The Emergence of WebAssembly (Wasm) in Scientific Computing

Armin Sobhani
asobhani@sharcnet.ca
https://staff.sharcnet.ca/asobhani
SHARCNET | Compute Ontario
HPC Technical Consultant
OntarioTech UNIVERSITY

SHARCNET™

Compute Ontario

In Today's Webinar…

- Why Wasm?

- Wasm 101

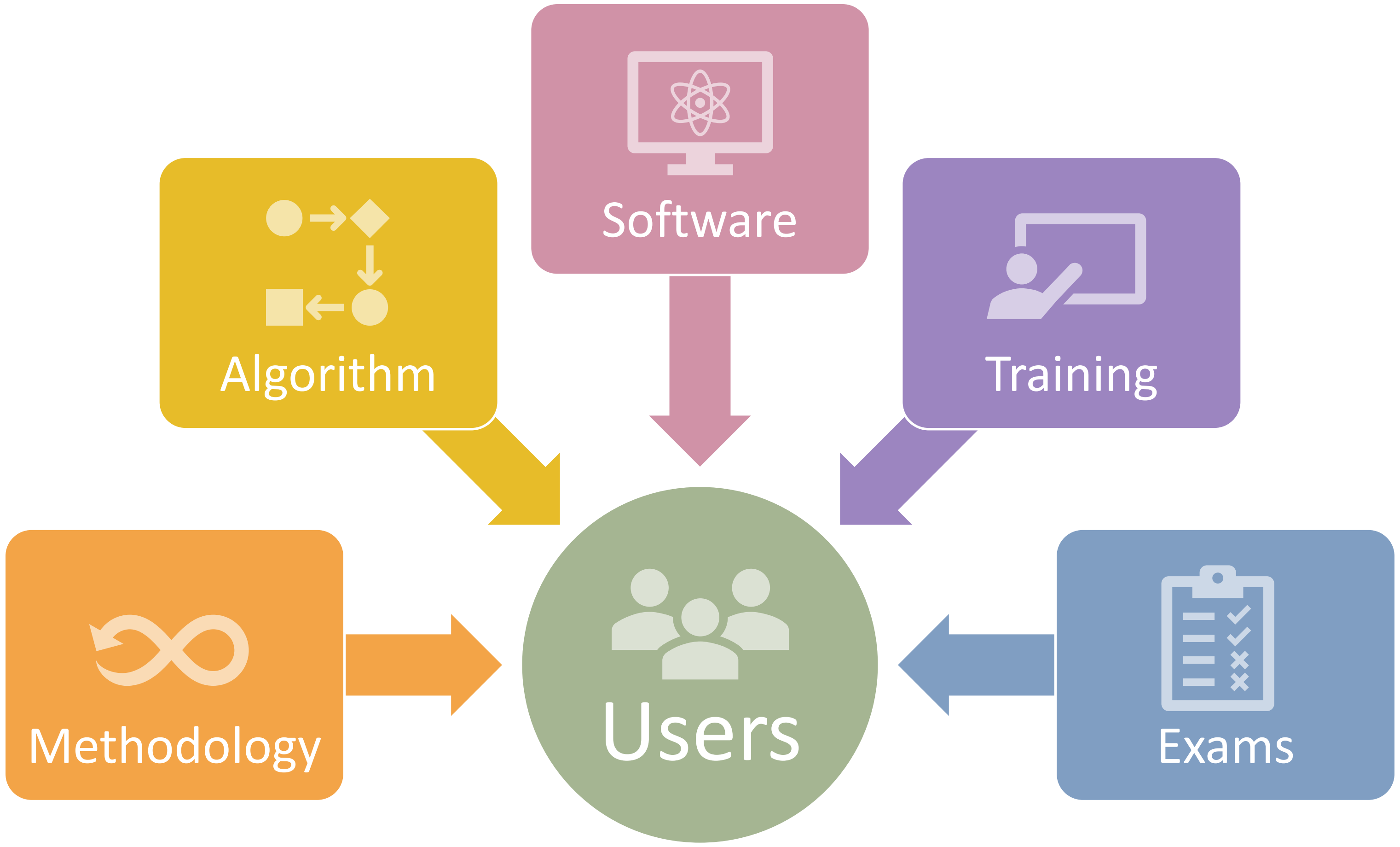
- Emscripten

- WebGPU

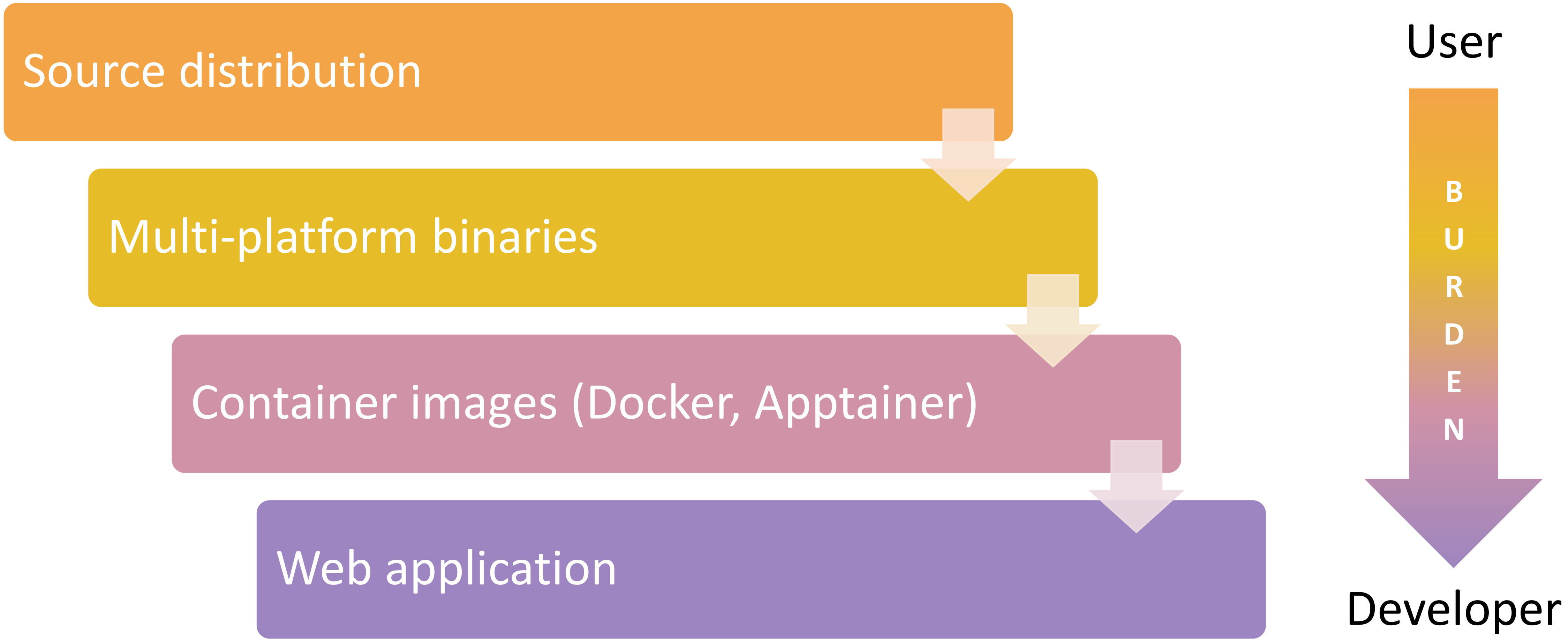
- Live Session

Why Wasm?

The State of the Problem



Solutions


Source distribution
Multi-platform binaries
Container images (Docker, Apptainer)
Web application
User
BURDEN
Developer

Providing a Web App – Challenges

Portability

Availability

Scalability

Challenge Accepted – Wasm Web App

Portability
Availability
Scalability

WebAssembly (Wasm) 101

What’s Wasm?

