Governor for DB2

Governor.py

- run()

init truth_manager: etcd3

init db2

init ha = Ha(db2, etcd3)

- init_setup() (startup.py)

  - ha.lock_updater() (ha.py)
    ↓ new thread
    keep_updating_lock() —no lock→ sleep(10)
    ↓ has lock
    update_lock()
    ↓
    update_leader() (etcd3.py)
    ↓
    put data into etcd (leader, host_name)

  - init(etcd, db2, ha)

    db2.start() (db2.py) db2start

    db2.get_role() db2 get db cfg for HADR. get role for it

    get leader timestamp: db2.fetch_leader_timestamp() → get timestamp from timestamp_file

    get truth timestamp: truth_manager.get_prev_leader_timestamp() → get value of "/prev"

    if leader_timestamp is none or truth_timestamp is none or truth_timestamp <= leader_timestamp

    yes:
    db2.is_primary()
    ↓ yes
    db2.start_as_primary()
    Rlock.acquire()
    db2start
    db2 start hadr on db HADRDB as primary by force
    Rlock.release()

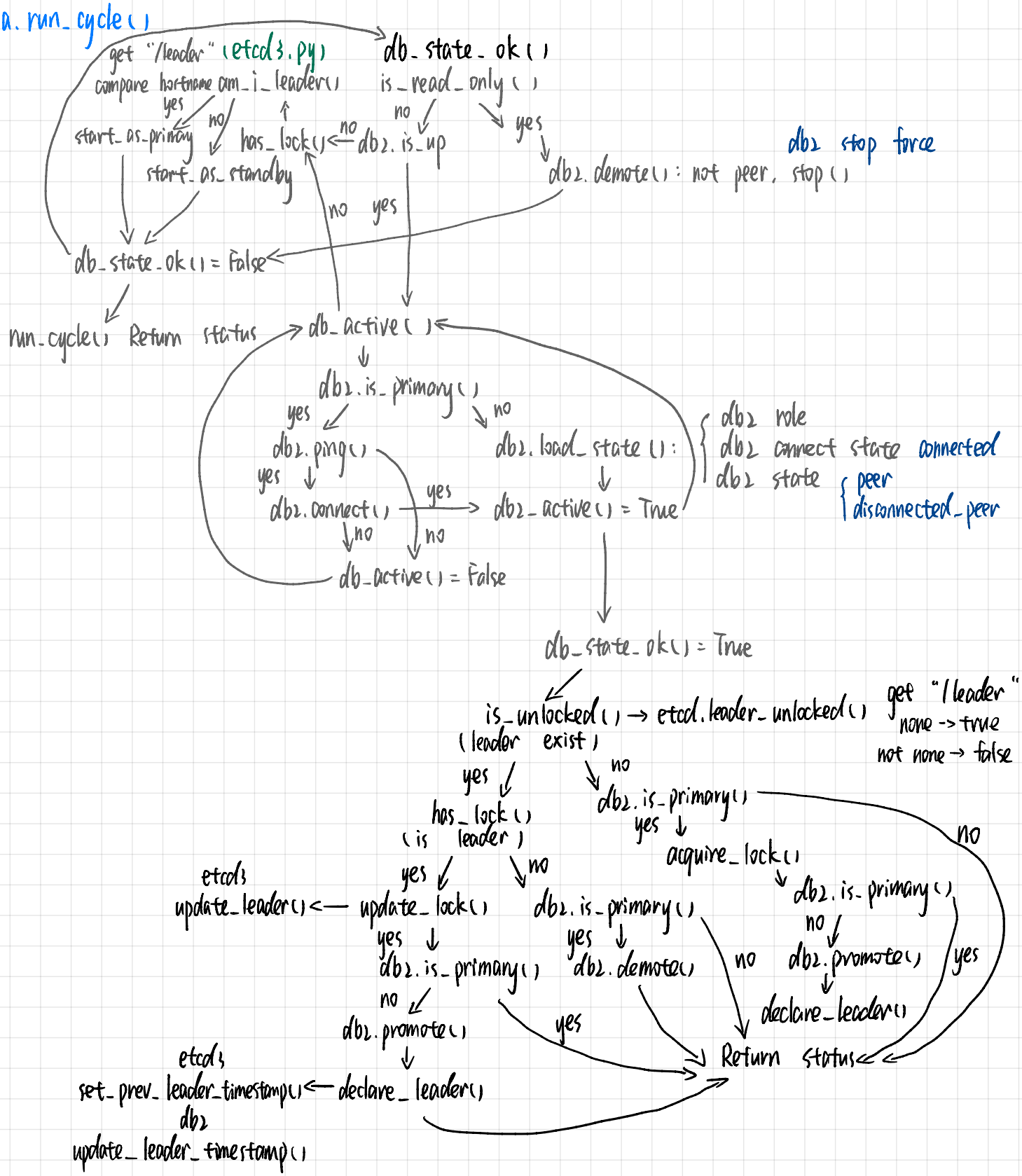
    no:
    - init_standby()
    ↓
    db2.start_as_standby()
    db2start
    db2 start hadr on db HADRDB as standby
    loop get current leader: truth_manager.current_leader()
    ↳ get value for "/leader"

