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 $c \in$ *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*
CurrentTerm, *CurrentTerm*[*s*]: current election term at server *s* → updated in *update_position*, heartbeat
ReadyToServe, equal to 0 before any primary is elected
SyncSource sync source of server node *s*

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

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

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

$InitPrimary \triangleq Primary =$ 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
$InitCurrentTerm \triangleq CurrentTerm = [s \in Server \mapsto 0]$
$InitReadyToServe \triangleq ReadyToServe = 0$
$InitSyncSource \triangleq SyncSource = [s \in Server \mapsto Nil]$

$Init \triangleq$
$\wedge\ InitPrimary \wedge InitSecondary \wedge InitOplog \wedge InitStore \wedge InitCt$
$\wedge\ InitOt \wedge InitPt \wedge InitCp \wedge InitCalState \wedge InitInMsgc \wedge InitInMsgs$
$\wedge\ InitServerMsg \wedge InitBlockedClient \wedge InitBlockedThread \wedge InitOpCount$

$\land$ *InitState* $\land$ *InitSnap* $\land$ *InitHistory* $\land$ *InitCurrentTerm* $\land$ *InitReadyToServe*
$\land$ *InitSyncSource*

---

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

DH helpers

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

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE IF } HLCLt(new,\ state[index]) \text{ THEN } SubSeq(state,\ 1,\ index-1) \circ \langle new \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } AddState(new,\ state,\ index+1) \\
\text{RECURSIVE } RemoveState(\_,\ \_,\ \_) \\
RemoveState(old,\ state,\ index) \triangleq \text{IF } state[index] = old \text{ THEN } SubSeq(state,\ 1,\ index-1) \circ SubSeq(state,\ ind \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{ELSE } RemoveState(old,\ state,\ index+1)
\end{array}
$$

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

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

There are majority nodes agree to elect node i to become primary

$$
\begin{array}{l}
ElectPrimary(i,\ majorNodes) \triangleq \\
\quad \land\ ReadyToServe > 0 \\
\quad \land\ \forall\, j \in majorNodes : \land\ NotBehind(i,\ j) \\
\qquad\qquad\qquad\qquad\qquad\qquad \land\ CurrentTerm[i] \geq CurrentTerm[j] \\
\quad \land\ IsMajority(majorNodes) \\
\qquad \text{voted nodes for } i \text{ cannot be primary anymore} \\
\quad \land\ Primary' = \text{LET } possiblePrimary \triangleq Primary \setminus majorNodes \\
\qquad\qquad\qquad\quad \text{IN } \quad possiblePrimary \cup \{i\} \\
\qquad \text{add voted nodes into secondaries} \\
\quad \land\ Secondary' = \text{LET } possibleSecondary \triangleq Secondary \cup majorNodes \\
\qquad\qquad\qquad\quad \text{IN } \quad possibleSecondary \setminus \{i\} \\
\quad \land\ CurrentTerm' = [index \in Server \mapsto \text{IF } index \in (majorNodes \cup \{i\}) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } CurrentTerm[i] + 1 \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } CurrentTerm[index]] \\
\quad \land\ \text{UNCHANGED } \langle Oplog,\ Store,\ Ct,\ Ot,\ InMsgc,\ InMsgs,\ ServerMsg,\ BlockedClient, \\
\qquad\qquad\qquad\qquad BlockedThread,\ OpCount,\ Pt,\ Cp,\ State,\ CalState,\ SnapshotTable, \\
\qquad\qquad\qquad\qquad History,\ CurrentTerm,\ ReadyToServe,\ SyncSource\rangle
\end{array}
$$

$$
\begin{array}{l}
TurnOnReadyToServe \triangleq \\
\quad \land\ ReadyToServe = 0 \\
\quad \land\ \exists\, s \in Primary : \\
\qquad \land\ CurrentTerm' = [CurrentTerm \text{ EXCEPT } ![s] = CurrentTerm[s] + 1] \\
\qquad \land\ ReadyToServe' = ReadyToServe + 1 \\
\quad \land\ \text{UNCHANGED } \langle Primary,\ \ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgc,\ InMsgs,\ ServerMsg,\ BlockedClient,
\end{array}
$$

$$
\begin{array}{l}
AdvanceCp \triangleq \\
\quad \land\ ReadyToServe > 0
\end{array}
$$

