```
──────────────── MODULE TunableMongoDB_Repl ────────────────
EXTENDS Naturals, FiniteSets, Sequences, TLC

 constants and variables
CONSTANTS Client, Server,     the set of clients and servers
          Key, Value,         the set of keys and values
          Nil,               model value, place holder
          PtStop,             max physical time
          Number              writeConcern number

VARIABLES Primary,           Primary node
          Secondary,         secondary nodes
          Oplog,             oplog[s]: oplog at server[s]
          Store,             store[s]: data stored at server[s]
          Ct,                Ct[s]: cluster time at node s
          Ot,                Ot[s]: the last applied operation time at server s
          ServerMsg,         ServerMsg[s]: the channel of heartbeat msgs at server s
          Pt,                Pt[s]: physical time at server s
          Cp,                Cp[s]: majority commit point at server s
          State,             State[s]: the latest Ot of all servers that server s knows
          CalState,          CalState: sorted State[Primary]
          CurrentTerm,       CurrentTerm[s]: current election term at server s
                             → updated in update_position, heartbeat and replicate
          ReadyToServe,      equal to 0 before any primary is elected
          SyncSource         sync source of server node s
├──────────────────────────────────────────────────────────┤
ASSUME Cardinality(Client) ≥ 1   at least one clinet
ASSUME Cardinality(Server) ≥ 2   at least one primary and one secondary
ASSUME Cardinality(Key) ≥ 1   at least one object
ASSUME Cardinality(Value) ≥ 2   at least two values to update

 Helpers
├──────────────────────────────────────────────────────────┤
HLCLt(x, y) ≜ IF x.p < y.p
                THEN TRUE
               ELSE IF x.p = y.p
                THEN IF x.l < y.l
                          THEN TRUE
                      ELSE FALSE
                ELSE FALSE

HLCMin(x, y) ≜ IF HLCLt(x, y) THEN x ELSE y
HLCMax(x, y) ≜ IF HLCLt(x, y) THEN y ELSE x
HLCType ≜ [p : Nat, l : Nat]
Min(x, y) ≜ IF x < y THEN x ELSE y
```

```
$Max(x, y) \triangleq$ IF $x > y$ THEN $x$ ELSE $y$

$vars \triangleq \langle Primary, Secondary, Oplog, Store, Ct, Ot, ServerMsg,$
          $Pt, Cp, CalState, State,$
          $CurrentTerm, ReadyToServe, SyncSource\rangle$

RECURSIVE $CreateState(\_, \_)$ init state
$CreateState(len, seq) \triangleq$
    IF $len = 0$ THEN $seq$
     ELSE $CreateState(len - 1, Append(seq, [p \mapsto 0, l \mapsto 0]))$

$LogTerm(i, index) \triangleq$ IF $index = 0$ THEN $0$ ELSE $Oplog[i][index].term$
$LastTerm(i) \triangleq CurrentTerm[i]$

 Is node $i$ ahead of node $j$
$NotBehind(i, j) \triangleq \lor LastTerm(i) > LastTerm(j)$
                      $\lor \land LastTerm(i) = LastTerm(j)$
                         $\land Len(Oplog[i]) \geq Len(Oplog[j])$

$IsMajority(servers) \triangleq Cardinality(servers) * 2 > Cardinality(Server)$

 Return the maximum value from a set, or undefined if the set is empty.
$MaxVal(s) \triangleq$ CHOOSE $x \in s : \forall y \in s : x \geq y$

 commit point
RECURSIVE $AddState(\_, \_, \_)$
$AddState(new, state, index) \triangleq$
    IF $index = 1 \land HLCLt(new, state[1])$
        THEN $\langle new\rangle \circ state$ less than the first
     ELSE IF $index = Len(state) + 1$
        THEN $state \circ \langle new\rangle$
     ELSE IF $HLCLt(new, state[index])$
        THEN $SubSeq(state, 1, index - 1) \circ \langle new\rangle \circ SubSeq(state, index, Len(state))$
     ELSE $AddState(new, state, index + 1)$

RECURSIVE $RemoveState(\_, \_, \_)$
$RemoveState(old, state, index) \triangleq$
    IF $state[index] = old$
        THEN $SubSeq(state, 1, index - 1) \circ SubSeq(state, index + 1, Len(state))$
     ELSE $RemoveState(old, state, index + 1)$

