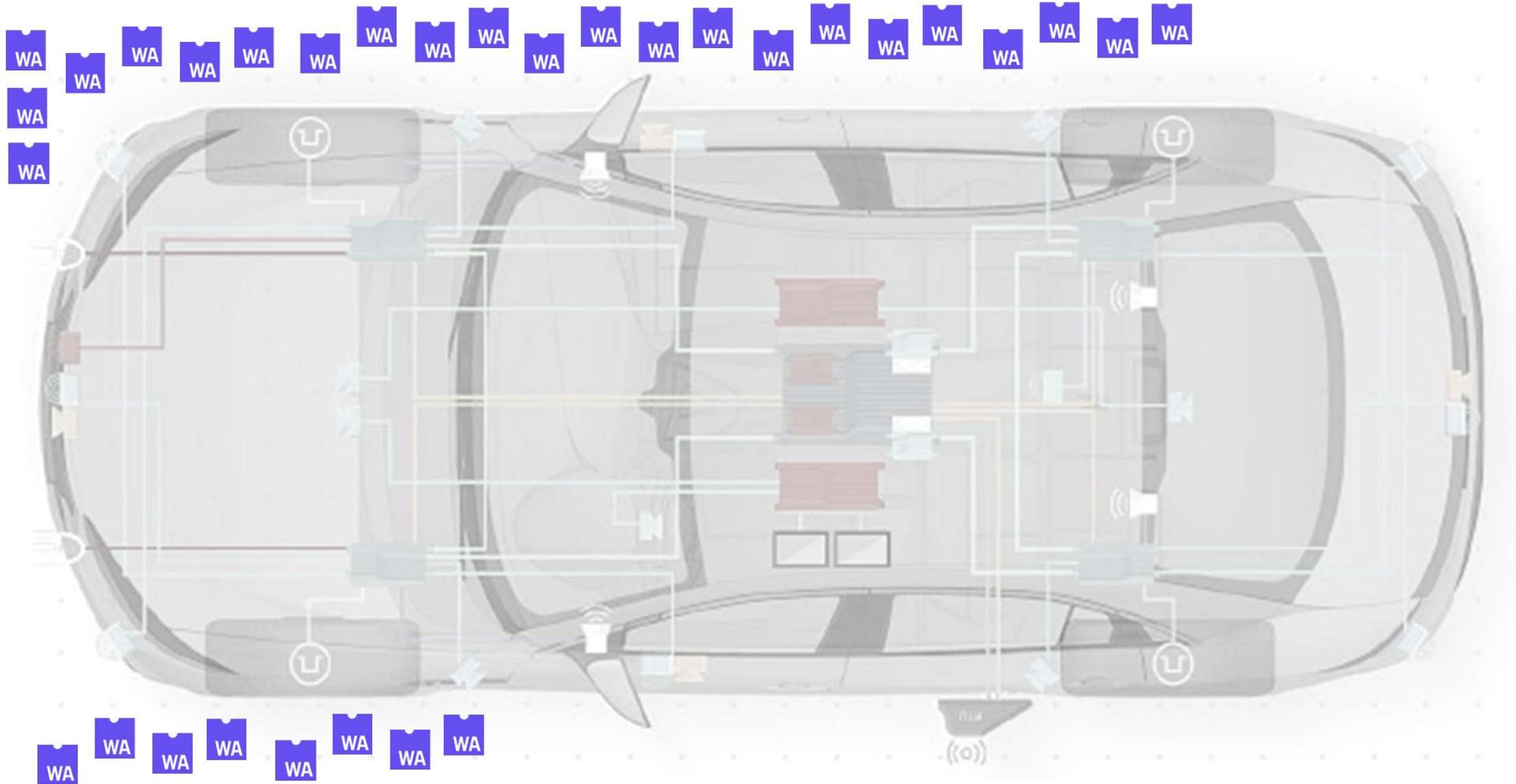
Classic containers are arch. and OS specific



Docker containers are arch. and OS specific



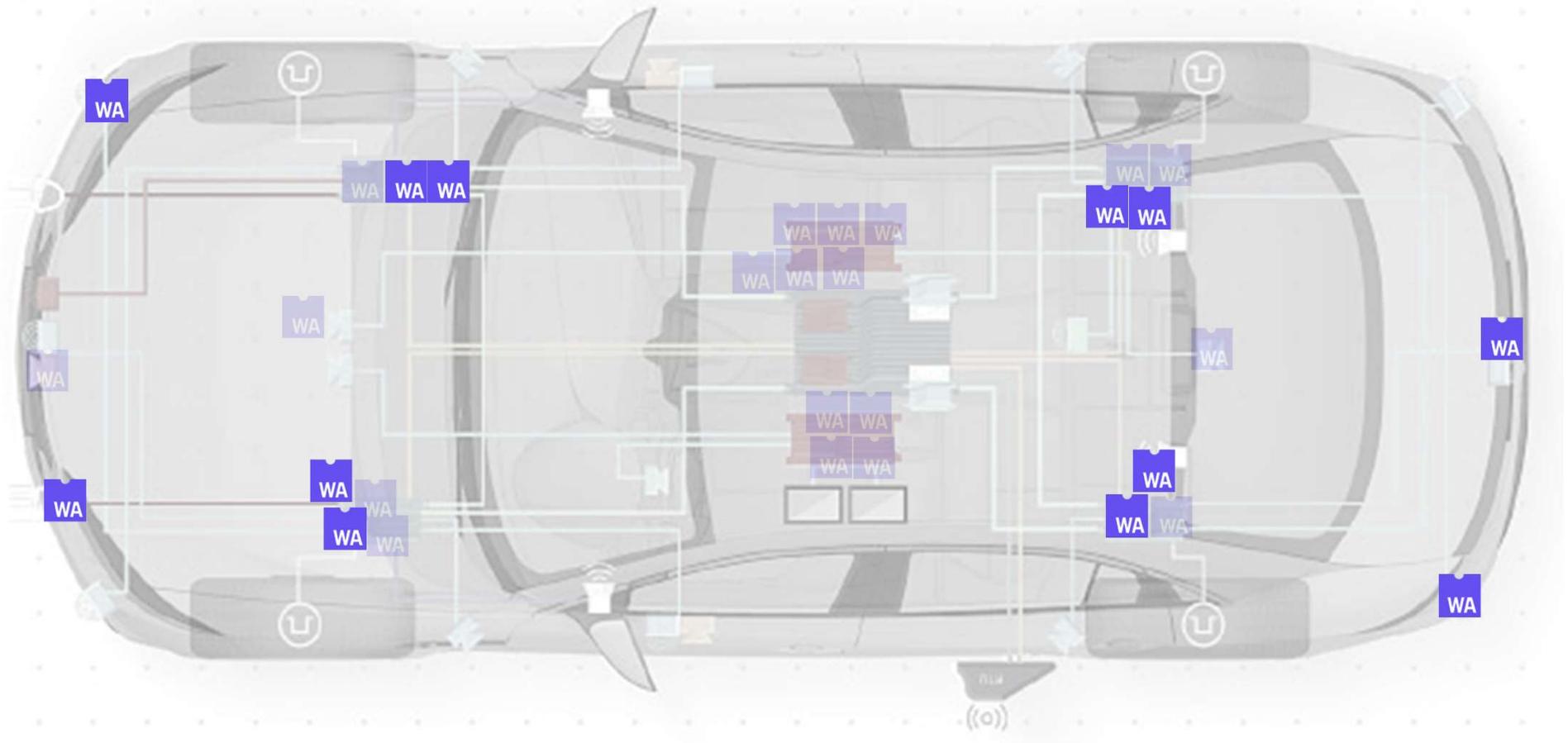
Wasm components fit anywhere



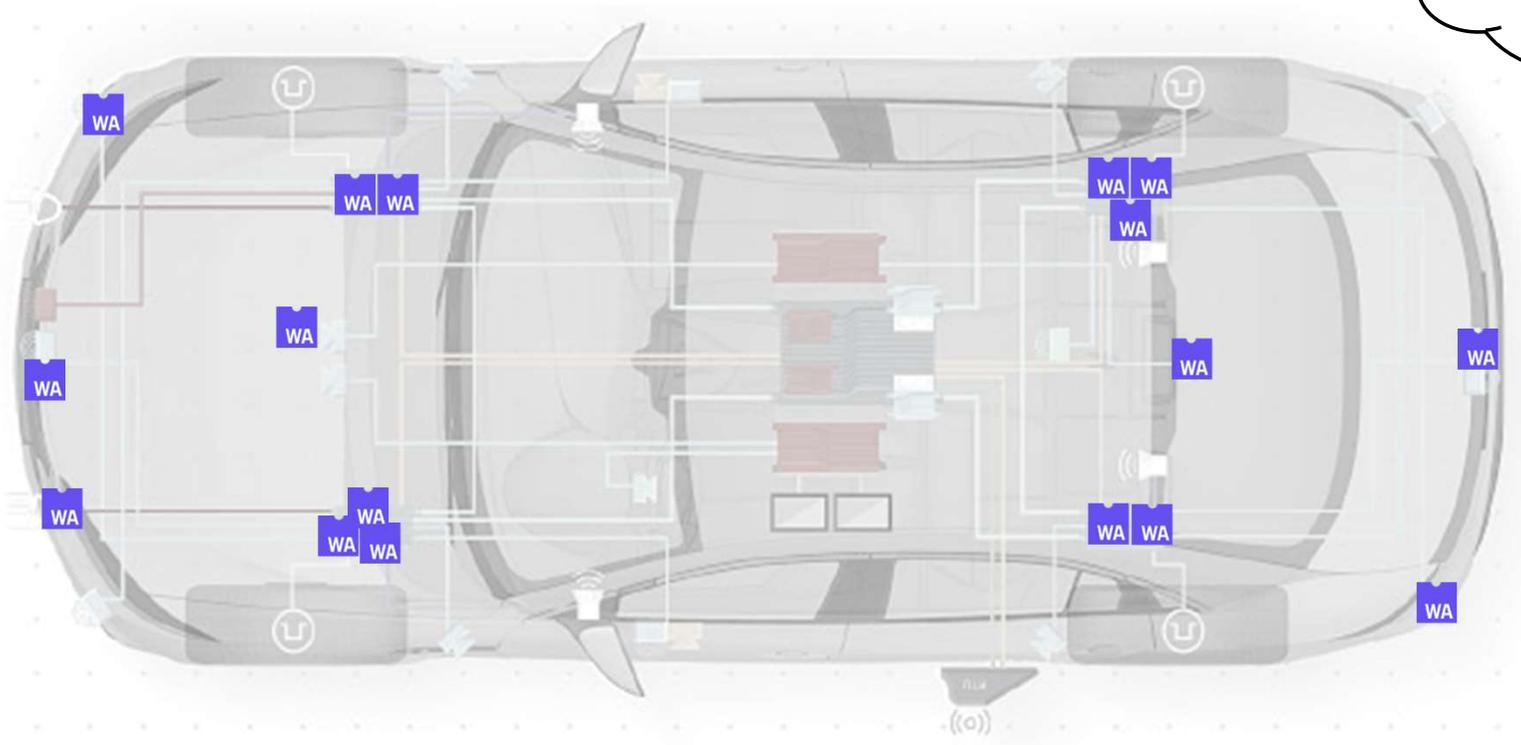
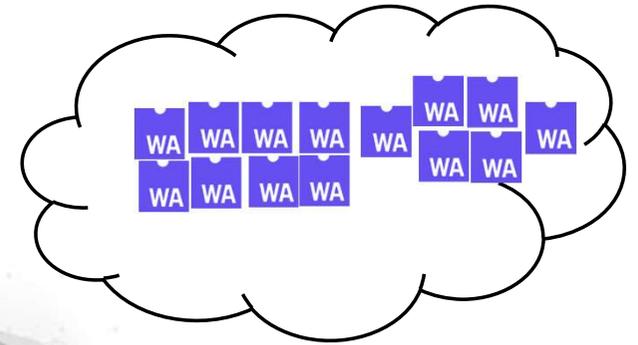
Wasm components fit anywhere



