```
---------------- MODULE TunableMongoDB_RBK ----------------
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
          BlockedThread,       BlockedThread: 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
          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

$HLCMin(x, y) \triangleq$ IF $HLCLt(x, y)$ THEN $x$ ELSE $y$
$HLCMax(x, y) \triangleq$ IF $HLCLt(x, y)$ THEN $y$ ELSE $x$
$HLCType \triangleq [p : Nat, l : Nat]$
$Min(x, y) \triangleq$ IF $x < y$ THEN $x$ ELSE $y$
$Max(x, y) \triangleq$ IF $x > y$ THEN $x$ ELSE $y$

$vars \triangleq \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc,$
          $InMsgs, ServerMsg, BlockedClient, BlockedThread,$
          $OpCount, Pt, Cp, CalState, State, SnapshotTable,$
          $History, CurrentTerm, ReadyToServe, SyncSource\rangle$

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

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

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

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

 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

```
$UnchangedExPt \triangleq \langle Primary, Secondary, InMsgc, InMsgs, ServerMsg, Oplog, Store,$
                    $Ct, Ot, BlockedClient, OpCount \rangle$
$UnchangedExCt \triangleq \langle Primary, Secondary, InMsgc, InMsgs, ServerMsg, Oplog, Store,$
                    $Pt, Ot, BlockedClient, OpCount \rangle$
$MaxPt \triangleq$ LET $x \triangleq$ CHOOSE $s \in Server : \forall s1 \in Server \setminus \{s\} :$
                          $Pt[s] \geq Pt[s1]$ IN $Pt[x]$

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

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.

```
$RollbackCommonPoint(i, j) \triangleq$
    $\text{LET } commonIndices \triangleq \{k \in \text{DOMAIN } Oplog[i] :$
                                  $\land k \leq Len(Oplog[j])$
                                  $\land Oplog[i][k] = Oplog[j][k]\} \text{IN}$
        $\text{IF } commonIndices = \{\} \text{ THEN } 0 \text{ ELSE } MaxVal(commonIndices)$
```

*Init* Part

---

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

$Init \triangleq$
   $\land InitPrimary \land InitSecondary \land InitOplog \land InitStore \land InitCt$
```

```
    $\land InitOt \land InitPt \land InitCp \land InitCalState \land InitInMsgc \land InitInMsgs$
    $\land InitServerMsg \land InitBlockedClient \land InitBlockedThread \land InitOpCount$
    $\land InitState \land InitSnap \land InitHistory \land InitCurrentTerm \quad \land InitReadyToServe$
    $\land InitSyncSource$
```

Next *State* Actions

Replication Protocol: possible actions

---

```
snapshot periodly
$Snapshot \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists s \in Server :$
        $SnapshotTable' = [SnapshotTable \text{ EXCEPT } ![s] =$
                              $Append(@, [ot \mapsto Ot[s], store \mapsto Store[s]])]$
                               create a new snapshot
    $\land \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc,$
                    $InMsgs, ServerMsg, BlockedClient, BlockedThread,$
                    $OpCount, Pt, Cp, CalState, State, History, CurrentTerm,$
                    $ReadyToServe, SyncSource\rangle$

$Stepdown \triangleq$
          $\land ReadyToServe > 0$
          $\land \exists s \in Primary :$
              $\land Primary' = Primary \setminus \{s\}$
              $\land Secondary' = Secondary \cup \{s\}$
          $\land \text{UNCHANGED } \langle Oplog, Store, Ct, Ot, InMsgc, InMsgs, ServerMsg,$
                          $BlockedClient, BlockedThread, OpCount, Pt, Cp,$
                          $State, CalState, SnapshotTable, History, CurrentTerm,$
                          $ReadyToServe, SyncSource\rangle$

There are majority nodes agree to elect node $i$ to become primary
$ElectPrimary(i, majorNodes) \triangleq$
    $\land ReadyToServe > 0$
    $\land \forall j \in majorNodes : \land NotBehind(i, j)$
                            $\land CurrentTerm[i] \geq CurrentTerm[j]$
    $\land IsMajority(majorNodes)$
        voted nodes for $i$ cannot be primary anymore
    $\land Primary' = \text{LET } possiblePrimary \triangleq Primary \setminus majorNodes$
                  $\text{IN } possiblePrimary \cup \{i\}$
        add voted nodes into secondaries
    $\land Secondary' = \text{LET } possibleSecondary \triangleq Secondary \cup majorNodes$
                    $\text{IN } possibleSecondary \setminus \{i\}$
    $\land CurrentTerm' = [index \in Server \mapsto \text{IF } index \in (majorNodes \cup \{i\})$
                                          $\text{THEN } CurrentTerm[i] + 1$
```

