```
──────────────────── MODULE TunableMongoDB ────────────────────
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
          OpTimes,               op count at most
          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
          InMsgc,                InMsgc[c]: the channel of messages at client c ∈ Client
          InMsgs,                InMsgc[s]: the channel of messages at server s ∈ Server
          ServerMsg,             ServerMsg[s]: the channel of heartbeat msgs at server s
          BlockedClient,         BlockedClient: Client operations in progress
          BlcokedThread,         BlcokedThread: blocked user thread and content
          OpCount,               OpCount[c]: op count for client c
          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]
          SnapshotTable,         SnapshotTable[s] : snapshot mapping table at server s
          History                History[c]: History sequence at client c
├──────────────────────────────────────────────────────────────┤
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]
```

$Min(x,\ y) \triangleq \text{IF } x < y \text{ THEN } x \text{ ELSE } y$
$Max(x,\ y) \triangleq \text{IF } x > y \text{ THEN } x \text{ ELSE } y$

$vars \triangleq \langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgc,\ InMsgs,\ ServerMsg,\ BlockedClient,\ BlcokedThre$

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

$InitPrimary \triangleq Primary = \text{CHOOSE } s \in Server : \text{TRUE}$
$InitSecondary \triangleq Secondary = Server \setminus \{Primary\}$
$InitOplog \triangleq Oplog = [s \in Server \mapsto \langle\rangle]$
$InitStore \triangleq Store = [n \in Server \cup Client \mapsto [k \in Key \mapsto Nil]]$
$InitCt \triangleq Ct = [n \in Server \cup Client \mapsto [p \mapsto 0,\ l \mapsto 0]]$
$InitOt \triangleq Ot = [n \in Server \cup Client \mapsto [p \mapsto 0,\ l \mapsto 0]]$
$InitInMsgc \triangleq InMsgc = [c \in Client \mapsto \langle\rangle]$
$InitInMsgs \triangleq InMsgs = [s \in Server \mapsto \langle\rangle]$
$InitServerMsg \triangleq ServerMsg = [s \in Server \mapsto \langle\rangle]$
$InitBlockedClient \triangleq BlockedClient = \{\}$
$InitBlcokedThread \triangleq BlcokedThread = [s \in Client \mapsto Nil]$
$InitOpCount \triangleq OpCount = [c \in Client \mapsto OpTimes]$
$InitPt \triangleq Pt = [s \in Server \mapsto 1]$
$InitCp \triangleq Cp = [n \in Server \cup Client \mapsto [p \mapsto 0,\ l \mapsto 0]]$
$InitCalState \triangleq CalState = CreateState(Cardinality(Server),\ \langle\rangle)$
create initial $state(for\ calculate)$
$InitState \triangleq State = [s \in Server \mapsto [s0 \in Server \mapsto$
$\qquad [p \mapsto 0,\ l \mapsto 0]]]$
$InitSnap \triangleq SnapshotTable = [s \in Server \mapsto \langle [ot \mapsto [p \mapsto 0,\ l \mapsto 0],$
$\qquad store \mapsto [k \in Key \mapsto Nil]]\rangle]$
$InitHistory \triangleq History = [c \in Client \mapsto \langle\rangle]$ History operation seq is empty

$Init \triangleq$
$\quad \land InitPrimary \land InitSecondary \land InitOplog \land InitStore \land InitCt$
$\quad \land InitOt \land InitPt \land InitCp \land InitCalState \land InitInMsgc \land InitInMsgs$
$\quad \land InitServerMsg \land InitBlockedClient \land InitBlcokedThread \land InitOpCount$
$\quad \land InitState \land InitSnap \land InitHistory$

---

snapshot
$\text{RECURSIVE } SelectSnapshot\_rec(\_,\ \_,\ \_)$
$SelectSnapshot\_rec(stable,\ cp,\ index) \triangleq$
$\quad \text{IF } HLCLt(cp,\ stable[index].ot) \text{ THEN } stable[index - 1].store$
$\quad \text{ELSE IF } index = Len(stable) \text{ THEN } stable[index].store$
$\quad \text{ELSE } SelectSnapshot\_rec(stable,\ cp,\ index + 1)$

$SelectSnapshot(stable,\ cp) \triangleq SelectSnapshot\_rec(stable,\ cp,\ 1)$

snapshot periodly

```
$Snapshot \triangleq$
    $\land \exists s \in Server :$
          $SnapshotTable' = [SnapshotTable$ EXCEPT $![s] =$
                                   $Append(@, [ot \mapsto Ot[s], store \mapsto Store[s]])]$
                                    create a new snapshot
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc,$
                      $InMsgs, ServerMsg, BlockedClient, BlcokedThread,$
                      $OpCount, Pt, Cp, CalState, State, History \rangle$
```

