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 $c \in Client$
InMsgs, *InMsgc*[*s*]: the channel of messages at server $s \in Server$
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*

ASSUME $Cardinality(Client) \geq 1$ at least one clinet
ASSUME $Cardinality(Server) \geq 2$ at least one primary and one secondary
ASSUME $Cardinality(Key) \geq 1$ at least one object
ASSUME $Cardinality(Value) \geq 2$ at least two values to update

helpers

$HLCLt(x, y) \triangleq$ 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

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

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

$InitPrimary \triangleq Primary =$ CHOOSE $s \in Server$: TRUE
$InitSecondary \triangleq Secondary = Server \setminus \{Primary\}$
$InitOplog \triangleq Oplog = [s \in Server \mapsto \langle\rangle]$
$InitStore \triangleq Store = [n \in Server \cup Client \mapsto [k \in Key \mapsto Nil]]$
$InitCt \triangleq Ct = [n \in Server \cup Client \mapsto [p \mapsto 0,\ l \mapsto 0]]$
$InitOt \triangleq Ot = [n \in Server \cup Client \mapsto [p \mapsto 0,\ l \mapsto 0]]$
$InitInMsgc \triangleq InMsgc = [c \in Client \mapsto \langle\rangle]$
$InitInMsgs \triangleq InMsgs = [s \in Server \mapsto \langle\rangle]$
$InitServerMsg \triangleq ServerMsg = [s \in Server \mapsto \langle\rangle]$
$InitBlockedClient \triangleq BlockedClient = \{\}$
$InitBlockedThread \triangleq BlockedThread = [s \in Client \mapsto Nil]$
$InitOpCount \triangleq OpCount = [c \in Client \mapsto OpTimes]$
$InitPt \triangleq Pt = [s \in Server \mapsto 1]$
$InitCp \triangleq Cp = [n \in Server \cup Client \mapsto [p \mapsto 0,\ l \mapsto 0]]$
$InitCalState \triangleq CalState = CreateState(Cardinality(Server),\ \langle\rangle)$
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

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

---

snapshot
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 $SelectSnapshot\_rec(stable,\ cp,\ index + 1)$

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

snapshot periodly

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

---

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

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

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

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

$ServerTakeHeartbeat \triangleq$
$\land \exists\, s \in Server :$
$\land Len(ServerMsg[s]) \neq 0$ message channel is not empty
$\land Ct' = [Ct \text{ EXCEPT } ![s] = HLCMax(Ct[s], ServerMsg[s][1].ct)]$
$\land State' =$
$\text{LET } SubHbState \triangleq State[s]$
$hb \triangleq [SubHbState \text{ EXCEPT } ![ServerMsg[s][1].s] =$
$ServerMsg[s][1].aot]$
$\text{IN } [State \text{ EXCEPT } ![s] = hb]$
$\land CalState' = \text{LET } newcal \triangleq$
$\text{IF } s = Primary$ primary node: update *CalState*

```
                    THEN  AdvanceState(ServerMsg[s][1].aot,
                                       State[s][ServerMsg[s][1].s], CalState)
                    ELSE  CalState IN  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' = [ServerMsg EXCEPT ![s] = Tail(@)]
  ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ot, InMsgc, InMsgs,
      BlockedClient, BlockedThread, OpCount, Pt, SnapshotTable, History⟩


 clock

UnchangedExPt ≜ ⟨Primary, Secondary, InMsgc, InMsgs, ServerMsg, Oplog, Store,
                 Ct, Ot, BlockedClient, OpCount⟩
UnchangedExCt ≜ ⟨Primary, Secondary, InMsgc, InMsgs, ServerMsg, Oplog, Store,
                 Pt, Ot, BlockedClient, OpCount⟩

MaxPt ≜ LET x ≜ CHOOSE s ∈ Server : ∀ s1 ∈ Server \ {s} :
                                      Pt[s] ≥ Pt[s1]IN  Pt[x]

NTPSync ≜  simplify NTP protocal
  ∧ Pt' = [s ∈ Server ↦ MaxPt]
  ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs,
               ServerMsg, BlockedClient, BlockedThread, OpCount, Cp,
               CalState, State, SnapshotTable, History⟩

AdvancePt ≜
  ∃ s ∈ Server :
    ∧ s = Primary                          for simplicity
    ∧ Pt[s] ≤ PtStop
    ∧ Pt' = [Pt EXCEPT ![s] = @ + 1]       advance physical time
    ∧ BroadcastHeartbeat(s)                broadcast heartbeat periodly
  ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs, State,
               BlockedClient, BlockedThread, OpCount, Cp, CalState, SnapshotTable, History⟩

Tick(s) ≜  Ct' = IF Ct[s].p ≥ Pt[s] THEN [Ct EXCEPT ![s] = [p ↦ @.p, l ↦ @.l + 1]]
                                    ELSE [Ct EXCEPT ![s] = [p ↦ Pt[s], l ↦ 0]]

 Replicate
```

