
MODULE *TunableMongoDB*

EXTENDS *Naturals, FiniteSets, Sequences, TLC*

constants and variables

CONSTANTS	<i>Client, Server,</i>	the set of clients and servers
	<i>Key, Value,</i>	the set of keys and values
	<i>Nil,</i>	model value, place holder
	<i>OpTimes,</i>	<i>op</i> count at most
	<i>PtStop,</i>	max physical time
	<i>Number</i>	<i>writeConcern</i> number
VARIABLES	<i>Primary,</i>	Primary node
	<i>Secondary,</i>	secondary nodes
	<i>Oplog,</i>	<i>oplog[s]</i> : <i>oplog</i> at <i>server[s]</i>
	<i>Store,</i>	<i>store[s]</i> : data stored at <i>server[s]</i>
	<i>Ct,</i>	<i>Ct[s]</i> : cluster time at node <i>s</i>
	<i>Ot,</i>	<i>Ot[s]</i> : the last applied operation time at server <i>s</i>
	<i>InMsgc,</i>	<i>InMsgc[c]</i> : the channel of messages at client <i>c</i> \in <i>Client</i>
	<i>InMsgs,</i>	<i>InMsgs[s]</i> : the channel of messages at server <i>s</i> \in <i>Server</i>
	<i>ServerMsg,</i>	<i>ServerMsg[s]</i> : the channel of heartbeat msgs at server <i>s</i>
	<i>BlockedClient,</i>	<i>BlockedClient</i> : <i>Client</i> operations in progress
	<i>BlockedThread,</i>	<i>BlockedThread</i> : blocked user thread and content
	<i>OpCount,</i>	<i>OpCount[c]</i> : <i>op</i> count for client <i>c</i>
	<i>Pt,</i>	<i>Pt[s]</i> : physical time at server <i>s</i>
	<i>Cp,</i>	<i>Cp[s]</i> : majority commit point at server <i>s</i>
	<i>State,</i>	<i>State[s]</i> : the latest <i>Ot</i> of all servers that server <i>s</i> knows
	<i>CalState,</i>	<i>CalState</i> : sorted <i>State[Primary]</i>
	<i>SnapshotTable,</i>	<i>SnapshotTable[s]</i> : snapshot mapping table at server <i>s</i>
	<i>History</i>	<i>History[c]</i> : <i>History</i> sequence at client <i>c</i>

ASSUME $\text{Cardinality}(\text{Client}) \geq 1$ at least one client

ASSUME $\text{Cardinality}(\text{Server}) \geq 2$ at least one primary and one secondary

ASSUME $\text{Cardinality}(\text{Key}) \geq 1$ at least one object

ASSUME $\text{Cardinality}(\text{Value}) \geq 2$ at least two values to update

helpers

$\text{HLCLt}(x, y) \triangleq$ 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

$\text{HLCMin}(x, y) \triangleq$ IF $\text{HLCLt}(x, y)$ THEN x ELSE y

$\text{HLCMax}(x, y) \triangleq$ IF $\text{HLCLt}(x, y)$ THEN y ELSE x

$\text{HLCType} \triangleq [p : \text{Nat}, l : \text{Nat}]$

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

RECURSIVE *CreateState*($-, -$) init state

$$InitPrimary \triangleq Primary = \text{CHOOSE } s \in Server : \text{TRUE}$$
$$InitOplog \stackrel{\Delta}{=} Oplog = [s \in Server \mapsto \langle \rangle]$$
$$InitCt \triangleq Ct = [n \in Server \cup Client \mapsto [p \mapsto 0, l \mapsto 0]]$$
$$InitInMsgc \triangleq InMsgc = [c \in Client \mapsto \langle \rangle]$$
$$InitServerMsg \triangleq ServerMsg = [s \in Server \mapsto \langle \rangle]$$
$$InitBlockedThread \equiv BlockedThread = [s \in Client \mapsto Nil]$$
$$InitPt \triangleq Pt = [s \in Server \mapsto 1]$$
$$InitCalState \stackrel{\Delta}{=} CalState = CreateState(Cardinality(Server), \langle \rangle)$$
$$InitState \stackrel{\Delta}{=} State = [s \in Server \mapsto [s0 \in Server \mapsto [$$
$$InitSnap \stackrel{\Delta}{=} SnapshotTable = [s \in Server \mapsto \langle [ot \mapsto [p \mapsto 0, l \mapsto 0], \\ store \mapsto [k \in Key \mapsto Nil]] \rangle]$$
$$Init \triangleq$$
$$\wedge \textit{InitOt} \wedge \textit{InitPt} \wedge \textit{InitCp} \wedge \textit{InitCalState} \wedge \textit{InitInMsgc} \wedge \textit{InitInMsgs}$$
$$\wedge \textit{InitState} \wedge \textit{InitSnap} \wedge \textit{InitHistory}$$
RECURSIVE *SelectSnapshot_rec*(-, -, -)