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

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

$AdvanceCp \triangleq$
    $\land Cp' = [Cp$ EXCEPT $![Primary] = CalState[Cardinality(Server) \div 2 + 1]]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs, ServerMsg, BlockedClient, B$
```

heartbeat

```
$BroadcastHeartbeat(s) \triangleq$
    LET $msg \triangleq [s \mapsto s, aot \mapsto Ot[s], ct \mapsto Ct[s], cp \mapsto Cp[s]]$
    IN  $ServerMsg' = [x \in Server \mapsto$ IF $x = s$ THEN $ServerMsg[x]$
                                                ELSE $Append(ServerMsg[x], msg)]$

$ServerTakeHeartbeat \triangleq$
    $\land \exists s \in Server :$
        $\land Len(ServerMsg[s]) \neq 0$ message channel is not empty
        $\land Ct' = [Ct$ 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 $s = Primary$ 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 $s = Primary$ 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 $![s] = newcp]$
        $\land ServerMsg' = [ServerMsg$ EXCEPT $![s] = Tail(@)]$
     $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ot, InMsgc, InMsgs,$
        $BlockedClient, BlcokedThread, OpCount, Pt, SnapshotTable, History\rangle$
```

clock

```
$UnchangedExPt \triangleq \langle Primary, Secondary, InMsgc, InMsgs, ServerMsg, Oplog, Store,$
                 $Ct, Ot, BlockedClient, OpCount\rangle$
$UnchangedExCt \triangleq \langle Primary, Secondary, InMsgc, InMsgs, ServerMsg, Oplog, Store,$
                 $Pt, Ot, BlockedClient, OpCount\rangle$

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

$NTPSync \triangleq$   simplify $NTP$ protocal
    $\land Pt' = [s \in Server \mapsto MaxPt]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs,$
                $ServerMsg, BlockedClient, BlcokedThread, OpCount, Cp,$
                $CalState, State, SnapshotTable, History\rangle$

$AdvancePt \triangleq$
    $\exists s \in Server :$
        $\land s = Primary$                    for simplicity
        $\land Pt[s] \leq PtStop$
        $\land Pt' = [Pt$ EXCEPT $![s] = @ + 1]$  advance physical time
        $\land BroadcastHeartbeat(s)$          broadcast heartbeat periodly
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs, State,$
      $BlockedClient, BlcokedThread, OpCount, Cp, CalState, SnapshotTable, History\rangle$

$Tick(s) \triangleq Ct' =$ IF $Ct[s].p \geq Pt[s]$ THEN $[Ct$ EXCEPT $![s] = [p \mapsto @.p, l \mapsto @.l + 1]]$
                                ELSE $[Ct$ EXCEPT $![s] = [p \mapsto Pt[s], l \mapsto 0]]$
```

Replicate

```
$ReplicateOplog(s) \triangleq$ LET $len\_s \triangleq Len(Oplog[s])$
                                   $len\_p \triangleq Len(Oplog[Primary])$
                               IN  IF $s \neq Primary \land len\_s < len\_p$
                                   THEN $SubSeq(Oplog[Primary], len\_s + 1, len\_p)$
                                   ELSE $\langle\rangle$

$Replicate \triangleq$
    $\land \exists s \in Secondary :$
        $\land ReplicateOplog(s) \neq \langle\rangle$
        $\land Oplog' = [Oplog$ EXCEPT $![s] = @ \circ ReplicateOplog(s)]$
        $\land Store' = [Store$ EXCEPT $![s] = Store[Primary]]$
        $\land Ct' = [Ct$ EXCEPT $![s] = HLCMax(Ct[s], Ct[Primary])]$   update $Ct[s]$
        $\land Ot' = [Ct$ EXCEPT $![s] = HLCMax(Ot[s], Ot[Primary])]$   update $Ot[s]$
        $\land Cp' = [Cp$ EXCEPT $![s] = HLCMax(Cp[s], Cp[Primary])]$   update $Cp[s]$
        $\land State' =$
            LET $SubHbState \triangleq State[s]$
                $hb \triangleq [SubHbState$ EXCEPT $![Primary] = Ot[Primary]]$
            IN  $[State$ EXCEPT $![s] = hb]$   update primary state $s$ knows
        $\land$ LET $msg \triangleq [s \mapsto s, aot \mapsto Ot'[s], ct \mapsto Ct'[s], cp \mapsto Cp'[s]]$
           IN  $ServerMsg' = [ServerMsg$ EXCEPT $![Primary]$
                             $= Append(ServerMsg[Primary], msg)]$
            we treat *replSetUpdatePosition* as a special heartbeat
        $\land$ UNCHANGED $\langle Primary, Secondary, InMsgc, InMsgs, BlockedClient,$
                 $BlcokedThread, OpCount, Pt, CalState, SnapshotTable, History\rangle$