$$
\begin{array}{l}
ReplicateOplog(s) \triangleq \text{LET } len\_s \triangleq Len(Oplog[s]) \\
\qquad\qquad\qquad\qquad\quad len\_p \triangleq Len(Oplog[Primary]) \\
\qquad\qquad\qquad\quad \text{IN IF } s \neq Primary \wedge len\_s < len\_p \\
\qquad\qquad\qquad\qquad\quad \text{THEN } SubSeq(Oplog[Primary],\ len\_s + 1,\ len\_p) \\
\qquad\qquad\qquad\qquad\quad \text{ELSE } \langle\rangle
\end{array}
$$

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

server get

$$
\begin{array}{l}
ServerGetReply\_local \triangleq \\
\quad \wedge \exists\, s \in Server: \\
\qquad \wedge Len(InMsgs[s]) \neq 0 \qquad \backslash * \text{ message channel is not empty} \\
\qquad \wedge InMsgs[s][1].op = \text{“get”} \backslash * msg \text{ type: get} \\
\qquad \wedge InMsgs[s][1].rc = \text{“local”} \backslash * \text{ Read Concern: local} \\
\qquad \wedge Ct' = [\ Ct \text{ EXCEPT } ![s] = HLCMax(Ct[s],\ InMsgs[s][1].ct)\ ] \\
\qquad \wedge InMsgc' = [\ InMsgc \text{ EXCEPT } ![InMsgs[s][1].c] = \\
\qquad\quad Append(@,\ [op \mapsto \text{“get”},\ k \mapsto InMsgs[s][1].k,\ v \mapsto \\
\qquad\qquad\quad Store[s][InMsgs[s][1].k],\ ct \mapsto Ct'[s],\ ot \mapsto Ot[s]])] \\
\qquad\quad \backslash * \text{ send } msg \text{ to client} \\
\qquad \wedge InMsgs' = [\ InMsgs \text{ EXCEPT } ![s] = Tail(@)\ ] \\
\quad \wedge \text{UNCHANGED } \langle Primary,\ Secondary,\ Oplog,\ Store,\ Ot,\ ServerMsg, \\
\qquad\quad BlockedClient,\ BlockedThread,\ OpCount,\ Pt,\ Cp, \\
\qquad\quad CalState,\ State,\ SnapshotTable,\ History\rangle
\end{array}
$$

$$
\begin{array}{l}
ServerGetReply\_local\_sleep \triangleq \\
\quad \wedge \exists\, s \in Server : \\
\qquad \wedge Len(InMsgs[s]) \neq 0 \quad \text{message channel is not empty} \\
\qquad \wedge InMsgs[s][1].op = \text{“get”} \quad msg \text{ type: get} \\
\qquad \wedge InMsgs[s][1].rc = \text{“local”} \quad \text{Read Concern: local}
\end{array}
$$

$\land\ Ct' = [Ct \text{ EXCEPT } ![s] = HLCMax(Ct[s],\ InMsgs[s][1].ct)]$ Update Ct according to $InMsg$
$\land\ BlockedThread' = [BlockedThread \text{ EXCEPT } ![InMsgs[s][1].c] =$
$[type \mapsto$ "read_local", $s \mapsto s,\ k \mapsto InMsgs[s][1].k,\ ot \mapsto InMsgs[s][1].ot]]$
$\land\ InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Tail(@)]$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ot,\ InMsgc,\ ServerMsg,$
$BlockedClient,\ OpCount,\ Pt,\ Cp,$
$CalState,\ State,\ SnapshotTable,\ History\rangle$