- A technology that allows running high-performance, low-level code in web browsers
- It’s a binary-code format that servers as a compilation target for other programming languages
- Binary-code can run on any platform that hosts a compliant Wasm virtual machine
- Can be executed at near-native code performance
- Wasm modules are isolated from the rest of browser’s runtime

Wasm – Features

- Hardware-independent
- Language-independent
- Platform-independent
- Fast
- Modular

Wasm – Features

- Hardware-independent
- Language-independent
- Platform-independent
- Fast
- Modular

compile once, and run anywhere

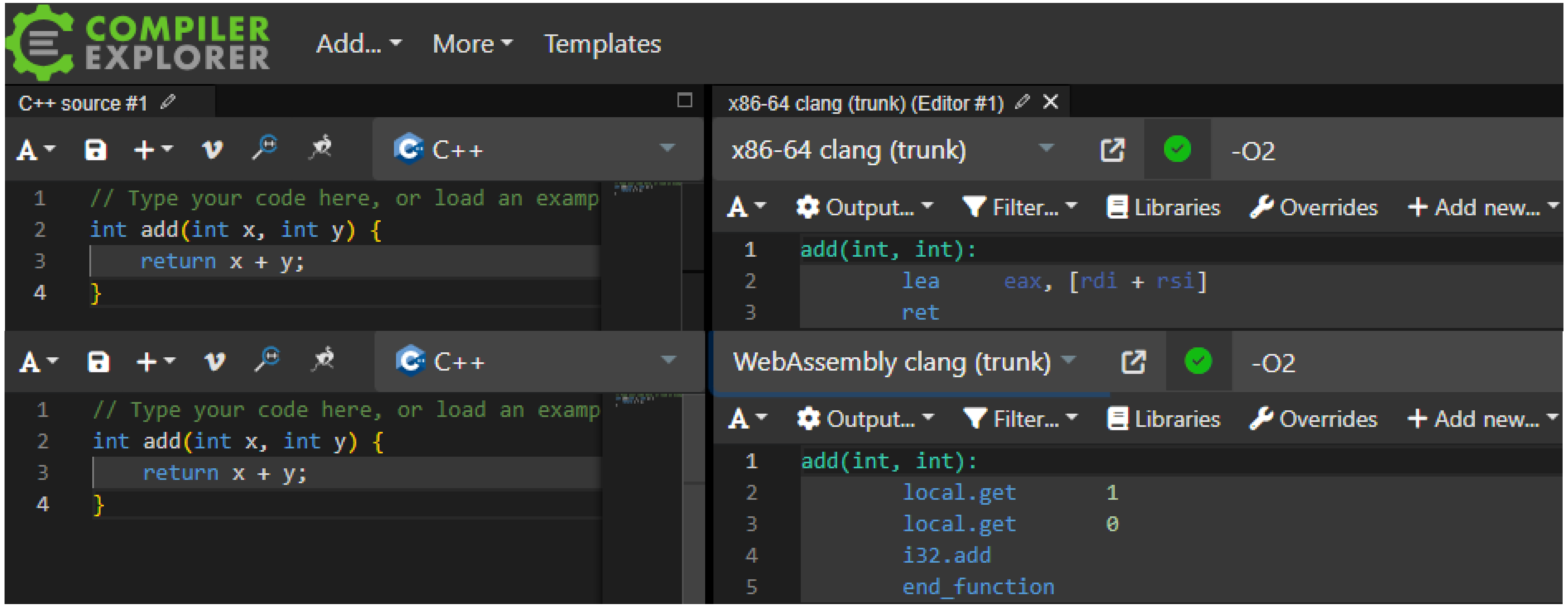
Wasm – Features

- Hardware-independent
- Language-independent
- Platform-independent
- Fast
- Modular

compile once, and run anywhere

good fit for scientific computing

Wasm Overview – Typed Stack-Based Machine



```cpp
// Type your code here, or load an examp
int add(int x, int y) {
    return x + y;
}
```

x86-64 clang (trunk) -O2

```
add(int, int):
        lea     eax, [rdi + rsi]
        ret
```

```cpp
// Type your code here, or load an examp
int add(int x, int y) {
    return x + y;
}
```

