

Embedded Ecosystem Requirements

Chris Woods

| “Thinking beyond an API”

The Embedded Ecosystem Requires:

- 1. Continued WASI-Preview 1 Support**
- 2. Validation of the Component Model for Embedded Systems**
- 3. Embedded Innovation**

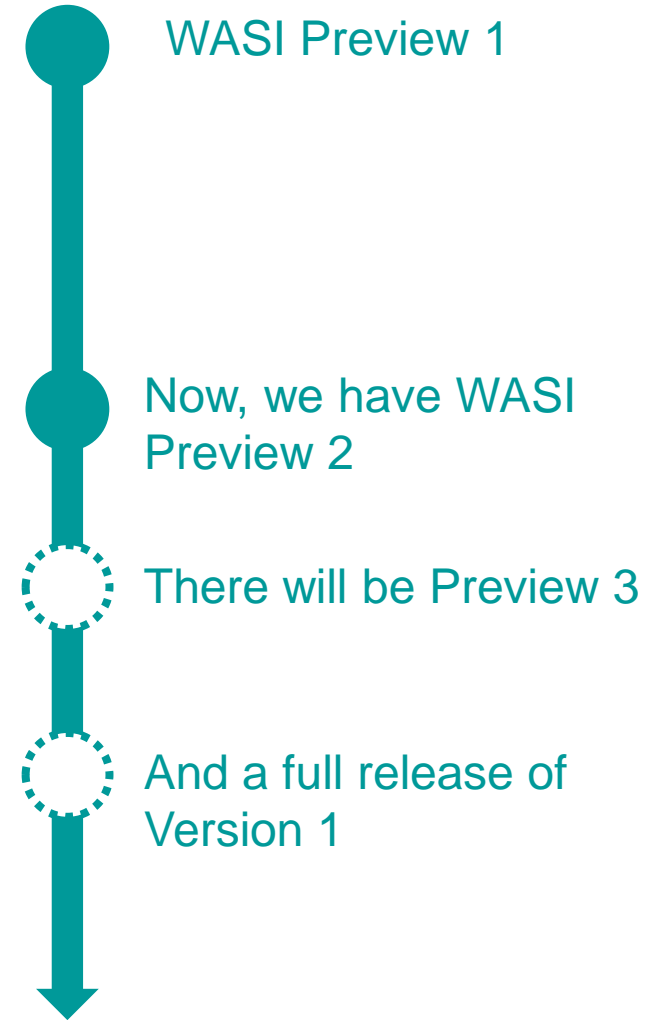
Millions of WASI preview-1 devices & Large ecosystem has emerged

There is a large existing embedded ecosystem which has emerged based on WASI Preview 1.

This includes **millions of individual devices** running the WAMR runtime.


These devices have been produced by companies like Amazon, Xiaomi and Sony / Midokura.

The wider ecosystem includes additional companies and organization looking to adopt WASM and WASI this includes companies like Siemens, Bosch, and Aptiv.





Scale...

The average full-scale data center is 100,000 square feet in size and runs around 100,000 servers, which are essentially powerful computers. Feb 1, 2023


 Institutional Real Estate, Inc.
<https://irei.com> > Articles

Cities and regions with the highest concentration of data centers

 About featured snippets •  Feedback

People also ask

How many servers does an AWS datacenter have?
AWS Data Centers Today: 100+ Locations, 1.5 Million Servers, and More. A single Amazon Web Services data center can host as many as 50,000 servers. Aug 9, 2023