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

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

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

$BlockedClient,\ OpCount,\ Pt,\ Cp,$
$CalState,\ State,\ SnapshotTable,\ History\rangle$

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

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

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 $\langle Primary, Secondary, InMsgc, ServerMsg, BlockedClient,$
$BlockedThread, OpCount, Pt, Cp, SnapshotTable, History\rangle$

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

$ServerPutReply\_number\_sleep \triangleq$
$\land \exists\, s \in Server :$
$\land$ $s = Primary$
$\land$ $Len(InMsgs[s]) \neq 0$ 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
$\land$ $CalState' = AdvanceState(Ot'[Primary], Ot[Primary], CalState)$
$\land$ $BlockedThread' = [BlockedThread$ 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*]
$\land$ $InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
$\land$ UNCHANGED $\langle Primary, Secondary, InMsgc, ServerMsg, BlockedClient,$

$OpCount$, Pt, Cp, $SnapshotTable$, $History\rangle$

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

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

$ServerPutReply\_majority\_sleep \triangleq$
$\land \exists\, s \in Server :$
$\land\ s = Primary$
$\land\ Len(InMsgs[s]) \neq 0$
$\land\ InMsgs[s][1].op =$ "put"
$\land\ InMsgs[s][1].wc =$ "major"
$\land\ Tick(s)$
$\land\ Ot' = [Ot$ EXCEPT $![Primary] = Ct'[Primary]]$
$\land\ Store' = [Store$ 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
$\land\ CalState' = AdvanceState(Ot'[Primary],\ Ot[Primary],\ CalState)$
$\land\ BlockedThread' = [BlockedThread$ EXCEPT $![InMsgs[s][1].c] = [type \mapsto$ "write_major", ot
$\mapsto Ot'[s],\ s \mapsto s,\ k \mapsto InMsgs[s][1].k,\ v \mapsto InMsgs[s][1].v]]$
$\land\ InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
$\land$ UNCHANGED $\langle Primary, Secondary, InMsgc, ServerMsg, BlockedClient, OpCount, Pt, Cp, SnapshotTab$

$ServerPutReply\_majority\_wake \triangleq$
$\land \exists\, c \in Client :$
$\land\ BlockedThread[c] \neq Nil$
$\land\ BlockedThread[c].type =$ "write_major"

$\land\ \neg HLCLt(Cp[Primary],\ BlockedThread[c].ot)$
$\land\ InMsgc' = [InMsgc \text{ EXCEPT } ![c] =$
$\quad Append(@,\ [op \mapsto \text{“put”},\ ct \mapsto Ct[BlockedThread[c].s],\ ot \mapsto BlockedThread[c].ot,\ k \mapsto BlockedThre$
$\land\ BlockedThread' = [BlockedThread \text{ EXCEPT } ![c] = Nil]$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgs,\ ServerMsg,\ BlockedClient,\ OpCoun$

client get

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

$ClientGetRequest\_local\_secondary \triangleq$
$\land\ \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Secondary :$
$\quad \land\ InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@,$
$\qquad [op \mapsto \text{“get”},\ c \mapsto c,\ rc \mapsto \text{“local”},\ k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
$\quad \land\ BlockedClient' = BlockedClient \cup \{c\}$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgc,\ ServerMsg,\ BlockedThread,\ OpCount$

$ClientGetRequest\_majority\_primary \triangleq$
$\land\ \exists\, k \in Key,\ c \in Client \setminus BlockedClient :$
$\quad \land\ InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] = Append(@,$
$\qquad [op \mapsto \text{“get”},\ c \mapsto c,\ rc \mapsto \text{“major”},\ k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
$\quad \land\ BlockedClient' = BlockedClient \cup \{c\}$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgc,\ ServerMsg,\ BlockedThread,\ OpCoun$

