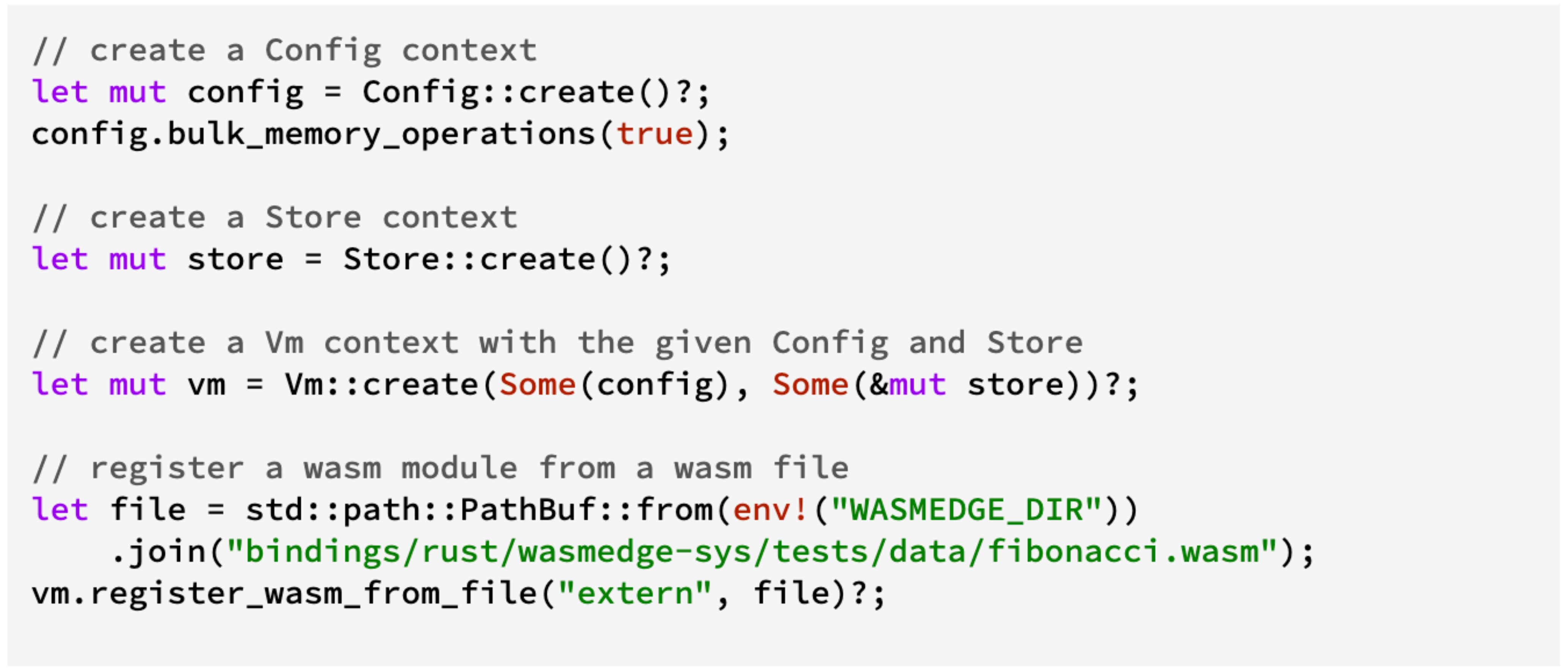
WasmEdge

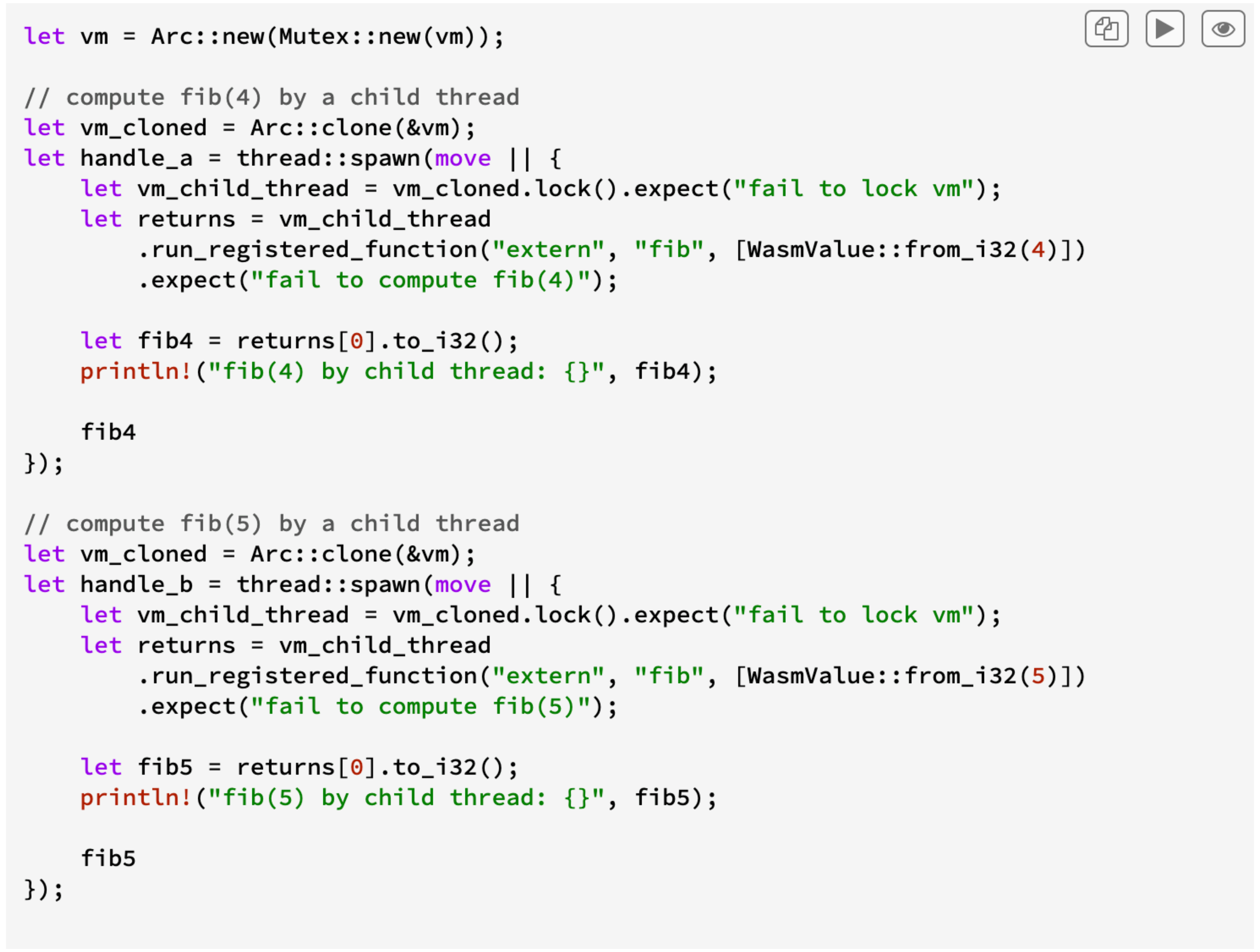
Compute Fibonacci numbers concurrently

Step 1: create a Vm context and register the WebAssembly module



```rust
// create a Config context
let mut config = Config::create()?;
config.bulk_memory_operations(true);

// create a Store context
let mut store = Store::create()?;

// create a Vm context with the given Config and Store
let mut vm = Vm::create(Some(config), Some(&mut store))?;

// register a wasm module from a wasm file
let file = std::path::PathBuf::from(env!("WASMEDGE_DIR"))
    .join("bindings/rust/wasmedge-sys/tests/data/fibonacci.wasm");
vm.register_wasm_from_file("extern", file)?;
```