```
                                    ELSE $CurrentTerm[index]]$
    $\land$ UNCHANGED $\langle Oplog, Store, Ct, Ot, InMsgc, InMsgs, ServerMsg, BlockedClient,$
                      $BlockedThread, OpCount, Pt, Cp, State, CalState, SnapshotTable,$
                      $History, CurrentTerm, ReadyToServe, SyncSource\rangle$

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

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

$ServerTakeHeartbeat \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists s \in Server :$
        $\land Len(ServerMsg[s]) \neq 0$  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  $[Cp \text{ EXCEPT } ![s] = newcp]$
          $\land\ ServerMsg' = [ServerMsg \text{ EXCEPT } ![s] = Tail(@)]$
          $\land\ CurrentTerm' = [CurrentTerm \text{ EXCEPT } ![s] = Max(CurrentTerm[s], ServerMsg[s][1].term)]$
       $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ot, InMsgc,$
                        $InMsgs, BlockedClient, BlockedThread, OpCount, Pt,$
                        $SnapshotTable, History, ReadyToServe, SyncSource\rangle$

$ServerTakeUpdatePosition \triangleq$
    $\land\ ReadyToServe > 0$
    $\land\ \exists\, s \in Server :$
        $\land\ Len(ServerMsg[s]) \neq 0$   message channel is not empty
        $\land\ ServerMsg[s].type\ = $ "update_position"
        $\land\ Ct' = [Ct \text{ EXCEPT } ![s] = HLCMax(Ct[s], ServerMsg[s][1].ct)]$   update $ct$ accordingly
        $\land\ State' =$
            LET $SubHbState \triangleq State[s]$
                $hb \triangleq [SubHbState \text{ EXCEPT } ![ServerMsg[s][1].s] =$
                        $ServerMsg[s][1].aot]$
            IN  $[State \text{ EXCEPT } ![s] = hb]$
        $\land\ CalState' =$ 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  $[Cp \text{ EXCEPT } ![s] = newcp]$
          $\land\ ServerMsg' =$ LET $newServerMsg \triangleq [ServerMsg \text{ EXCEPT } ![s] = Tail(@)]$
                            $appendMsg \triangleq [type \mapsto$ "update_position", $s \mapsto s,\ aot \mapsto ServerMsg[s][1].aot,$
                                       $ct \mapsto ServerMsg[s][1].ct,\ cp \mapsto ServerMsg[s][1].cp,\ term \mapsto Server$
                            $newMsg \triangleq$ IF $s \in Primary$
                                       THEN $newServerMsg$ If $s$ is primary, accept the $msg$, else forward it to its sy
                                       ELSE $[newServerMsg \text{ EXCEPT } ![SyncSource[s]] = Append(newServer$
                        IN  $newMsg$
          $\land\ CurrentTerm' = [CurrentTerm \text{ EXCEPT } ![s] = Max(CurrentTerm[s], ServerMsg[s][1].term)]$
       $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ot, InMsgc, InMsgs,$
                        $BlockedClient, BlockedThread, OpCount, Pt, SnapshotTable,$
                        $History, ReadyToServe, SyncSource\rangle$

$NTPPrimary$
```

```
$NTPSync \triangleq$   simplify $NTP$ protocal
    $\land\ ReadyToServe > 0$
    $\land\ Pt' = [s \in Server \mapsto MaxPt]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs,$
                   $ServerMsg, BlockedClient, BlockedThread, OpCount, Cp,$
                   $CalState, State, SnapshotTable, History, CurrentTerm, ReadyToServe, SyncSource\rangle$

$AdvancePt \triangleq$
    $\land\ ReadyToServe > 0$
    $\land\ \exists\, s \in Server :$
         $\land\ s = Primary$                          for simplicity
         $\land\ Pt[s] \leq PtStop$
         $\land\ Pt' = [Pt$ EXCEPT $![s] = @ + 1]$     advance physical time
         $\land\ BroadcastHeartbeat(s)$                broadcast heartbeat periodly
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs, State,$
      $BlockedClient, BlockedThread, OpCount, Cp, CalState, SnapshotTable, History, CurrentTerm,$
      $ReadyToServe, SyncSource\rangle$
```

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

```
$Replicate(i, j) \triangleq$
    $\land\ ReadyToServe > 0$
    $\land\ CanSyncFrom(i, j)$   $i$ can sync from $j$ only need not to rollback
    $\land\ i \in Secondary$
    $\land\ ReplicateOplog(i, j) \neq \langle\rangle$
    $\land\ Oplog' = [Oplog$ EXCEPT $![i] = @ \circ ReplicateOplog(i, j)]$
    $\land\ Store' = [Store$ 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 $Cp[i]$
    $\land\ CurrentTerm' = [CurrentTerm$ EXCEPT $![i] = Max(CurrentTerm[i], CurrentTerm[j])]$   update $CurrentTer$
    $\land\ State' =$
           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 $\langle Primary, Secondary, InMsgc, InMsgs, BlockedClient,$
                    $BlockedThread, OpCount, Pt, CalState, SnapshotTable,$
                    $History, ReadyToServe\rangle$
```