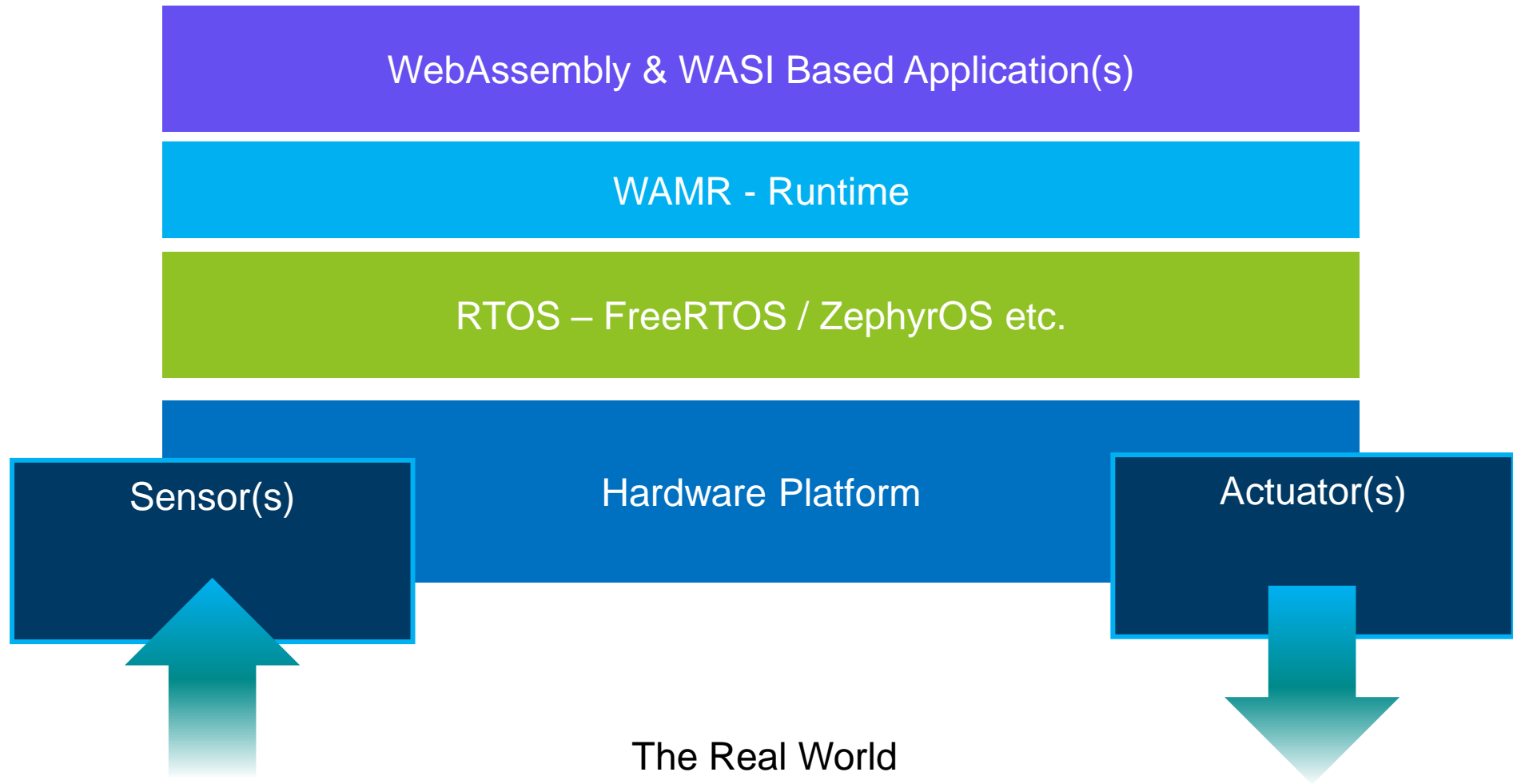
 cloudzero.com
<https://www.cloudzero.com> > blog > aws-data-center-loc...

AWS Data Centers Today: 100+ Locations, 1.5 Million Servers, and

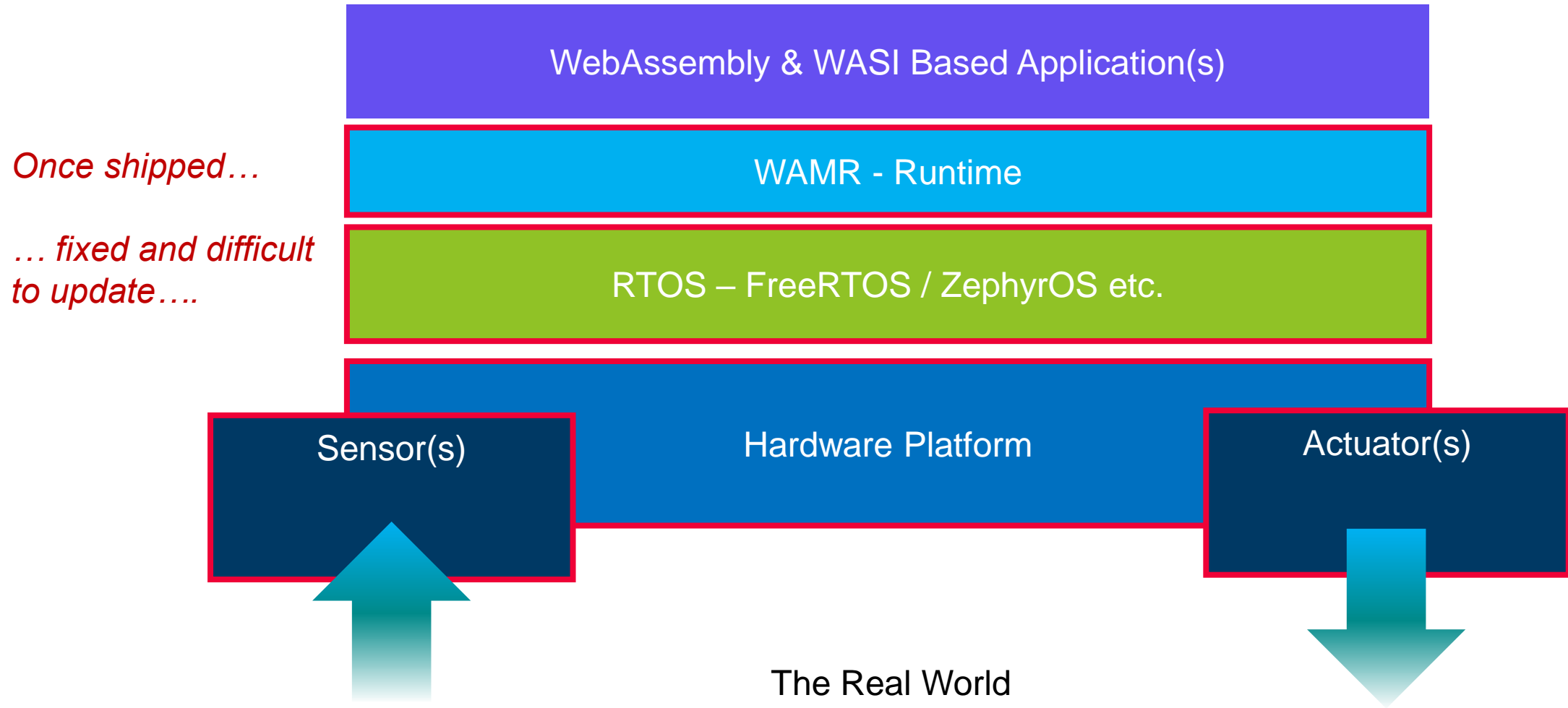
...