 server get
$ServerGetReply\_local \triangleq$
    $\land \exists s \in Server :$
        $\land Len(InMsgs[s]) \neq 0$          message channel is not empty
        $\land InMsgs[s][1].op =$ "get"       *msg* type: get
        $\land InMsgs[s][1].rc =$ "local"     Read Concern: local
        $\land Ct' = [Ct$ EXCEPT $![s] = HLCMax(Ct[s], InMsgs[s][1].ct)]$
        $\land InMsgc' = [InMsgc$ EXCEPT $![InMsgs[s][1].c] =$
            $Append(@, [op \mapsto$ "get", $k \mapsto InMsgs[s][1].k, v \mapsto$
                      $Store[s][InMsgs[s][1].k], ct \mapsto Ct'[s], ot \mapsto Ot[s]])]$
            send *msg* to client
        $\land InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ot, ServerMsg,$
                    $BlockedClient, BlcokedThread, OpCount, Pt, Cp,$
                    $CalState, State, SnapshotTable, History\rangle$

$ServerGetReply\_majority \triangleq$
    $\land \exists s \in Server :$
        $\land Len(InMsgs[s]) \neq 0$          message channel is not empty
        $\land InMsgs[s][1].op =$ "get"       *msg* type: get
        $\land InMsgs[s][1].rc =$ "major"     Read Concern: majority
        ∧ $Ct' = [Ct$ EXCEPT $![s] = HLCMax(Ct[s], InMsgs[s][1].ct)]$
        ∧ $InMsgc' = [InMsgc$ EXCEPT $![InMsgs[s][1].c] =$
            $Append(@, [op \mapsto$ "get", $k \mapsto InMsgs[s][1].k, v \mapsto$
                        $SelectSnapshot(SnapshotTable[s], Cp[s])[InMsgs[s][1].k], ct$
                        $\mapsto Ct'[s], ot \mapsto Cp[s]])]$
          send $msg$ to client
        ∧ $InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
    ∧ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ot, ServerMsg,$
                $BlockedClient, BlcokedThread, OpCount, Pt, Cp,$
                $CalState, State, SnapshotTable, History\rangle$

$ServerGetReply\_linearizable\_sleep \triangleq$
    ∧ $\exists s \in Server :$
        ∧ $s = Primary$
        ∧ $Len(InMsgs[s]) \neq 0$
        ∧ $InMsgs[s][1].op =$ "get"
        ∧ $InMsgs[s][1].rc =$ "linea"  Read Concern: linearizable
        ∧ $Tick(s)$  advance cluster time
        ∧ $Oplog' = [Oplog$ EXCEPT $![Primary] =$
                  $Append(@, \langle Nil, Nil, Ct'[s]\rangle)]$
                  append noop operation to $oplog[s]$
        ∧ $Ot' = [Ot$ EXCEPT $![s] = Ct'[s]]$
                advance the last applied operation time $Ot[s]$
        ∧ $State' =$
            LET $SubHbState \triangleq State[s]$
                $hb \triangleq [SubHbState$ EXCEPT $![Primary] = Ot'[Primary]]$
            IN  $[State$ EXCEPT $![s] = hb]$  update primary state
        ∧ $CalState' = AdvanceState(Ot'[Primary], Ot[Primary], CalState)$
        ∧ $InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
        ∧ $BlcokedThread' = [BlcokedThread$ EXCEPT $![InMsgs[s][1].c] =$
                        $[type \mapsto$ "read_linea", $ot \mapsto Ct'[s], s \mapsto s, k$
                        $\mapsto InMsgs[s][1].k, v \mapsto Store[s][InMsgs[s][1].k]]]$
                  add the user thread to $BlcokedThread[c]$
    ∧ UNCHANGED $\langle Primary, Secondary, Store, InMsgc, ServerMsg, BlockedClient,$
                $OpCount, Pt, Cp, SnapshotTable, History\rangle$