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

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 $ServerMsg' = [x \in Server \mapsto \text{IF } x = s \text{ THEN } ServerMsg[x]$
ELSE $Append(ServerMsg[x], msg)]$

$ServerTakeHeartbeat \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, s \in Server :$
$\land\ Len(ServerMsg[s]) \neq 0$ message channel is not empty
$\land\ ServerMsg[s].type =$ "heartbeat"
$\land\ Ct' = [Ct \text{ EXCEPT } ![s] = HLCMax(Ct[s], ServerMsg[s][1].ct)]$
$\land\ State' =$
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 $[Cp \text{ EXCEPT } ![s] = newcp]$
$\land\ ServerMsg' = [ServerMsg \text{ EXCEPT } ![s] = Tail(@)]$
$\land\ CurrentTerm' = [CurrentTerm \text{ EXCEPT } ![s] = Max(CurrentTerm[s], ServerMsg[s][1].term)]$
$\land$ 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 $SubHbState \triangleq State[s]$
$hb \triangleq [SubHbState$ EXCEPT $![ServerMsg[s][1].s] =$
$ServerMsg[s][1].aot]$
IN $[State$ EXCEPT $![s] = hb]$
$\land\ CalState' =$ LET $newcal \triangleq$
IF $s \in Primary$ primary node: update *CalState*
THEN $AdvanceState(ServerMsg[s][1].aot,$
$State[s][ServerMsg[s][1].s],\ CalState)$
ELSE $CalState$IN $newcal$
$\land\ Cp' =$ LET $newcp \triangleq$
primary node: compute new mcp
IF $s \in Primary$ THEN $CalState'[Cardinality(Server) \div 2 + 1]$
secondary node: update mcp
ELSE IF $\neg HLCLt(ServerMsg[s][1].cp,\ Cp[s])$
$\land \neg HLCLt(Ot[s],\ ServerMsg[s][1].cp)$
THEN $ServerMsg[s][1].cp$
ELSE $Cp[s]$
IN $[Cp$ EXCEPT $![s] = newcp]$
$\land\ ServerMsg' =$ LET $newServerMsg \triangleq [ServerMsg$ EXCEPT $![s] = Tail(@)]$
$appendMsg \triangleq [type \mapsto$ "update_position", $s \mapsto s,\ aot \mapsto ServerMsg[s][1].aot,\ ct \mapsto$
$newMsg \triangleq$ IF $s \in Primary$
THEN $newServerMsg$ If s is primary, accept the *msg*, else forward it to its sy
ELSE $[newServerMsg$ EXCEPT $![SyncSource[s]] = Append(newServer$
IN $newMsg$
$\land\ CurrentTerm' = [CurrentTerm$ EXCEPT $![s] = Max(CurrentTerm[s],\ ServerMsg[s][1].term)]$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ot,\ InMsgc,\ InMsgs,$
$BlockedClient,\ BlockedThread,\ OpCount,\ Pt,\ SnapshotTable,\ History,\ ReadyToServe,\ SyncSource\rangle$

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 $Pt[x]$

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
    $\land\ Pt[s] \leq PtStop$
    $\land\ Pt' = [Pt \text{ EXCEPT } ![s] = @ + 1]$ advance physical time
    $\land\ BroadcastHeartbeat(s)$ broadcast heartbeat periodly
  $\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, InMsgs, State, BlockedClient, BlockedThread, OpCount, Cp, CalState, SnapshotTable, History, CurrentTerm, ReadyToServe, SyncSource\rangle$

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

Replication
Idea: replicate$canSyncFrom log$term
$SyncSource[s]SyncSource UpdatePosition$
$UpdatePosition$actiontype$updatePosition$