  - ha.start_daemon_pool()

    - health_checker() —new thread→ check_health() → is primary → db2 connect to HADRDB user username using password

- lock_updater() —new thread→ keep_updating_lock()
- db_watchdog() —new thread→ check_db() → db2.is_running() ps -ef | grep -i db2sysc | grep -v grep

while true
- ha.monitor_daemon() → health_checker, lock_updater, db_watchdog
- ha.run_cycle()



failover

situation 1:

hadr_node1

db2 role : primary

db2 connect state : connected

db2 state : peer

hadr_node2

db2 role : standby

db2 connect state : connected

db2 state : peer

situation 2:

hadr_node1

stopped

governor:

- db_state_ok
  - ↓
  - db_active()
    - ↓ no
    - is leader
      - ↓ yes
      - start_as_primary
        - ↓ operation failed, release lock
        - discard_lock:
          - ↓
          - Return status
          - (database software was stopped on primary. starting again)

etcd.attempt_to_delete_leader

↓

delete "/leader"

hadr_node2

db2 role : standby

db2 connect state : disconnected

db2 state : disconnected_peer

governor:

ha.run_cycle()

- db_state_ok = True
- update memership
  - ↓
  - leader exists
    - ↓ no
    - is leader
      - ↓ no
      - is primary
        - ↓ no
        - acquire_lock
          - ↓
          - promote db2 takeover hadr on db HADRDB by force
            - ↓
            - declare_leader
              - ↓
              - Return status
              - (promoted self to leader by acquiring session lock)

situation 3 :

hadr_node1

restart

db2 role : primary

db2 connect state : disconnected

db2 state : disconnected_peer()

hadr_node2

db2 role : standby

db2 connect state : connected

db2 state : peer

governor :

ha.run_cycle() (first loop)

- db_state_ok = True
  - ↓
  - update membership
    - ↓
    - leader exist
      - ↓ yes
      - is leader
        - ↓ no
        - is primary
          - ↓ yes
          - db2.demote() db2 stop force
            - ↓
            - Return status (demoting self because I do not have the lock and I was a leader)

ha.run_cycle() (second loop)

- db_state_ok
  - ↓
  - db_active
    - ↓
    - is primary
      - ↓ yes
      - db2ping
        - ↓ no
        - db_active_Return = False
          - ↓
          - is leader
            - ↓ no
            - stop() db2 stop
          - start_as_standby() db2 start hadr on db HADRDB as standby
            - ↓
            - db_state_ok_Return = False
              - ↓
              - Return status (database software was stopped on standby. starting again)