Step 2: create two child threads to compute `Fib(4)` and `Fib(5)` respectively



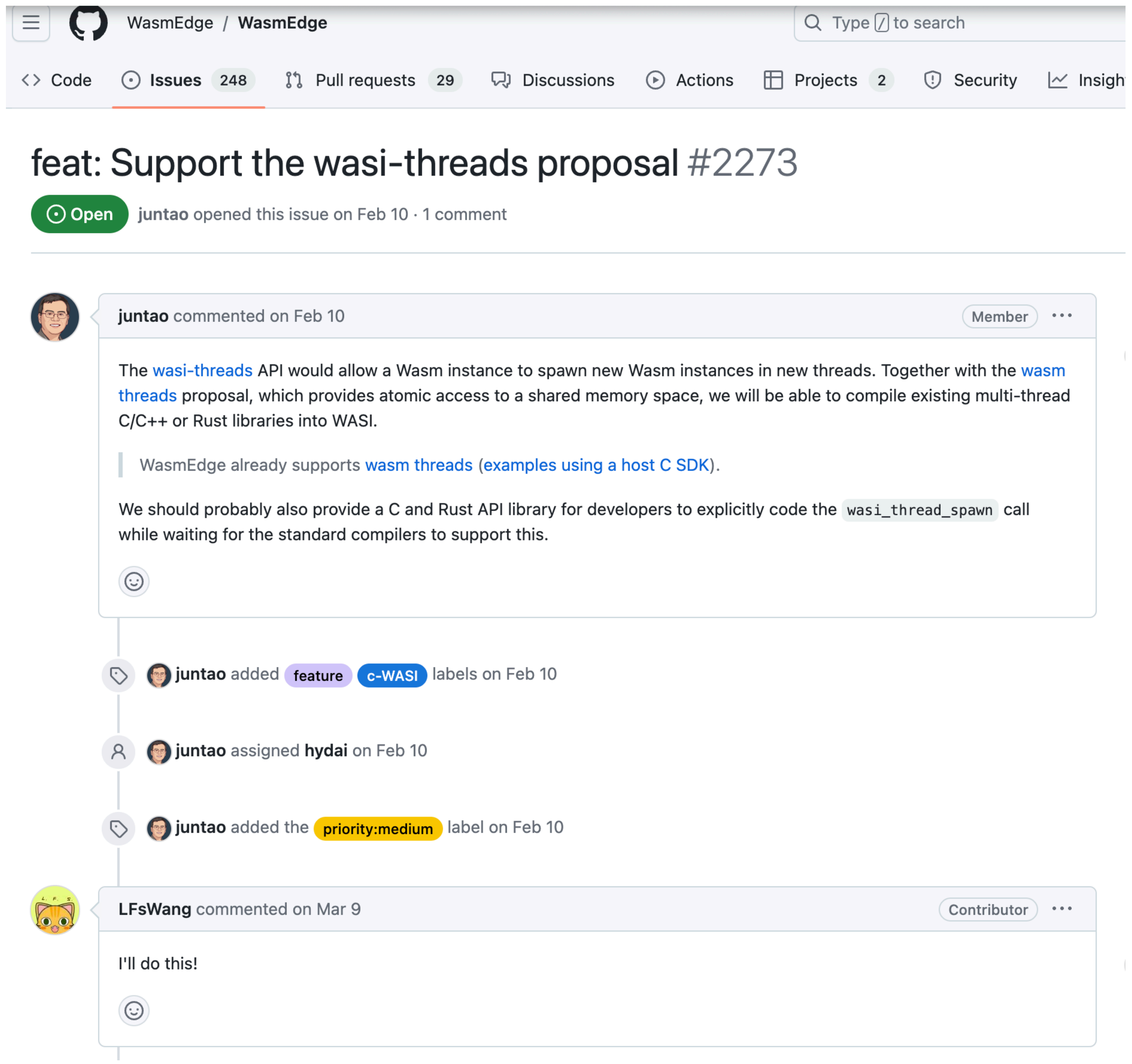
```rust
let vm = Arc::new(Mutex::new(vm));

// compute fib(4) by a child thread
let vm_cloned = Arc::clone(&vm);
let handle_a = thread::spawn(move || {
    let vm_child_thread = vm_cloned.lock().expect("fail to lock vm");
    let returns = vm_child_thread
        .run_registered_function("extern", "fib", [WasmValue::from_i32(4)])
        .expect("fail to compute fib(4)");

    let fib4 = returns[0].to_i32();
    println!("fib(4) by child thread: {}", fib4);

    fib4
});

// compute fib(5) by a child thread
let vm_cloned = Arc::clone(&vm);
let handle_b = thread::spawn(move || {
    let vm_child_thread = vm_cloned.lock().expect("fail to lock vm");
    let returns = vm_child_thread
        .run_registered_function("extern", "fib", [WasmValue::from_i32(5)])
        .expect("fail to compute fib(5)");

    let fib5 = returns[0].to_i32();
    println!("fib(5) by child thread: {}", fib5);

    fib5
});
```

Wasm Thread proposals

- wasm threads
  - 提供了共享内存、原子操作和等待/通知所需的基本操作
  - The responsibility of creating and joining threads is deferred to the embedder. embedder指WebAssembly运行时的宿主环境
- wasi-threads
  - 这是一个**WASI**级别的提案，属于wasm提案的增强版；
  - 仅仅提供了一种**生成线程**的机制， 其他类似于线程的操作（如线程加入、锁定等）都将使用来自wasm级别proposal中基本操作；

Wasmedge will support thread proposals

- The wasi-threads API would allow a Wasm instance to spawn new Wasm instances in new threads.
- Together with the wasm threads proposal, which provides atomic access to a shared memory space, we will be able to **compile existing multi-thread C/C++ or Rust libraries into WASI.**
- "We would work on thinking how to design **Multi Thread per Wasm Instance** in the future."



feat: Support the wasi-threads proposal #2273

Open juntao opened this issue on Feb 10 · 1 comment

juntao commented on Feb 10 Member

The wasi-threads API would allow a Wasm instance to spawn new Wasm instances in new threads. Together with the wasm threads proposal, which provides atomic access to a shared memory space, we will be able to compile existing multi-thread C/C++ or Rust libraries into WASI.

> WasmEdge already supports wasm threads (examples using a host C SDK).

We should probably also provide a C and Rust API library for developers to explicitly code the `wasi_thread_spawn` call while waiting for the standard compilers to support this.

juntao added feature c-WASI labels on Feb 10

juntao assigned hydai on Feb 10

juntao added the priority:medium label on Feb 10

LFsWang commented on Mar 9 Contributor

I'll do this!

目前的wasm并发方式与系统结合的问题

- 需要一个外部的c/rust函数来实现多线程，并且最后打包成一个可执行文件；
  - 运行形式从wasmedge xxx.wasm 改成 ./xxx，xxx.wasm是跨平台的，而./xxx不是
  - 执行过程还需要增加一个编译的步骤；
  - ...