```
Rollback i's oplog and recover it to j's state
Recover to j's state immediately to prevent internal client request
$RollbackAndRecover(i, j) \triangleq$
    $\land ReadyToServe > 0$
    $\land i \in Secondary$
    $\land CanRollback(i, j)$
    $\land$ 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 $CurrentTer$
    $\land Store' = [Store$ 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 $\langle Primary, Secondary, InMsgc, InMsgs, BlockedClient,$
                     $BlockedThread, OpCount, Pt, CalState, SnapshotTable,$
                     $History, ReadyToServe\rangle$
```

Tunable Protocol: *Server* Actions

---

Server Get

```
$ServerGetReply\_local\_sleep \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists s \in Server :$
        $\land Len(InMsgs[s]) \neq 0$           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 $![s] = HLCMax(Ct[s], InMsgs[s][1].ct)]$  Update $Ct$ according to $InMsg$
        $\land BlockedThread' = [BlockedThread$ EXCEPT $![InMsgs[s][1].c] =$
                    $[type \mapsto$ "read_local", $s \mapsto s, k \mapsto InMsgs[s][1].k, ot \mapsto InMsgs[s][1].ot]]$
        $\land InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ot, InMsgc, ServerMsg,$
                    $BlockedClient, OpCount, Pt, Cp,$
                    $CalState, State, SnapshotTable, History,$
                    $CurrentTerm, ReadyToServe, SyncSource\rangle$
```

```
$ServerGetReply\_local\_wake \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists c \in Client :$
        $\land BlockedThread[c] \neq Nil$
        $\land BlockedThread[c].type = \text{"read\_local"}$
        $\land \neg HLCLt(Ot[BlockedThread[c].s], BlockedThread[c].ot)$ wait until $Ot[s] \geq$ target $ot$
        $\land InMsgc' = [InMsgc$ EXCEPT $![c] = Append(@, [op \mapsto \text{"get"}, k \mapsto BlockedThread[c].k, v \mapsto$
                        $Store[BlockedThread[c].s][BlockedThread[c].k],$
                        $ct \mapsto Ct[BlockedThread[c].s], ot \mapsto Ot[BlockedThread[c].s]])]$
            send $msg$ to client
        $\land BlockedThread' = [BlockedThread$ EXCEPT $![c] = Nil]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs, ServerMsg,$
                    $BlockedClient, OpCount, Pt, Cp,$
                    $CalState, State, SnapshotTable, History,$
                    $CurrentTerm, ReadyToServe, SyncSource\rangle$

$ServerGetReply\_majority\_sleep \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists s \in Server :$
        $\land Len(InMsgs[s]) \neq 0$            message channel is not empty
        $\land InMsgs[s][1].op = \text{"get"}$        $msg$ type: get
        $\land InMsgs[s][1].rc = \text{"major"}$      Read Concern: majority
        $\land Ct' = [Ct$ EXCEPT $![s] = HLCMax(Ct[s], InMsgs[s][1].ct)]$
        $\land BlockedThread' = [BlockedThread$ EXCEPT $![InMsgs[s][1].c] =$
                            $[type \mapsto \text{"read\_major"}, s \mapsto s, k \mapsto InMsgs[s][1].k, ot \mapsto InMsgs[s][1].ot]]$
        $\land InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ot, InMsgc, ServerMsg,$
                    $BlockedClient, OpCount, Pt, Cp,$
                    $CalState, State, SnapshotTable, History,$
                    $CurrentTerm, ReadyToServe, SyncSource\rangle$

$ServerGetReply\_majority\_wake \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists c \in Client :$
        $\land BlockedThread[c] \neq Nil$
        $\land BlockedThread[c].type = \text{"read\_major"}$
        $\land \neg HLCLt(Ot[BlockedThread[c].s], BlockedThread[c].ot)$ wait until $Ot[s] \geq$ target $ot$
        $\land InMsgc' = [InMsgc$ EXCEPT $![c] = Append(@, [op \mapsto \text{"get"}, k \mapsto BlockedThread[c].k, v \mapsto$
                        $SelectSnapshot(SnapshotTable[BlockedThread[c].s], Cp[BlockedThread[c].s])[BlockedThr$
                        $ct \mapsto Ct[BlockedThread[c].s], ot \mapsto Cp[BlockedThread[c].s]])]$
            send $msg$ to client
        $\land BlockedThread' = [BlockedThread$ EXCEPT $![c] = Nil]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs, ServerMsg,$
                    $BlockedClient, OpCount, Pt, Cp,$
                    $CalState, State, SnapshotTable, History,$
```