$ServerGetReply\_linearizable\_wake \triangleq$
      ∧ $\exists c \in Client :$
        ∧ $BlcokedThread[c] \neq Nil$
        ∧ $BlcokedThread[c].type =$ "read_linea"
        ∧ $\neg HLCLt(Cp[BlcokedThread[c].s], BlcokedThread[c].ot)$  $cp[s] \geq$ target $ot$
        ∧ $InMsgc' = [InMsgc$ EXCEPT $![c] = Append(@, [op \mapsto$ "get", $k$
                  $\mapsto BlcokedThread[c].k, v \mapsto BlcokedThread[c].v, ct$
                  $\mapsto Ct[BlcokedThread[c].s], ot \mapsto BlcokedThread[c].ot])]$
        ∧ $BlcokedThread' = [BlcokedThread$ EXCEPT $![c] = Nil]$  remove blocked state
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs,$
                      $ServerMsg, BlockedClient, OpCount, Pt, Cp,$
                      $CalState, State, SnapshotTable, History\rangle$
```

server put

```
$ServerPutReply\_zero \triangleq$
    $\land \exists s \in Server :$
        $\land s = Primary$
        $\land Len(InMsgs[s]) \neq 0$              message channel is not empty
        $\land InMsgs[s][1].op =$ "put"            $msg$ type: put
        $\land InMsgs[s][1].wc =$ "zero"           Write Concern: 0
        $\land Tick(s)$                            advance cluster time
        $\land Ot' = [Ot$ EXCEPT $![Primary] = Ct'[Primary]]$
                              advance the last applied operation time $Ot[Primary]$
        $\land Store' = [Store$ EXCEPT $![Primary][InMsgs[s][1].k] = InMsgs[s][1].v]$
        $\land Oplog' = [Oplog$ EXCEPT $![Primary] =$
                    $Append(@, \langle InMsgs[s][1].k, InMsgs[s][1].v, Ot'[Primary]\rangle)]$
                    append operation to $oplog[primary]$
        $\land State' =$
            LET $SubHbState \triangleq State[s]$
                $hb \triangleq [SubHbState$ EXCEPT $![Primary] = Ot'[Primary]]$
            IN  $[State$ EXCEPT $![s] = hb]$  update primary state
        $\land CalState' = AdvanceState(Ot'[Primary], Ot[Primary], CalState)$
        $\land InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
    $\land$ UNCHANGED $\langle Primary, Secondary, InMsgc, ServerMsg, BlockedClient,$
                $BlcokedThread, OpCount, Pt, Cp, SnapshotTable, History\rangle$
```

```
$ServerPutReply\_number\_sleep \triangleq$
    $\land \exists s \in Server :$
        $\land s = Primary$
        $\land Len(InMsgs[s]) \neq 0$              message channel is not empty
        $\land InMsgs[s][1].op =$ "put"            $msg$ type: put
        $\land InMsgs[s][1].wc =$ "num"            Write Concern: $num$
        $\land Tick(s)$                            advance cluster time
        $\land Ot' = [Ot$ EXCEPT $![Primary] = Ct'[Primary]]$
                              advance the last applied operation time $Ot[Primary]$
        $\land Store' = [Store$ EXCEPT $![Primary][InMsgs[s][1].k] = InMsgs[s][1].v]$
        $\land Oplog' = [Oplog$ EXCEPT $![Primary] =$
                    $Append(@, \langle InMsgs[s][1].k, InMsgs[s][1].v, Ot'[Primary]\rangle)]$
        $\land State' =$
            LET $SubHbState \triangleq State[s]$
                $hb \triangleq [SubHbState$ EXCEPT $![Primary] = Ot'[Primary]]$
            IN  $[State$ EXCEPT $![s] = hb]$  update primary state
        $\land CalState' = AdvanceState(Ot'[Primary], Ot[Primary], CalState)$
        $\land\ BlcokedThread' = [BlcokedThread\ \text{EXCEPT}\ ![InMsgs[s][1].c] = [type$
            $\mapsto \text{“write\_num”},\ ot \mapsto Ct'[s],\ s \mapsto s,\ numnode \mapsto InMsgs[s][1].num]]$
                        add the user $thHistory$ to $BlcokedThread[c]$
        $\land\ InMsgs' = [InMsgs\ \text{EXCEPT}\ ![s] = Tail(@)]$
    $\land\ \text{UNCHANGED}\ \langle Primary,\ Secondary,\ InMsgc,\ ServerMsg,\ BlockedClient,$
                        $OpCount,\ Pt,\ Cp,\ SnapshotTable,\ History\rangle$