$AdvanceState(new, old, state) \triangleq AddState(new, RemoveState(old, state, 1), 1)$

 clock

$MaxPt \triangleq$ LET $x \triangleq$ CHOOSE $s \in Server : \forall s1 \in Server \setminus \{s\} :$
                              $Pt[s] \geq Pt[s1]$IN $Pt[x]$

$Tick(s) \triangleq Ct' =$ IF $Ct[s].p \geq Pt[s]$ THEN $[Ct$ EXCEPT $![s] = [p \mapsto @.p, l \mapsto @.l + 1]]$
```

```
                                        ELSE  $[Ct \text{ EXCEPT } ![s] = [p \mapsto Pt[s], l \mapsto 0]]$

heartbeat
Only $Primary$ node sends heartbeat once advance pt
$BroadcastHeartbeat(s) \triangleq$
    LET $msg \triangleq [type \mapsto$ "heartbeat", $s \mapsto s, aot \mapsto Ot[s],$
                   $ct \mapsto Ct[s], cp \mapsto Cp[s], term \mapsto CurrentTerm[s]]$
    IN  $ServerMsg' = [x \in Server \mapsto$ IF $x = s$ THEN $ServerMsg[x]$
                                              ELSE $Append(ServerMsg[x], msg)]$

Can node $i$ sync from node $j$?
$CanSyncFrom(i, j) \triangleq$
    $\land Len(Oplog[i]) < Len(Oplog[j])$
    $\land LastTerm(i) = LogTerm(j, Len(Oplog[i]))$

$Oplog$ entries needed to replicate from $j$ to $i$
$ReplicateOplog(i, j) \triangleq$
    LET $len\_i \triangleq Len(Oplog[i])$
        $len\_j \triangleq Len(Oplog[j])$
    IN  IF $i \neq Primary \land len\_i < len\_j$
                    THEN $SubSeq(Oplog[j], len\_i + 1, len\_j)$
                    ELSE $\langle\rangle$

Can node $i$ rollback its $log$ based on $j$'s $log$
$CanRollback(i, j) \triangleq$  $\land Len(Oplog[i]) > 0$
                   $\land Len(Oplog[j]) > 0$
                   $\land CurrentTerm[i] < CurrentTerm[j]$
                   $\land$
                       $\lor Len(Oplog[i]) > Len(Oplog[j])$
                       $\lor \land Len(Oplog[i]) \leq Len(Oplog[j])$
                         $\land CurrentTerm[i] \neq LogTerm(j, Len(Oplog[i]))$

Returns the highest common index between two divergent logs.
If there is no common index between the logs, returns 0.
$RollbackCommonPoint(i, j) \triangleq$
    LET $commonIndices \triangleq \{k \in$ DOMAIN $Oplog[i]$ :
                           $\land k \leq Len(Oplog[j])$
                           $\land Oplog[i][k] = Oplog[j][k]\}$IN
        IF $commonIndices = \{\}$ THEN 0 ELSE $MaxVal(commonIndices)$

$Init$ Part
--------------------------------------------------------------------------------
$InitPrimary \triangleq Primary =$ CHOOSE $s \in Server$ : TRUE
$InitSecondary \triangleq Secondary = Server \setminus \{Primary\}$
$InitOplog \triangleq Oplog = [s \in Server \mapsto \langle\rangle]$
$InitStore \triangleq Store = [n \in Server \cup Client \mapsto [k \in Key \mapsto Nil]]$
$InitCt \triangleq Ct = [n \in Server \cup Client \mapsto [p \mapsto 0, l \mapsto 0]]$
```

```
$InitOt \triangleq Ot = [n \in Server \cup Client \mapsto [p \mapsto 0, l \mapsto 0]]$
$InitServerMsg \triangleq ServerMsg = [s \in Server \mapsto \langle\rangle]$
$InitPt \triangleq Pt = [s \in Server \mapsto 1]$
$InitCp \triangleq Cp = [n \in Server \cup Client \mapsto [p \mapsto 0, l \mapsto 0]]$
$InitCalState \triangleq CalState = CreateState(Cardinality(Server), \langle\rangle)$
                                          create initial state(for calculate)
$InitState \triangleq State = [s \in Server \mapsto [s0 \in Server \mapsto$
                                                  $[p \mapsto 0, l \mapsto 0]]]$
$InitCurrentTerm \triangleq CurrentTerm = [s \in Server \mapsto 0]$
$InitReadyToServe \triangleq ReadyToServe = 0$
$InitSyncSource \triangleq SyncSource = [s \in Server \mapsto Nil]$

$Init \triangleq$
    $\land InitPrimary \land InitSecondary \land InitOplog \land InitStore \land InitCt$
    $\land InitOt \land InitPt \land InitCp \land InitCalState$
    $\land InitServerMsg$
    $\land InitState \land InitCurrentTerm \land InitReadyToServe$
    $\land InitSyncSource$
```