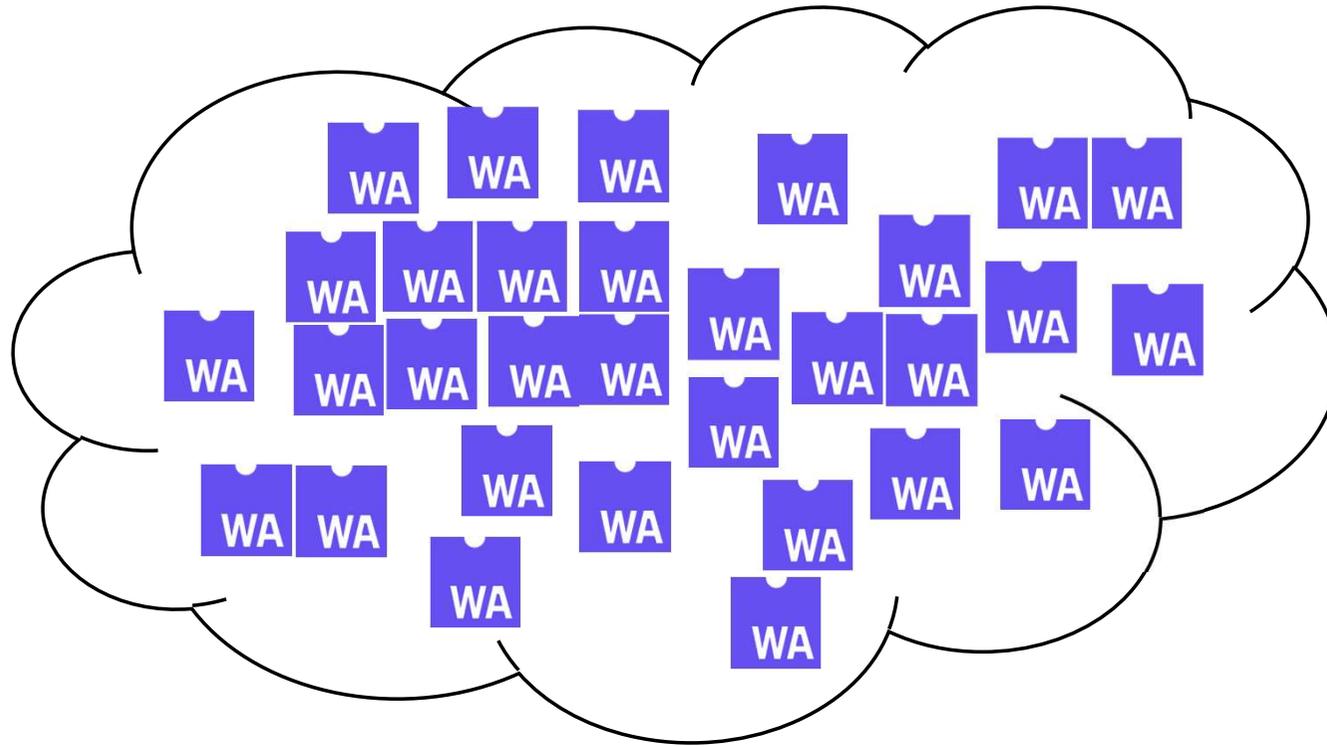
You can move them on powerdown



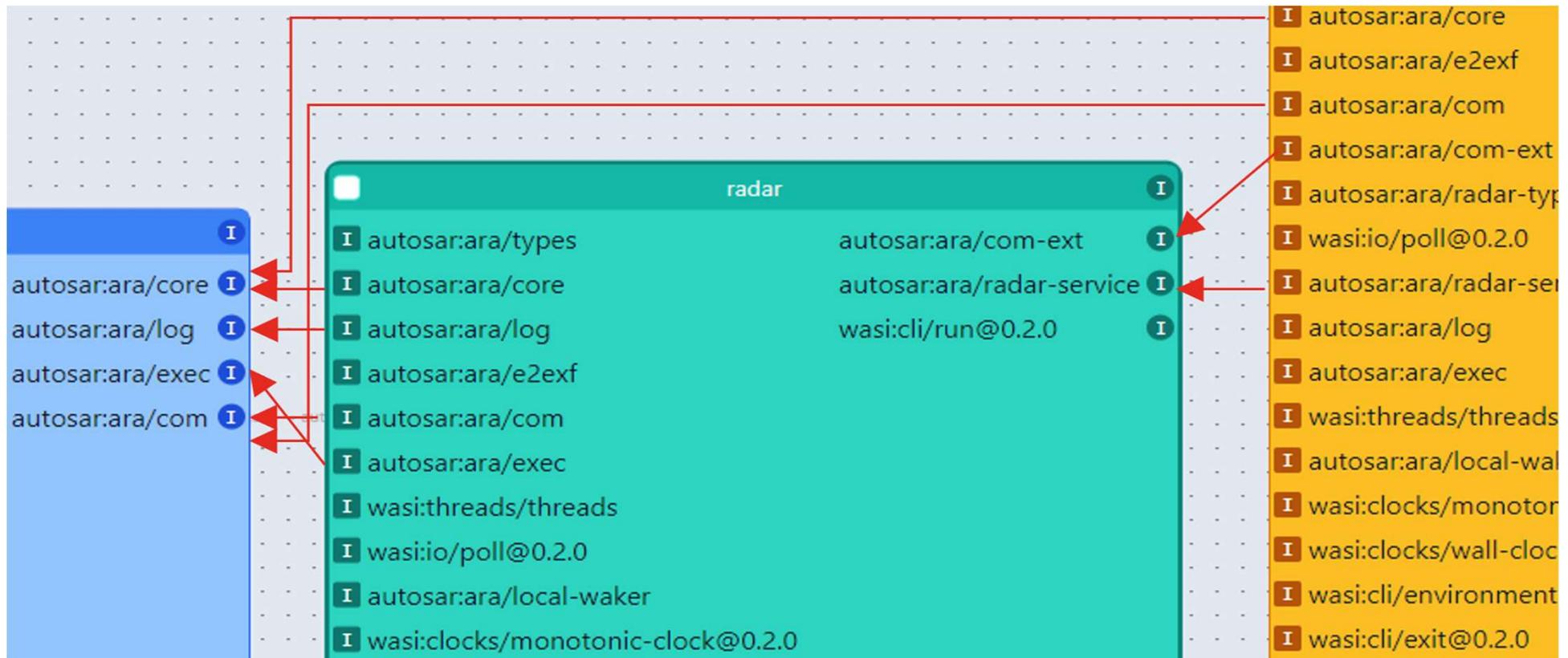
You can run components on the edge



... or fully simulate the vehicle



Component Model offers graphical composition



... and debugging in the browser

UpdateRate updated
FrontObjectDistance updated 5
radar active
brakeEvent sent
parkingBrakeEvent sent
FrontObjectDistance updated 39
METHODS: Target position isx, y, z (4,2,3), adjusting position ...
METHODS: Adjusting position was successful, effective position equals target position
METHODS: Effective position isx, y, z (4,2,3)
radar active
brakeEvent sent
parkingBrakeEvent sent

Adjust 4, 2, 3
Calibrate config
Echo text
Update rate 0
Front distance
Rear distance
Limit 0
{"success":true,"effectivePos

Application Output

I/O values & Triggers = fusion app

Breakpoint

Single stepping

Local variables

Call stack

```
318 // FIX for possible threading problem in vSomeIP which led to SEGFALT
319 Dstd::this_thread::sleep_for(Dstd::chrono::milliseconds(10));
320
321 // send sample
322 auto send_result = m_skeleton->parkingBrakeEvent.Send(std::move(l_sampleParkingBrake));
323 if (send_result) {
324     m_logger_ctx3.LogInfo() << "parkingBrakeEvent sent";
325 } else {
326     m_logger_ctx3.LogError() << "parkingBrakeEvent.Send with error: " << send_re
327 }
328
329 // Update Field Value and send notification.
330 if (i % 5 == 0) {
331     m_update_rate = i * 1000;
332     auto update_result = m_skeleton->UpdateRate.Update
```

Line 318, Column 79 (From component.core.wasm) Coverage: n/a

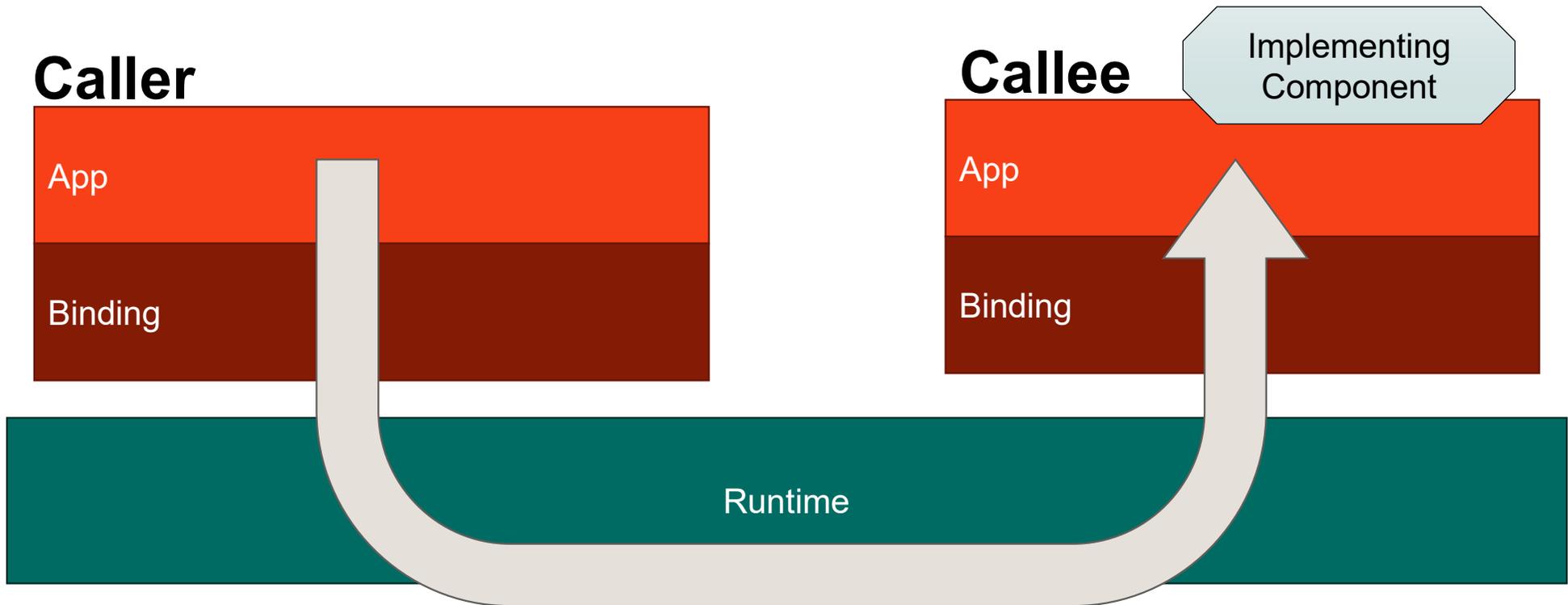
Enabled by the Component Model

- **Language neutral definition**
Wasm Interface Types
- **High level data types**
Option, Result, String, Vector, Future, Stream
- **Standardized calling convention (canonical ABI)**
and system calls (WASI)

Perfect match with
AUTOSAR
Adaptive Platform

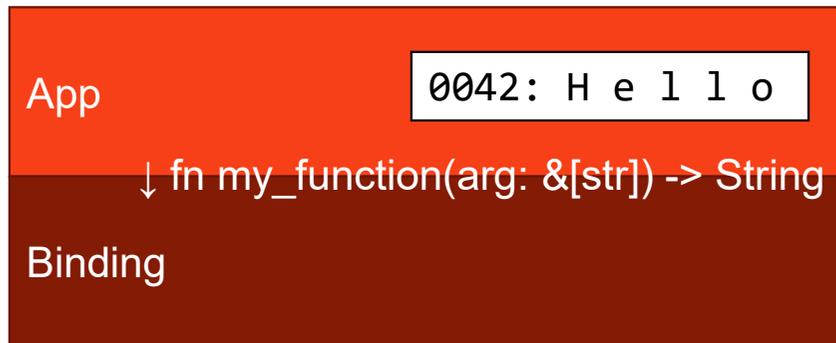
Lifting and Lowering: Canonical ABI

my-function: func(arg: string) -> string

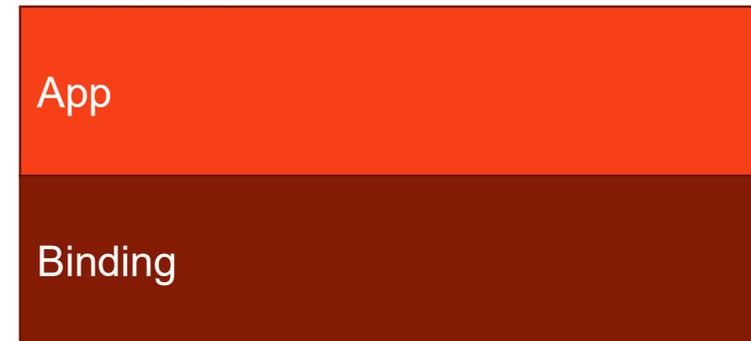


App starts with a string in memory

Caller



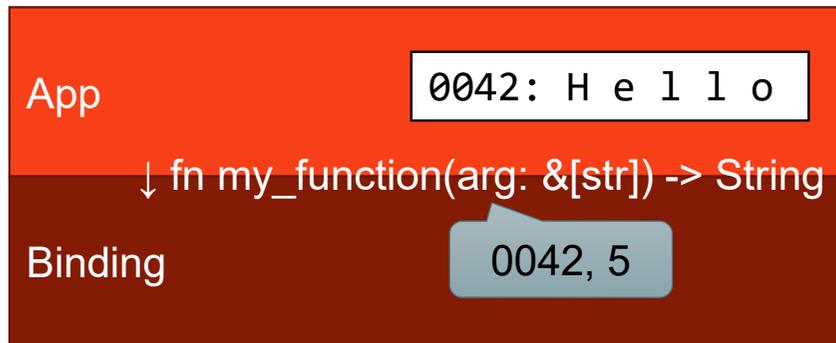
Callee



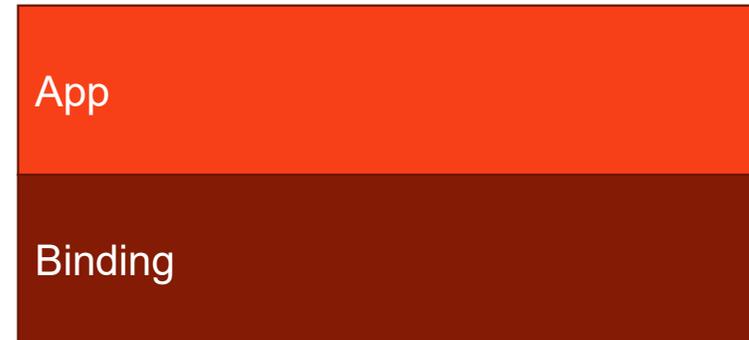
Runtime

App calls binding

Caller



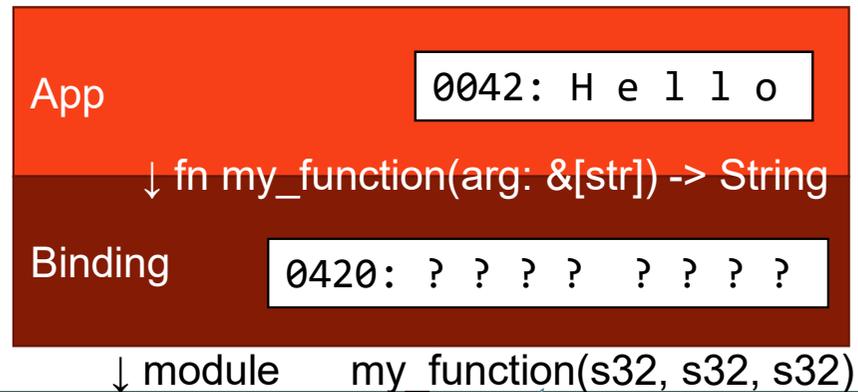
Callee



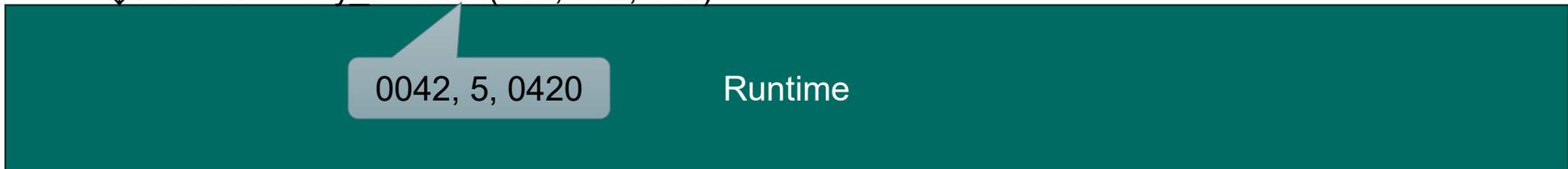
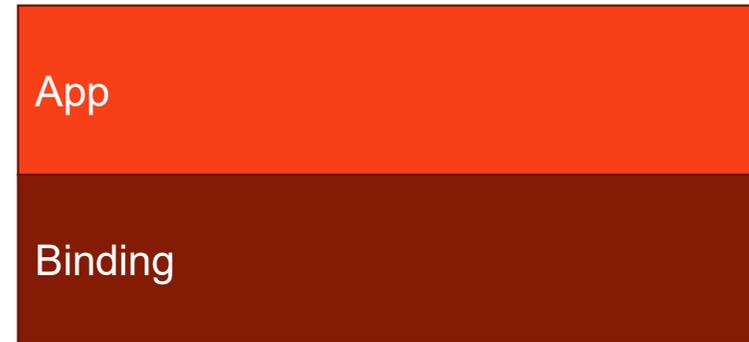
Runtime