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 $\langle\rangle$

Replicate oplog from node j to node i, and update related structures accordingly
$Replicate(i, j) \triangleq$
  $\land\ ReadyToServe > 0$
  $\land\ CanSyncFrom(i, j)$
  $\land\ i \in Secondary$
  $\land\ ReplicateOplog(i, j) \neq \langle\rangle$
  $\land\ Oplog' = [Oplog \text{ EXCEPT } ![i] = @ \circ ReplicateOplog(i, j)]$
  $\land\ Store' = [Store \text{ EXCEPT } ![i] = Store[j]]$
  $\land\ Ct' = [Ct \text{ EXCEPT } ![i] = HLCMax(Ct[i], Ct[j])]$ update $Ct[i]$
  $\land\ Ot' = [Ot \text{ EXCEPT } ![i] = HLCMax(Ot[i], Ot[j])]$ update $Ot[i]$
  $\land\ Cp' = [Cp \text{ EXCEPT } ![i] = HLCMax(Cp[i], Cp[j])]$ update $Cp[i]$
  $\land\ CurrentTerm' = [CurrentTerm \text{ EXCEPT } ![i] = Max(CurrentTerm[i], CurrentTerm[j])]$ update $CurrentTer$
  $\land\ State' =$

$\quad\quad$ LET $SubHbState \triangleq State[i]$
$\quad\quad\quad\quad hb \triangleq [SubHbState$ EXCEPT $![j] = Ot[j]]$
$\quad\quad$ IN $[State$ EXCEPT $![i] = hb]$ update j's state i knows
$\land$ LET $msg \triangleq [type \mapsto$ "update_position", $s \mapsto i$, $aot \mapsto Ot'[i]$, $ct \mapsto Ct'[i]$, $cp \mapsto Cp'[i]]$
$\quad$ IN $ServerMsg' = [ServerMsg$ EXCEPT $![j] = Append(ServerMsg[j], msg)]$
$\land\ SyncSource' = [SyncSource$ EXCEPT $![i] = j]$
$\land$ UNCHANGED $\langle Primary, Secondary, InMsgc, InMsgs, BlockedClient,$
$\quad\quad\quad\quad\quad\quad BlockedThread, OpCount, Pt, CalState, SnapshotTable,$
$\quad\quad\quad\quad\quad\quad History, ReadyToServe\rangle$

server get

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

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

$ServerGetReply\_majority\_sleep \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, s \in Server :$

$\land\ Len(InMsgs[s]) \neq 0$ 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
$\land\ Ct' = [Ct \text{ EXCEPT } ![s] = HLCMax(Ct[s],\ InMsgs[s][1].ct)]$
$\land\ BlockedThread' = [BlockedThread \text{ EXCEPT } ![InMsgs[s][1].c] =$
$[type \mapsto \text{“read\_major”},\ s \mapsto s,\ k \mapsto InMsgs[s][1].k,\ ot \mapsto InMsgs[s][1].ot]]$
$\land\ InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Tail(@)]$
$\land$ 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*
$\land\ InMsgc' = [InMsgc \text{ EXCEPT } ![c] = Append(@,\ [op \mapsto \text{“get”},\ k \mapsto BlockedThread[c].k,\ v \mapsto$
$SelectSnapshot(SnapshotTable[BlockedThread[c].s],\ Cp[BlockedThread[c].s])[BlockedThr$
$\mapsto Ct[BlockedThread[c].s],\ ot \mapsto Cp[BlockedThread[c].s]])]$
send *msg* to client
$\land\ BlockedThread' = [BlockedThread \text{ EXCEPT } ![c] = Nil]$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Oplog,\ Store,\ Ct,\ Ot,\ InMsgs,\ ServerMsg,$
$BlockedClient,\ OpCount,\ Pt,\ Cp,$
$CalState,\ State,\ SnapshotTable,\ History,$
$CurrentTerm,\ ReadyToServe,\ SyncSource\rangle$