IF $HLCLt(cp, stable[index].ot)$ THEN $stable[index - 1].store$

```
ELSE SelectSnapshot_rec(stable, cp, index + 1)
```

2

snapshot periodically

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

commit point

RECURSIVE $AddState(-, -, -)$
 $AddState(new, state, index) \triangleq$ IF $index = 1 \wedge HLCLt(new, state[1])$ THEN $\langle new \rangle \circ state$ $\text{less than the first}$
 ELSE IF $index = Len(state) + 1$ THEN $state \circ \langle new \rangle$
 ELSE IF $HLCLt(new, state[index])$ THEN $SubSeq(state, 1, index - 1) \circ \langle new \rangle$
 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 $RemoveState(old, state, index + 1)$
 $AdvanceState(new, old, state) \triangleq AddState(new, RemoveState(old, state, 1), 1)$
 $AdvanceCp \triangleq$
 $\wedge Cp' = [Cp \text{ EXCEPT } ![Primary] = CalState[Cardinality(Server) \div 2 + 1]]$
 $\wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs, ServerMsg, BlockedClient, L$

heartbeat

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

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

THEN $AdvanceState(ServerMsg[s][1].aot,$
 $State[s][ServerMsg[s][1].s], CalState)$
 ELSE $CalState$ IN $newcal$
 $\wedge Cp' = LET newcp \triangleq$
 $\quad primary\ node: compute\ new\ mcp$
 IF $s = Primary$ THEN $CalState'[Cardinality(Server) \div 2 + 1]$
 $\quad secondary\ node: update\ mcp$
 ELSE IF $\neg HLClt(ServerMsg[s][1].cp, Cp[s])$
 $\quad \wedge \neg HLClt(Ot[s], ServerMsg[s][1].cp)$
 THEN $ServerMsg[s][1].cp$
 ELSE $Cp[s]$
 IN $[Cp\ EXCEPT ![s] = newcp]$
 $\wedge ServerMsg' = [ServerMsg\ EXCEPT ![s] = Tail(@)]$
 $\wedge UNCHANGED \langle Primary, Secondary, Oplog, Store, Ot, InMsgc, InMsgs,$
 $BlockedClient, BlockedThread, OpCount, Pt, SnapshotTable, History \rangle$

clock

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

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

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

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

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

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

Replicate

$$\begin{aligned}
\text{ReplicateOplog}(s) &\triangleq \text{LET } \text{len_s} \triangleq \text{Len}(\text{Oplog}[s]) \\
&\quad \text{len_p} \triangleq \text{Len}(\text{Oplog}[\text{Primary}]) \\
&\quad \text{IN IF } s \neq \text{Primary} \wedge \text{len_s} < \text{len_p} \\
&\quad \quad \text{THEN } \text{SubSeq}(\text{Oplog}[\text{Primary}], \text{len_s} + 1, \text{len_p}) \\
&\quad \quad \text{ELSE } \langle \rangle
\end{aligned}$$

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

server get

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

$$\begin{aligned}
\text{ServerGetReply_local_sleep} &\triangleq \\
&\wedge \exists s \in \text{Server} : \\
&\quad \wedge \text{Len}(\text{InMsgs}[s]) \neq 0 \quad \text{message channel is not empty} \\
&\quad \wedge \text{InMsgs}[s][1].\text{op} = \text{"get"} \quad \text{msg type: get} \\
&\quad \wedge \text{InMsgs}[s][1].\text{rc} = \text{"local"} \quad \text{Read Concern: local}
\end{aligned}$$

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

$ServerGetReply_local_wake \triangleq$

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

$ServerGetReply_majority_sleep \triangleq$

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

$ServerGetReply_majority_wake \triangleq$

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

BlockedClient, OpCount, Pt, Cp,
CalState, State, SnapshotTable, History

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

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

server put

ServerPutReply_zero \triangleq
 $\wedge \exists s \in \text{Server} :$
 $\wedge s = \text{Primary}$