Google search for “How many servers does an AWS datacenter have”

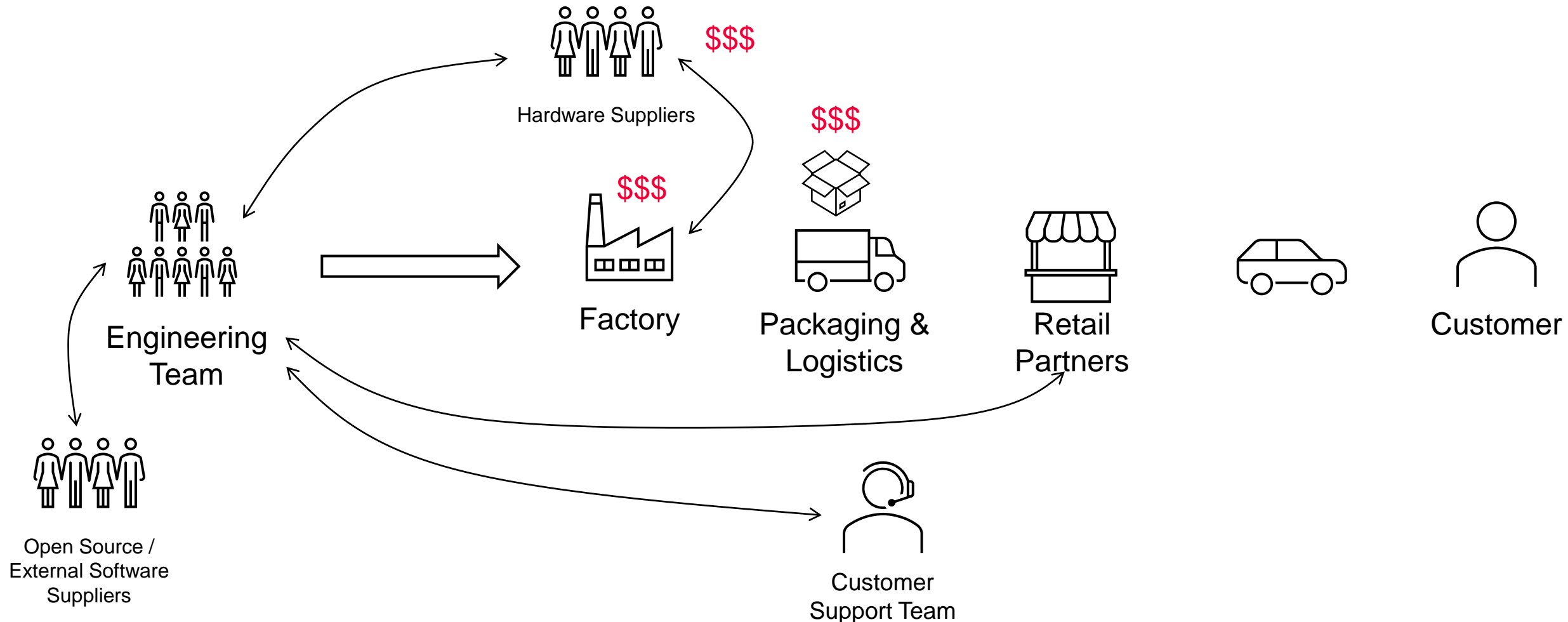
Typical Embedded System with WASI



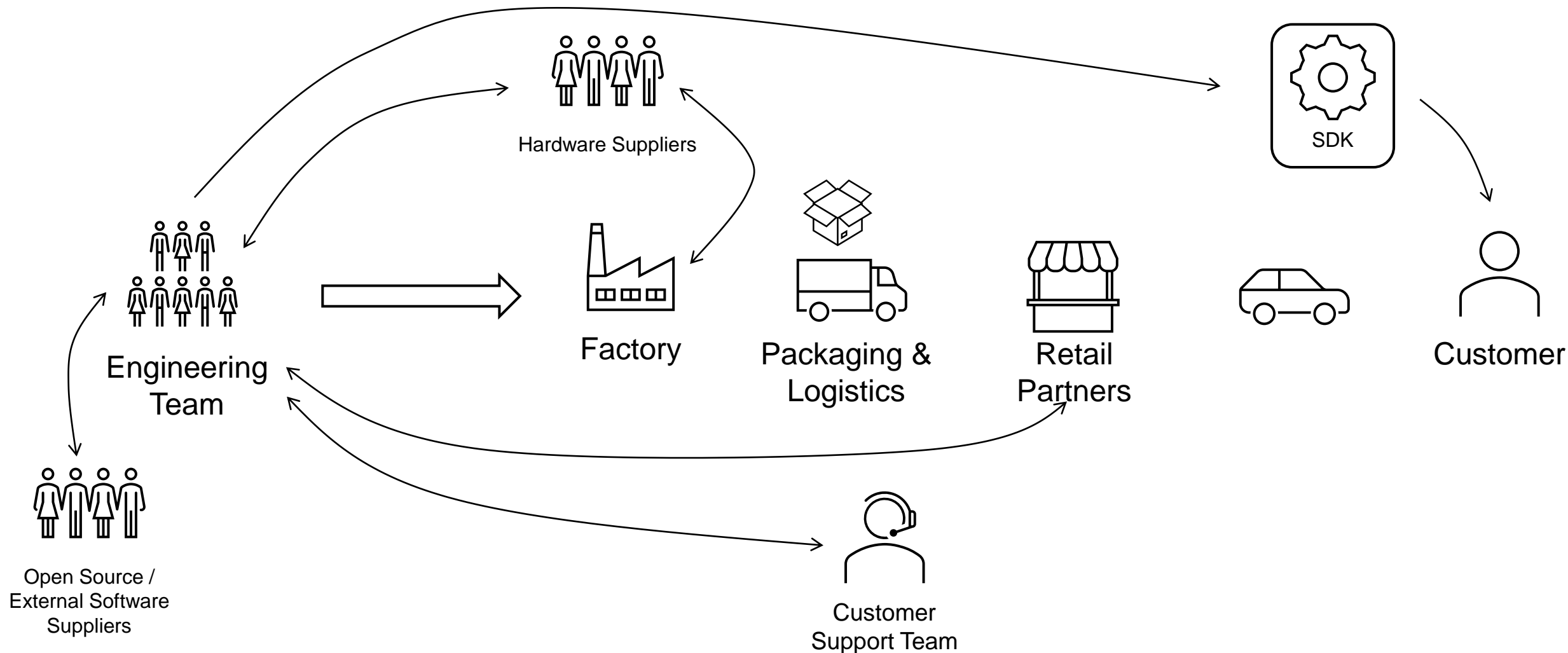