$ServerGetReply\_linearizable\_sleep \triangleq$
$\land\ ReadyToServe > 0$
$\land\ \exists\, s \in Server :$
$\land\ s = Primary$
$\land\ Len(InMsgs[s]) \neq 0$
$\land\ InMsgs[s][1].op = \text{“get”}$
$\land\ InMsgs[s][1].rc = \text{“linea”}$ 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 $Ot[s]$
$\land\ State' =$
LET $SubHbState \triangleq State[s]$
$hb \triangleq [SubHbState \text{ EXCEPT } ![Primary] = Ot'[Primary]]$
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 $BlockedThread[c]$
$\land$ UNCHANGED $\langle Primary,\ Secondary,\ Store,\ InMsgc,\ ServerMsg,\ BlockedClient,$
$OpCount,\ Pt,\ Cp,\ SnapshotTable,\ History,$
$CurrentTerm,\ ReadyToServe,\ SyncSource\rangle$

$ServerGetReply\_linearizable\_wake \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, c \in Client :$
$\land\ BlockedThread[c] \neq Nil$
$\land\ BlockedThread[c].type = \text{“read\_linea”}$
$\land\ \neg HLCLt(Cp[BlockedThread[c].s],\ BlockedThread[c].ot)$ $cp[s] \geq$ 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
$\land$ 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 $oplog[primary]$
$\land\ Oplog' = \text{LET } entry \triangleq [k \mapsto InMsgs[s][1].k,\ v \mapsto InMsgs[s][1].v,\ ot \mapsto Ot'[s],\ term \mapsto CurrentTerm$
$newLog \triangleq Append(Oplog[s],\ entry)$
$\text{IN } [Oplog \text{ EXCEPT } ![s] = newLog]$
$\land\ State' =$
$\text{LET } SubHbState \triangleq State[s]$
$hb \triangleq [SubHbState \text{ EXCEPT } ![s] = Ot'[s]]$
$\text{IN } [State \text{ EXCEPT } ![s] = hb]$ update primary state

$\land$ $CalState' = AdvanceState(Ot'[s],\ Ot[s],\ CalState)$
$\land$ $InMsgs' = [InMsgs$ EXCEPT $![s] = Tail(@)]$
$\land$ 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
$\land$ $InMsgs[s][1].op =$ "put" *msg* type: put
$\land$ $InMsgs[s][1].wc =$ "num" Write Concern: *num*
$\land$ $Tick(s)$ advance cluster time
$\land$ $Ot' = [Ot$ EXCEPT $![s] = Ct'[s]]$
advance the last applied operation time $Ot[Primary]$
$\land$ $Store' = [Store$ EXCEPT $![s][InMsgs[s][1].k] = InMsgs[s][1].v]$
$\land$ $Oplog' = [\ Oplog$ EXCEPT $![s] =$
$Append(@, \langle InMsgs[s][1].k,\ InMsgs[s][1].v,\ Ot'[s],\ CurrentTerm[s]\rangle)]$
$\land$ 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 $Oplog' = [Oplog$ EXCEPT $![s] = newLog]$
$\land$ $State' =$
LET $SubHbState \triangleq State[s]$
$hb \triangleq [SubHbState$ EXCEPT $![s] = Ot'[s]]$
IN $[State$ EXCEPT $![s] = hb]$ update primary state
$\land$ $CalState' = AdvanceState(Ot'[s],\ Ot[s],\ CalState)$
$\land$ $BlockedThread' = [BlockedThread$ 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 $\langle Primary,\ Secondary,\ InMsgc,\ ServerMsg,\ BlockedClient,$
$OpCount,\ Pt,\ Cp,\ SnapshotTable,$
$History,\ CurrentTerm,\ ReadyToServe,\ SyncSource\rangle$

$ServerPutReply\_number\_wake \triangleq$
$\land$ $ReadyToServe > 0$
$\land \exists\, c \in Client :$
$\land$ $BlockedThread[c] \neq Nil$
$\land$ $BlockedThread[c].type =$ "write_num"
$\land$ $\neg HLCLt(CalState[Cardinality(Server) - BlockedThread[c].numnode + 1],$

$$BlockedThread[c].ot) \quad \text{CalState}[s][n - num + 1] \geq \text{ target } ot$$

