NEAR

NEAR 中文开发者分享
Wasm Runtime 工作原理

near.org

Bo Yao
Technical Lead
@ailisp

内容提要

1. 什么是Wasm Runtime
2. Wasm Runtime基本工作原理
3. Promise与跨合约调用
4. Gas模型
5. 实际应用

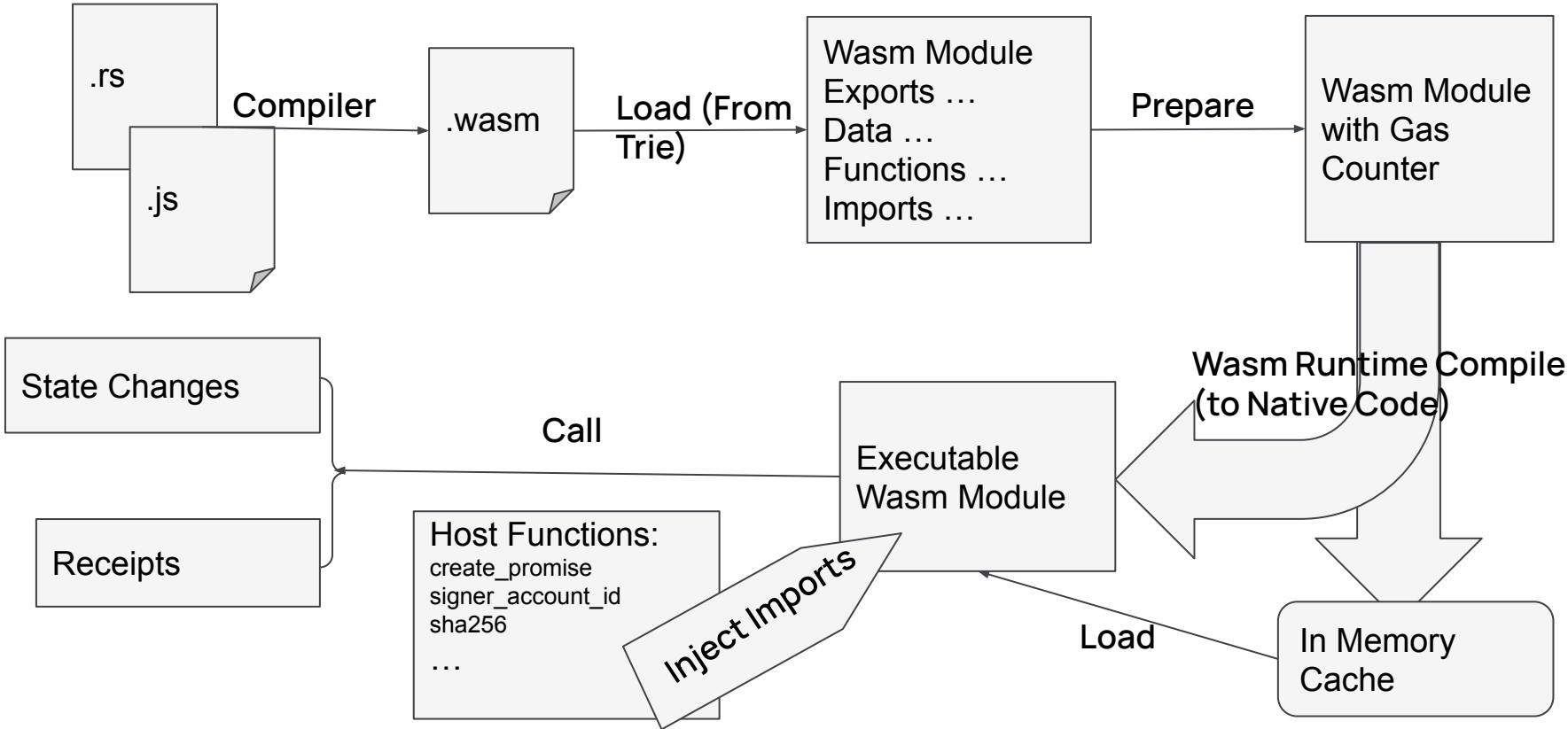
1. 什么是Wasm Runtime

- Runtime: NEAR节点的运行环境。处理创建账户，转账，合约调用等Tx中的各种Action
- Wasm Contract Runtime:
  - Runtime的子系统，仅处理智能合约调用
  - 运行智能合约程序的虚拟机，输入为区块链state + 用户输入参数输出为state changes + 其他要执行的Action（转账，调用其他合约等）
  - NEAR为WebAssembly VM，其他主流区块链为EVM或WebAssembly VM