$ServerPutReply\_number\_wake\ \triangleq$
      $\land\ \exists\, c \in Client :$
        $\land\ BlcokedThread[c] \neq Nil$
        $\land\ BlcokedThread[c].type = \text{“write\_num”}$
        $\land\ \neg HLCLt(CalState[Cardinality(Server) - BlcokedThread[c].numnode + 1],$
                        $BlcokedThread[c].ot)$   $CalState[s][n - num + 1] \geq$ target $ot$
        $\land\ InMsgc' = [InMsgc\ \text{EXCEPT}\ ![c] = Append(@,\ [op \mapsto \text{“put”},\ ct$
                          $\mapsto Ct[Primary],\ ot \mapsto BlcokedThread[c].ot])]$
        $\land\ BlcokedThread' = [BlcokedThread\ \text{EXCEPT}\ ![c] = Nil]$
                                                    remove blocked state
      $\land\ \text{UNCHANGED}\ \langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgs,$
                        $ServerMsg,\ BlockedClient,\ OpCount,\ Pt,\ Cp,$
                        $CalState,\ State,\ SnapshotTable,\ History\rangle$

$ServerPutReply\_majority\_sleep\ \triangleq$
    $\land\ \exists\, s \in Server :$
        $\land\ s = Primary$
        $\land\ Len(InMsgs[s]) \neq 0$
        $\land\ InMsgs[s][1].op = \text{“put”}$
        $\land\ InMsgs[s][1].wc = \text{“major”}$
        $\land\ Tick(s)$
        $\land\ Ot' = [Ot\ \text{EXCEPT}\ ![Primary] = Ct'[Primary]]$
        $\land\ Store' = [Store\ \text{EXCEPT}\ ![Primary][InMsgs[s][1].k] = InMsgs[s][1].v]$
        $\land\ Oplog' = [Oplog\ \text{EXCEPT}\ ![Primary] =$
                    $Append(@,\ \langle InMsgs[s][1].k,\ InMsgs[s][1].v,\ Ot'[Primary]\rangle)]$
        $\land\ State' =$
            $\text{LET}\ SubHbState \triangleq State[s]$
                  $hb \triangleq [SubHbState\ \text{EXCEPT}\ ![Primary] = Ot'[Primary]]$
            $\text{IN}\quad [State\ \text{EXCEPT}\ ![s] = hb]$   update primary state
        $\land\ CalState' = AdvanceState(Ot'[Primary],\ Ot[Primary],\ CalState)$
        $\land\ BlcokedThread' = [BlcokedThread\ \text{EXCEPT}\ ![InMsgs[s][1].c] = [type \mapsto \text{“write\_major”},\ ot$
                    $\mapsto Ct'[s],\ s \mapsto s]]$
        $\land\ InMsgs' = [InMsgs\ \text{EXCEPT}\ ![s] = Tail(@)]$
    $\land\ \text{UNCHANGED}\ \langle Primary,\ Secondary,\ InMsgc,\ ServerMsg,\ BlockedClient,\ OpCount,\ Pt,\ Cp,\ SnapshotTab$

$ServerPutReply\_majority\_wake\ \triangleq$
      $\land\ \exists\, c \in Client :$
        $\land\ BlcokedThread[c] \neq Nil$
        ∧ BlcokedThread[c].type = “write_major”
        ∧ ¬HLCLt(Cp[Primary], BlcokedThread[c].ot)
        ∧ InMsgc′ = [InMsgc EXCEPT ![c] =
            Append(@, [op ↦ “put”, ct ↦ Ct[BlcokedThread[c].s]])]
        ∧ BlcokedThread′ = [BlcokedThread EXCEPT ![c] = Nil]
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs, ServerMsg, BlockedClient, OpCoun
```

client get

```
ClientGetRequest_local_primary ≜
    ∧ ∃ k ∈ Key, c ∈ Client \ BlockedClient :
        ∧ InMsgs′ = [InMsgs EXCEPT ![Primary] = Append(@,
            [op ↦ “get”, c ↦ c, rc ↦ “local”, k ↦ k, ct ↦ Ct[c]])]
        ∧ BlockedClient′ = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,
                 BlcokedThread, OpCount, Pt, Cp, CalState,
                 State, SnapshotTable, History⟩