```
                $CurrentTerm, ReadyToServe, SyncSource\rangle$

$ServerGetReply\_linearizable\_sleep \triangleq$
    $\land ReadyToServe > 0$
    $\land \exists s \in Server :$
        $\land s = Primary$
        $\land Len(InMsgs[s]) \neq 0$
        $\land InMsgs[s][1].op =$ "get"
        $\land InMsgs[s][1].rc =$ "linea"  Read Concern: linearizable
        $\land Tick(s)$  advance cluster time
        $\land Oplog' = [Oplog$ EXCEPT $![Primary] =$
                    $Append(@, \langle Nil, Nil, Ct'[s]\rangle)]$
                     append noop operation to $oplog[s]$
        $\land Ot' = [Ot$ EXCEPT $![s] = Ct'[s]]$
                    advance the last applied operation time $Ot[s]$
        $\land State' =$
            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 $![s] = Tail(@)]$
        $\land BlockedThread' = [BlockedThread$ 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 $BlockedThread[c]$
    $\land$ UNCHANGED $\langle Primary, Secondary, Store, InMsgc, ServerMsg, BlockedClient,$
                $OpCount, Pt, Cp, SnapshotTable, History,$
                $CurrentTerm, ReadyToServe, SyncSource\rangle$

$ServerGetReply\_linearizable\_wake \triangleq$
      $\land ReadyToServe > 0$
      $\land \exists c \in Client :$
        $\land BlockedThread[c] \neq Nil$
        $\land BlockedThread[c].type =$ "read_linea"
        $\land \neg HLCLt(Cp[BlockedThread[c].s], BlockedThread[c].ot)$  $cp[s] \geq$ target $ot$
        $\land InMsgc' = [InMsgc$ EXCEPT $![c] = Append(@, [op \mapsto$ "get", $k$
                    $\mapsto BlockedThread[c].k, v \mapsto BlockedThread[c].v, ct$
                    $\mapsto Ct[BlockedThread[c].s], ot \mapsto BlockedThread[c].ot])]$
        $\land BlockedThread' = [BlockedThread$ EXCEPT $![c] = Nil]$  remove blocked state
      $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs,$
                  $ServerMsg, BlockedClient, OpCount, Pt, Cp,$
                  $CalState, State, SnapshotTable, History,$
                  $CurrentTerm, ReadyToServe, SyncSource\rangle$
```

Server Put
serveroplog

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

***************************************************************************
*DH* modified: Add $k$ and $v$ message when block thread, and return them when wake
***************************************************************************

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

$ServerPutReply\_number\_wake\ \triangleq$
    $\wedge\ ReadyToServe > 0$
    $\wedge\ \exists\, c \in Client :$
      $\wedge\ BlockedThread[c] \neq Nil$
      $\wedge\ BlockedThread[c].type =$ “write_num”
      $\wedge\ \neg HLCLt(CalState[Cardinality(Server) - BlockedThread[c].numnode + 1],$
                  $BlockedThread[c].ot)$   $CalState[s][n - num + 1] \geq$ target $ot$
      $\wedge\ InMsgc' = [InMsgc$ EXCEPT $![c] = Append(@, [op \mapsto$ “put”, $ct$
                   $\mapsto Ct[Primary],\ ot \mapsto BlockedThread[c].ot,\ k \mapsto BlockedThread[c].k,\ v \mapsto BlockedThrea$
      $\wedge\ BlockedThread' = [BlockedThread$ EXCEPT $![c] = Nil]$
                                              remove blocked state
    $\wedge$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgs,$
                    $ServerMsg,\ BlockedClient,\ OpCount,\ Pt,\ Cp,$
                    $CalState,\ State,\ SnapshotTable,\ History,$
                    $CurrentTerm,\ ReadyToServe,\ SyncSource\rangle$
```