Typical Embedded System with WASI



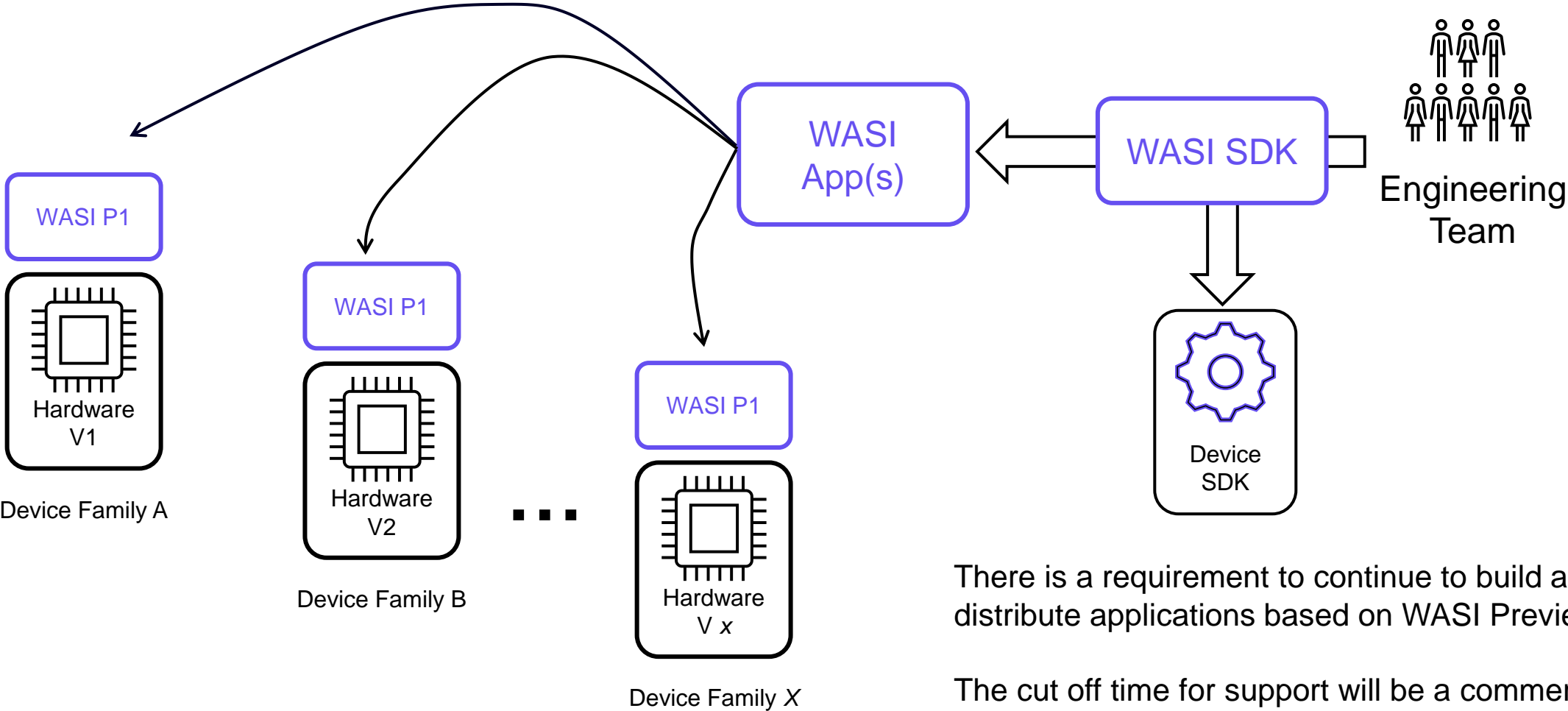
Physical Distribution is Hard; Rate of Change Limited and Expensive