Binding elects return area and calls runtime

Caller

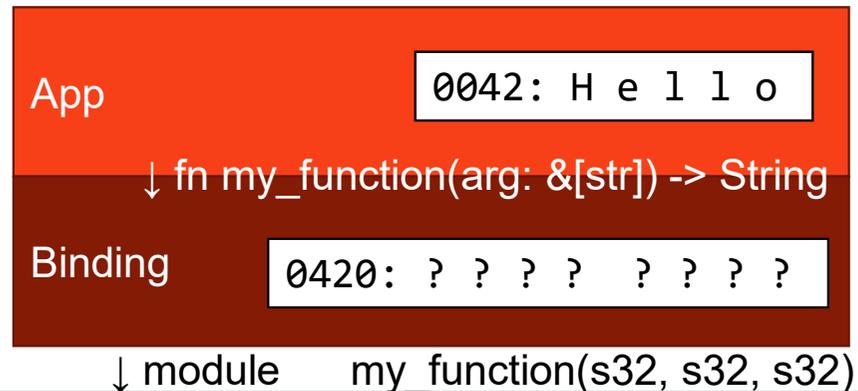


Callee

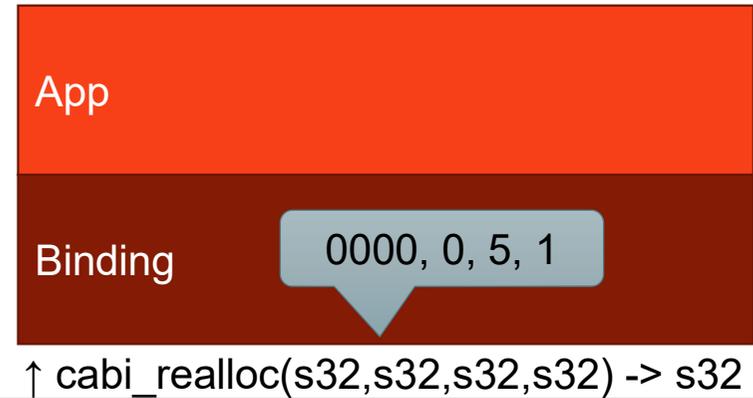


Runtime allocates memory in callee

Caller



Callee

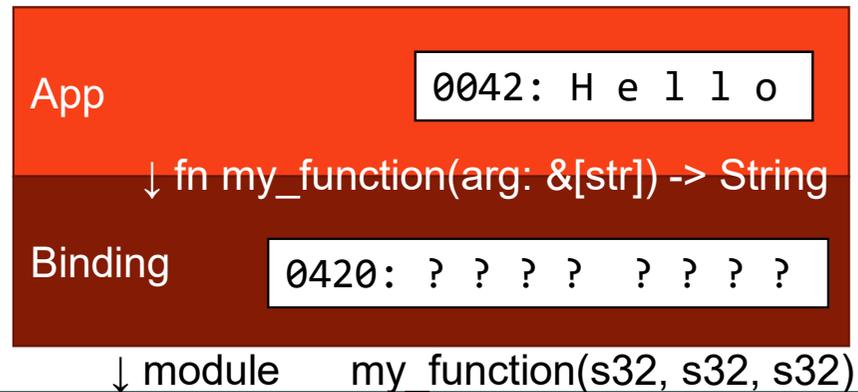


`0042, 5, 0420`

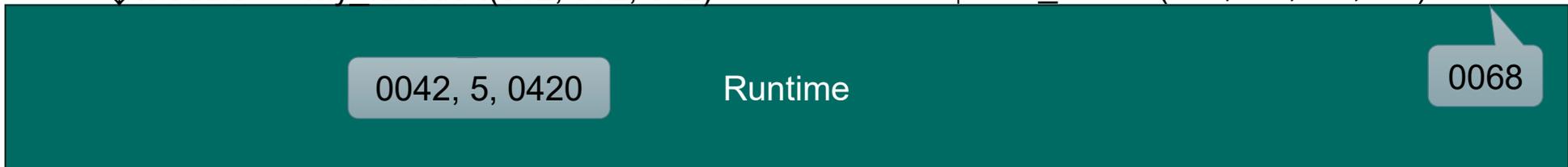
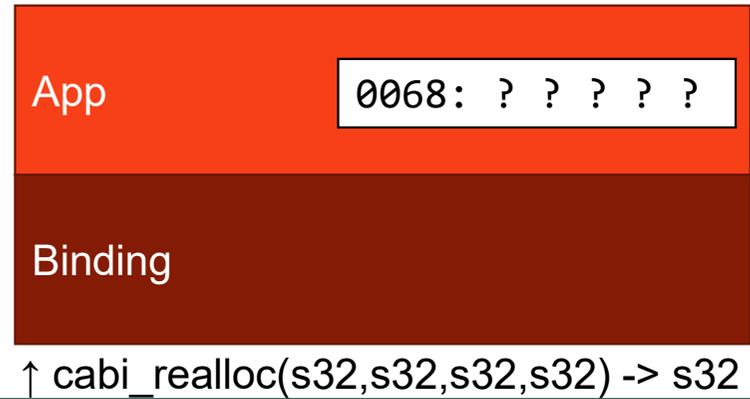
Runtime