内容提要

1. 什么是Wasm Runtime
2. Wasm Runtime基本工作原理
3. Promise与跨合约调用
4. Gas模型
5. 实际应用

1. Wasm Runtime基本工作原理



1. Wasm Runtime基本工作原理

- Prepare: https://github.com/near/nearcore/blob/master/runtime/near-vm-runner/src/prepare.rs
  - Validate，检查恶意Contract
  - 加入Gas Counter, Stack Height Limiter
  - 检查Import
- Compile/Cache:https://github.com/near/nearcore/blob/master/runtime/near-vm-runner/src/cache.rs
- Inject imports: https://github.com/near/nearcore/blob/master/runtime/near-vm-runner/src/imports.rs

内容提要

1. 什么是Wasm Runtime
2. Wasm Runtime基本工作原理
3. Promise与跨合约调用
4. Gas模型
5. 实际应用

3. Promise与跨合约调用

- 合约调用(Contract Call)本身为一个单步操作(Action)，不能同时完成其他单步操作
- 单步操作包括:创建/删除账户，添加/删除Key，部署/调用合约，转账/质押(NEAR)
- 如果希望完成其他单步操作，必须创建Promise
    - 创建的Promise效果是令节点创建Receipt，每个单步操作创建相应的一个或多个Receipt
    - Receipt会在下个block time交给Runtime，其中Function Call类型的Receipt会传给Wasm Runtime。
    - 所有的Receipt都处理完后，才会被加入block
    - 开发时要特别注意包含失败Receipt的Transaction仍然可以成功，promise_then要检查。

内容提要

1. 什么是Wasm Runtime
2. Wasm Runtime基本工作原理
3. Promise与跨合约调用
4. Gas模型
5. 实际应用

4. Gas模型

- Gas Counter在加载完合约之后，通过修改Wasm Module的方式注入
- 具体Gas计算方法为：
  - 每一个wasm instruction记一个gas
  - 每一个Host Function，如sha256，storage_read等，单独计算gas，数值在 ProtocolConfig.runtime_config.wasm_config.ext_costs
  - 创建Promise从而产生其他单步操作的，需要为创建操作付出gas，数值在 ProtocolConfig.runtime_config.transaction_costs

4. Gas模型

- 当前的数值是通过runtime-params-estimator计算得到的。原理是在QEMU中benchmark各种操作所用的CPU Instructions数。有些参数会随着Protocol Upgrade更新，一般是变小
- 可以通过RPC查询当前的protocol config得到gas的各项参数：https://docs.near.org/api/rpc/protocol
- 单个Action最大的Gas为300T，Transaction的所有Action也为300T
- 尽可能减少Number of wasm instructions，减少跨合约调用的次数也会减Gas使用
- 每一项Action和host function用到的gas会在workspace中显示，可以据此优化Gas

4. Gas模型

NEAR workspaces-js

中使用callRaw，输出结果

可以查看每一项gas用量：


```
"receipts_outcome": [
  {
    "block_hash": "9wWB2Py55P8Hw4B6gffoHTNbks2t5XQPF2UaXs9yVHJo",
    "id": "B8LZjjE1tCw5TagjwAQB3XcBEJuUfzNaqf58Wv4injXx",
    "outcome": {
      "executor_id": "dev-11695.test.near",
      "gas_burnt": 7304727873471,
      "logs": [],
      "metadata": {
        "gas_profile": [
          {
            "cost": "BASE",
            "cost_category": "WASM_HOST_COST",
            "gas_used": "1059072444"
          },
          {
            "cost": "CONTRACT_LOADING_BASE",
            "cost_category": "WASM_HOST_COST",
            "gas_used": "35445963"
          },
          {
            "cost": "CONTRACT_LOADING_BYTES",
            "cost_category": "WASM_HOST_COST",
            "gas_used": "112080341250"
          },
          {
            "cost": "READ_MEMORY_BASE",
            "cost_category": "WASM_HOST_COST",
            "gas_used": "2609863200"
          },
          {
            "cost": "READ_MEMORY_BYTE",
```

内容提要

1. 什么是Wasm Runtime
2. Wasm Runtime基本工作原理
3. Promise与跨合约调用
4. Gas模型
5. 实际应用

5. 实际应用

- Gas优化
- 增加新的Host Function实现昂贵的cryptographics primitives，比如ZKP中用到的bls12-381曲线
- 在NEAR生态开发新的SDK，如JavaScript，或使用非官方语言开发，如Zig，C，Golang，Python等。例如：https://github.com/austinabell/near-zig-hw/blob/main/src/main.zig

感谢参与！

参考资料：

1. https://github.com/near/nearcore
2. https://docs.near.org
3. https://nomicon.io
4. https://github.com/austinabell/near-zig-hw/blob/main/src/main.zig
5. https://github.com/near/near-sdk-js

问题、建议
bo@near.org