Physical Distribution is Hard; Rate of Change Limited and Expensive



The Result : Legacy Support for P1 is needed



There is a requirement to continue to build and distribute applications based on WASI Preview 1

The cut off time for support will be a commercial decision based on each embedded company's internal requirements.

Hardware formfactor & WASI/WASM on embedded devices competes with Docker / containers

Expensive BoM

↑

Cheap BoM

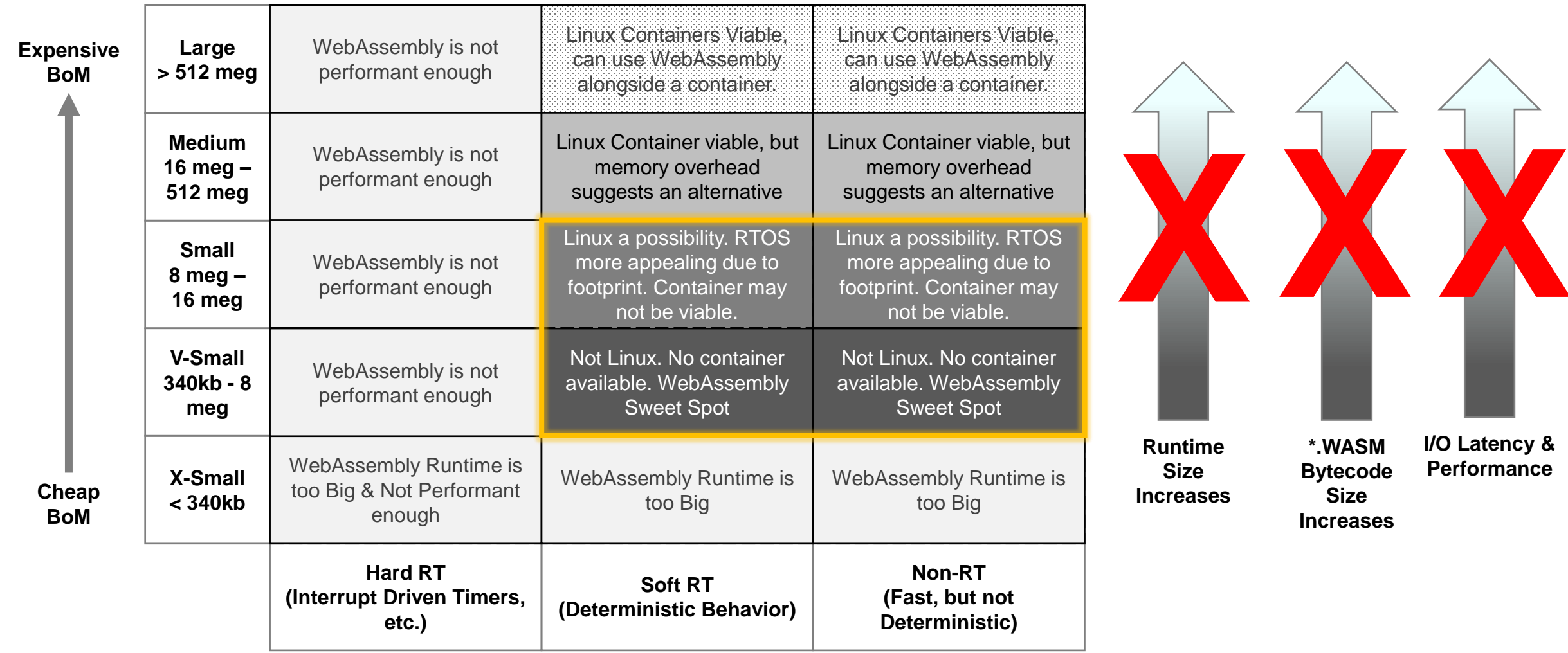
Large > 512 meg	WebAssembly is not performant enough	Linux Containers Viable, can use WebAssembly alongside a container.	Linux Containers Viable, can use WebAssembly alongside a container.
Medium 16 meg – 512 meg	WebAssembly is not performant enough	Linux Container viable, but memory overhead suggests an alternative	Linux Container viable, but memory overhead suggests an alternative
Small 8 meg – 16 meg	WebAssembly is not performant enough	Linux a possibility. RTOS more appealing due to footprint. Container may not be viable.	Linux a possibility. RTOS more appealing due to footprint. Container may not be viable.
V-Small 340kb - 8 meg	WebAssembly is not performant enough	Not Linux. No container available. WebAssembly Sweet Spot	Not Linux. No container available. WebAssembly Sweet Spot
X-Small < 340kb	WebAssembly Runtime is too Big & Not Performant enough	WebAssembly Runtime is too Big	WebAssembly Runtime is too Big
	Hard RT (Interrupt Driven Timers, etc.)	Soft RT (Deterministic Behavior)	Non-RT (Fast, but not Deterministic)

NB: In an IoT device which will sell 10,000’s – millions of units the hardware Bill of Materials (BoM) is more important than the Software Development costs.