Next *State* Actions

Replication Protocol: possible actions

```
$Stepdown \triangleq$
            $\land ReadyToServe > 0$
            $\land \exists s \in Primary :$
                $\land Primary' = Primary \setminus \{s\}$
                $\land Secondary' = Secondary \cup \{s\}$
            $\land$ UNCHANGED $\langle Oplog, Store, Ct, Ot, ServerMsg,$
                          $Pt, Cp,$
                          $State, CalState, CurrentTerm,$
                          $ReadyToServe, SyncSource\rangle$

 There are majority nodes agree to elect node $i$ to become primary
$ElectPrimary \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists i \in Server : \exists majorNodes \in$ SUBSET $(Server) :$
        $\land \forall j \in majorNodes : \land NotBehind(i, j)$
                                  $\land CurrentTerm[i] \geq CurrentTerm[j]$
        $\land IsMajority(majorNodes)$
        voted nodes for $i$ cannot be primary anymore
        $\land Primary' =$ LET $possiblePrimary \triangleq Primary \setminus majorNodes$
                      IN  $possiblePrimary \cup \{i\}$
        add voted nodes into secondaries
        $\land Secondary' =$ LET $possibleSecondary \triangleq Secondary \cup majorNodes$
                        IN  $possibleSecondary \setminus \{i\}$
        $\land CurrentTerm' = [index \in Server \mapsto$ IF $index \in (majorNodes \cup \{i\})$
```

```
                                    THEN  $CurrentTerm[i] + 1$
                                    ELSE  $CurrentTerm[index]]$
           $\land$ UNCHANGED $\langle Oplog, Store, Ct, Ot, ServerMsg, Pt, Cp, State, CalState,$
                         $CurrentTerm, ReadyToServe, SyncSource\rangle$

$TurnOnReadyToServe \triangleq$
    $\land ReadyToServe = 0$
    $\land \exists s \in Primary :$
        $\land CurrentTerm' = [CurrentTerm$ EXCEPT $![s] = CurrentTerm[s] + 1]$
        $\land ReadyToServe' = ReadyToServe + 1$
    $\land$ UNCHANGED $\langle Primary,\ Secondary, Oplog, Store, Ct, Ot,$
                    $ServerMsg, Pt, Cp,$
                    $State, CalState, SyncSource\rangle$

$AdvanceCp \triangleq$
    $\land ReadyToServe > 0$
    $\land Cp' = [Cp$ EXCEPT $![Primary] = CalState[Cardinality(Server) \div 2 + 1]]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot,$
                    $ServerMsg,\ Pt, CalState,$
                    $State, CurrentTerm, ReadyToServe, SyncSource\rangle$

$ServerTakeHeartbeat \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists s \in Server :$
        $\land Len(ServerMsg[s]) \neq 0$   message channel is not empty
        $\land ServerMsg[s].type\ =$ "heartbeat"
        $\land Ct' = [Ct$ EXCEPT $![s] = HLCMax(Ct[s], ServerMsg[s][1].ct)]$
        $\land State' =$
            LET $SubHbState \triangleq State[s]$
                $hb \triangleq [SubHbState$ EXCEPT $![ServerMsg[s][1].s] =$
                        $ServerMsg[s][1].aot]$
            IN  $[State$ EXCEPT $![s] = hb]$
        $\land CalState' =$ LET $newcal \triangleq$
                          IF $s \in Primary$ primary node: update $CalState$
                           THEN  $AdvanceState(ServerMsg[s][1].aot,$
                                         $State[s][ServerMsg[s][1].s], CalState)$
                           ELSE  $CalState$IN   $newcal$
        $\land Cp' =$ LET $newcp \triangleq$
                      primary node: compute new mcp