Callee provides memory address

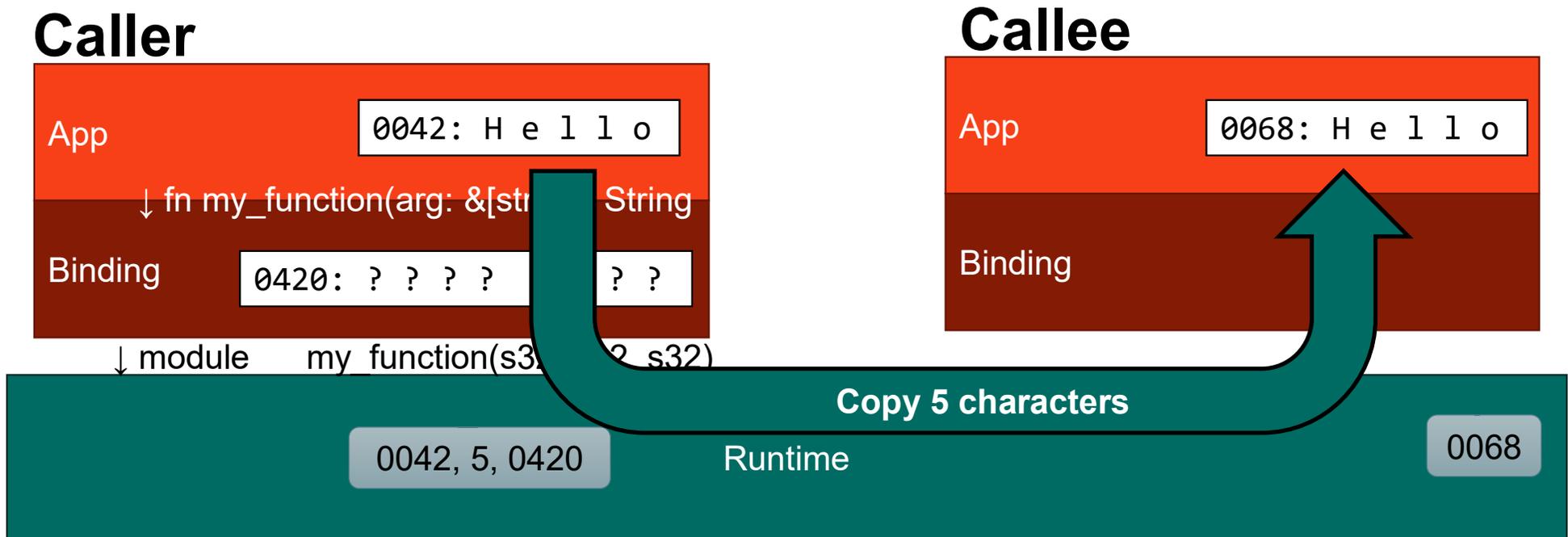
Caller



Callee

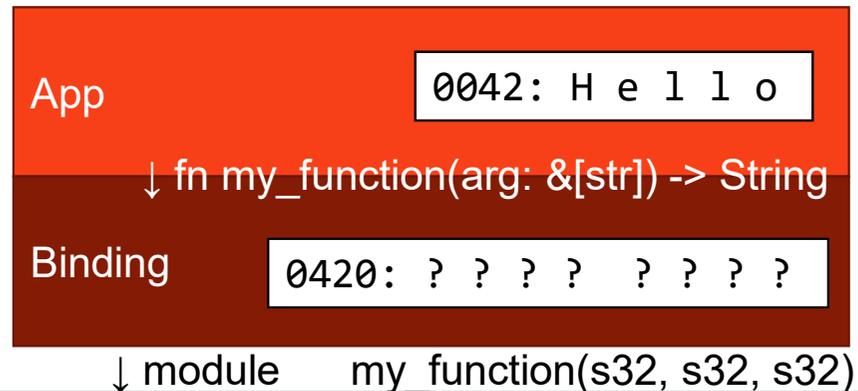


Runtime copies five characters

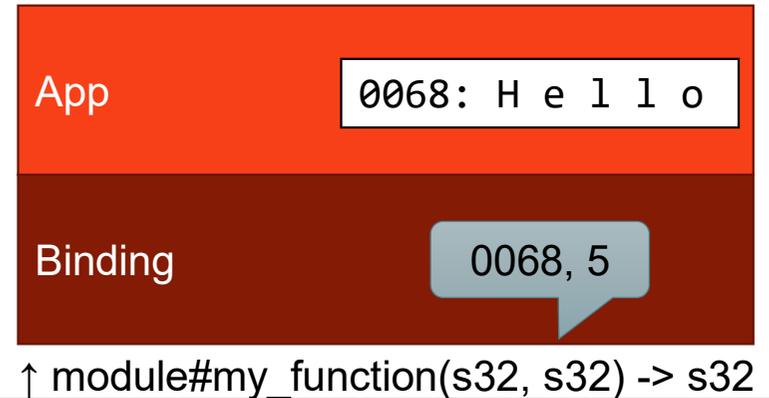


Runtime calls binding

Caller



Callee

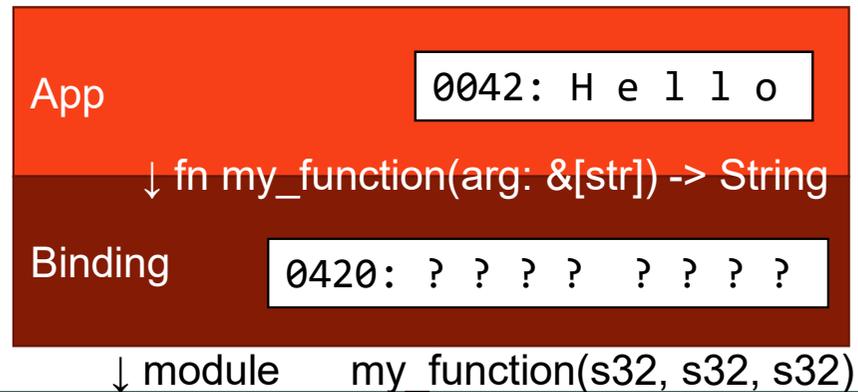


`0042, 5, 0420`

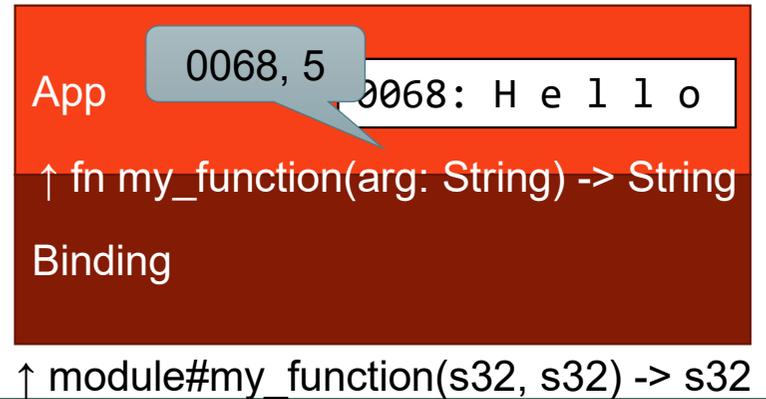
Runtime

Binding calls implementation

Caller



Callee

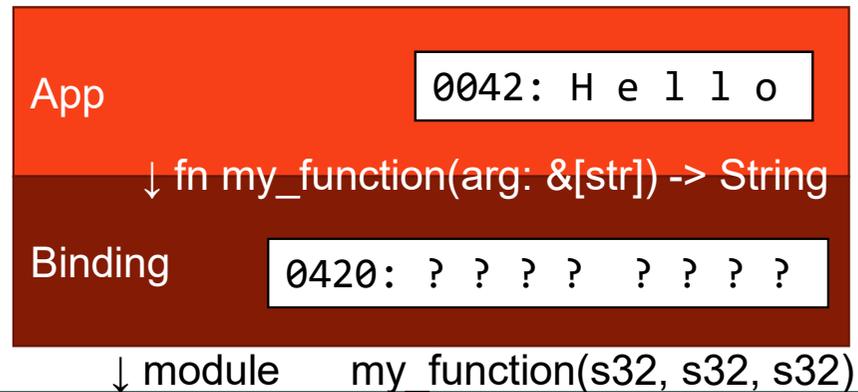


0042, 5, 0420

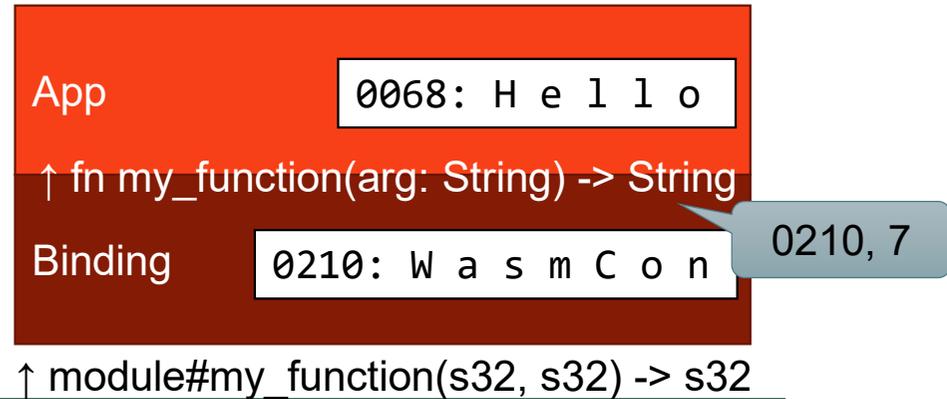
Runtime

Implementation returns String

Caller

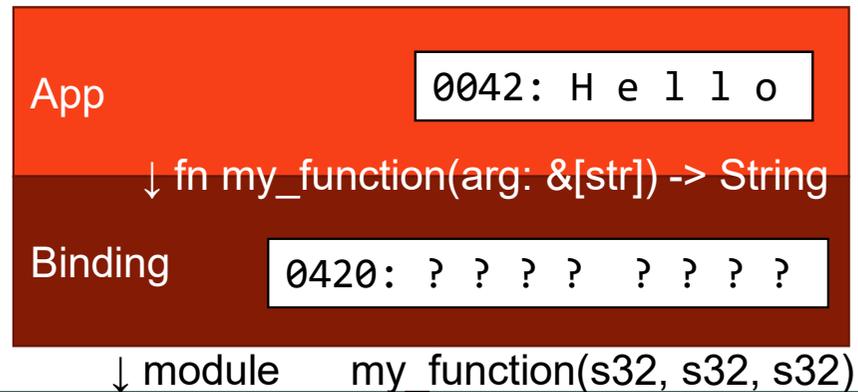


Callee

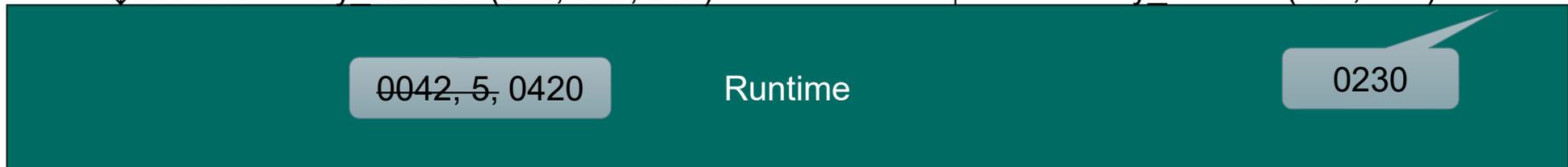
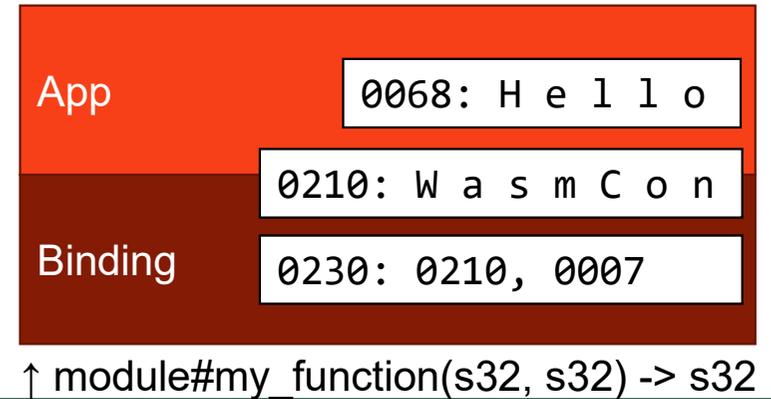


Binding returns static return area

Caller

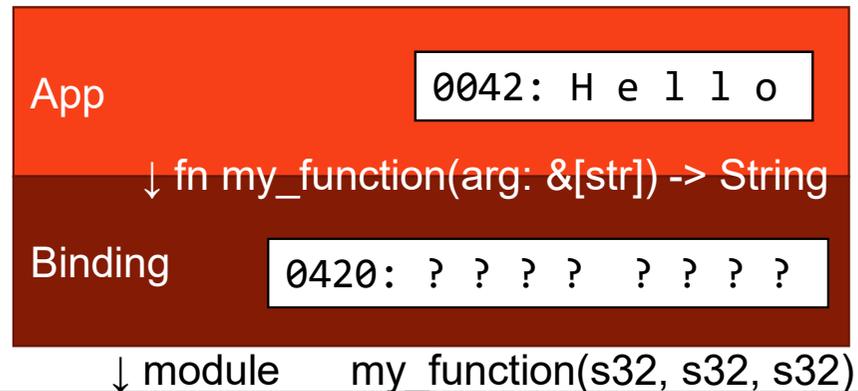


Callee

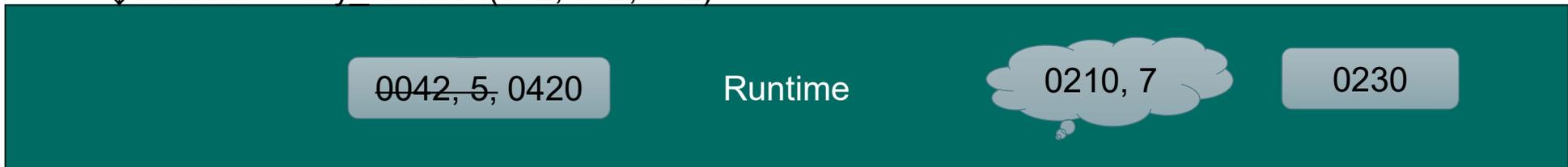
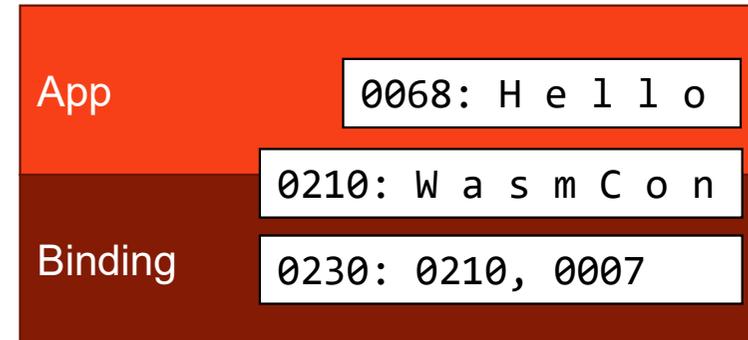