WebAssembly clang (trunk) -O2

```
add(int, int):
        local.get       1
        local.get       0
        i32.add
        end_function
```

Wasm Overview – Modules

units of deployment, loading, and compilation

stateless and side-effect free

declares imports and exports

multiple modules can interact within single application

Wasm Overview – Memory Model

- Harvard architecture (linear memory separate from code)
- A single *linear memory* per module
- Flat array of bytes
- Can grow by a multiple of the page size (64K)
- Cannot be shrunk

What about Standard Library?

emscripten

WASI

For web embedding

- Relies on Web APIs and JavaScript
- More mature
- Provides POSIX and libc/libc++ on top
- Also sound, graphics, etc. via SDL

For standalone runtimes

- Has POSIX like interface
- Work in progress
- Provides libc/libc++

Wasm Data Flow Architecture

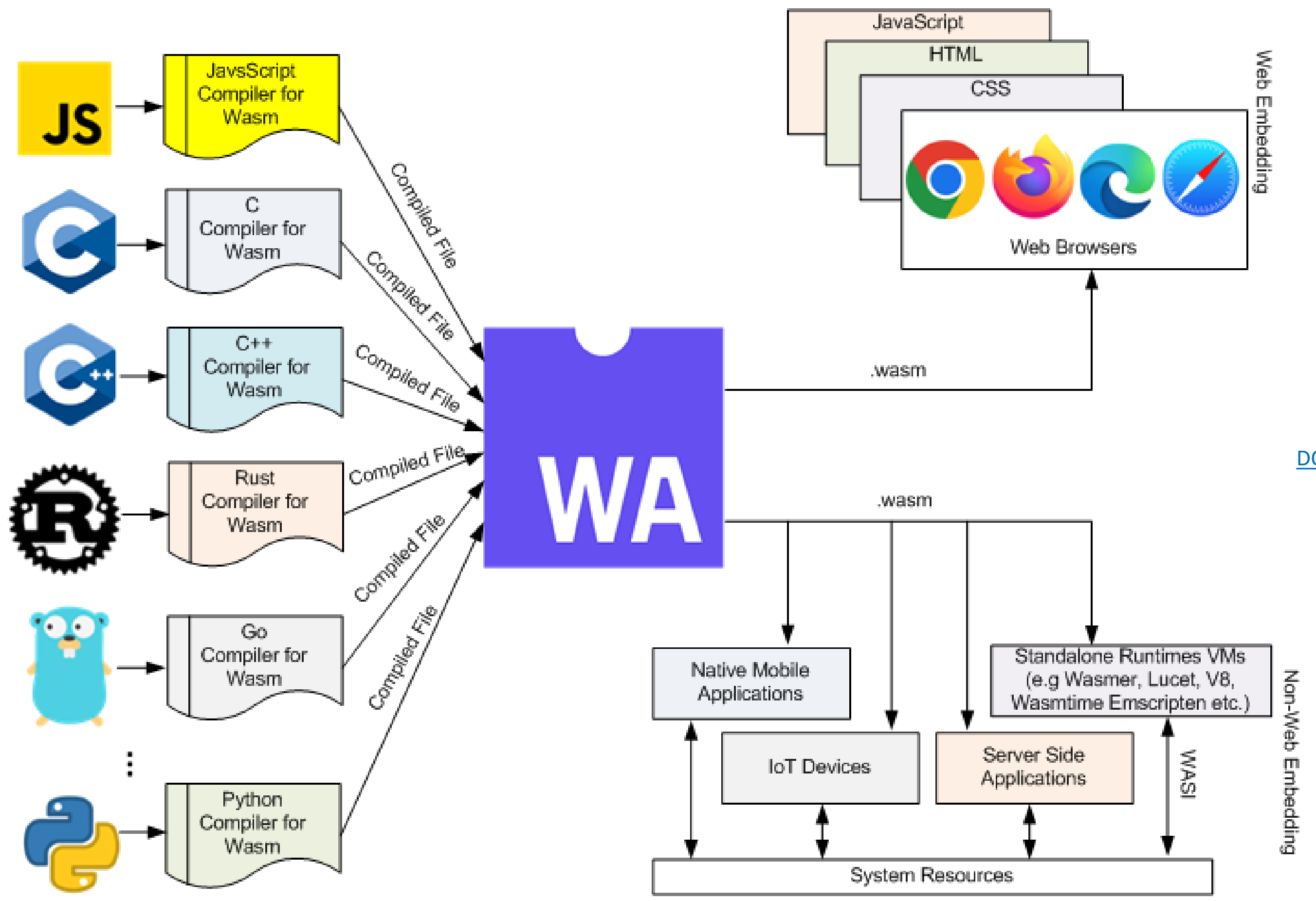


Image courtesy of
DOI:10.3390/fi15080275

Wasm Compilers for Scientific Computing

Language	Compiler	Optimization	Ecosystem	Performance	Link (QR)
C/C++	Emscripten	High	Wide	Moderate	
C/C++	Clang 8 and higher	High	Wide	Moderate	
Fortran	Full-Stack-Fortran	Low	Limited	Low	
Rust	RustC	High	Growing	High	
Python	Pyodide	Low	Limited	Low	
Jupyter	JupyterLite	Low	Growing	Low	
R	webR	Low	Growing	Low	

Challenge Accepted – Wasm Web App

Portability
Availability
Scalability

New Challenges

The whole dependency must be ported

May require some web development (e.g. JS) for user interaction

SHARCNET™

Compute Ontario

Limitations

Wasm32 is limited to 32 bits and can access only 2^{32} bytes (4 GB)

Currently allocating more than ~300MB of memory is not reliable

Multithreading on hold (*Spectre* and *Meltdown* security vulnerabilities)

Has no direct DOM (Document Object Model) access

emscripten

SHARCNET™

Compute Ontario

Emscripten