                     IF $s \in Primary$ THEN $CalState'[Cardinality(Server) \div 2 + 1]$
                      secondary node: update mcp
                      ELSE  IF $\neg HLCLt(ServerMsg[s][1].cp, Cp[s])$
                               $\land \neg HLCLt(Ot[s], ServerMsg[s][1].cp)$
                      THEN $ServerMsg[s][1].cp$
                      ELSE  $Cp[s]$
                     IN   $[Cp$ EXCEPT $![s] = newcp]$
        ∧ ServerMsg′ = [ServerMsg EXCEPT ![s] = Tail(@)]
        ∧ CurrentTerm′ = [CurrentTerm EXCEPT ![s] = Max(CurrentTerm[s], ServerMsg[s][1].term)]
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ot, Pt,
                    ReadyToServe, SyncSource⟩

ServerTakeUpdatePosition ≜
    ∧ ReadyToServe > 0
    ∧ ∃ s ∈ Server :
        ∧ Len(ServerMsg[s]) ≠ 0   message channel is not empty
        ∧ ServerMsg[s].type  = “update_position”
        ∧ Ct′ = [Ct EXCEPT ![s] = HLCMax(Ct[s], ServerMsg[s][1].ct)]  update ct accordingly
        ∧ State′ =
            LET SubHbState ≜ State[s]
                hb ≜ [SubHbState EXCEPT ![ServerMsg[s][1].s] =
                        ServerMsg[s][1].aot]
            IN  [State EXCEPT ![s] = hb]
        ∧ CalState′ = LET newcal ≜
                        IF s ∈ Primary  primary node: update CalState
                         THEN  AdvanceState(ServerMsg[s][1].aot,
                                    State[s][ServerMsg[s][1].s], CalState)
                         ELSE  CalStateIN   newcal
        ∧ Cp′ = LET newcp ≜
                  primary node: compute new mcp
                  IF s ∈ Primary THEN CalState′[Cardinality(Server) ÷ 2 + 1]
                  secondary node: update mcp
                  ELSE IF ¬HLCLt(ServerMsg[s][1].cp, Cp[s])
                         ∧ ¬HLCLt(Ot[s], ServerMsg[s][1].cp)
                  THEN ServerMsg[s][1].cp
                  ELSE Cp[s]
                IN  [Cp EXCEPT ![s] = newcp]
        ∧ ServerMsg′ = LET newServerMsg ≜ [ServerMsg EXCEPT ![s] = Tail(@)]
                           appendMsg ≜ [type ↦ “update_position”, s ↦ s, aot ↦ ServerMsg[s][1].aot,
                                         ct ↦ ServerMsg[s][1].ct, cp ↦ ServerMsg[s][1].cp, term ↦ Server
                           newMsg ≜ IF s ∈ Primary
                                     THEN newServerMsg  If s is primary, accept the msg, else forward it to its sy
                                     ELSE  [newServerMsg EXCEPT ![SyncSource[s]] = Append(newServer
                        IN   newMsg
        ∧ CurrentTerm′ = [CurrentTerm EXCEPT ![s] = Max(CurrentTerm[s], ServerMsg[s][1].term)]
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ot,
                    Pt,
                    ReadyToServe, SyncSource⟩

 NTPPrimary
NTPSync ≜   simplify NTP protocal
    ∧ ReadyToServe > 0
    ∧ $Pt' = [s \in Server \mapsto MaxPt]$
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot,
                 ServerMsg, Cp,
                 CalState, State, CurrentTerm, ReadyToServe, SyncSource⟩

AdvancePt ≜
    ∧ $ReadyToServe > 0$
    ∧ $\exists s \in Server$ :
        ∧ $s = Primary$                       for simplicity
        ∧ $Pt[s] \leq PtStop$
        ∧ $Pt' = [Pt$ EXCEPT $![s] = @ + 1]$   advance physical time
        ∧ $BroadcastHeartbeat(s)$             broadcast heartbeat periodly
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, State,
                 Cp, CalState, CurrentTerm,
                 ReadyToServe, SyncSource⟩
```