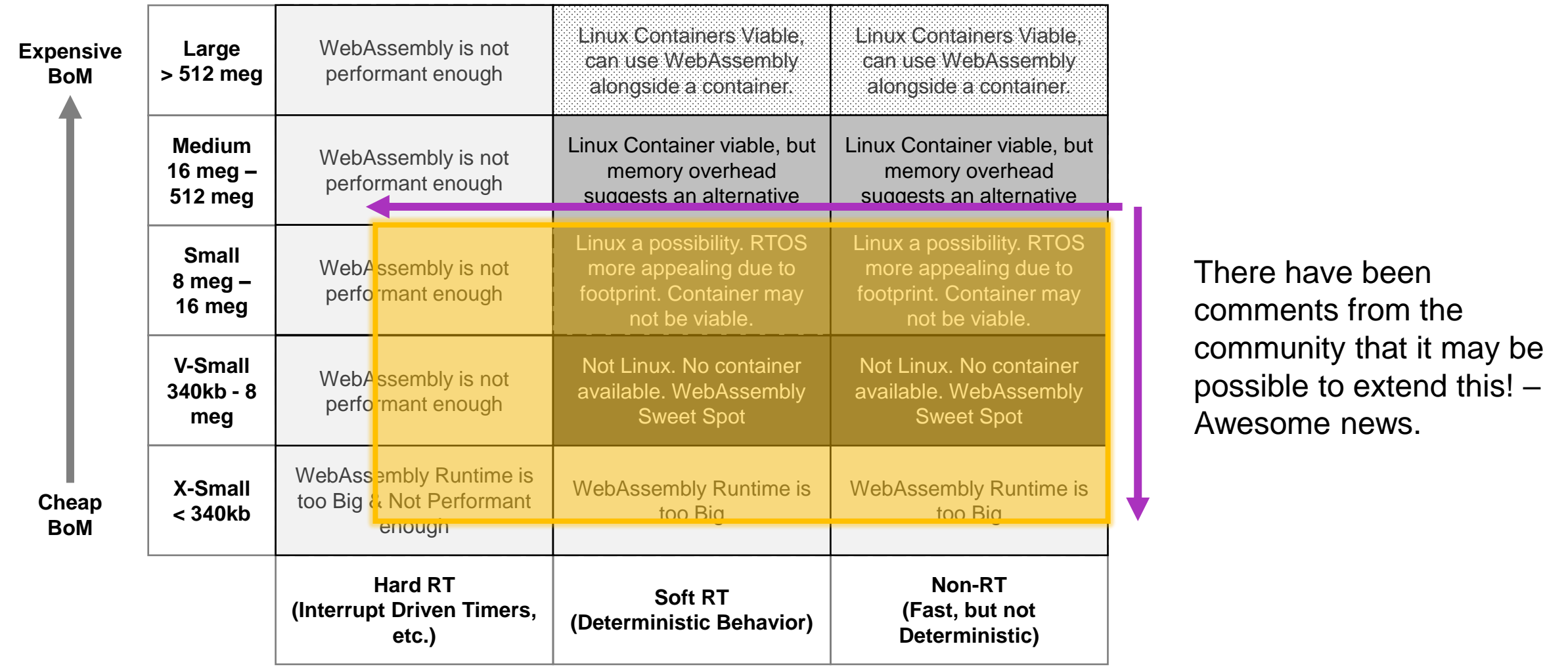
Software costs can be offset over number of units sold.

Therefore, increased development difficulty and on occasion delay are accepted to provide more competitive products in the market.

Hardware formfactor & WASI/WASM on embedded devices competes with Docker / containers



Hardware formfactor & WASI/WASM on embedded devices competes with Docker / containers



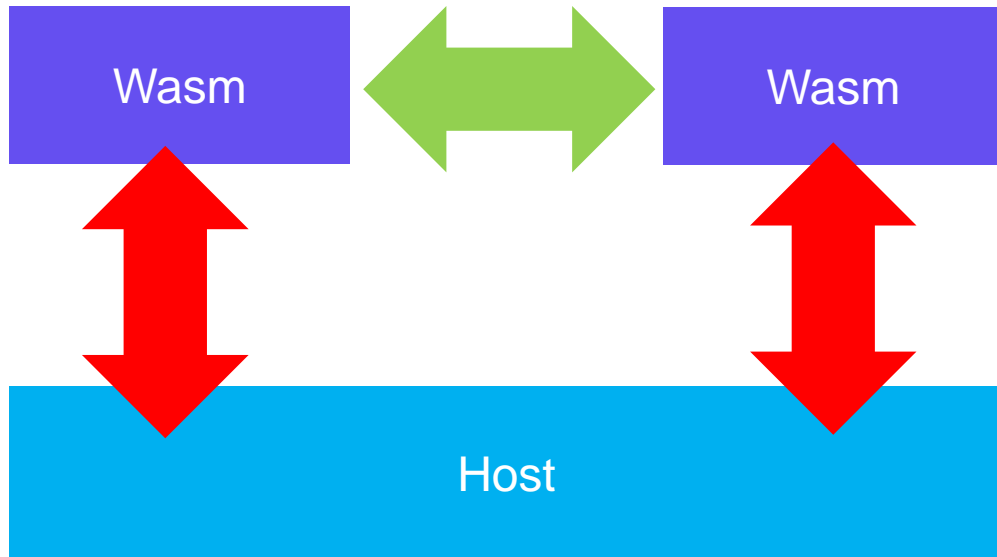
Typical Embedded System with WASI : IO Performance is Critical