$ClientGetRequest\_majority\_secondary \triangleq$
$\land\ \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Secondary :$
$\quad \land\ InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@,$
$\qquad [op \mapsto \text{“get”},\ c \mapsto c,\ rc \mapsto \text{“major”},\ k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
$\quad \land\ BlockedClient' = BlockedClient \cup \{c\}$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgc,\ ServerMsg,\ BlockedThread,\ OpCoun$

$ClientGetRequest\_linearizable \triangleq$
$\land\ \exists\, k \in Key,\ c \in Client \setminus BlockedClient :$
$\quad \land\ InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] = Append(@,$
$\qquad [op \mapsto \text{“get”},\ c \mapsto c,\ rc \mapsto \text{“linea”},\ k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
$\quad \land\ BlockedClient' = BlockedClient \cup \{c\}$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgc,\ ServerMsg,\ BlockedThread,\ OpCoun$

client put

```
 ****************************************************************************
DH modified: change the state of history when write with w:0
 ****************************************************************************
ClientPutRequest_zero ≜
    ∧ ∃ k ∈ Key, v ∈ Value, c ∈ Client \ BlockedClient :
        ∧ OpCount[c] ≠ 0
        ∧ InMsgs' = [InMsgs EXCEPT ![Primary] =
            Append(@, [op ↦ "put", c ↦ c, wc ↦ "zero", k
                       ↦ k, v ↦ v, ct ↦ Ct[c]])]
        ∧ OpCount' = [OpCount EXCEPT ![c] = @ − 1]
        ∧ History'  = [History EXCEPT ![c] = Append(@, [op ↦ "put", ts ↦ InMsgc[c][1].ot, k ↦ k, v ↦ v])]
    ∧ UNCHANGED ⟨Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc,
                 ServerMsg, BlockedClient, BlockedThread, Pt, Cp,
                 CalState, State, SnapshotTable⟩

ClientPutRequest_number ≜
    ∧ ∃ k ∈ Key, v ∈ Value, c ∈ Client \ BlockedClient, num ∈ Number :
        ∧ InMsgs' = [InMsgs EXCEPT ![Primary] =
            Append(@, [op ↦ "put", c ↦ c, wc ↦ "num", num ↦ num, k ↦ k, v ↦ v, ct ↦ Ct[c]])]
        ∧ BlockedClient' = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨OpCount, Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,
                 BlockedThread, Pt, Cp, CalState, State, SnapshotTable, History⟩

ClientPutRequest_majority ≜
    ∧ ∃ k ∈ Key, v ∈ Value, c ∈ Client \ BlockedClient :
        ∧ InMsgs' = [InMsgs EXCEPT ![Primary] =
            Append(@, [op ↦ "put", c ↦ c, wc ↦ "major", k ↦ k, v ↦ v, ct ↦ Ct[c]])]
        ∧ BlockedClient' = BlockedClient ∪ {c}
    ∧ UNCHANGED ⟨OpCount, Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlockedThread

 ****************************************************************************
DH modified: record the k and v in msg to history
 ****************************************************************************
ClientGetResponse ≜
    ∧ ∃ c ∈ Client :
        ∧ OpCount[c] ≠ 0            client c has operation times
        ∧ Len(InMsgc[c]) ≠ 0        message channel is not empty
        ∧ InMsgc[c][1].op = "get"   msg type: get
        ∧ Store' = [Store EXCEPT ![c][InMsgc[c][1].k] = InMsgc[c][1].v]
            store data
        ∧ History' = [History EXCEPT ![c] = Append(@, [op
                       ↦ "get", 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, Ct, Ot, InMsgs, ServerMsg,
               BlockedThread, Pt, Cp, CalState, State, SnapshotTable⟩

 ****************************************************************************
DH modified: record the k and v in msg to history record ot from server
 ****************************************************************************
ClientPutResponse ≜
  ∧ ∃ 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⟩



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
```