Runtime reads address and length

Caller

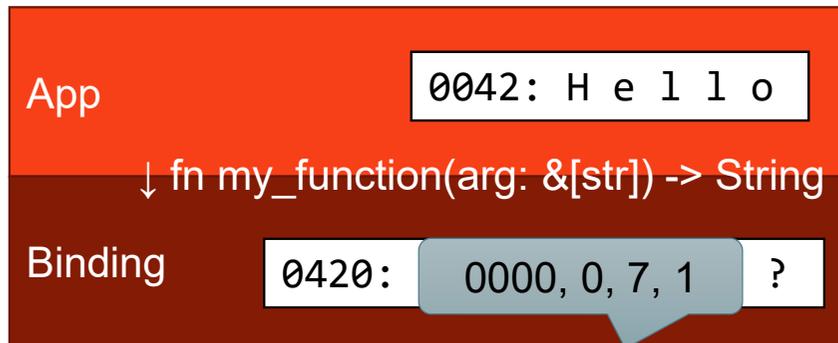


Callee

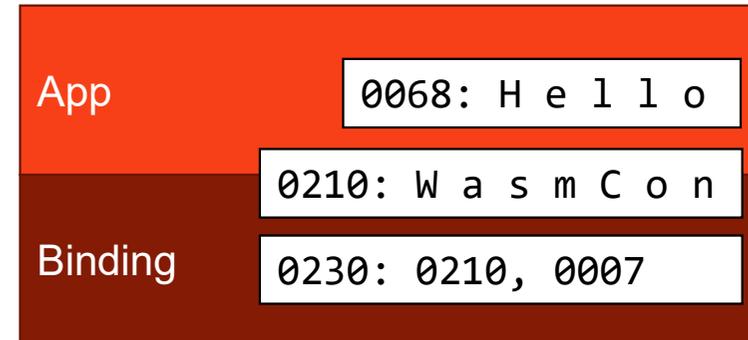


Runtime allocates memory in caller

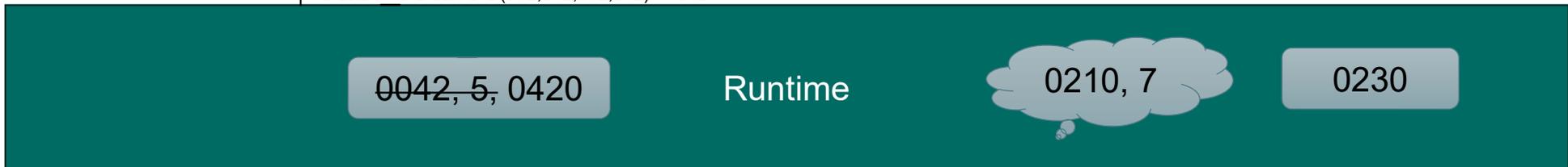
Caller



Callee

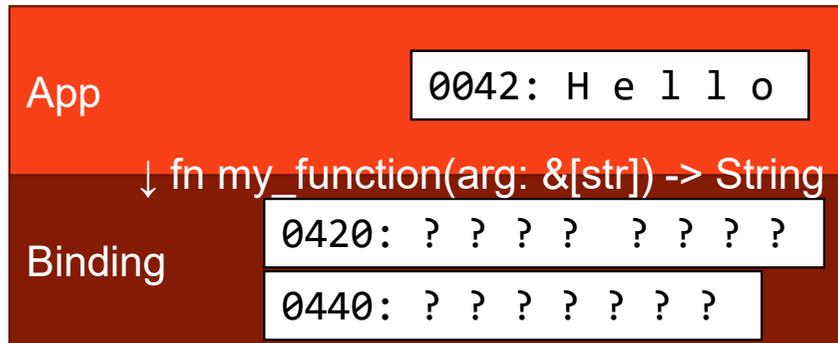


module ↓ my_function(s32, s32, s32) ↑ cabi realloc(s32,s32,s32,s32) -> s32



Caller returns address

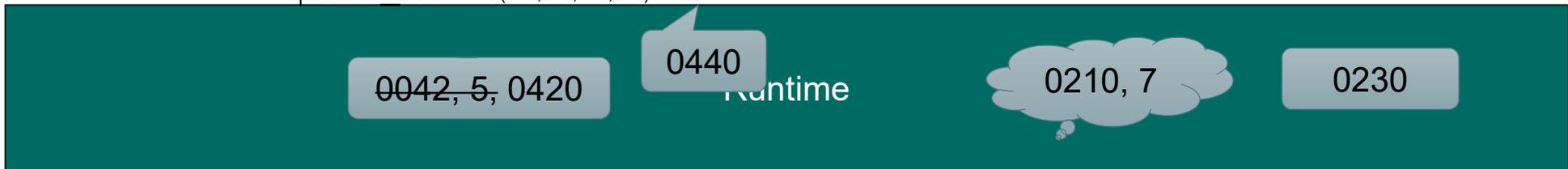
Caller



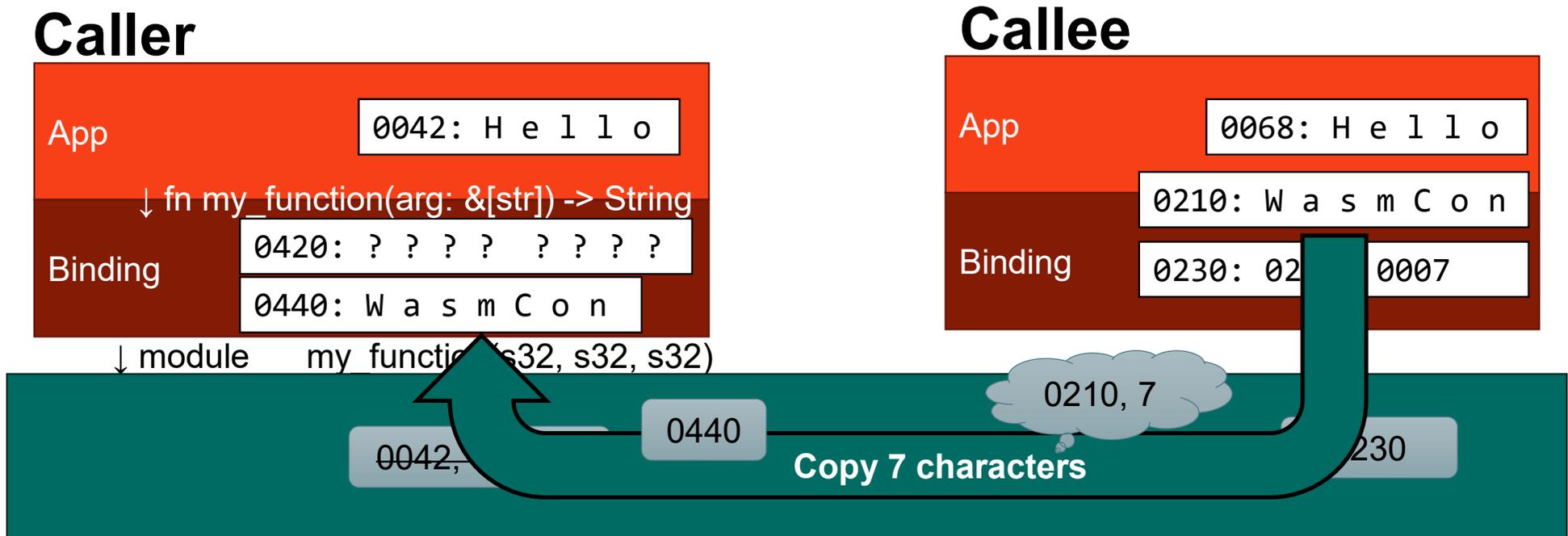
Callee



module ↓ my_function(s32, s32, s32) ↑ cabi realloc(s32,s32,s32,s32) -> s32

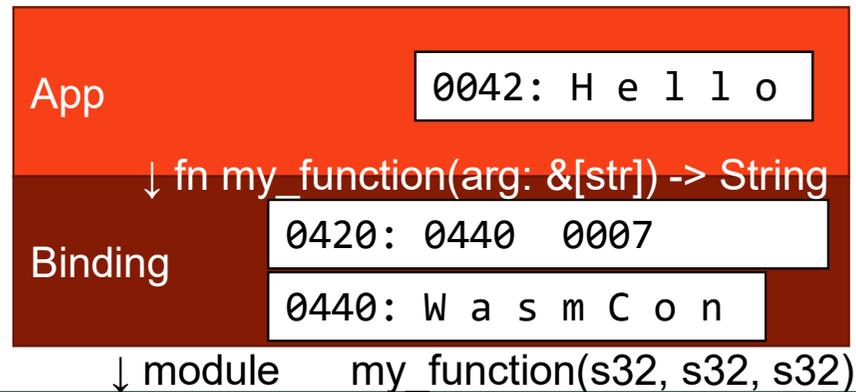


Runtime copies seven characters

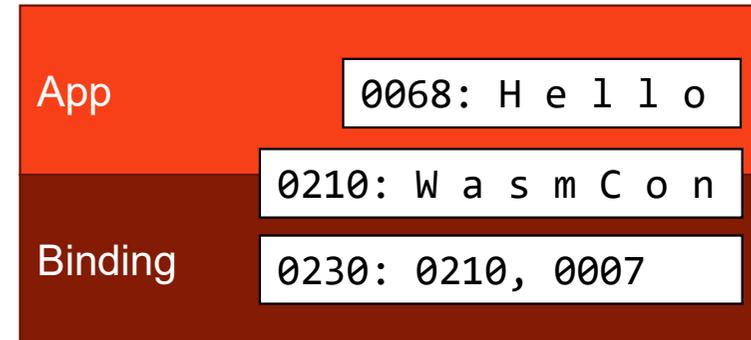


Runtime fills return area in caller

Caller

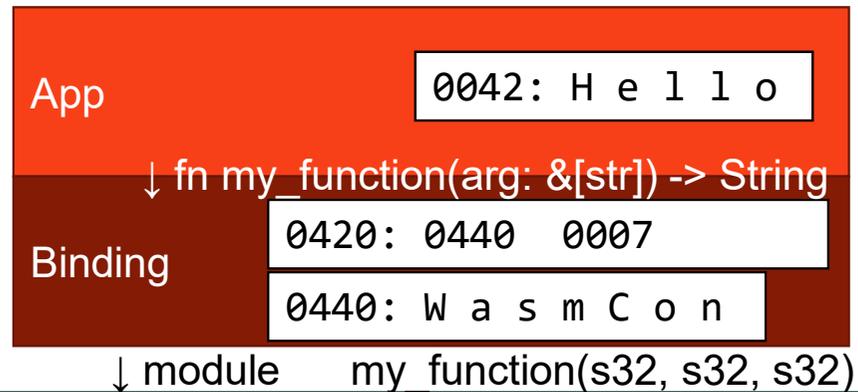


Callee

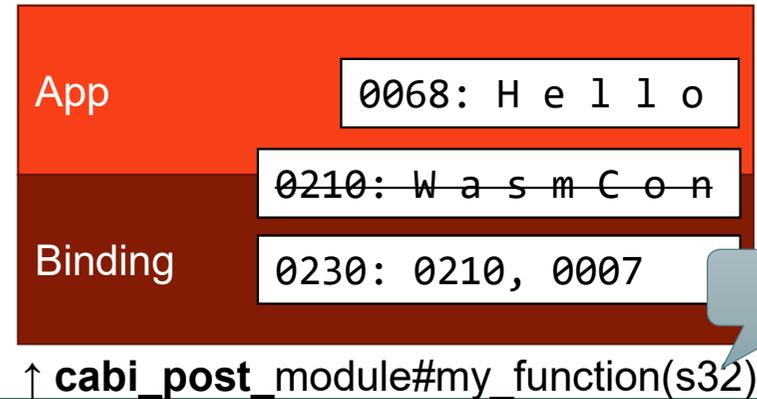


Runtime cleans up

Caller



Callee

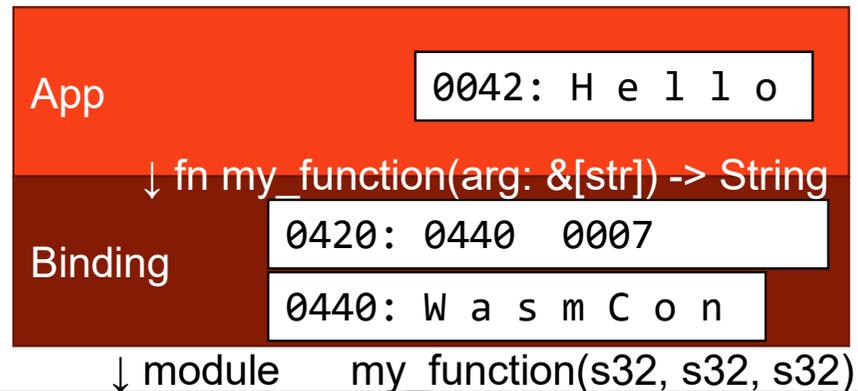


0230

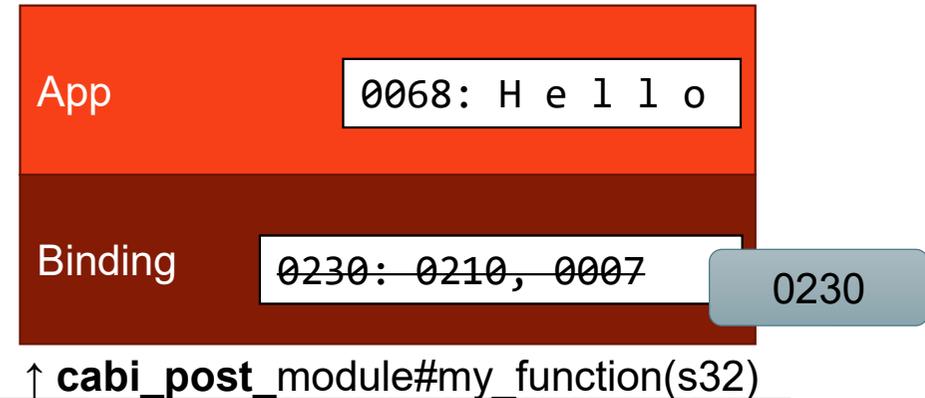


Binding cleans up

Caller



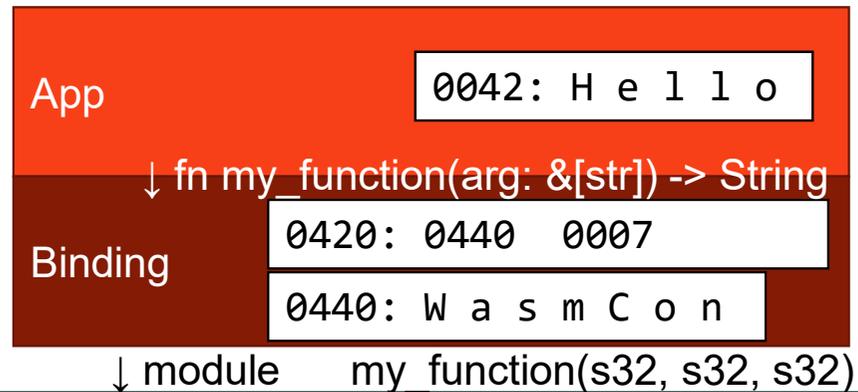
Callee



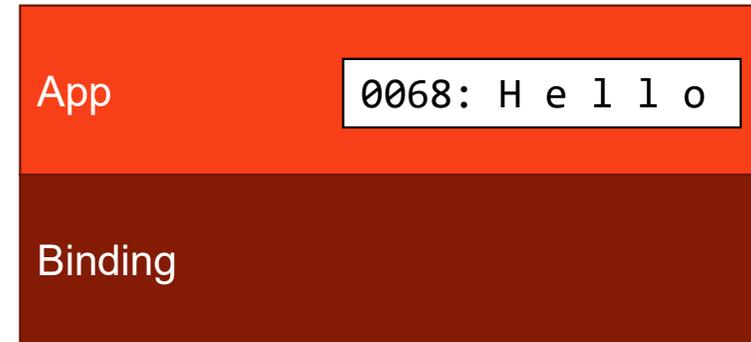
Runtime

Return to caller

Caller



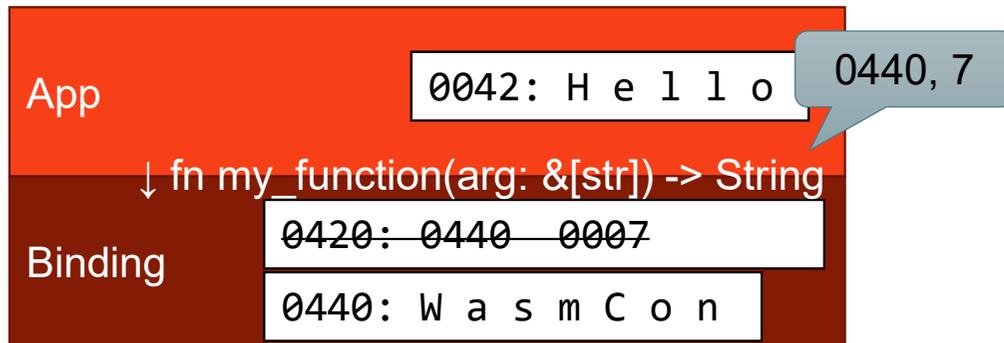
Callee



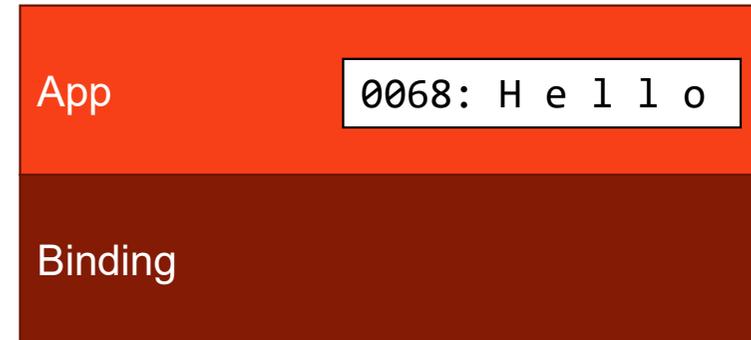
Runtime

Return to app

Caller



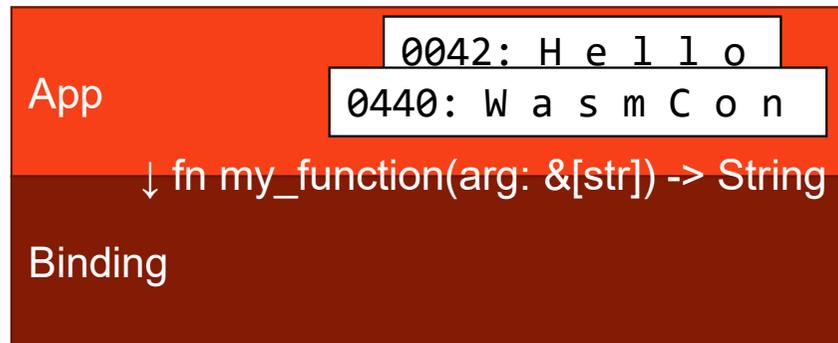
Callee



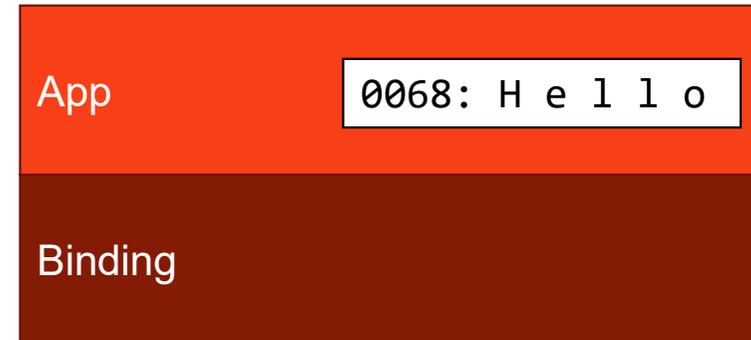
Runtime

Both apps own their strings

Caller



Callee



Runtime

Recap: Calls and symbols

Caller



Callee



Runtime

Native compilation

⚠ ~~speed sandbox~~

WIT for native binaries in SDV

- **Native binaries maximize CPU efficiency**
Unchanged debugging and deployment experience
- **Mixing languages with minimal overhead**
Optimized ABI for shared everything
- **Transition to and from wasm by recompilation**
Transparently move between native and wasm components

Native compilation

Executable

App

↓ fn my_function(arg: &[str]) -> String

Binding

↓ module **my_function**(s32, s32, s32)