Key IO Requirements

- “clamp to clamp time” (aka response latency)
- Data throughput
- Reliability

Based on published information / Validation of Component Model for Embedded Use cases



Concerns on Implementation

Anything that impacts I/O throughput needs to be addressed:

- “...*this requires the Canonical ABI to call realloc ...*”
- Unnecessary memory copies
- Unnecessary data transformations

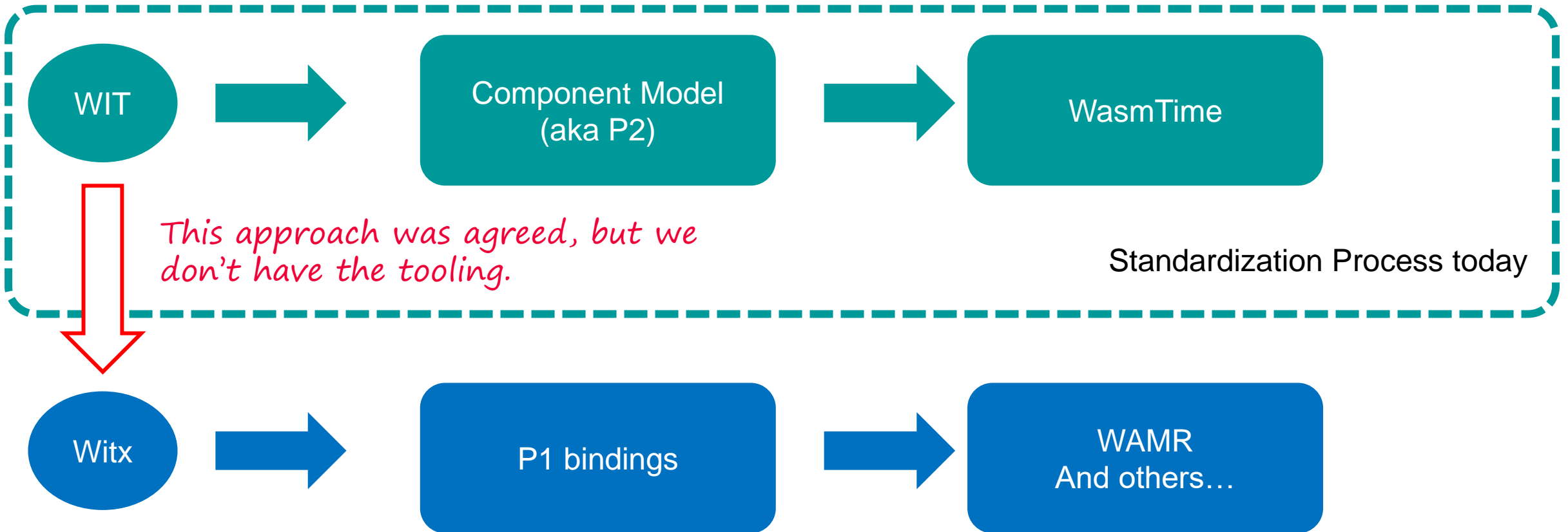
Solutions

Initial research shows it should be possible to engineer around these issues, but this work has not been completed yet.

Needs to be validated, particularly if people's lives are on the line.

Micro-benchmarks help to do this.

Transition mechanics from P1 to P2; Source Compatibility then ABI Compatibility



Source Compatibility Reduces Fragmentation and Aids Adoption

Preview 1 Runtime Specific
Networking interfaces:

- Lunatic



- WasmEdge



- Wamr

Preview 2 Networking Interface

- Socket Definition
- HTTP

Source Compatibility Reduces Fragmentation and Aids Adoption

Preview 1 Runtime Specific Networking interfaces:

- Lunatic



- WasmEdge



- Wamr

Backport

Preview 2 Networking Interface

- Socket Definition
- HTTP

When to End of Life P1?

Not an easy question to answer:

- Commercial motivations by different ecosystem partners
- Feature parity is important (threads, sockets, etc.) + Also around tool chain and supporting development tooling, debugging, compile time, etc.
- Validation of P2 technology - NFRs around runtime performance and implementations of P2