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

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

$\text{ServerPutReply_number_sleep} \triangleq$

$$\begin{aligned}
& \wedge \exists s \in \text{Server} : \\
& \quad \wedge s = \text{Primary} \\
& \quad \wedge \text{Len}(\text{InMsgs}[s]) \neq 0 && \text{message channel is not empty} \\
& \quad \wedge \text{InMsgs}[s][1].\text{op} = \text{"put"} && \text{msg type: put} \\
& \quad \wedge \text{InMsgs}[s][1].\text{wc} = \text{"num"} && \text{Write Concern: num} \\
& \quad \wedge \text{Tick}(s) && \text{advance cluster time} \\
& \quad \wedge \text{Ot}' = [\text{Ot} \text{ EXCEPT } ![\text{Primary}] = \text{Ct}'[\text{Primary}]] \\
& \quad && \text{advance the last applied operation time } \text{Ot}[\text{Primary}] \\
& \quad \wedge \text{Store}' = [\text{Store} \text{ EXCEPT } ![\text{Primary}][\text{InMsgs}[s][1].k] = \text{InMsgs}[s][1].v] \\
& \quad \wedge \text{Oplog}' = [\text{Oplog} \text{ EXCEPT } ![\text{Primary}] = \\
& \quad \quad \text{Append}(\text{@}, (\text{InMsgs}[s][1].k, \text{InMsgs}[s][1].v, \text{Ot}'[\text{Primary}]))) \\
& \quad \wedge \text{State}' = \\
& \quad \quad \text{LET } \text{SubHbState} \triangleq \text{State}[s] \\
& \quad \quad \quad \text{hb} \triangleq [\text{SubHbState} \text{ EXCEPT } ![\text{Primary}] = \text{Ot}'[\text{Primary}]] \\
& \quad \quad \text{IN } [\text{State} \text{ EXCEPT } ![s] = \text{hb}] \text{ update primary state} \\
& \quad \wedge \text{CalState}' = \text{AdvanceState}(\text{Ot}'[\text{Primary}], \text{Ot}[\text{Primary}], \text{CalState}) \\
& \quad \wedge \text{BlockedThread}' = [\text{BlockedThread} \text{ EXCEPT } ![\text{InMsgs}[s][1].c] = [\text{type} \\
& \quad \quad \mapsto \text{"write_num"}, \text{ot} \mapsto \text{Ot}'[s], s \mapsto s, \text{numnode} \mapsto \text{InMsgs}[s][1].\text{num}, \\
& \quad \quad k \mapsto \text{InMsgs}[s][1].k, v \mapsto \text{InMsgs}[s][1].v]] \\
& \quad \quad \text{add the user thHistory to BlockedThread[c]} \\
& \quad \wedge \text{InMsgs}' = [\text{InMsgs} \text{ EXCEPT } ![s] = \text{Tail}(\text{@})] \\
& \quad \wedge \text{UNCHANGED } \langle \text{Primary}, \text{Secondary}, \text{InMsgc}, \text{ServerMsg}, \text{BlockedClient},
\end{aligned}$$

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

$ServerPutReply_number_wake \triangleq$

$\wedge \exists c \in Client :$

$\wedge BlockedThread[c] \neq Nil$

$\wedge BlockedThread[c].type = \text{"write_num"}$

$\wedge \neg HLClt(CalState[Cardinality(Server) - BlockedThread[c].numnode + 1],$
 $BlockedThread[c].ot) \quad CalState[s][n - num + 1] \geq target\ ot$

$\wedge InMsgc' = [InMsgc \text{ EXCEPT } ![c] = Append(@, [op \mapsto \text{"put"}, ct$
 $\mapsto Ct[Primary], ot \mapsto BlockedThread[c].ot, k \mapsto BlockedThread[c].k, v \mapsto BlockedThread[c].v]$

$\wedge BlockedThread' = [BlockedThread \text{ EXCEPT } ![c] = Nil]$

$\text{remove blocked state}$

$\wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs,$
 $ServerMsg, BlockedClient, OpCount, Pt, Cp,$
 $CalState, State, SnapshotTable, History \rangle$