- The oldest (since 2010) compiler and toolchain targeting Wasm
- Primarily for web embedding
- Uses LLVM and Binaryen
- Supports POSIX threads (pthreads) using SharedArrayBuffer
- Converts OpenGL to WebGL

Emscripten Ports

```
$ emcc --show-ports
Available official ports:
    boost_headers - Boost headers v1.70.0 (--use-port=boost_headers)
    bullet (-sUSE_BULLET=1 or --use-port=bullet; zlib license)
    bzip2 (-sUSE_BZIP2=1 or --use-port=bzip2; BSD license)
    cocos2d (-sUSE_COCOS2D=3 or --use-port=cocos2d)
    freetype (-sUSE_FREETYPE=1 or --use-port=freetype; freetype license)
    giflib (-sUSE_GIFLIB=1 or --use-port=giflib; MIT license)
    harfbuzz (-sUSE_HARFBUZZ=1 or --use-port=harfbuzz; MIT license)
    icu (-sUSE_ICU=1 or --use-port=icu; Unicode License)
    libjpeg (-sUSE_LIBJPEG=1 or --use-port=libjpeg; BSD license)
    libmodplug (-sUSE_MODPLUG=1 or --use-port=libmodplug; public domain)
    libpng (-sUSE_LIBPNG or --use-port=libpng; zlib license)
    mpg123 (-sUSE_MPG123=1 or --use-port=mpg123; zlib license)
    ogg (-sUSE_OGG=1 or --use-port=ogg; zlib license)
    regal (-sUSE_REGAL=1 or --use-port=regal; Regal license)
```

```
    sdl2 (-sUSE_SDL=2 or --use-port=sdl2; zlib license)
    sdl2_gfx (-sUSE_SDL_GFX=2 or --use-port=sdl2_gfx; zlib license)
    sdl2_image (-sUSE_SDL_IMAGE=2 or --use-port=sdl2_image; zlib license)
    sdl2_mixer (-sUSE_SDL_MIXER=2 or --use-port=sdl2_mixer; zlib license)
    sdl2_net (-sUSE_SDL_NET=2 or --use-port=sdl2_net; zlib license)
    sdl2_ttf (-sUSE_SDL_TTF=2 or --use-port=sdl2_ttf; zlib license)
    sqlite3 (-sUSE_SQLITE3=1 or --use-port=sqlite3); public domain)
    vorbis (-sUSE_VORBIS or --use-port=vorbis; zlib license)
    zlib (-sUSE_ZLIB=1 or --use-port=zlib; zlib license)
Available contrib ports:
    contrib.glfw3 (--use-port=contrib.glfw3; Apache 2.0 license)
```