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

remove blocked state

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

****

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

****

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

$$ServerPutReply\_majority\_wake \triangleq$$
$$\land\ ReadyToServe > 0$$
$$\land\ \exists\, c \in Client :$$
$$\land\ BlockedThread[c] \neq \ Nil$$
$$\land\ BlockedThread[c].type = \text{"write\_major"}$$
$$\land\ \neg HLCLt(Cp[Primary], BlockedThread[c].ot)$$
$$\land\ InMsgc' = [InMsgc \text{ EXCEPT } ![c] =$$
$$Append(@, [op \mapsto \text{"put"}, ct \mapsto Ct[BlockedThread[c].s], ot \mapsto BlockedThread[c].ot, k \mapsto BlockedThre$$

$\land\ BlockedThread' = [BlockedThread \text{ EXCEPT } ![c] = Nil]$
$\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs, ServerMsg, BlockedClient, OpCoun$

client get

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

$ClientGetRequest\_local\_secondary \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Secondary :$
$\land\ InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@,$
$[op \mapsto$ "get", $c \mapsto c,\ rc \mapsto$ "local", $k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
$\land\ BlockedClient' = BlockedClient \cup \{c\}$
$\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlockedThread, OpCount$

$ClientGetRequest\_majority\_primary \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Primary :$
$\land\ InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@,$
$[op \mapsto$ "get", $c \mapsto c,\ rc \mapsto$ "major", $k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
$\land\ BlockedClient' = BlockedClient \cup \{c\}$
$\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlockedThread, OpCoun$

$ClientGetRequest\_majority\_secondary \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Secondary :$
$\land\ InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@,$
$[op \mapsto$ "get", $c \mapsto c,\ rc \mapsto$ "major", $k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
$\land\ BlockedClient' = BlockedClient \cup \{c\}$
$\land$ UNCHANGED $\langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg, BlockedThread, OpCoun$

$ClientGetRequest\_linearizable \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, k \in Key,\ c \in Client \setminus BlockedClient,\ s \in Primary :$
$\land\ InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@,$
$[op \mapsto$ "get", $c \mapsto c,\ rc \mapsto$ "linea", $k \mapsto k,\ ct \mapsto Ct[c],\ ot \mapsto Ot[c]])]$
$\land\ BlockedClient' = BlockedClient \cup \{c\}$

$\land$ 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
****

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

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

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

$ClientGetResponse \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, c \in Client :$
$\land\ OpCount[c] \neq 0$ client c has operation times
$\land\ Len(InMsgc[c]) \neq 0$ message channel is not empty

$\land\ InMsgc[c][1].op = \text{“get”}$ *msg* type: get
$\land\ Store' = [Store \text{ EXCEPT } ![c][InMsgc[c][1].k] = InMsgc[c][1].v]$
store data
$\land\ History' = [History \text{ EXCEPT } ![c] = Append(@, [op \mapsto \text{“get”}, ts \mapsto InMsgc[c][1].ot, k \mapsto InMsgc[c][1].k, v \mapsto InMsgc[c][1].v])]$
$\land\ InMsgc' = [InMsgc \text{ EXCEPT } ![c] = Tail(@)]$
$\land\ BlockedClient' = \text{IF } Len(InMsgc'[c]) = 0$
$\text{THEN } BlockedClient \setminus \{c\}$
$\text{ELSE } BlockedClient$ remove blocked state
$\land\ OpCount' = [OpCount \text{ EXCEPT } ![c] = @ - 1]$
$\land \text{ UNCHANGED } \langle Primary, Secondary, Oplog, Ct, Ot, InMsgs, ServerMsg, BlockedThread, Pt, Cp, CalState, State, SnapshotTable, CurrentTerm, ReadyToServe, SyncSource\rangle$