***************************************************************************
*DH* modified: Add $k$ and $v$ message when block thread, and return them when wake
***************************************************************************

```
$ServerPutReply\_majority\_sleep\ \triangleq$
  $\wedge\ ReadyToServe > 0$
  $\wedge\ \exists\, s \in Primary :$
    $\wedge\ Len(InMsgs[s]) \neq 0$
    $\wedge\ InMsgs[s][1].op =$ “put”
    $\wedge\ InMsgs[s][1].wc =$ “major”
    $\wedge\ Tick(s)$
    $\wedge\ Ot' = [Ot$ EXCEPT $![s] = Ct'[s]]$
    $\wedge\ Store' = [Store$ EXCEPT $![s][InMsgs[s][1].k] = InMsgs[s][1].v]$
    $\wedge\ Oplog' =$
        LET $entry \triangleq [k \mapsto InMsgs[s][1].k,\ v \mapsto InMsgs[s][1].v,\ ot \mapsto Ot'[s],\ term \mapsto CurrentTerm[s]]$
            $newLog \triangleq Append(Oplog[s],\ entry)$
        IN  $[Oplog$ EXCEPT $![s] = newLog]$
    $\wedge\ State' =$
        LET $SubHbState \triangleq State[s]$
```

```
            hb ≜ [SubHbState EXCEPT ![s] = Ot′[s]]
          IN   [State EXCEPT ![s] = hb]   update primary state
       ∧ CalState′ = AdvanceState(Ot′[s], Ot[s], CalState)
       ∧ BlockedThread′ = [BlockedThread EXCEPT ![InMsgs[s][1].c] = [type ↦ "write_major", ot
                    ↦ Ot′[s], s ↦ s, k ↦ InMsgs[s][1].k, v ↦ InMsgs[s][1].v]]
       ∧ InMsgs′ = [InMsgs EXCEPT ![s] = Tail(@)]
    ∧ UNCHANGED ⟨Primary, Secondary, InMsgc, ServerMsg, BlockedClient, OpCount,
                    Pt, Cp, SnapshotTable, History, CurrentTerm, ReadyToServe, SyncSource⟩

ServerPutReply_majority_wake ≜
      ∧ ReadyToServe > 0
      ∧ ∃ c ∈ Client :
        ∧ BlockedThread[c] ≠ Nil
        ∧ BlockedThread[c].type = "write_major"
        ∧ ¬HLCLt(Cp[Primary], BlockedThread[c].ot)
        ∧ InMsgc′ = [InMsgc EXCEPT ![c] =
            Append(@, [op ↦ "put", ct ↦ Ct[BlockedThread[c].s], ot ↦ BlockedThread[c].ot,
                      k ↦ BlockedThread[c].k, v ↦ BlockedThread[c].v])]
        ∧ BlockedThread′ = [BlockedThread EXCEPT ![c] = Nil]
      ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs, ServerMsg,
                      BlockedClient, OpCount, Pt, Cp, CalState, State, SnapshotTable,
                      History, CurrentTerm, ReadyToServe, SyncSource⟩
```

Tunable Protocol: *Client* Actions

---

Client Get

```
ClientGetRequest_local_primary ≜
    ∧ ReadyToServe > 0
    ∧ ∃ k ∈ Key, c ∈ Client \ BlockedClient, s ∈ Primary :
        ∧ InMsgs′ = [InMsgs EXCEPT ![s] = Append(@,
            [op ↦ "get", c ↦ c, rc ↦ "local", k ↦ k, ct ↦ Ct[c], ot ↦ Ot[c]])]
        ∧ BlockedClient′ = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,
                    BlockedThread, OpCount, Pt, Cp, CalState,
                    State, SnapshotTable, History,
                    CurrentTerm, ReadyToServe, SyncSource⟩

ClientGetRequest_local_secondary ≜
    ∧ ReadyToServe > 0
    ∧ ∃ k ∈ Key, c ∈ Client \ BlockedClient, s ∈ Secondary :
        ∧ InMsgs′ = [InMsgs EXCEPT ![s] = Append(@,
            [op ↦ "get", c ↦ c, rc ↦ "local", k ↦ k, ct ↦ Ct[c], ot ↦ Ot[c]])]
        ∧ BlockedClient′ = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,
                  $BlockedThread, OpCount, Pt, Cp, CalState, State, SnapshotTable,$
                  $History, CurrentTerm, ReadyToServe, SyncSource\rangle$

$ClientGetRequest\_majority\_primary \triangleq$
    $\land\ ReadyToServe > 0$
    $\land\ \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Primary :$
        $\land\ InMsgs' = [InMsgs\ \text{EXCEPT}\ ![s] = Append(@,$
            $[op \mapsto \text{“get”},\ c \mapsto c,\ rc \mapsto \text{“major”},\ k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
        $\land\ BlockedClient' = BlockedClient \cup \{c\}$
    $\land\ \text{UNCHANGED}\ \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,$
                  $BlockedThread, OpCount, Pt, Cp, CalState, State, SnapshotTable,$
                  $History, CurrentTerm, ReadyToServe, SyncSource\rangle$

$ClientGetRequest\_majority\_secondary \triangleq$
    $\land\ ReadyToServe > 0$
    $\land\ \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Secondary :$
        $\land\ InMsgs' = [InMsgs\ \text{EXCEPT}\ ![s] = Append(@,$
            $[op \mapsto \text{“get”},\ c \mapsto c,\ rc \mapsto \text{“major”},\ k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
        $\land\ BlockedClient' = BlockedClient \cup \{c\}$
    $\land\ \text{UNCHANGED}\ \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,$
                  $BlockedThread, OpCount, Pt, Cp, CalState, State, SnapshotTable,$
                  $History, CurrentTerm, ReadyToServe, SyncSource\rangle$

$ClientGetRequest\_linearizable \triangleq$
    $\land\ ReadyToServe > 0$
    $\land\ \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Primary :$
        $\land\ InMsgs' = [InMsgs\ \text{EXCEPT}\ ![s] = Append(@,$
            $[op \mapsto \text{“get”},\ c \mapsto c,\ rc \mapsto \text{“linea”},\ k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
        $\land\ BlockedClient' = BlockedClient \cup \{c\}$
    $\land\ \text{UNCHANGED}\ \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,$
                  $BlockedThread, OpCount, Pt, Cp, CalState, State, SnapshotTable,$
                  $History, CurrentTerm, ReadyToServe, SyncSource\rangle$
```