Emscripten Ports

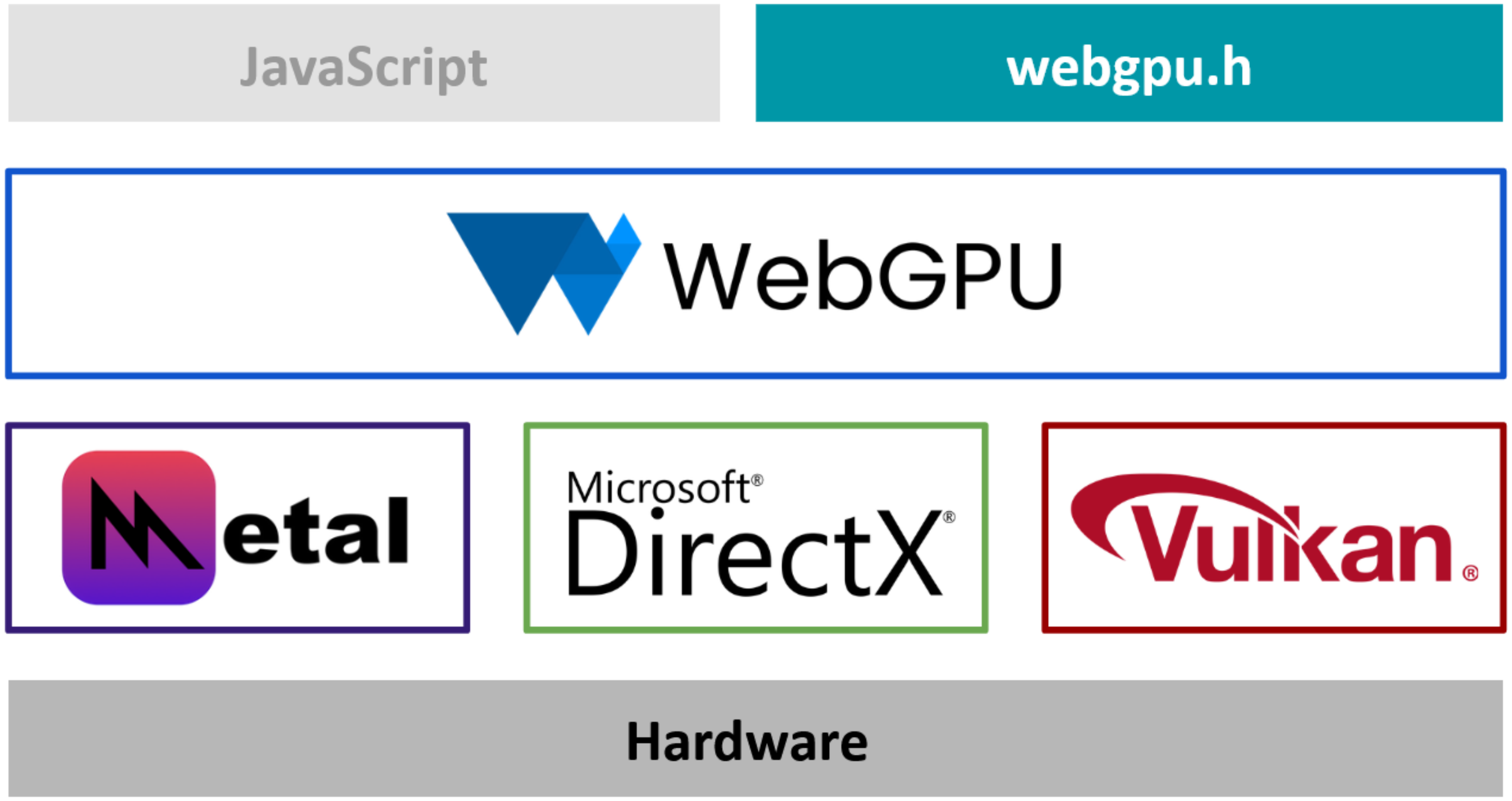
```
$ emcc --show-ports
Available official ports:
    boost_headers - Boost headers v1.70.0 (--use-port=boost_headers)
    bullet (-sUSE_BULLET=1 or --use-port=bullet; zlib license)
    bzip2 (-sUSE_BZIP2=1 or --use-port=bzip2; BSD license)
    cocos2d (-sUSE_COCOS2D=3 or --use-port=cocos2d)
    freetype (-sUSE_FREETYPE=1 or --use-port=freetype; freetype license)
    giflib (-sUSE_GIFLIB=1 or --use-port=giflib; MIT license)
    harfbuzz (-sUSE_HARFBUZZ=1 or --use-port=harfbuzz; MIT license)
    icu (-sUSE_ICU=1 or --use-port=icu; Unicode License)
    libjpeg (-sUSE_LIBJPEG=1 or --use-port=libjpeg; BSD license)
    libmodplug (-sUSE_MODPLUG=1 or --use-port=libmodplug; public domain)
    libpng (-sUSE_LIBPNG or --use-port=libpng; zlib license)
    mpg123 (-sUSE_MPG123=1 or --use-port=mpg123; zlib license)
    ogg (-sUSE_OGG=1 or --use-port=ogg; zlib license)
    regal (-sUSE_REGAL=1 or --use-port=regal; Regal license)