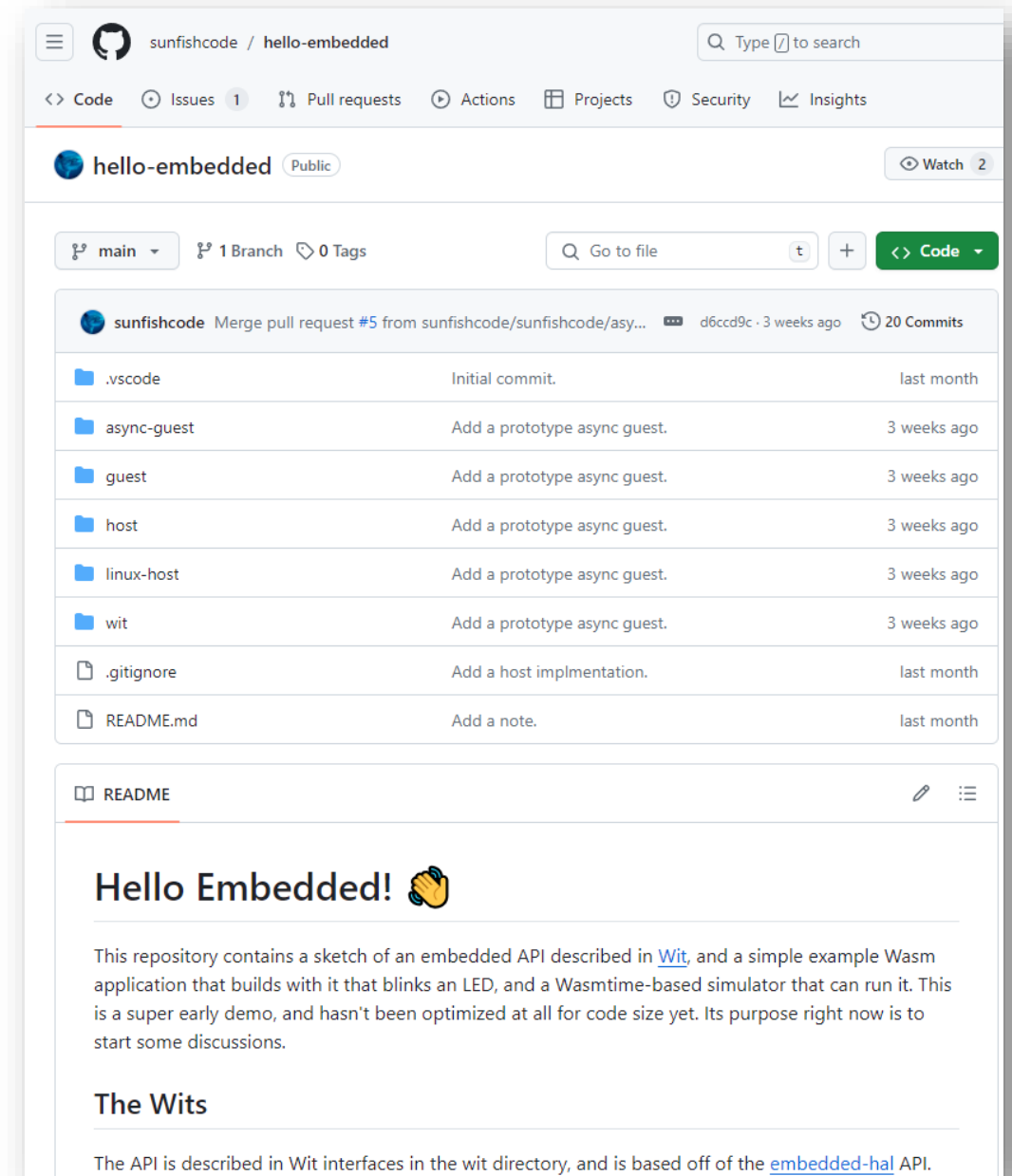
Open Question : How do we share the cost of continued P1 support?

Source Compatibility Aids Innovation

Innovation like the embedded API work is awesome.

But due to the physical limitations of the embedded world it is hard to keep up with the standard process.

We need some way to technical contribute and validate these ideas on the existing systems. → which means having some P1 backwards rendering.



“There are millions of WASI preview-1 devices in production and a large ecosystem has emerged in the embedded space. There are multiple things that this ecosystem requires in order to continue to be successful. They are driven by the logistical challenges in physically distributing a runtime, and the hardware formfactor upon which it operates. Additionally, WASI/WASM on embedded devices competes with other solutions, including Docker / containers for larger embedded systems. It needs to compete effectively against this incumbent technology. This imposes cost and physical limits which increase the need for legacy support for preview1, guide the transition mechanics for preview2, impose performance characteristics for the component model, and inspire new projects and proposed improvements both to WASI and WASM”

Millions of WASI preview-1 devices & Large ecosystem has emerged

Logistical challenges in physically distributing a runtime

Legacy support for preview1

Hardware formfactor & WASI/WASM on embedded devices competes with Docker / containers

Validation of Performance characteristics for the component model

Transition mechanics for preview2

New projects

The Embedded Ecosystem Requires:

- 1. Continued WASI-Preview 1 Support**
- 2. Validation of the Component Model for Embedded Systems**
- 3. Embedded Innovation**

The Embedded Ecosystem Requires:

1. Continued WASI-Preview 1 Support

- WASI-SDK Tooling
- WIT -> WITx
- Threading, Sockets / Networking

2. Validation of the Component Model for Embedded Systems

Specifically for WASM \leftrightarrow Host Interface

1. Reliable and deterministic
2. As performance equivalent or near to P1

3. Embedded Innovation

1. I2C / GPIO / SPI Interface definition
2. Multiple memories (WASI / Wasm)
3. Read only memory (WASI / Wasm)
4. Dynamic-loading