A Different World: a Blockchain-Focused, General-Purpose Applicable Software Sandbox System Based on RISC-V

Xuejie Xiao (x@nervos.org)
“Your title is too long!”
Let me explain
A Different World: a Blockchain-Focused, General-Purpose Applicable Software Sandbox System Based on RISC-V

Xuejie Xiao (x@nervos.org)
A Different World: a Blockchain-Focused, General-Purpose Applicable Software Sandbox System Based on RISC-V

Xuejie Xiao (x@nervos.org)
A Different World: a Blockchain-Focused, General-Purpose Applicable Software Sandbox System Based on RISC-V

Xuejie Xiao (x@nervos.org)
A Different World: a Blockchain-Focused, General-Purpose Applicable Software Sandbox System Based on RISC-V

Xuejie Xiao (x@nervos.org)
Executes UNTRUSTED code

- JavaScript engines
  - v8
- Cloud provider products
  - Docker
  - AWS Lambda, Cloudflare Workers
Requirements for a software sandbox

- Security
  - Code in the sandbox should not escape the sandbox!
- (Loose) Determinism
  - Code in the sandbox should achieve the same result no matter where we run it.
- Performance
Additional requirements for blockchain VM

- **(Strict) Determinism**
  - No external environment (such as current time) should affect code running in the sandbox.

- **Runtime cost model**
  - A blockchain way of solving halting problem.

- **Future proof**
  - Upgrading a blockchain is extremely hard!
“Blockchain is hardware-like software which is hard to upgrade.”
Existing sandbox solution

- Docker
  - OS level virtualization with the help of Linux features (cgroups, kernel namespaces)
  - Limited by the CPU architecture of the underlying machine
- JavaScript
  - It’s actually a decent sandbox!
  - Too high level, optimizations can be hard
  - Bloated
- WebAssembly
WebAssembly

- A binary format for a stack-based virtual machine
- Mainly for the web, adopted as a sandbox technology, very popular among blockchain world
- Agreed as a spec amongst major browser vendors.
- A decent attempt
## WebAssembly

<table>
<thead>
<tr>
<th>C input source</th>
<th>Linear assembly bytecode (intermediate representation)</th>
<th>Wasm binary encoding (hexadecimal bytes)</th>
</tr>
</thead>
</table>
| int factorial(int n) {  
  if (n == 0)  
    return 1;  
  else  
    return n * factorial(n-1);  
} | get_local 0  
  i64.eqz  
  if (result i64)  
    i64.const 1  
  else  
    get_local 0  
    get_local 0  
    i64.const 1  
    i64.sub  
    call 0  
    i64.mul  
  end | 20 00  
  50  
  04 7E  
  42 01  
  05  
  20 00  
  20 00  
  42 01  
  7D  
  10 00  
  7E  
  0B |

Source: https://en.wikipedia.org/wiki/WebAssembly
WebAssembly: quirks

- Only a software spec, not a hardware spec
  - It only takes major browser vendors to agree to introduce breaking changes!
- Non-determinism
- Too many high level features harm runtime cost model as well as future compatibility
  - GC
  - Threading
WebAssembly: quirks

- Only a software spec, not a hardware spec
  - It only takes major browser vendors to agree to introduce breaking changes!
- Non-determinism
- Too many high level features harm runtime cost model as well as future compatibility
  - GC
  - Threading

An okay choice for general sandbox, but not a good blockchain sandbox solution.
Can we do better?
Introducing: CKB-VM

- A RV64IMC standard compliant software implementation of RISC-V
  - GCC is directly used to build binaries for CKB-VM
  - We might introduce V or P extensions in the future
- Real CPU cycles as runtime cost model (think CKB-VM as an in-order CPU)
- Ship algorithms with your code
  - Cryptographic algorithms
  - GC
```
size_t strlen(char *ptr) {
    char *curr = ptr;
    while (*curr != 0) {
        curr++;
    }
    return (curr - ptr);
}
```

```
function strlen(ptr) {
    ptr = ptr|0;
    var curr = 0;
    curr = ptr;
    while (MEM8[curr]|0 != 0) {
        curr = (curr + 1)|0;
    }
    return (curr - ptr)|0;
}
```

Source: https://en.wikipedia.org/wiki/Asm.js
lower level operations in a platform independent way

```
strlen(char*):
    cmp BYTE PTR [rdi], 0
    je .L4
    mov rax, rdi
 .L3:
    inc rax
    cmp BYTE PTR [rax], 0
    jne .L3
    sub rax, rdi
    ret
 .L4:
    xor eax, eax
    ret
```
RISC-V has the best of both worlds

```assembly
strlen(char*):
    lbu a5, 0(a0)
    mv a4, a0
    beqz a5, 14 .L4
.L3:
    addi a0, a0, 1
    lbu a5, 0(a0)
    bnez a5, 8 .L3
    sub a0, a0, a4
    ret
.L4:
    li a0, 0
    ret
```
V & P extensions can help as well!
Insight into the VM enables more use cases

- Adapt algorithms to budget
- Real time process migration
Biggest obstacle: performance
Inspiration

Huge thanks to Michael Clark & Bruce Hoult!
Biggest obstacle: performance

- Reasonably fast is enough
- JIT to the rescue
  - Due to the simplicity of RISC-V, it’s actually not very hard to translate to other architectures.
  - TOOWTDI: There’s Only One Way To Do It.
- And there’s more ...
Hardware support: why not?

- x86_64 CPU
- RISC-V CPU
- PCI-E or USB
- Outside world
Expand CKB-VM beyond blockchain: Edge Computing

“Blockchain is hardware-like software which is hard to upgrade.”
“Therefore we adopted RISC-V for CKB-VM because of its simplicity and flexibility.”