Client Put

```
*************************************************************************
```
*DH* modified: change the state of history when write with w:0
```
*************************************************************************
```

```
$ClientPutRequest\_zero \triangleq$
    $\land\ ReadyToServe > 0$
    $\land\ \exists\, k \in Key,\ v \in Value,\ c \in Client \setminus BlockedClient,\ s \in Primary :$
        $\land\ OpCount[c] \neq 0$
        $\land\ InMsgs' = [InMsgs\ \text{EXCEPT}\ ![s] =$
            $Append(@, [op \mapsto \text{“put”},\ c \mapsto c,\ wc \mapsto \text{“zero”},\ k$
                    $\mapsto k,\ v \mapsto v,\ ct \mapsto Ct[c]])]$
        $\land\ OpCount' = [OpCount\ \text{EXCEPT}\ ![c] = @ - 1]$
        $\land\ History' \ = [History\ \text{EXCEPT}\ ![c] = Append(@, [op \mapsto \text{“put”},\ ts \mapsto InMsgc[c][1].ot,\ k \mapsto k,\ v \mapsto v])]$
```

```
  ∧ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,$
              $BlockedClient, BlockedThread, Pt, Cp, CalState, State, SnapshotTable,$
              $CurrentTerm, ReadyToServe, SyncSource\rangle$

$ClientPutRequest\_number \triangleq$
  $\land ReadyToServe > 0$
  $\land \exists k \in Key, v \in Value, c \in Client \setminus BlockedClient, num \in Number, s \in Primary :$
      $\land InMsgs' = [InMsgs$ EXCEPT $![s] =$
          $Append(@, [op \mapsto$ "put", $c \mapsto c, wc \mapsto$ "num", $num \mapsto num, k \mapsto k, v \mapsto v, ct \mapsto Ct[c]])]$
      $\land BlockedClient' = BlockedClient \cup \{c\}$
  ∧ UNCHANGED $\langle OpCount, Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc,$
              $ServerMsg, BlockedThread, Pt, Cp, CalState, State, SnapshotTable,$
              $History, CurrentTerm, ReadyToServe, SyncSource\rangle$

$ClientPutRequest\_majority \triangleq$
  $\land ReadyToServe > 0$
  $\land \exists k \in Key, v \in Value, c \in Client \setminus BlockedClient, s \in Primary :$
      $\land InMsgs' = [InMsgs$ EXCEPT $![s] =$
          $Append(@, [op \mapsto$ "put", $c \mapsto c, wc \mapsto$ "major", $k \mapsto k, v \mapsto v, ct \mapsto Ct[c]])]$
      $\land BlockedClient' = BlockedClient \cup \{c\}$
  ∧ UNCHANGED $\langle OpCount, Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc,$
              $ServerMsg, BlockedThread, Pt, Cp, CalState, State, SnapshotTable,$
              $History, CurrentTerm, ReadyToServe, SyncSource\rangle$
```