Replication

Idea: replicate*canSyncFromlog*term

SyncSource[*s*]*SyncSource UpdatePosition*

*UpdatePosition*actiontype*updatePosition*

Replicate oplog from node j to node i, and update related structures accordingly

```
Replicate ≜
    ∧ $ReadyToServe > 0$
    ∧ $\exists i, j \in Server$ :
       ∧ $CanSyncFrom(i, j)$   $i$ can sync from $j$ only need not to rollback
       ∧ $i \in Secondary$
       ∧ $ReplicateOplog(i, j) \neq \langle\rangle$
       ∧ $Oplog' = [Oplog$ EXCEPT $![i] = @ \circ ReplicateOplog(i, j)]$
       ∧ $Store' = [Store$ EXCEPT $![i] = Store[j]]$
       ∧ $Ct' = [Ct$ EXCEPT $![i] = HLCMax(Ct[i], Ct[j])]$   update $Ct[i]$
       ∧ $Ot' = [Ot$ EXCEPT $![i] = HLCMax(Ot[i], Ot[j])]$   update $Ot[i]$
       ∧ $Cp' = [Cp$ EXCEPT $![i] = HLCMax(Cp[i], Cp[j])]$   update $Cp[i]$
       ∧ $CurrentTerm' = [CurrentTerm$ EXCEPT $![i] = Max(CurrentTerm[i], CurrentTerm[j])]$   update Curren
       ∧ $State' =$
              LET $SubHbState \triangleq State[i]$
                  $hb \triangleq [SubHbState$ EXCEPT $![j] = Ot[j]]$
              IN  $[State$ EXCEPT $![i] = hb]$   update $j$'s state $i$ knows
       ∧ LET $msg \triangleq [type \mapsto$ "update_position", $s \mapsto i, aot \mapsto Ot'[i], ct \mapsto Ct'[i], cp \mapsto Cp'[i]]$
         IN  $ServerMsg' = [ServerMsg$ EXCEPT $![j] = Append(ServerMsg[j], msg)]$
       ∧ $SyncSource' = [SyncSource$ EXCEPT $![i] = j]$
       ∧ UNCHANGED ⟨Primary, Secondary, Pt, CalState,
                    ReadyToServe⟩
```