↑ cabi_realloc(s32,s32,s32,s32) -> s32

Implementing shared library

App

↑ fn my_function(arg: String) -> String

Binding

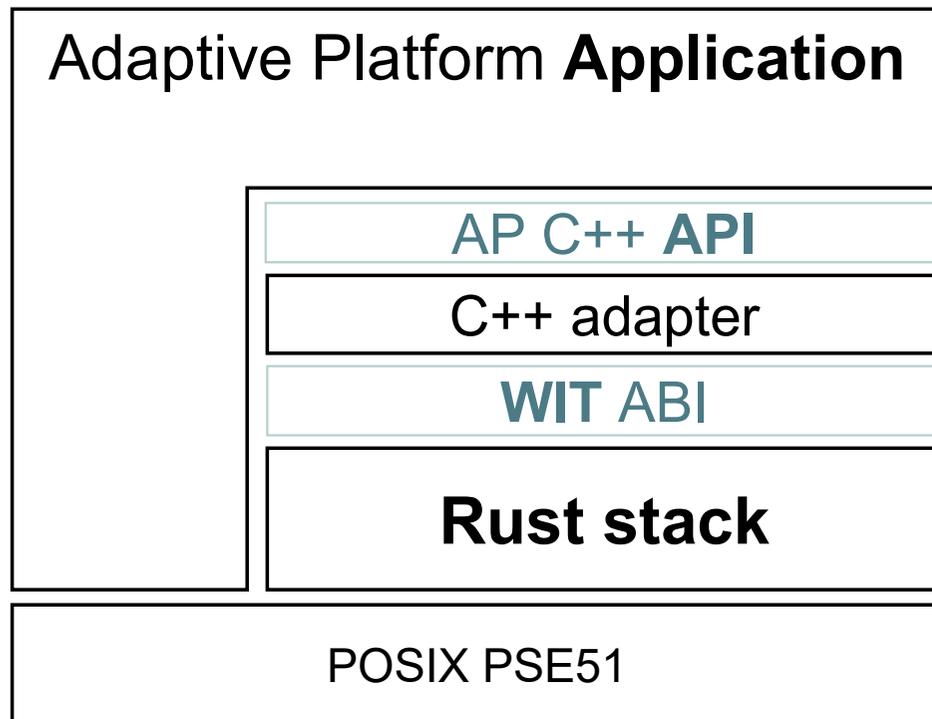
↑ cabi_realloc(s32,s32,s32,s32) -> s32

↑ module#my_function(s32, s32) -> s32

↑ cabi_post_module#my_function(s32)

Connecting shared library (mesh)

The Component Model enables mixing languages



- **Unmodified C++ application**
- **Async methods and events via WASI 0.2**
- **Rust AUTOSAR adaptive stack**
- **Native binary**

Publisher subscriber demo application

```
FACT Info fusion alive
FACT Info not subscribed to brakeEvent yet
FACT Info Callback registered.
FACT Info brakeEvent subscription complete
FACT Info parkingBrakeEvent SetReceiveHandler() complete
FACT Info Callback registered.
FACT Info parkingBrakeEvent subscription
FACT Info Subscribe to Update Rate Field
FACT Info updateRateSubscription()
FACT Info Callback registered.
FACT Info Subscribe to Front Object Dista
FACT Info frontObjectDistanceSubscription
FACT Info Callback registered.
FACT Info Testing Field Getter.
FACT Info FIELDS: Trying to get values for UpdateRate and RearObjectDistance
CTX3 Info FIELDS: Getting the field Update Rate value :87
FACT Verbose ParkingBrakeEvent E2E state:ok (NoData)
FACT Verbose Callback: ParkingBrakeEvent - Radar: NOT active Object:255
      E2E checkStatus:ok (NotAvailable)
CTX3 Info FIELDS: Getting the Rear Object Distance Distance :-1
FACT Verbose FIELDS: Current Rear Object Distance is 65535
FACT Verbose FIELDS: Current Update Rate is 87
FACT Info Testing Field Setter.
FACT Info FIELDS: Trying to set UpdateRate and ObjectDetectionLimit
FACT Info FIELDS: UpdateRate setting skipped
FACT Info FIELDS: ObjectDetectionLimit setting skipped
FACT Info Testing Synchronous Method Call.
CTX3 Info radar active
CTX3 Info brakeEvent sent
CTX3 Info parkingBrakeEvent sent
```

Application
Output

libwasi.so: WASI clocks, poll



libstack.so: ara::core exec log

libradar.so: Publisher



libfusion.so: Subscriber

target/.../main: Calls 2xrun



libradar.so symbols

D *UND* **autosar**:ara/log|[**constructor**]logger

D *UND* autosar:ara/log|[method]logger.**report**

D *UND* autosar:ara/log|[resource-**drop**]logger

D *UND* **wasi:clocks**/monotonic-clock@0.2.0|**now**

g DF .text radar-wasi:cli/run@0.2.0#**run**

g DF .text autosar:ara/radar-service|[**method**]instance.**adjust**

g DF .text autosar:ara/radar-service|[method]**future**-result-adjust-output-error-code.**subscribe**

g DF .text autosar:ara/radar-service|[static]future-result-adjust-output-error-code.**get-value**

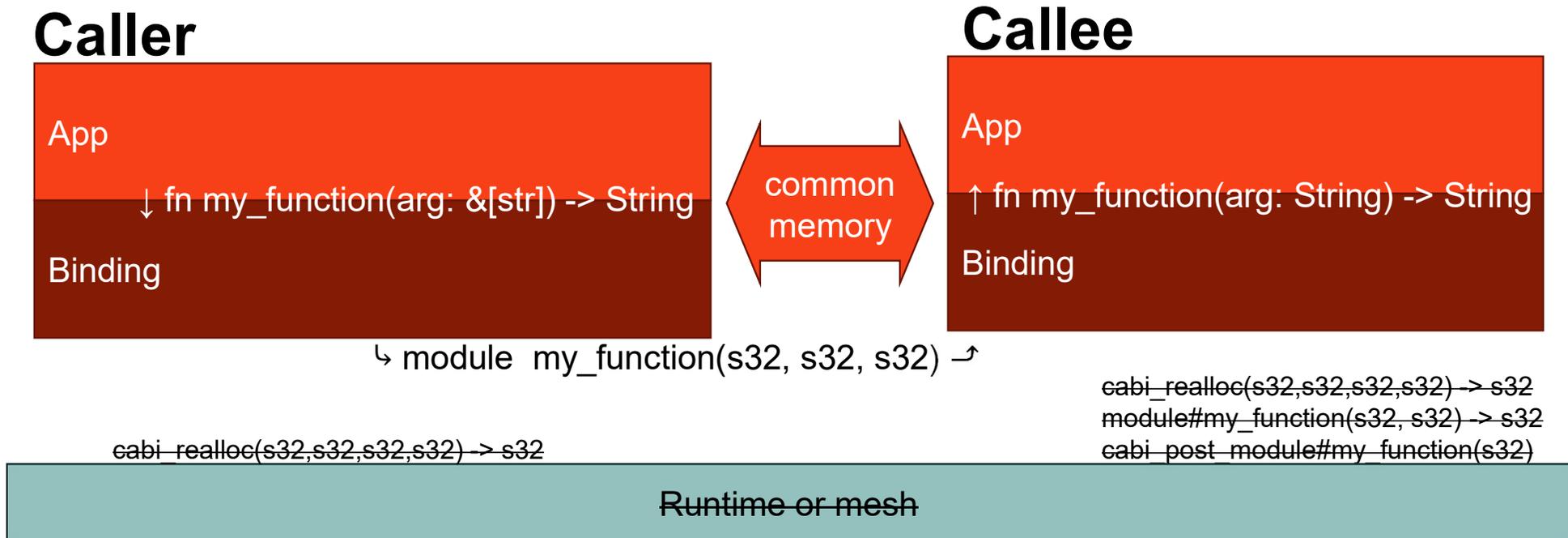
Symbol mangling for native environment

```
autosar:ara/radar-service|[static]  
future-result-adjust-output-error-code  
.get-value
```

```
autosarX3AaraX2Fradar_serviceX00X5BstaticX5D  
future_result_adjust_output_error_code  
X2Eget_value
```

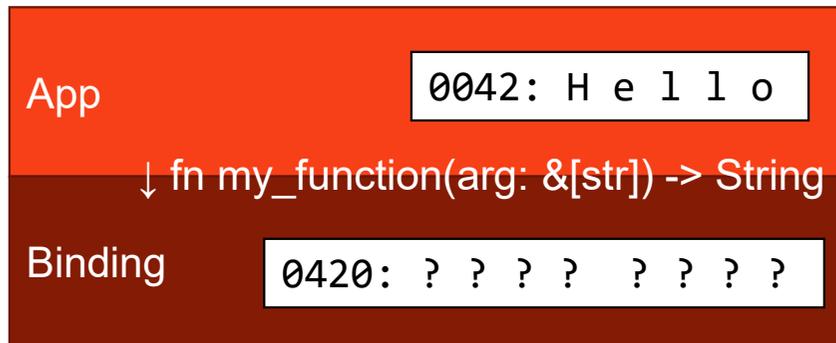
Symmetric ABI

Symmetric Idea: Can we remove the runtime?

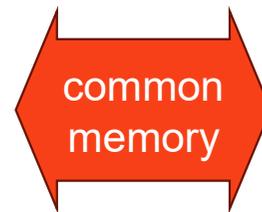
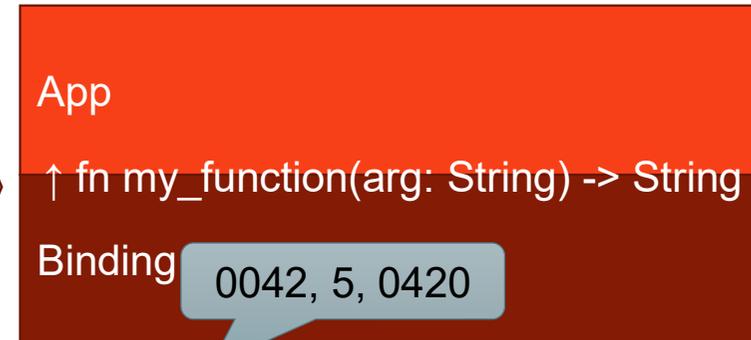


Symmetric: Caller binding directly calls callee

Caller



Callee

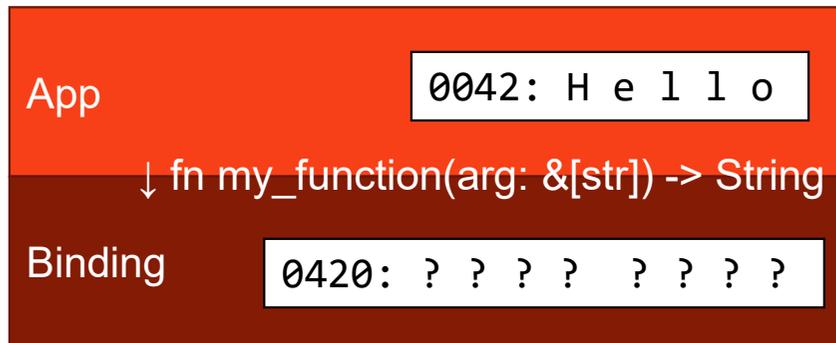


↳ `moduleX00my_function(s32, s32, s32)` ↗

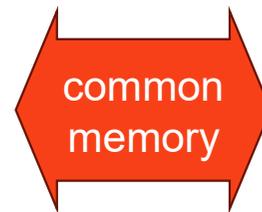
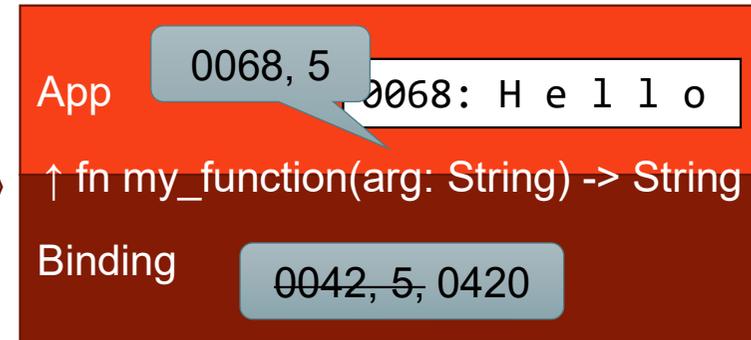
Direct call

Symmetric: Binding allocates, copies, and calls

Caller



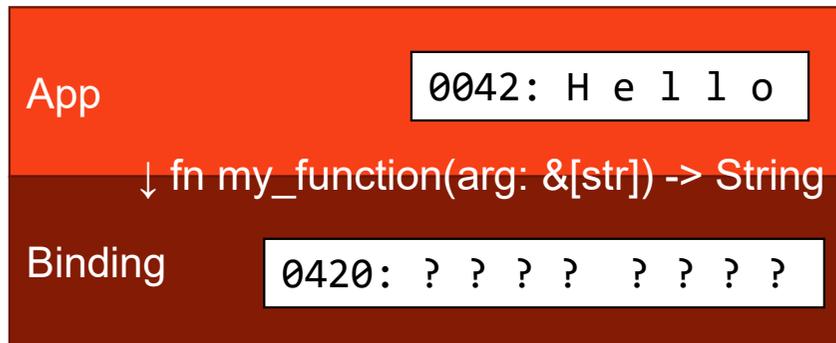
Callee



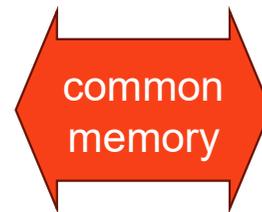
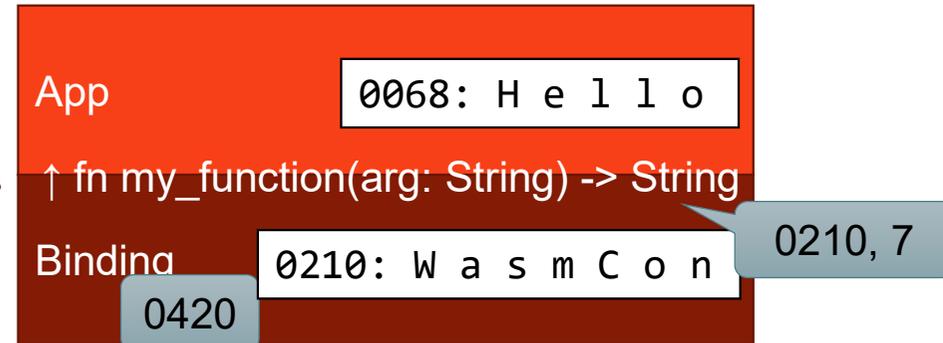
↳ moduleX00my_function(s32, s32, s32) ↗