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

$ClientPutResponse \triangleq$
$\land\ ReadyToServe > 0$
$\land \exists\, c \in Client :$
$\land\ OpCount[c] \neq 0$ client c has operation times
$\land\ Len(InMsgc[c]) \neq 0$ message channel is not empty
$\land\ InMsgc[c][1].op = \text{“put”}$ *msg* type: put
$\land\ Ct' = [Ct \text{ EXCEPT } ![c] = HLCMax(@, InMsgc[c][1].ct)]$
$\land\ Ot' = [Ot \text{ EXCEPT } ![c] = HLCMax(@, InMsgc[c][1].ot)]$ Update *Ot* to record “my write” *ot*
$\land\ History' = [History \text{ EXCEPT } ![c] = Append(@, [op \mapsto \text{“put”}, ts \mapsto InMsgc[c][1].ot, k \mapsto InMsgc[c][1].k, v \mapsto InMsgc[c][1].v])]$
$\land\ InMsgc' = [InMsgc \text{ EXCEPT } ![c] = Tail(@)]$
$\land\ BlockedClient' = \text{IF } Len(InMsgc'[c]) = 0$
$\text{THEN } BlockedClient \setminus \{c\}$
$\text{ELSE } BlockedClient$ remove blocked state
$\land\ OpCount' = [OpCount \text{ EXCEPT } ![c] = @ - 1]$
$\land \text{ UNCHANGED } \langle Primary, Secondary, Oplog, Store, InMsgs, ServerMsg, BlockedThread, Pt, Cp, CalState, State, SnapshotTable, CurrentTerm, ReadyToServe, SyncSource\rangle$

$ClientGetRequest\_local \triangleq \lor ClientGetRequest\_local\_primary$
$\lor ClientGetRequest\_local\_secondary$
$ClientGetRequest\_majority \triangleq \lor ClientGetRequest\_majority\_primary$
$\lor ClientGetRequest\_majority\_secondary$

all possible client get actions
$ClientGetRequest \triangleq \lor ClientGetRequest\_local$

$\lor$ *ClientGetRequest_majority*
$\lor$ *ClientGetRequest_linearizable*

all possible clent put actions

ClientPutRequest $\triangleq$ $\lor$ *ClientPutRequest_zero*
$\lor$ *ClientPutRequest_number*
$\lor$ *ClientPutRequest_majority*

all possible server get actions

ServerGetReply $\triangleq$ $\lor$ *ServerGetReply_local_sleep*
$\lor$ *ServerGetReply_local_wake*
$\lor$ *ServerGetReply_majority_sleep*
$\lor$ *ServerGetReply_majority_wake*
$\lor$ *ServerGetReply_linearizable_sleep*
$\lor$ *ServerGetReply_linearizable_wake*

all possible server put actions

ServerPutReply $\triangleq$ $\lor$ *ServerPutReply_zero*
$\lor$ *ServerPutReply_number_sleep*
$\lor$ *ServerPutReply_majority_sleep*
$\lor$ *ServerPutReply_number_wake*
$\lor$ *ServerPutReply_majority_wake*

Can node i rollback its *log* based on j's *log*

$$
\begin{aligned}
\mathit{CanRollback}(i,\, j) \triangleq\ & \land \mathit{Len}(\mathit{Oplog}[i]) > 0 \\
& \land \mathit{Len}(\mathit{Oplog}[j]) > 0 \\
& \land \mathit{LastTerm}(i) < \mathit{LastTerm}(j) \\
& \land \\
& \quad \lor \mathit{Len}(\mathit{Oplog}[i]) > \mathit{Len}(\mathit{Oplog}[j]) \\
& \quad \lor \land \mathit{Len}(\mathit{Oplog}[i]) \leq \mathit{Len}(\mathit{Oplog}[j]) \\
& \qquad \land \mathit{LastTerm}(i) \neq \mathit{LogTerm}(j,\, \mathit{Len}(\mathit{Oplog}[i])) \\
& \qquad\ \mathit{Len}(\mathit{Oplog}[i])\mathit{Len}(\mathit{Oplog}[i] + 1? - 1?)
\end{aligned}
$$