Rollback i's oplog and recover it to j's state

Recover to j's state immediately to prevent internal client request

```
$RollbackAndRecover \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists i, j \in Server :$
        $\land i \in Secondary$
        $\land CanRollback(i, j)$
        $\land$ LET $cmp \triangleq RollbackCommonPoint(i, j)$ IN
          LET $commonLog \triangleq SubSeq(Oplog[i], 1, cmp)$
              $appendLog \triangleq SubSeq(Oplog[j], cmp + 1, Len(Oplog[j]))$
          IN  $Oplog' = [Oplog$ EXCEPT $![i] = commonLog \circ appendLog]$
        $\land CurrentTerm' = [CurrentTerm$ EXCEPT $![i] = Max(CurrentTerm[i], CurrentTerm[j])]$   update Curren
        $\land Store' = [Store$ EXCEPT $![i] = Store[j]]$
        $\land Ct' = [Ct$ EXCEPT $![i] = HLCMax(Ct[i], Ct[j])]$   update $Ct[i]$
        $\land Ot' = [Ot$ EXCEPT $![i] = HLCMax(Ot[i], Ot[j])]$   update $Ot[i]$
        $\land Cp' = [Cp$ EXCEPT $![i] = HLCMax(Cp[i], Cp[j])]$   update $Cp[i]$
        $\land State' =$ LET $SubHbState \triangleq State[i]$
                          $hb \triangleq [SubHbState$ EXCEPT $![j] = Ot[j]]$
                      IN  $[State$ EXCEPT $![i] = hb]$   update $j$'s state $i$ knows
        $\land$ LET $msg \triangleq [type \mapsto$ "update_position", $s \mapsto i, aot \mapsto Ot'[i], ct \mapsto Ct'[i], cp \mapsto Cp'[i]]$
          IN  $ServerMsg' = [ServerMsg$ EXCEPT $![j] = Append(ServerMsg[j], msg)]$
        $\land SyncSource' = [SyncSource$ EXCEPT $![i] = j]$
        $\land$ UNCHANGED $\langle Primary, Secondary, Pt, CalState,$
                  $ReadyToServe \rangle$

$ClientRequest \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists s \in Server, k \in Key, v \in Value :$
        $\land s \in Primary$
        $\land Tick(s)$
        $\land Ot' = [Ot$ EXCEPT $![s] = Ct'[s]]$
        $\land Store' = [Store$ EXCEPT $![s][k] = v]$
        $\land Oplog' =$ LET $entry \triangleq [k \mapsto k, v \mapsto v, ot \mapsto Ot'[s], term \mapsto CurrentTerm[s]]$
                           $newLog \triangleq Append(Oplog[s], entry)$
                       IN  $[Oplog$ EXCEPT $![s] = newLog]$
        $\land State' =$ LET $SubHbState \triangleq State[s]$
                          $hb \triangleq [SubHbState$ EXCEPT $![s] = Ot'[s]]$
                      IN  $[State$ EXCEPT $![s] = hb]$
        $\land CalState' = AdvanceState(Ot'[s], Ot[s], CalState)$
        $\land$ UNCHANGED $\langle Primary, Secondary, ServerMsg,$
                     $Pt, Cp,$
                     $CurrentTerm, ReadyToServe, SyncSource \rangle$
```

Action Wrapper

Next state for all configurations

$$\begin{aligned} Next \triangleq\ & \lor\ Replicate \\ & \lor\ AdvancePt \\ & \lor\ ServerTakeHeartbeat \\ & \lor\ ServerTakeUpdatePosition \\ & \lor\ Stepdown \\ & \lor\ RollbackAndRecover \\ & \lor\ TurnOnReadyToServe \\ & \lor\ ElectPrimary \\ & \lor\ ClientRequest \end{aligned}$$

$$Spec \triangleq Init \land \Box[Next]_{vars}$$

\\ $*$ Modification *History*
\\ $*$ Last modified *Mon Apr* 18 22:16:19 *CST* 2022 by *dh*
\\ $*$ Created *Mon Apr* 18 11:38:53 *CST* 2022 by *dh*