```

```
    sdl2 (-sUSE_SDL=2 or --use-port=sdl2; zlib license)
    sdl2_gfx (-sUSE_SDL_GFX=2 or --use-port=sdl2_gfx; zlib license)
    sdl2_image (-sUSE_SDL_IMAGE=2 or --use-port=sdl2_image; zlib license)
    sdl2_mixer (-sUSE_SDL_MIXER=2 or --use-port=sdl2_mixer; zlib license)
    sdl2_net (-sUSE_SDL_NET=2 or --use-port=sdl2_net; zlib license)
    sdl2_ttf (-sUSE_SDL_TTF=2 or --use-port=sdl2_ttf; zlib license)
    sqlite3 (-sUSE_SQLITE3=1 or --use-port=sqlite3); public domain)
    vorbis (-sUSE_VORBIS or --use-port=vorbis; zlib license)
    zlib (-sUSE_ZLIB=1 or --use-port=zlib; zlib license)
Available contrib ports:
    contrib.glfw3 (--use-port=contrib.glfw3; Apache 2.0 license)
```

WebGPU

WebGPU

Image courtesy of
eliemichel.github.io/LearnWebGPU



`webgpu.h` Implemetations (Bindings)

Emscripten

Javascript

Web

`-sUSE_WEBGPU`

Dawn

C++

Standalone

wgpu-native

Rust

Standalone

Live Session

https://github.com/arminms/gol2p