Motoko Stable Memory API

运行环境

- OS : Ubuntu 20.04.3 LTS
- DFX Version : 0.8.3

前提说明

Canister 本质上是WebAssembly模块，内存分配也和WASM的内存模型一样

WASM的内存模型是分页内存管理模型。每个内存页64KiB(65536 byte)。共65536页（即4G）。

Function

- func grow(size : Nat32) -> Nat32
    - 参数： 要新增加的 WASM page 数[只能从1-65535]， 初始时Stable WASM没有内存页，直接读会报错。
        - 注意， WASM page总量应该是65536页，但是直接分配65536会报StableMemory range out of bounds。应该是有一个内存页被创始时占用了。
    - 意义： 增加 page_number * 64KiB(65536 byte) 的wasm 内存空间（可以理解为C语言中的allocate）。grow一个内存页后， 内存页内的数据会全部被初始化为Blob \00 。
    - 返回之前的内存页数。当前可控的Stable Memory WASM仍然是4G，而不是8G，这个后面可能会优化。具体原因在IC Storage中说过了， 现在的WASM还是受制于32位限制， 不能扩容到更大的内存。之Canister Stable内存扩容的提案是在执行层做了中间件，让Canister可以无感访问（不用管指针）比4G更大的内存。
- func storeBlob(offset : Nat32， value : Blob) -> （）
    - 参数:
        - offset : 指针地址起始位置
        - value : Blob数据

        - 意义： 自动从offset开始存数据， 然后stable memory wasm中的起始位置变为offset+value.size()， offset的值目前需要人为控制
        - 不返回值。如果offset超过了当前的page数* 65536， 就会报错， out of bound。
- func loadBlob(offset : Nat32, size : Nat32) -> Blob
    - 参数：
        - offset： 指针开始位置【0-65535】
        - size： 要读的大小【0-65535】
        - 每个Page从0开始，到65535为止，共65536个byte。
        - 如果offset+size超过了当前分配的内存， 那么就会报错， out of bound。
    - 意义：
        - 可以自己控制offset，即从哪里开始读， size 控制读的大小。
        - 可以将offset理解为数组头指针（或者某一位指针）， size为偏移量(在C语言中是sizeof(Int/Char/…))。通过offset+size可以访问数组的某个元素。
- func size -> Nat32
    - 返回内存页的数量
    - 意义： 当前实际Stable内存为： 已经分配的内存页的数量*65536 byte
    - 具体计算表达式： stable_memory_size = size()*65536

建议：

- 由于当前Canister的限制， 我们只能将存储放在运行时Canister的内存中。这就造成了：如果存的东西多， 那么堆内存就小，能做的事就少。存储就变成一个很掣肘的事情。[1]
- 数据盒子项目的优化接下来要做的主要有两个：
    - 继续完善Databox和AutoScale-Storage Container 两个平台。前者为用户做存储组件，后者目标为搭建一个通用的DAPP存储平台。
    - 持续跟进Stable Memory API， 用Rust和Motoko两个语言实现Canister存储
- 每个DAPP需要用上分配给自己的Stable内存， 现在基本上没用，加上StableAPI后就可以实现运行时内存和Stable内存进行交换。就可以让运行时内存更多的服务于业务而不是存储，存储则交给Stable内存。这是一个很大的进步， 之前的WASM32应用就是缺少Stable+运行时这种内存模型， 导致[1]的问题。

Forum讨论跟进：

https://forum.dfinity.org/t/experimental-stable-memory-module/8277/3