Symmetric: App returns string

Caller



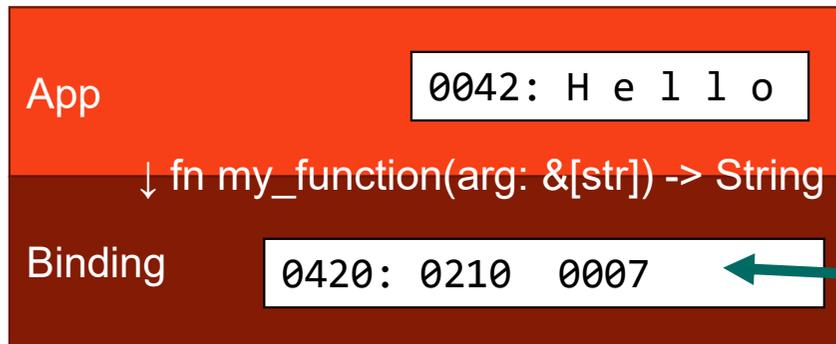
Callee



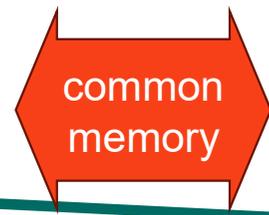
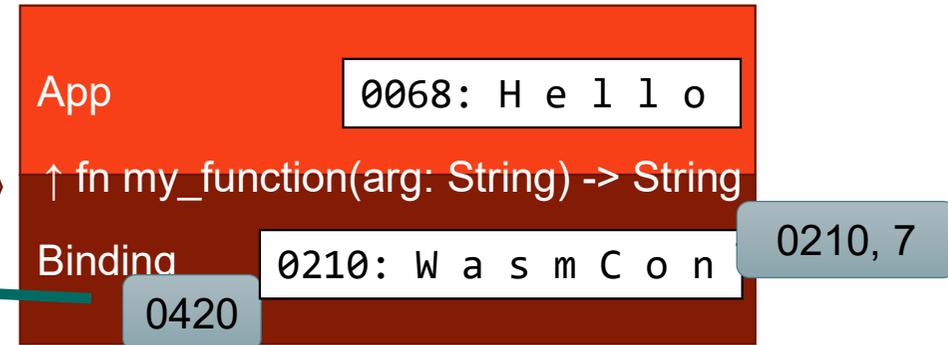
↳ `moduleX00my_function(s32, s32, s32)` ↗

Symmetric: Binding fills return area

Caller



Callee



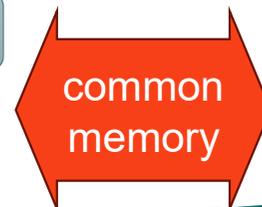
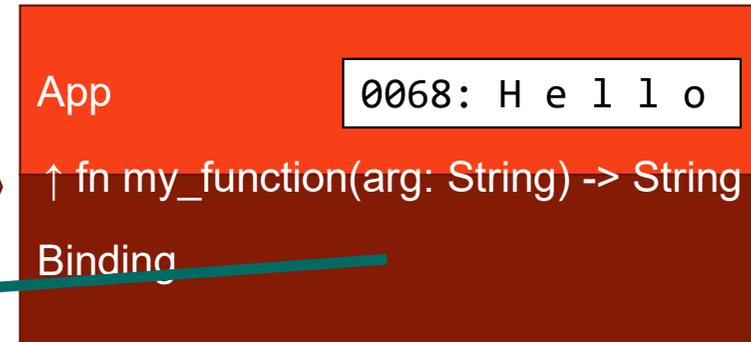
↳ moduleX00my_function(s32, s32, s32) ↗

Symmetric: Binding returns re-owned string

Caller



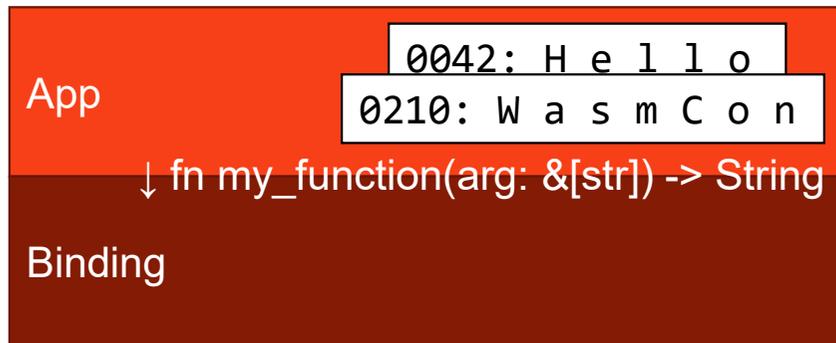
Callee



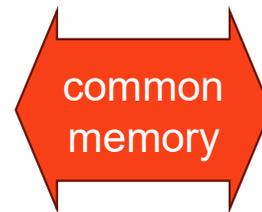
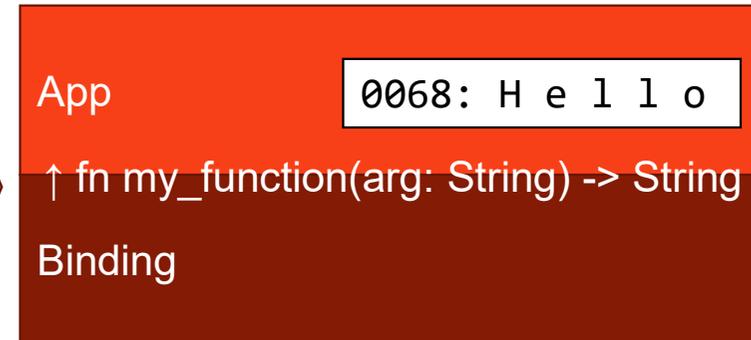
↳ moduleX00my_function(s32, s32, s32) ↗

Symmetric: Neither app sees a difference

Caller



Callee

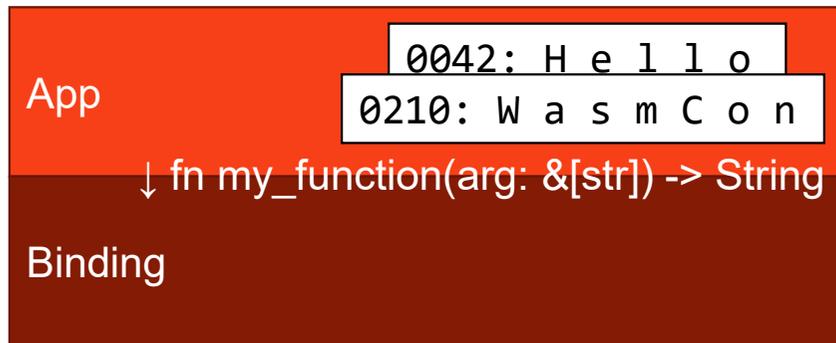


↳ `moduleX00my_function(s32, s32, s32)` ↗

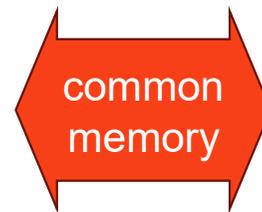
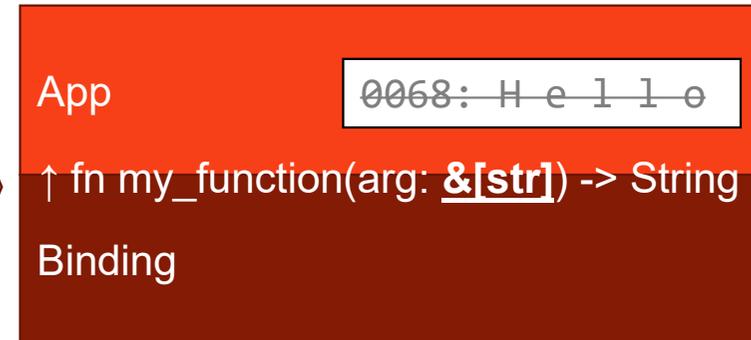
Unchanged
import ABI!

Symmetric API: Simpler and no copy

Caller



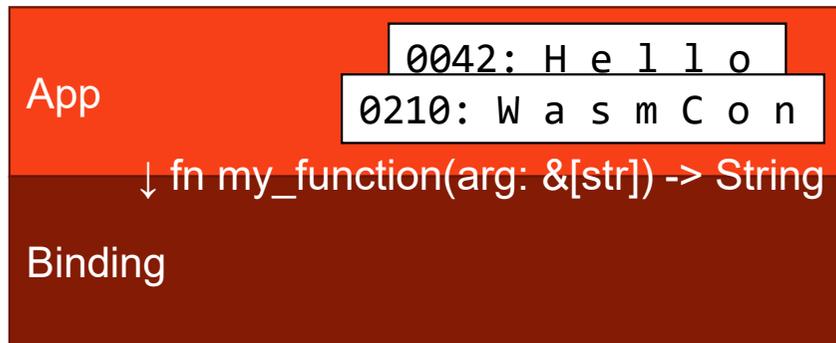
Callee



↳ moduleX00my_function(s32, s32, s32) ↗

Full insulation is a runtime option

Caller.exe



↓ moduleX00my_function(s32, s32, s32)
↑ cabi_realloc() + copy result

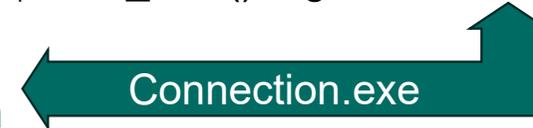


Some form of communication

Callee.so

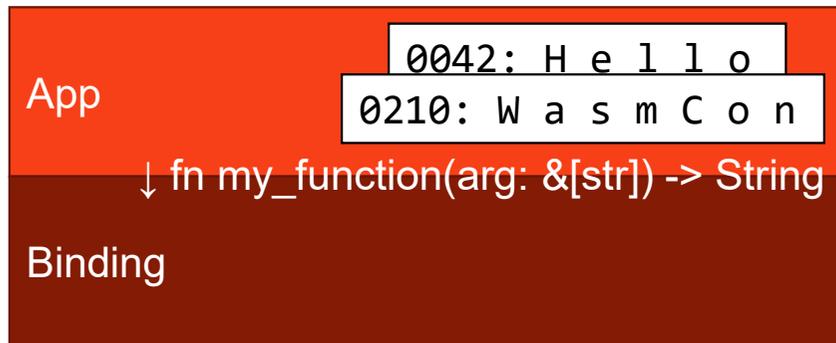


↑ cabi_realloc() + copy argument
↑ moduleX00my_function(s32, s32, s32)
copy result
↑ **cabi_free()** argument and result

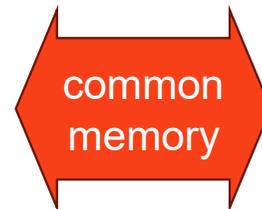
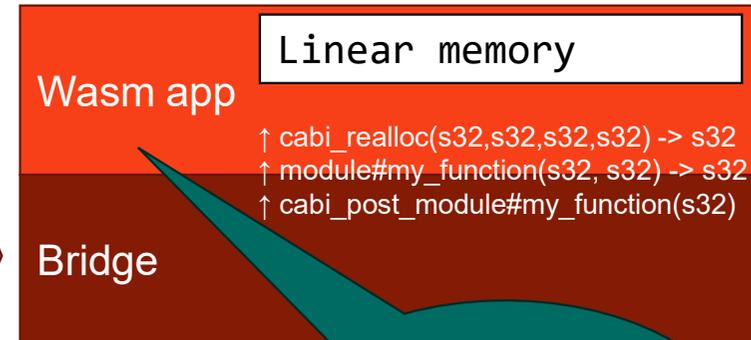


Symmetric: Wasm remains a transparent option

Caller



Callee



↳ moduleX00my_function(s32, s32, s32) →

Interpreter
Just In Time
Ahead of Time
wasm2c

Outlook

Further use for WIT without wasm

- **Use WIT for zero-overhead plugins e.g. in wasmtime or embedded**
- **A stable binary interface between Rust modules**
- **No sandbox, plugins can use all of POSIX (both good and bad)**

Additional topics

- **Multithreading & Exceptions: C++**
- **Symmetric resource ABI**
- **Async calls: WASI 0.3**
- **Functional safety: Zero-allocation**
- **Zero-copy sandbox**

Where to get more information

Fork of wasm-tools and wit-bindgen at

<https://github.com/cpetig/>

Issue for Symmetric ABI

<https://github.com/WebAssembly/component-model/issues/386>

Flat data types <https://github.com/cpetig/flat-types-rust>

TL;DR

- **Symmetric ABI optimizes the component model for shared everything**
- **Mixing sandboxed and shared everything leverages their complimentary strengths**

Thank you!

Especially to all creators of WASI!

Questions?