****************************************************************************
*DH* modified: record the *k* and *v* in *msg* to history
****************************************************************************

```
$ClientGetResponse \triangleq$
  $\land ReadyToServe > 0$
  $\land \exists c \in Client :$
      $\land OpCount[c] \neq 0$                client $c$ has operation times
      $\land Len(InMsgc[c]) \neq 0$            message channel is not empty
      $\land InMsgc[c][1].op =$ "get"          $msg$ type: get
      $\land Store' = [Store$ EXCEPT $![c][InMsgc[c][1].k] = InMsgc[c][1].v]$
          store data
      $\land History' = [History$ EXCEPT $![c] = Append(@, [op \mapsto$ "get",
          $ts \mapsto InMsgc[c][1].ot, k \mapsto InMsgc[c][1].k, v \mapsto InMsgc[c][1].v])]$

      $\land InMsgc' = [InMsgc$ EXCEPT $![c] = Tail(@)]$
      $\land BlockedClient' =$ IF $Len(InMsgc'[c]) = 0$
                             THEN $BlockedClient \setminus \{c\}$
                             ELSE $BlockedClient$   remove blocked state
      $\land OpCount' = [OpCount$ EXCEPT $![c] = @ - 1]$
  ∧ UNCHANGED $\langle Primary, Secondary, Oplog, Ct, Ot, InMsgs, ServerMsg,$
              $BlockedThread, Pt, Cp, CalState, State, SnapshotTable,$
```

```
                    CurrentTerm, ReadyToServe, SyncSource⟩

***************************************************************************
DH modified: record the k and v in msg to history record ot from server
***************************************************************************
ClientPutResponse ≜
    ∧ ReadyToServe > 0
    ∧ ∃ c ∈ Client :
        ∧ OpCount[c] ≠ 0                  client c has operation times
        ∧ Len(InMsgc[c]) ≠ 0              message channel is not empty
        ∧ InMsgc[c][1].op = "put"         msg type: put
        ∧ Ct' = [Ct EXCEPT ![c] = HLCMax(@, InMsgc[c][1].ct)]
        ∧ Ot' = [Ot EXCEPT ![c] = HLCMax(@, InMsgc[c][1].ot)]  Update Ot to record "my write" ot
        ∧ History' = [History EXCEPT ![c] = Append(@, [op
                ↦ "put", ts ↦ InMsgc[c][1].ot, k ↦ InMsgc[c][1].k, v ↦ InMsgc[c][1].v])]
        ∧ InMsgc' = [InMsgc EXCEPT ![c] = Tail(@)]
        ∧ BlockedClient' = IF Len(InMsgc'[c]) = 0
                             THEN BlockedClient \ {c}
                             ELSE BlockedClient   remove blocked state
        ∧ OpCount' = [OpCount EXCEPT ![c] = @ − 1]
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, InMsgs, ServerMsg,
                  BlockedThread, Pt, Cp, CalState, State, SnapshotTable,
                  CurrentTerm, ReadyToServe, SyncSource⟩

 Action Wrapper
├──────────────────────────────────────────────────────────────────────────┤

ClientGetRequest_local ≜ ∨ ClientGetRequest_local_primary
                         ∨ ClientGetRequest_local_secondary
ClientGetRequest_majority ≜ ∨ ClientGetRequest_majority_primary
                            ∨ ClientGetRequest_majority_secondary

 all possible client get actions
ClientGetRequest ≜ ∨ ClientGetRequest_local
                   ∨ ClientGetRequest_majority
                   ∨ ClientGetRequest_linearizable

 all possible clent put actions
ClientPutRequest ≜ ∨ ClientPutRequest_zero
                   ∨ ClientPutRequest_number
                   ∨ ClientPutRequest_majority

 all possible server get actions
ServerGetReply ≜ ∨ ServerGetReply_local_sleep
                 ∨ ServerGetReply_local_wake
                 ∨ ServerGetReply_majority_sleep
                 ∨ ServerGetReply_majority_wake
```

$\lor \mathit{ServerGetReply\_linearizable\_sleep}$
$\lor \mathit{ServerGetReply\_linearizable\_wake}$

all possible server put actions

$\mathit{ServerPutReply} \triangleq \lor \mathit{ServerPutReply\_zero}$
$\lor \mathit{ServerPutReply\_number\_sleep}$
$\lor \mathit{ServerPutReply\_majority\_sleep}$
$\lor \mathit{ServerPutReply\_number\_wake}$
$\lor \mathit{ServerPutReply\_majority\_wake}$

$\mathit{RollbackOplogAction} \triangleq \exists i, j \in \mathit{Server} : \mathit{RollbackAndRecover}(i, j)$

$\mathit{ReplicateAction} \triangleq \exists i, j \in \mathit{Server} : \mathit{Replicate}(i, j)$

$\mathit{ElectPrimaryAction} \triangleq$
$\exists i \in \mathit{Server} : \exists \mathit{majorNodes} \in \textsc{subset}\ (\mathit{Server}) : \mathit{ElectPrimary}(i, \mathit{majorNodes})$