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

$ServerPutReply_majority_sleep \triangleq$

$\wedge \exists s \in Server :$

$\wedge s = Primary$

$\wedge Len(InMsgs[s]) \neq 0$

$\wedge InMsgs[s][1].op = \text{"put"}$

$\wedge InMsgs[s][1].wc = \text{"major"}$

$\wedge Tick(s)$

$\wedge Ot' = [Ot \text{ EXCEPT } ![Primary] = Ct'[Primary]]$

$\wedge Store' = [Store \text{ EXCEPT } ![Primary][InMsgs[s][1].k] = InMsgs[s][1].v]$

$\wedge Oplog' = [Oplog \text{ EXCEPT } ![Primary] =$
 $Append(@, \langle InMsgs[s][1].k, InMsgs[s][1].v, Ot'[Primary] \rangle)]$

$\wedge State' =$

$LET\ SubHbState \triangleq State[s]$

$hb \triangleq [SubHbState \text{ EXCEPT } ![Primary] = Ot'[Primary]]$

$IN\ [State \text{ EXCEPT } ![s] = hb] \quad \text{update primary state}$

$\wedge CalState' = AdvanceState(Ot'[Primary], Ot[Primary], CalState)$

$\wedge BlockedThread' = [BlockedThread \text{ EXCEPT } ![InMsgs[s][1].c] = [type \mapsto \text{"write_major"}, ot$
 $\mapsto Ot'[s], s \mapsto s, k \mapsto InMsgs[s][1].k, v \mapsto InMsgs[s][1].v]]$

$\wedge InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Tail(@)]$

$\wedge \text{UNCHANGED } \langle Primary, Secondary, InMsgc, ServerMsg, BlockedClient, OpCount, Pt, Cp, SnapshotTable, History \rangle$

$ServerPutReply_majority_wake \triangleq$

$\wedge \exists c \in Client :$

$\wedge BlockedThread[c] \neq Nil$

$\wedge BlockedThread[c].type = \text{"write_major"}$

$$\begin{aligned}
& \wedge \neg HLCLt(Cp[Primary], BlockedThread[c].ot) \\
& \wedge InMsgc' = [InMsgc \text{ EXCEPT } ![c] = \\
& \quad Append(@, [op \mapsto \text{"put"}, ct \mapsto Ct[BlockedThread[c].s], ot \mapsto BlockedThread[c].ot, k \mapsto BlockedThread[c].k]) \\
& \wedge BlockedThread' = [BlockedThread \text{ EXCEPT } ![c] = Nil] \\
& \wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs, ServerMsg, BlockedClient, OpCount \rangle
\end{aligned}$$

client get

$$\begin{aligned}
ClientGetRequest_local_primary & \triangleq \\
& \wedge \exists k \in Key, c \in Client \setminus BlockedClient : \\
& \quad \wedge InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] = Append(@, \\
& \quad \quad [op \mapsto \text{"get"}, c \mapsto c, rc \mapsto \text{"local"}, k \mapsto k, ct \mapsto Ct[c], ot \mapsto Ot[c]])] \\
& \quad \wedge BlockedClient' = BlockedClient \cup \{c\} \\
& \wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, \\
& \quad BlockedThread, OpCount, Pt, Cp, CalState, \\
& \quad State, SnapshotTable, History \rangle
\end{aligned}$$

$$\begin{aligned}
ClientGetRequest_local_secondary & \triangleq \\
& \wedge \exists k \in Key, c \in Client \setminus BlockedClient, s \in Secondary : \\
& \quad \wedge InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@, \\
& \quad \quad [op \mapsto \text{"get"}, c \mapsto c, rc \mapsto \text{"local"}, k \mapsto k, ct \mapsto Ct[c], ot \mapsto Ot[c]])] \\
& \quad \wedge BlockedClient' = BlockedClient \cup \{c\} \\
& \wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlockedThread, OpCount \rangle
\end{aligned}$$

$$\begin{aligned}
ClientGetRequest_majority_primary & \triangleq \\
& \wedge \exists k \in Key, c \in Client \setminus BlockedClient : \\
& \quad \wedge InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] = Append(@, \\
& \quad \quad [op \mapsto \text{"get"}, c \mapsto c, rc \mapsto \text{"major"}, k \mapsto k, ct \mapsto Ct[c], ot \mapsto Ot[c]])] \\
& \quad \wedge BlockedClient' = BlockedClient \cup \{c\} \\
& \wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlockedThread, OpCount \rangle
\end{aligned}$$