$\lor$ *ServerGetReply_majority_sleep*
$\lor$ *ServerGetReply_majority_wake*
$\lor$ *ServerGetReply_linearizable_sleep*
$\lor$ *ServerGetReply_linearizable_wake*

all possible server put actions

$$\begin{aligned} \mathit{ServerPutReply} \triangleq\ & \lor \mathit{ServerPutReply\_zero} \\ & \lor \mathit{ServerPutReply\_number\_sleep} \\ & \lor \mathit{ServerPutReply\_majority\_sleep} \\ & \lor \mathit{ServerPutReply\_number\_wake} \\ & \lor \mathit{ServerPutReply\_majority\_wake} \end{aligned}$$

---

next state for all configurations

$$\begin{aligned} \mathit{Next} \triangleq\ & \lor \mathit{ClientGetRequest} \lor \mathit{ClientPutRequest} \\ & \lor \mathit{ClientGetResponse} \lor \mathit{ClientPutResponse} \\ & \lor \mathit{ServerGetReply} \lor \mathit{ServerPutReply} \\ & \lor \mathit{Replicate} \\ & \lor \mathit{AdvancePt} \\ & \lor \mathit{ServerTakeHeartbeat} \\ & \lor \mathit{Snapshot} \end{aligned}$$

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

$$\begin{aligned} \mathit{Next\_Except\_ClientRequset} \triangleq\ & \lor \mathit{ClientGetResponse} \\ & \lor \mathit{ClientPutResponse} \\ & \lor \mathit{ServerGetReply} \\ & \lor \mathit{ServerPutReply} \\ & \lor \mathit{Replicate} \\ & \lor \mathit{AdvancePt} \\ & \lor \mathit{ServerTakeHeartbeat} \\ & \lor \mathit{Snapshot} \end{aligned}$$

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

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

$$\begin{aligned} \mathit{ClientRequset\_3} \triangleq\ & \lor \mathit{ClientPutRequest\_majority} \\ & \lor \mathit{ClientGetRequest\_majority\_primary} \end{aligned}$$

$$\begin{aligned} \mathit{ClientRequset\_4} \triangleq\ & \lor \mathit{ClientPutRequest\_majority} \\ & \lor \mathit{ClientGetRequest\_majority\_secondary} \end{aligned}$$

$$\mathit{ClientRequset\_5} \triangleq \lor \mathit{ClientPutRequest\_majority}$$

$$\lor\ ClientGetRequest\_linearizable$$

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

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

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

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

$$ClientRequset\_10 \triangleq \lor\ ClientPutRequest\_number \lor\ ClientGetRequest\_linearizable$$

$$Next1 \triangleq \lor\ Next\_Except\_ClientRequset \lor\ ClientRequset\_1$$

$$Next2 \triangleq \lor\ Next\_Except\_ClientRequset \lor\ ClientRequset\_2$$

$$Next3 \triangleq \lor\ Next\_Except\_ClientRequset \lor\ ClientRequset\_3$$

$$Next4 \triangleq \lor\ Next\_Except\_ClientRequset \lor\ ClientRequset\_4$$

$$Next5 \triangleq \lor\ Next\_Except\_ClientRequset \lor\ ClientRequset\_5$$

$$Next6 \triangleq \lor\ Next\_Except\_ClientRequset \lor\ ClientRequset\_6$$

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

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

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

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

$$Spec1 \triangleq Init \land \Box[Next1]_{vars}$$
$$Spec2 \triangleq Init \land \Box[Next2]_{vars}$$
$$Spec3 \triangleq Init \land \Box[Next3]_{vars}$$

$Spec4 \triangleq Init \land \Box[Next4]_{vars}$
$Spec5 \triangleq Init \land \Box[Next5]_{vars}$
$Spec6 \triangleq Init \land \Box[Next6]_{vars}$
$Spec7 \triangleq Init \land \Box[Next7]_{vars}$
$Spec8 \triangleq Init \land \Box[Next8]_{vars}$
$Spec9 \triangleq Init \land \Box[Next9]_{vars}$
$Spec10 \triangleq Init \land \Box[Next10]_{vars}$

Idea:*Primary*check

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

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

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

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

\ ∗ Modification *History*
\ ∗ Last modified *Thu Mar* 31 18:02:24 *CST* 2022 by *dh*
\ ∗ Created *Wed Mar* 16 11:44:03 *CST* 2022 by *dh*