ClientGetRequest_local_secondary ≜
    ∧ ∃ k ∈ Key, c ∈ Client \ BlockedClient, s ∈ Secondary :
        ∧ InMsgs′ = [InMsgs EXCEPT ![s] = Append(@,
            [op ↦ “get”, c ↦ c, rc ↦ “local”, k ↦ k, ct ↦ Ct[c]])]
        ∧ BlockedClient′ = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlcokedThread, OpCount

ClientGetRequest_majority_primary ≜
    ∧ ∃ k ∈ Key, c ∈ Client \ BlockedClient :
        ∧ InMsgs′ = [InMsgs EXCEPT ![Primary] = Append(@,
            [op ↦ “get”, c ↦ c, rc ↦ “major”, k ↦ k, ct ↦ Ct[c]])]
        ∧ BlockedClient′ = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlcokedThread, OpCoun

ClientGetRequest_majority_secondary ≜
    ∧ ∃ k ∈ Key, c ∈ Client \ BlockedClient, s ∈ Secondary :
        ∧ InMsgs′ = [InMsgs EXCEPT ![s] = Append(@,
            [op ↦ “get”, c ↦ c, rc ↦ “major”, k ↦ k, ct ↦ Ct[c]])]
        ∧ BlockedClient′ = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlcokedThread, OpCoun

ClientGetRequest_linearizable ≜
    ∧ ∃ k ∈ Key, c ∈ Client \ BlockedClient :
        ∧ InMsgs′ = [InMsgs EXCEPT ![Primary] = Append(@,
            [op ↦ “get”, c ↦ c, rc ↦ “linea”, k ↦ k, ct ↦ Ct[c]])]
        ∧ BlockedClient′ = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlcokedThread, OpCoun
```

```
client put
$ClientPutRequest\_zero \triangleq$
    $\land \exists k \in Key, v \in Value, c \in Client \setminus BlockedClient :$
        $\land OpCount[c] \neq 0$
        $\land InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] =$
            $Append(@, [op \mapsto \text{“put”}, c \mapsto c, wc \mapsto \text{“zero”}, k$
                        $\mapsto k, v \mapsto v, ct \mapsto Ct[c]])]$
        $\land OpCount' = [OpCount \text{ EXCEPT } ![c] = @ - 1]$
    $\land \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc,$
                        $ServerMsg, BlockedClient, BlcokedThread, Pt, Cp,$
                        $CalState, State, SnapshotTable, History \rangle$

$ClientPutRequest\_number \triangleq$
    $\land \exists k \in Key, v \in Value, c \in Client \setminus BlockedClient, num \in Number :$
        $\land InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] =$
            $Append(@, [op \mapsto \text{“put”}, c \mapsto c, wc \mapsto \text{“num”}, num \mapsto num, k \mapsto k, v \mapsto v, ct \mapsto Ct[c]])]$
        $\land BlockedClient' = BlockedClient \cup \{c\}$
    $\land \text{UNCHANGED } \langle OpCount, Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,$
                        $BlcokedThread, Pt, Cp, CalState, State, SnapshotTable, History \rangle$

$ClientPutRequest\_majority \triangleq$
    $\land \exists k \in Key, v \in Value, c \in Client \setminus BlockedClient :$
        $\land InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] =$
            $Append(@, [op \mapsto \text{“put”}, c \mapsto c, wc \mapsto \text{“major”}, k \mapsto k, v \mapsto v, ct \mapsto Ct[c]])]$
        $\land BlockedClient' = BlockedClient \cup \{c\}$
    $\land \text{UNCHANGED } \langle OpCount, Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlcokedThrea$

$ClientGetResponse \triangleq$
    $\land \exists c \in Client :$
        $\land OpCount[c] \neq 0$              client $c$ has operation times
        $\land Len(InMsgc[c]) \neq 0$          message channel is not empty
        $\land InMsgc[c][1].op = \text{“get”}$     $msg$ type: get
        $\land Store' = [Store \text{ EXCEPT } ![c][InMsgc[c][1].k] = InMsgc[c][1].v]$
            store data
        $\land History' = [History \text{ EXCEPT } ![c] = Append(@, [op$
                        $\mapsto \text{“get”}, ts \mapsto InMsgc[c][1].ot])]$
        $\land InMsgc' = [InMsgc \text{ EXCEPT } ![c] = Tail(@)]$
        $\land BlockedClient' = \text{IF } Len(InMsgc'[c]) = 0$
                                $\text{THEN } BlockedClient \setminus \{c\}$
                                $\text{ELSE } BlockedClient$   remove blocked state
        $\land OpCount' = [OpCount \text{ EXCEPT } ![c] = @ - 1]$
    $\land \text{UNCHANGED } \langle Primary, Secondary, Oplog, Ct, Ot, InMsgs, ServerMsg,$
                        $BlcokedThread, Pt, Cp, CalState, State, SnapshotTable \rangle$