$$\begin{aligned}
ClientGetRequest_majority_secondary & \triangleq \\
& \wedge \exists k \in Key, c \in Client \setminus BlockedClient, s \in Secondary : \\
& \quad \wedge InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@, \\
& \quad \quad [op \mapsto \text{"get"}, c \mapsto c, rc \mapsto \text{"major"}, k \mapsto k, ct \mapsto Ct[c], ot \mapsto Ot[c]])] \\
& \quad \wedge BlockedClient' = BlockedClient \cup \{c\} \\
& \wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlockedThread, OpCount \rangle
\end{aligned}$$

$$\begin{aligned}
ClientGetRequest_linearizable & \triangleq \\
& \wedge \exists k \in Key, c \in Client \setminus BlockedClient : \\
& \quad \wedge InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] = Append(@, \\
& \quad \quad [op \mapsto \text{"get"}, c \mapsto c, rc \mapsto \text{"linea"}, k \mapsto k, ct \mapsto Ct[c], ot \mapsto Ot[c]])] \\
& \quad \wedge BlockedClient' = BlockedClient \cup \{c\} \\
& \wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlockedThread, OpCount \rangle
\end{aligned}$$

client put

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

$ClientPutRequest_zero \triangleq$
 $\wedge \exists k \in Key, v \in Value, c \in Client \setminus BlockedClient :$
 $\wedge OpCount[c] \neq 0$
 $\wedge InMsgs' = [InMsgs \text{ EXCEPT } ![Primary] =$
 $\quad Append(@, [op \mapsto \text{"put"}, c \mapsto c, wc \mapsto \text{"zero"}, k$
 $\quad \mapsto k, v \mapsto v, ct \mapsto Ct[c]])]$
 $\wedge OpCount' = [OpCount \text{ EXCEPT } ![c] = @ - 1]$
 $\wedge History' = [History \text{ EXCEPT } ![c] = Append(@, [op \mapsto \text{"put"}, ts \mapsto InMsgc[c][1].ot, k \mapsto k, v \mapsto v])]$
 $\wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc,$
 $\quad ServerMsg, BlockedClient, BlockedThread, Pt, Cp,$
 $\quad CalState, State, SnapshotTable \rangle$

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

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

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

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

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 \triangleq
 ∧ ∃ *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 \triangleq ∨ *ClientGetRequest_local_primary*
 ∨ *ClientGetRequest_local_secondary*
ClientGetRequest_majority \triangleq ∨ *ClientGetRequest_majority_primary*
 ∨ *ClientGetRequest_majority_secondary*

all possible client get actions
ClientGetRequest \triangleq ∨ *ClientGetRequest_local*
 ∨ *ClientGetRequest_majority*
 ∨ *ClientGetRequest_linearizable*

all possible client put actions
ClientPutRequest \triangleq ∨ *ClientPutRequest_zero*
 ∨ *ClientPutRequest_number*
 ∨ *ClientPutRequest_majority*

all possible server get actions
ServerGetReply \triangleq ∨ *ServerGetReply_local_sleep*
 ∨ *ServerGetReply_local_wake*

$\vee \text{ServerGetReply_majority_sleep}$
 $\vee \text{ServerGetReply_majority_wake}$
 $\vee \text{ServerGetReply_linearizable_sleep}$
 $\vee \text{ServerGetReply_linearizable_wake}$

all possible server put actions

$\text{ServerPutReply} \triangleq \vee \text{ServerPutReply_zero}$
 $\vee \text{ServerPutReply_number_sleep}$
 $\vee \text{ServerPutReply_majority_sleep}$
 $\vee \text{ServerPutReply_number_wake}$
 $\vee \text{ServerPutReply_majority_wake}$

next state for all configurations

$\text{Next} \triangleq \vee \text{ClientGetRequest} \vee \text{ClientPutRequest}$
 $\vee \text{ClientGetResponse} \vee \text{ClientPutResponse}$
 $\vee \text{ServerGetReply} \vee \text{ServerPutReply}$
 $\vee \text{Replicate}$
 $\vee \text{AdvancePt}$
 $\vee \text{ServerTakeHeartbeat}$
 $\vee \text{Snapshot}$

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

$\text{Next_Except_ClientRequest} \triangleq \vee \text{ClientGetResponse}$
 $\vee \text{ClientPutResponse}$
 $\vee \text{ServerGetReply}$
 $\vee \text{ServerPutReply}$
 $\vee \text{Replicate}$
 $\vee \text{AdvancePt}$
 $\vee \text{ServerTakeHeartbeat}$
 $\vee \text{Snapshot}$