Backlog

C++ challenges WASI 0.2

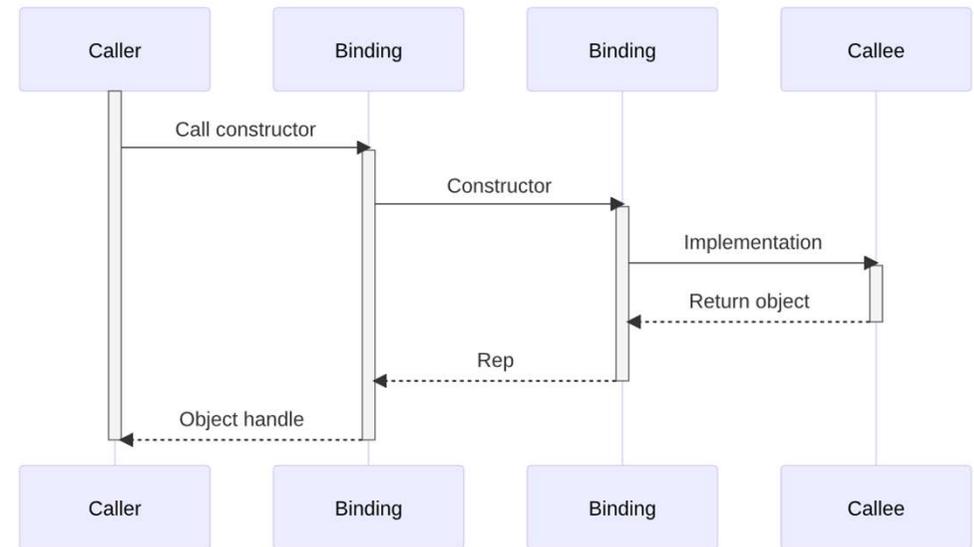
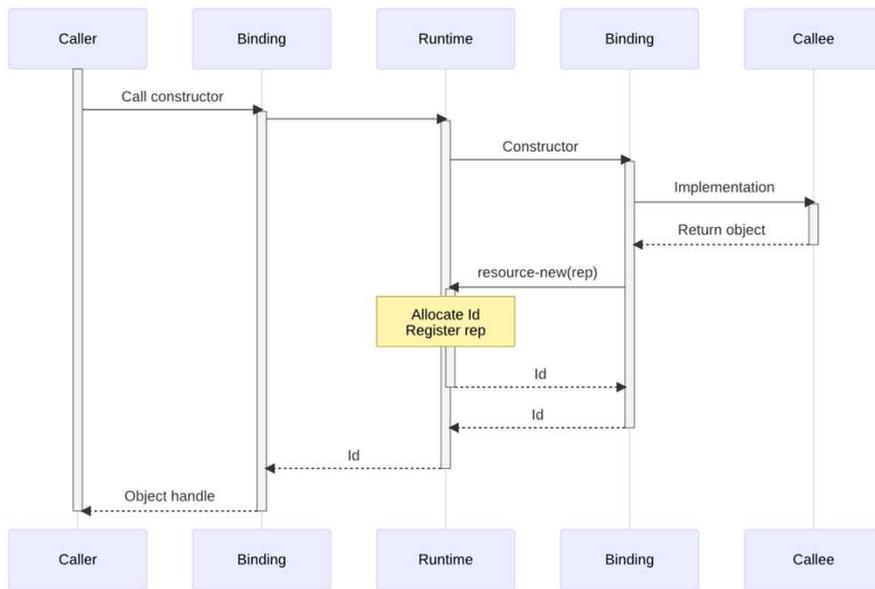
- **Async (promise & future) needs multi-threading**
... or concurrency TS's `.then()` or C++-20's `co_await`
- **Throw & catch is common needs exceptions**

Symmetric: Resources and async

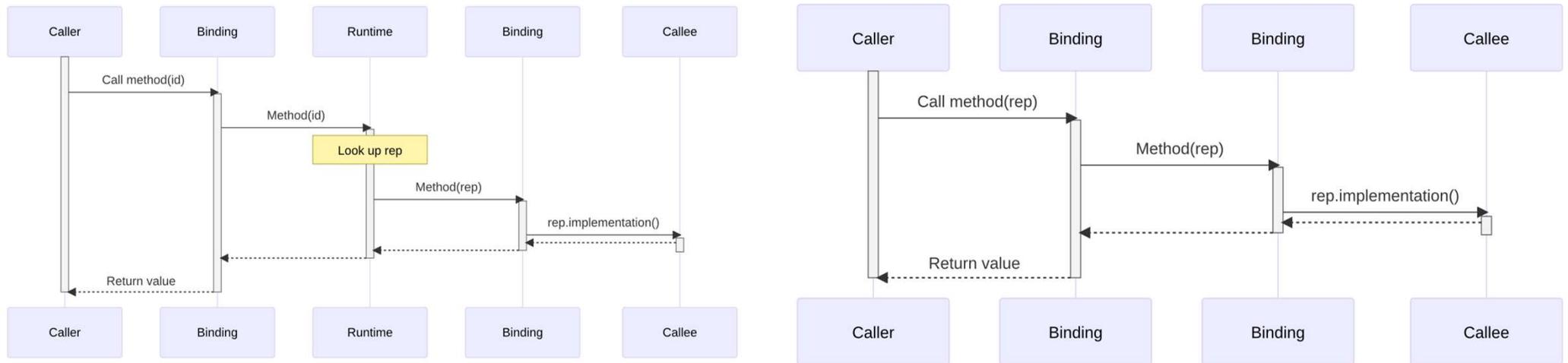
- **Resources: Use import ABI**
- Much simpler ABI, no resource-new/drop/rep
- A pointer sized resource ID obviates callee side resource-tables

- **Async: Use import ABI**
- Needs shared event & executor
- Event awaiting callee registers callback with executor

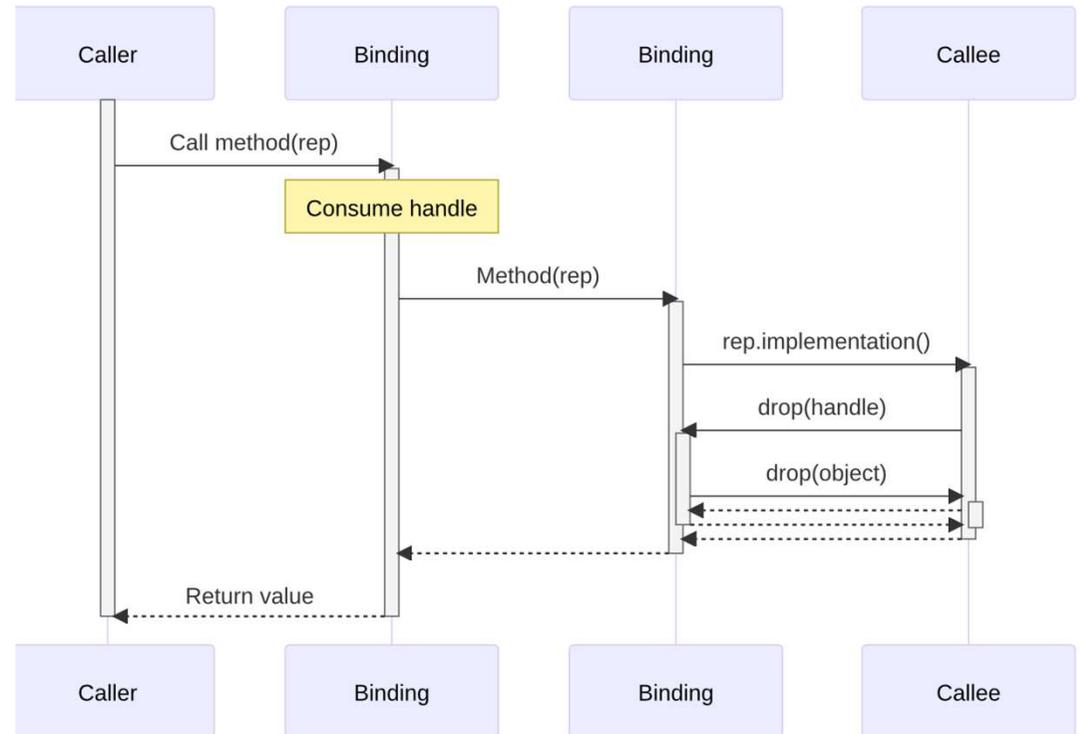
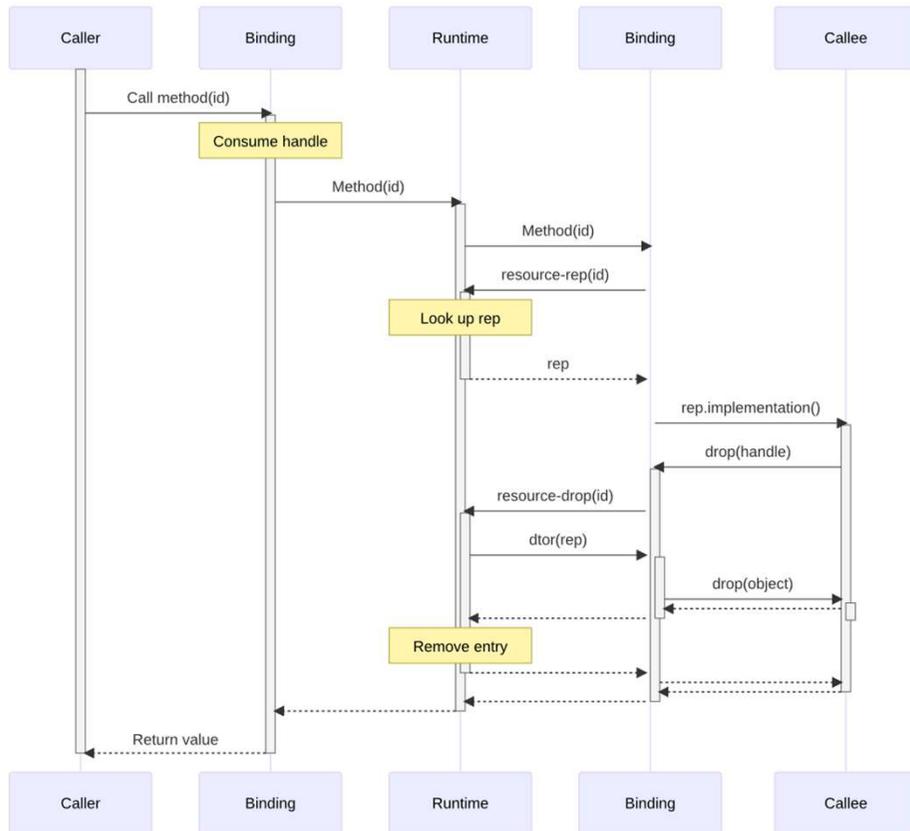
Symmetric Resources: Create



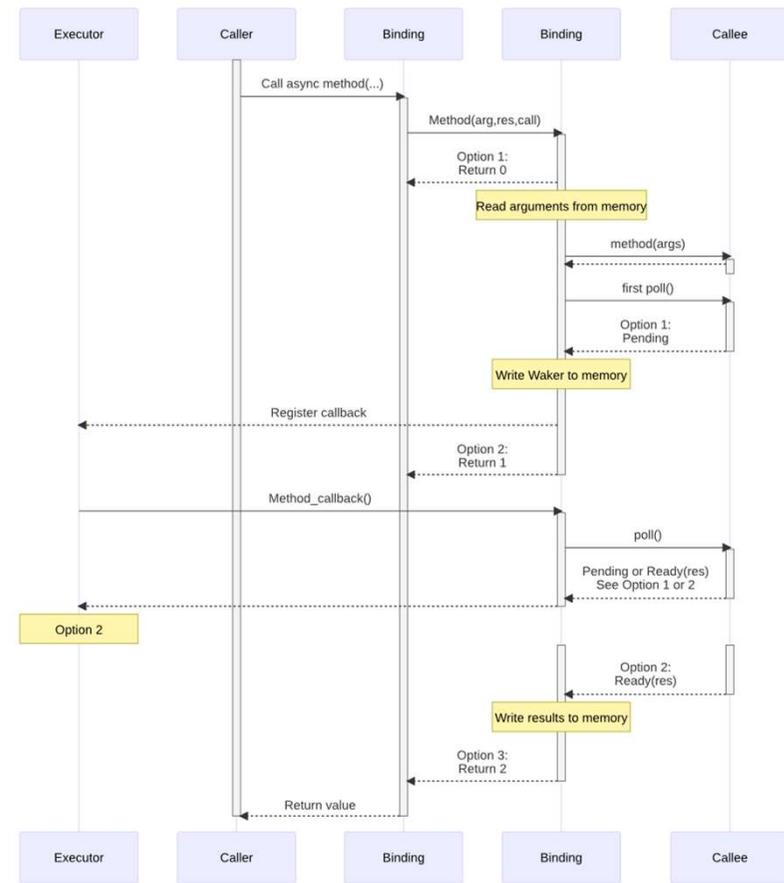
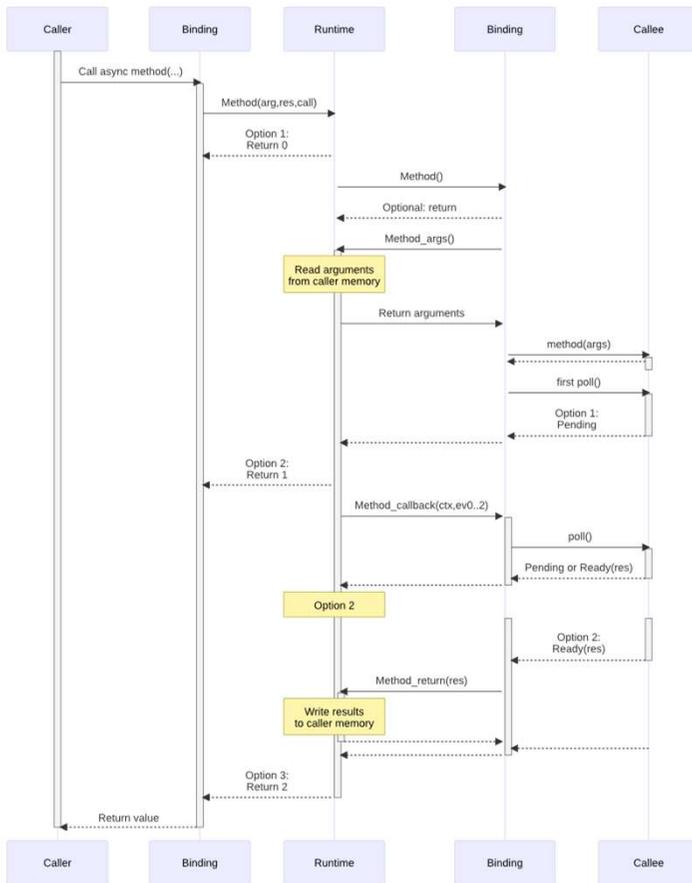
Symmetric Resources: Method



Symmetric Resources: Owned



Symmetric Async



Functional safety

- **Sandboxing is freedom from interference**
- **Constant time allocations**
- **Deterministic allocator or no allocations at run-time**
- **Qualification of bindings code**

Zero-copy between sandboxes

- **Needed for AI and image processing**
- **Shared memory**
- **One writer exclusive or many readers per element**
- **Flat data structures: Position independent**
- **Multi-memory or memory mapping**
- **More versatile if caller provides buffer**

Zero-copy inter-process via shared memory

```
record data {  
    id: u32,  
    name: list<list<string>>,  
}
```

Layout:

id₃₂ a₈ l₈ | a₈ l₈ a₈ l₈ | a₈ l₈ a₈ l₈ | "string" "data" | a₈ l₈ | "test"

Hex

0000 78 56 34 12, 02, 02 | 04, 02, 10 01, | 04, 06, 08 04 | 73 74 ...
0010 ... 72 69 6e 67 | 64 61 74 61 | 02 04 | 74 65 73 74