Returns the highest common index between two divergent logs, 'li' and 'lj'.
If there is no common index between the logs, returns 0.

$$
\begin{aligned}
&\mathit{RollbackCommonPoint}(i,\, j) \triangleq \\
&\quad \text{LET } \mathit{commonIndices} \triangleq \{k \in \text{DOMAIN } \mathit{Oplog}[i] : \\
&\qquad\qquad \land k \leq \mathit{Len}(\mathit{Oplog}[j]) \\
&\qquad\qquad \land \mathit{Oplog}[i][k] = \mathit{Oplog}[j][k]\}\text{IN} \\
&\qquad \text{IF } \mathit{commonIndices} = \{\} \text{ THEN } 0 \text{ ELSE } \mathit{MaxVal}(\mathit{commonIndices})
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{RollbackOplog}(i,\, j) \triangleq \\
&\quad \land i \in \mathit{Secondary} \\
&\quad \land \mathit{CanRollback}(i,\, j)
\end{aligned}
$$

$\land$ LET $cmp \triangleq RollbackCommonPoint(i, j)$IN
$\land \quad Oplog' = [Oplog \text{ EXCEPT } ![i] = SubSeq(Oplog[i], 1, cmp)]$
$\land \quad CurrentTerm' = [CurrentTerm \text{ EXCEPT } ![i] = LogTerm(j, cmp)]$
$\land$ UNCHANGED $\langle Primary, Secondary, Store, Ct, Ot, InMsgc, InMsgs, ServerMsg, BlockedClient, BlockedThread, OpCount, Pt, Cp, State, CalState, SnapshotTable, History, ReadyToServe, SyncSource\rangle$

$RollbackOplogAction \triangleq \exists\, i, j \in Server : RollbackOplog(i, j)$

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

$ElectPrimaryAction \triangleq$
$\exists\, i \in Server : \exists\, majorNodes \in \text{SUBSET } (Server) : ElectPrimary(i, majorNodes)$

---

next state for all configurations

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

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

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

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

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

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

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

$$
\begin{aligned}
\mathit{ClientRequset\_5} \triangleq\ & \lor \mathit{ClientPutRequest\_majority} \\
& \lor \mathit{ClientGetRequest\_linearizable}
\end{aligned}
$$

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

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

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

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

$$
\begin{aligned}
\mathit{ClientRequset\_10} \triangleq\ & \lor \mathit{ClientPutRequest\_number} \\
& \lor \mathit{ClientGetRequest\_linearizable}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Next}1 \triangleq\ & \lor \mathit{Next\_Except\_ClientRequset} \\
& \lor \mathit{ClientRequset\_}1
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Next}2 \triangleq\ & \lor \mathit{Next\_Except\_ClientRequset} \\
& \lor \mathit{ClientRequset\_}2
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Next}3 \triangleq\ & \lor \mathit{Next\_Except\_ClientRequset} \\
& \lor \mathit{ClientRequset\_}3
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Next}4 \triangleq\ & \lor \mathit{Next\_Except\_ClientRequset} \\
& \lor \mathit{ClientRequset\_}4
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Next}5 \triangleq\ & \lor \mathit{Next\_Except\_ClientRequset} \\
& \lor \mathit{ClientRequset\_}5
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Next}6 \triangleq\ & \lor \mathit{Next\_Except\_ClientRequset} \\
& \lor \mathit{ClientRequset\_}6
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Next}7 \triangleq\ & \lor \mathit{Next\_Except\_ClientRequset} \\
& \lor \mathit{ClientRequset\_}7
\end{aligned}
$$

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

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

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

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

Idea:*Primary*check

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

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

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

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

\ ∗ Modification *History*
\ ∗ Last modified *Thu Apr* 07 01:04:14 *CST* 2022 by *dh*
\ ∗ Created *Thu Mar* 31 20:33:19 *CST* 2022 by *dh*