$\text{ClientRequest_1} \triangleq \vee \text{ClientPutRequest_majority}$
 $\vee \text{ClientGetRequest_local_primary}$

$\text{ClientRequest_2} \triangleq \vee \text{ClientPutRequest_majority}$
 $\vee \text{ClientGetRequest_local_secondary}$

$\text{ClientRequest_3} \triangleq \vee \text{ClientPutRequest_majority}$
 $\vee \text{ClientGetRequest_majority_primary}$

$\text{ClientRequest_4} \triangleq \vee \text{ClientPutRequest_majority}$
 $\vee \text{ClientGetRequest_majority_secondary}$

$\text{ClientRequest_5} \triangleq \vee \text{ClientPutRequest_majority}$

$$\begin{aligned}
& \vee \textit{ClientGetRequest_linearizable} \\
\textit{ClientRequest_6} & \triangleq \vee \textit{ClientPutRequest_number} \\
& \vee \textit{ClientGetRequest_local_primary} \\
\textit{ClientRequest_7} & \triangleq \vee \textit{ClientPutRequest_number} \\
& \vee \textit{ClientGetRequest_local_secondary} \\
\textit{ClientRequest_8} & \triangleq \vee \textit{ClientPutRequest_number} \\
& \vee \textit{ClientGetRequest_majority_primary} \\
\textit{ClientRequest_9} & \triangleq \vee \textit{ClientPutRequest_number} \\
& \vee \textit{ClientGetRequest_majority_secondary} \\
\textit{ClientRequest_10} & \triangleq \vee \textit{ClientPutRequest_number} \\
& \vee \textit{ClientGetRequest_linearizable} \\
\textit{Next1} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_1} \\
\textit{Next2} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_2} \\
\textit{Next3} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_3} \\
\textit{Next4} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_4} \\
\textit{Next5} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_5} \\
\textit{Next6} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_6} \\
\textit{Next7} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_7} \\
\textit{Next8} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_8} \\
\textit{Next9} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_9} \\
\textit{Next10} & \triangleq \vee \textit{Next_Except_ClientRequest} \\
& \vee \textit{ClientRequest_10} \\
\textit{Spec1} & \triangleq \textit{Init} \wedge \Box[\textit{Next1}]_{vars} \\
\textit{Spec2} & \triangleq \textit{Init} \wedge \Box[\textit{Next2}]_{vars} \\
\textit{Spec3} & \triangleq \textit{Init} \wedge \Box[\textit{Next3}]_{vars}
\end{aligned}$$

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

Idea: *Primarycheck*

$MonotonicRead \triangleq \forall c \in Client : \forall i, j \in \text{DOMAIN } History[c] :$
 $\quad \wedge i < j$
 $\quad \wedge History[c][i].op = \text{"get"}$
 $\quad \wedge History[c][j].op = \text{"get"}$
 $\quad \Rightarrow \neg HLCLt(History[c][j].ts, History[c][i].ts)$

$MonotonicWrite \triangleq \forall c \in Client : \forall i, j \in \text{DOMAIN } History[c] :$
 $\quad \wedge i < j$
 $\quad \wedge History[c][i].op = \text{"put"}$
 $\quad \wedge History[c][j].op = \text{"put"}$
 $\quad \Rightarrow \neg HLCLt(History[c][j].ts, History[c][i].ts)$

$ReadYourWrite \triangleq \forall c \in Client : \forall i, j \in \text{DOMAIN } History[c] :$
 $\quad \wedge i < j$
 $\quad \wedge History[c][i].op = \text{"put"}$
 $\quad \wedge History[c][j].op = \text{"get"}$
 $\quad \Rightarrow \neg HLCLt(History[c][j].ts, History[c][i].ts)$

$WriteFollowRead \triangleq \forall c \in Client : \forall i, j \in \text{DOMAIN } History[c] :$
 $\quad \wedge i < j$
 $\quad \wedge History[c][i].op = \text{"get"}$
 $\quad \wedge History[c][j].op = \text{"put"}$
 $\quad \Rightarrow \neg HLCLt(History[c][j].ts, History[c][i].ts)$

\backslash * Modification *History*
 \backslash * Last modified *Tue Mar 22 15:52:26 CST 2022* by *dh*
 \backslash * Created *Wed Mar 16 11:44:03 CST 2022* by *dh*