MOTR API

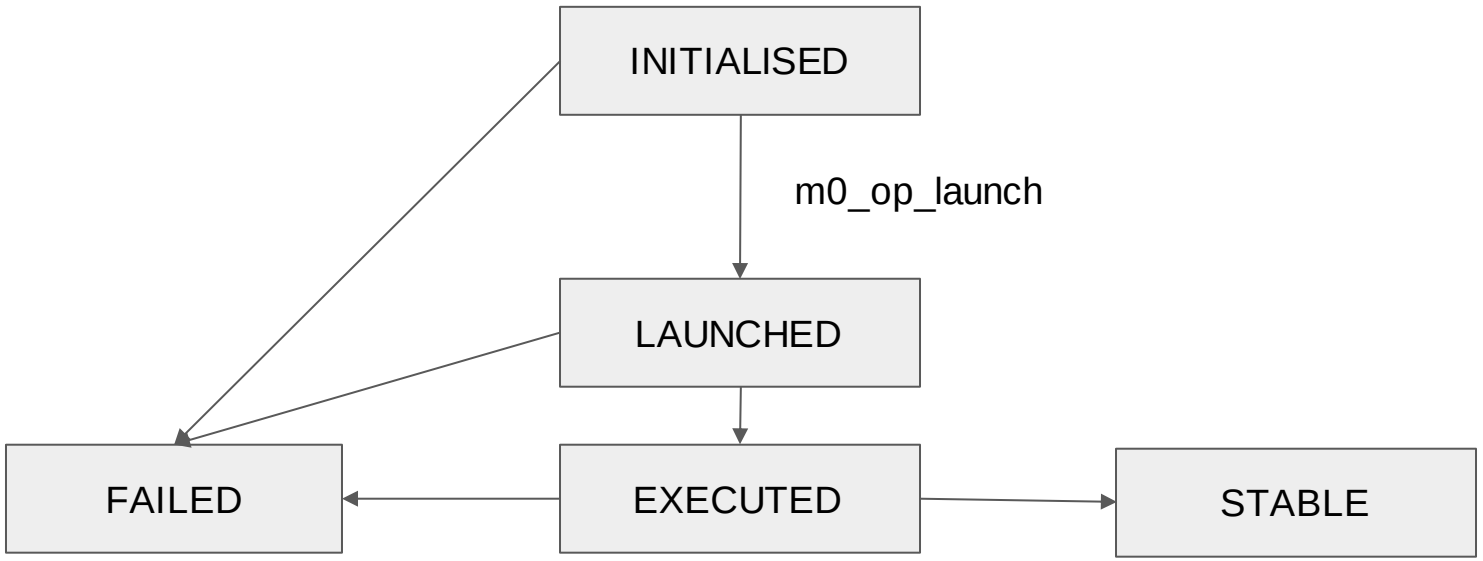
Source/Footnote:

Motr Access Interface Overview

- Operation
  - asynchronous
- Object
  - 128-bit persistent identifiers, assigned by user
  - Create/Delete an Object.
  - Read/Write/Sync an Object.
- Index: distributed key-value store
  - 128-bit persistent identifiers, assigned by user
  - Create/Delete and Lookup an Index.
  - GET/PUT/DEL/NEXT key-value pairs of an Index.
- Others such as APIs for composite objects (useful for applications such as HSM)

Part I: Operation

- A state machine that represents an ongoing request to Motr
- Asynchronous



Operation	m0_op_launch()	Launch an operation
	m0_op_wait()	Wait on an operation
	m0_op_setup()	Set callback functions for an operation
	m0_op_cancel()	Cancel operations
	m0_op_fini/free()	Finalise/free an operation

Entity Operations (for both object and index)	M0_EO_CREATE M0_EO_DELETE M0_EO_OPEN M0_EO_SYNC M0_EO_GETATTR M0_EO_SETATTR M0_EO_LAYOUT_GET M0_EO_LAYOUT_SET	Create/delete an entity Open an entity Flush entity data to storage devices. For Motr internal uses only For composite layout only
Object Operations	M0_OC_READ M0_OC_WRITE	IOs in scatter-gather-scatter fashion
Index Operations	M0_IC_GET M0_IC_PUT M0_IC_DEL M0_IC_NEXT M0_IC_LOOKUP M0_IC_LIST	Index queries, including GET, PUT, DEL and NEXT Check an index for an existence Given an index id, get the list of next indices

Part II: Object

- An object is viewed as an array of data-blocks, [0, 2^64), initially a hole
  - Flat name space
  - Network striped (Parity declustered data layout)
- No usual metadata (attributes, etc.) exposed to users, such as object size
  - Maximize IO performance by avoiding updating object attributes.
  - Mero has internal object attributes such as data layout.
- CREATE, DELETE, READ, WRITE, ALLOC, FREE operations
- scatter-gather-scatter READ and WRITE operations

Object Access	m0_obj_init/fini() m0_entity_create() m0_entity_delete() m0_entity_open() m0_obj_op() m0_sync_op_init() m0_sync_entity_add() m0_sync_op_add()	Initialise/finalise in-memory object data structure. Create and initialise operation for object creation and deletion. Once opened, the object data structure can be used in later READ/WRITE ops. Create and initialise an object operation specified by the opcode Create and initialise an SYNC operation. Add an entity to SYNC operation Add an `op` to SYNC operation.
Object Lock	m0_obj_lock_init() m0_obj_lock_fini() m0_obj_lock_get_sync() m0_obj_lock_get() m0_obj_lock_put()	Currently Motr only supports exclusive “whole object” lock in a group.

Part III: Index

- Distributed key-value store
- Key and value
  - Keys and values are bit-strings
  - Values accessed as a whole
  - Iteration in the order of keys
- Operations
  - CREATE, DELETE, LOOKUP and LIST index
  - GET, PUT, DEL and NEXT for index query
  - Scatter-gather-scatter operations
- User uses indices to build meta-data structures: namespace, file attributes

Motr Client Index APIs

Index (Key-value) Access	m0_idx_init/fini()	Initialise/finalise in-memory object data structure
	m0_entity_create() m0_entity_delete()	Create and initialise index creation/deletion operations.
	m0_idx_op()	Create and initialise index query operation. Clovis support GET, PUT, DEL and NEXT queries.
	m0_sync_op_init m0_sync_entity_add m0_sync_op_add	Flush index key and value to persistent storage devices.

Motr Access Interface Specifications

https://github.com/Seagate/cortx-motr/blob/main/motr/client.h

Examples

https://github.com/Seagate/cortx-motr-apps

https://github.com/Seagate/cortx-motr/tree/main/hsm

https://github.com/Seagate/cortx-mio

Questions?

Motr Internal Architecture