---

Next state for all configurations

$\mathit{Next} \triangleq \lor \mathit{ClientGetRequest} \lor \mathit{ClientPutRequest}$
$\lor \mathit{ClientGetResponse} \lor \mathit{ClientPutResponse}$
$\lor \mathit{ServerGetReply} \lor \mathit{ServerPutReply}$
$\lor \mathit{ReplicateAction}$
$\lor \mathit{AdvancePt}$
$\lor \mathit{ServerTakeHeartbeat}$
$\lor \mathit{ServerTakeUpdatePosition}$
$\lor \mathit{Snapshot}$
$\lor \mathit{Stepdown}$
$\lor \mathit{RollbackOplogAction}$
$\lor \mathit{TurnOnReadyToServe}$
$\lor \mathit{ElectPrimaryAction}$

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

$\mathit{Next\_Except\_ClientRequset} \triangleq \lor \mathit{ClientGetResponse}$
$\lor \mathit{ClientPutResponse}$
$\lor \mathit{ServerGetReply}$
$\lor \mathit{ServerPutReply}$
$\lor \mathit{ReplicateAction}$
$\lor \mathit{AdvancePt}$
$\lor \mathit{ServerTakeHeartbeat}$
$\lor \mathit{ServerTakeUpdatePosition}$
$\lor \mathit{Snapshot}$
$\lor \mathit{Stepdown}$
$\lor \mathit{RollbackOplogAction}$
$\lor \mathit{TurnOnReadyToServe}$

$\lor \mathit{ElectPrimaryAction}$

$$\mathit{ClientRequset\_1} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_majority} \\ \lor \mathit{ClientGetRequest\_local\_primary} \end{array}$$

$$\mathit{ClientRequset\_2} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_majority} \\ \lor \mathit{ClientGetRequest\_local\_secondary} \end{array}$$

$$\mathit{ClientRequset\_3} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_majority} \\ \lor \mathit{ClientGetRequest\_majority\_primary} \end{array}$$

$$\mathit{ClientRequset\_4} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_majority} \\ \lor \mathit{ClientGetRequest\_majority\_secondary} \end{array}$$

$$\mathit{ClientRequset\_5} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_majority} \\ \lor \mathit{ClientGetRequest\_linearizable} \end{array}$$

$$\mathit{ClientRequset\_6} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_number} \\ \lor \mathit{ClientGetRequest\_local\_primary} \end{array}$$

$$\mathit{ClientRequset\_7} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_number} \\ \lor \mathit{ClientGetRequest\_local\_secondary} \end{array}$$

$$\mathit{ClientRequset\_8} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_number} \\ \lor \mathit{ClientGetRequest\_majority\_primary} \end{array}$$

$$\mathit{ClientRequset\_9} \triangleq \begin{array}{l} \lor \mathit{ClientPutRequest\_number} \\ \lor \mathit{ClientGetRequest\_majority\_secondary} \end{array}$$

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

$$\mathit{Next}1 \triangleq \begin{array}{l} \lor \mathit{Next\_Except\_ClientRequset} \\ \lor \mathit{ClientRequset\_1} \end{array}$$

$$\mathit{Next}2 \triangleq \begin{array}{l} \lor \mathit{Next\_Except\_ClientRequset} \\ \lor \mathit{ClientRequset\_2} \end{array}$$

$$\mathit{Next}3 \triangleq \begin{array}{l} \lor \mathit{Next\_Except\_ClientRequset} \\ \lor \mathit{ClientRequset\_3} \end{array}$$

$$\mathit{Next}4 \triangleq \begin{array}{l} \lor \mathit{Next\_Except\_ClientRequset} \\ \lor \mathit{ClientRequset\_4} \end{array}$$

$$\mathit{Next}5 \triangleq \begin{array}{l} \lor \mathit{Next\_Except\_ClientRequset} \\ \lor \mathit{ClientRequset\_5} \end{array}$$

$$\mathit{Next}6 \triangleq \begin{array}{l} \lor \mathit{Next\_Except\_ClientRequset} \\ \lor \mathit{ClientRequset\_6} \end{array}$$

$$Next7 \triangleq \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_7$$

$$Next8 \triangleq \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_8$$

$$Next9 \triangleq \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_9$$

$$Next10 \triangleq \lor Next\_Except\_ClientRequset \\ \lor ClientRequset\_10$$

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

Causal Specifications

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

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

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

$$\begin{aligned}
WriteFollowRead \triangleq\ & \forall c \in Client : \forall i, j \in \text{DOMAIN } History[c] :\\
& \land \quad i < j\\
& \land \quad History[c][i].op = \text{“get”}\\
& \land \quad History[c][j].op = \text{“put”}
\end{aligned}$$

$$\Rightarrow \neg HLCLt(History[c][j].ts,\ History[c][i].ts)$$

\ ∗ Modification *History*
\ ∗ Last modified Sat *Apr* 09 00:52:42 *CST* 2022 by *dh*
\ ∗ Created *Thu Mar* 31 20:33:19 *CST* 2022 by *dh*