$ClientPutResponse \triangleq$
```

```
$\land \exists\, c \in \mathit{Client}:$
    $\land \mathit{OpCount}[c] \neq 0$                                    client $c$ has operation times
    $\land \mathit{Len}(\mathit{InMsgc}[c]) \neq 0$                        message channel is not empty
    $\land \mathit{InMsgc}[c][1].\mathit{op} = \text{“put”}$               $\mathit{msg}$ type: put
    $\land \mathit{Ct}' = [\mathit{Ct}\ \textsc{except}\ ![c] = \mathit{HLCMax}(@, \mathit{InMsgc}[c][1].\mathit{ct})]$
    $\land \mathit{History}' = [\mathit{History}\ \textsc{except}\ ![c] = \mathit{Append}(@, [\mathit{op}$
                    $\mapsto \text{“put”}, \mathit{ts} \mapsto \mathit{InMsgc}[c][1].\mathit{ot}])]$
    $\land \mathit{InMsgc}' = [\mathit{InMsgc}\ \textsc{except}\ ![c] = \mathit{Tail}(@)]$
    $\land \mathit{BlockedClient}' = \textsc{if}\ \mathit{Len}(\mathit{InMsgc}'[c]) = 0$
                        $\textsc{then}\ \mathit{BlockedClient} \setminus \{c\}$
                        $\textsc{else}\ \mathit{BlockedClient}$      remove blocked state
    $\land \mathit{OpCount}' = [\mathit{OpCount}\ \textsc{except}\ ![c] = @ - 1]$
$\land \textsc{unchanged}\ \langle \mathit{Primary}, \mathit{Secondary}, \mathit{Oplog}, \mathit{Store}, \mathit{Ot}, \mathit{InMsgs}, \mathit{ServerMsg},$
                 $\mathit{BlcokedThread}, \mathit{Pt}, \mathit{Cp}, \mathit{CalState}, \mathit{State}, \mathit{SnapshotTable} \rangle$


$\mathit{ClientGetRequest\_local} \triangleq \lor \mathit{ClientGetRequest\_local\_primary}$
                                 $\lor \mathit{ClientGetRequest\_local\_secondary}$
$\mathit{ClientGetRequest\_majority} \triangleq \lor \mathit{ClientGetRequest\_majority\_primary}$
                                    $\lor \mathit{ClientGetRequest\_majority\_secondary}$

all possible client get actions
$\mathit{ClientGetRequest} \triangleq \lor \mathit{ClientGetRequest\_local}$
                       $\lor \mathit{ClientGetRequest\_majority}$
                       $\lor \mathit{ClientGetRequest\_linearizable}$

all possible clent put actions
$\mathit{ClientPutRequest} \triangleq \lor \mathit{ClientPutRequest\_zero}$
                       $\lor \mathit{ClientPutRequest\_number}$
                       $\lor \mathit{ClientPutRequest\_majority}$

all possible server get actions
$\mathit{ServerGetReply} \triangleq \lor \mathit{ServerGetReply\_local}$
                     $\lor \mathit{ServerGetReply\_majority}$
                     $\lor \mathit{ServerGetReply\_linearizable\_sleep}$
                     $\lor \mathit{ServerGetReply\_linearizable\_wake}$

all possible server put actions
$\mathit{ServerPutReply} \triangleq \lor \mathit{ServerPutReply\_zero}$
                     $\lor \mathit{ServerPutReply\_number\_sleep}$
                     $\lor \mathit{ServerPutReply\_majority\_sleep}$
                     $\lor \mathit{ServerPutReply\_number\_wake}$
                     $\lor \mathit{ServerPutReply\_majority\_wake}$
```

next state for all configurations

$Next \triangleq \lor ClientGetRequest \lor ClientPutRequest$
$\quad \lor ClientGetResponse \lor ClientPutResponse$
$\quad \lor ServerGetReply \lor ServerPutReply$
$\quad \lor Replicate$
$\quad \lor AdvancePt$
$\quad \lor ServerTakeHeartbeat$
$\quad \lor Snapshot$

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

$Next\_Except\_ClientRequset \triangleq \lor ClientGetResponse$
$\quad \lor ClientPutResponse$
$\quad \lor ServerGetReply$
$\quad \lor ServerPutReply$
$\quad \lor Replicate$
$\quad \lor AdvancePt$
$\quad \lor ServerTakeHeartbeat$
$\quad \lor Snapshot$

$ClientRequset\_1 \triangleq \lor ClientPutRequest\_majority$
$\quad \lor ClientGetRequest\_local\_primary$

$ClientRequset\_2 \triangleq \lor ClientPutRequest\_majority$
$\quad \lor ClientGetRequest\_local\_secondary$

$ClientRequset\_3 \triangleq \lor ClientPutRequest\_majority$
$\quad \lor ClientGetRequest\_majority\_primary$

$ClientRequset\_4 \triangleq \lor ClientPutRequest\_majority$
$\quad \lor ClientGetRequest\_majority\_secondary$

$ClientRequset\_5 \triangleq \lor ClientPutRequest\_majority$
$\quad \lor ClientGetRequest\_linearizable$

$ClientRequset\_6 \triangleq \lor ClientPutRequest\_number$
$\quad \lor ClientGetRequest\_local\_primary$

$ClientRequset\_7 \triangleq \lor ClientPutRequest\_number$
$\quad \lor ClientGetRequest\_local\_secondary$

$ClientRequset\_8 \triangleq \lor ClientPutRequest\_number$
$\quad \lor ClientGetRequest\_majority\_primary$

$ClientRequset\_9 \triangleq \lor ClientPutRequest\_number$
$\quad \lor ClientGetRequest\_majority\_secondary$

$$ClientRequset\_10 \triangleq \begin{array}{l} \lor ClientPutRequest\_number \\ \lor ClientGetRequest\_linearizable \end{array}$$

$$Next1 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_1 \end{array}$$

$$Next2 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_2 \end{array}$$

$$Next3 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_3 \end{array}$$

$$Next4 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_4 \end{array}$$

$$Next5 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_5 \end{array}$$

$$Next6 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_6 \end{array}$$

$$Next7 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_7 \end{array}$$

$$Next8 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_8 \end{array}$$

$$Next9 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_9 \end{array}$$

$$Next10 \triangleq \begin{array}{l} \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_10 \end{array}$$

$$\begin{array}{l}
Spec1 \triangleq Init \land \Box[Next1]_{vars} \\
Spec2 \triangleq Init \land \Box[Next2]_{vars} \\
Spec3 \triangleq Init \land \Box[Next3]_{vars} \\
Spec4 \triangleq Init \land \Box[Next4]_{vars} \\
Spec5 \triangleq Init \land \Box[Next5]_{vars} \\
Spec6 \triangleq Init \land \Box[Next6]_{vars} \\
Spec7 \triangleq Init \land \Box[Next7]_{vars} \\
Spec8 \triangleq Init \land \Box[Next8]_{vars} \\
Spec9 \triangleq Init \land \Box[Next9]_{vars} \\
Spec10 \triangleq Init \land \Box[Next10]_{vars}
\end{array}$$

$$
\begin{aligned}
\mathit{MonotonicRead} \triangleq\ & \forall\, c \in \mathit{Client} : \forall\, i, j \in \textsc{domain}\ \mathit{History}[c] : \\
& \quad \land\ i < j \\
& \quad \land\ \mathit{History}[c][i].op = \text{“get”} \\
& \quad \land\ \mathit{History}[c][j].op = \text{“get”} \\
& \quad \Rightarrow \neg \mathit{HLCLt}(\mathit{History}[c][j].ts,\ \mathit{History}[c][i].ts)
\end{aligned}
$$

$$
\begin{aligned}
\mathit{MonotonicWrite} \triangleq\ & \forall\, c \in \mathit{Client} : \forall\, i, j \in \textsc{domain}\ \mathit{History}[c] : \\
& \quad \land\ i < j \\
& \quad \land\ \mathit{History}[c][i].op = \text{“put”} \\
& \quad \land\ \mathit{History}[c][j].op = \text{“put”} \\
& \quad \Rightarrow \neg \mathit{HLCLt}(\mathit{History}[c][j].ts,\ \mathit{History}[c][i].ts)
\end{aligned}
$$

\* Modification *History*
\* Last modified *Mon* May 17 21:30:02 *CST* 2021 by *JYwellin*
\* Created *Fri Mar* 13 15:53:03 *CST* 2020 by *JYwellin*