
MODULE *TunableMongoDB_RBK*

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>InMsgc[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>
	<i>CurrentTerm,</i>	<i>CurrentTerm[s]</i> : current election term at server <i>s</i> → updated in <i>update_position</i> , heartbeat and replicate
	<i>ReadyToServe,</i>	equal to 0 before any primary is elected
	<i>SyncSource</i>	sync source of server node <i>s</i>

ASSUME *Cardinality(Client)* ≥ 1 at least one client

ASSUME *Cardinality(Server)* ≥ 2 at least one primary and one secondary

ASSUME *Cardinality(Key)* ≥ 1 at least one object

ASSUME *Cardinality(Value)* ≥ 2 at least two values to update

Helpers

$HLCLt(x, y) \triangleq$ 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

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

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

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

snapshot helpers
 RECURSIVE $SelectSnapshot_rec(-, -, -)$
 $SelectSnapshot_rec(stable, cp, index) \triangleq$
 $\text{IF } HLClt(cp, stable[index].ot) \text{ THEN } stable[index - 1].store$
 $\text{ELSE IF } index = Len(stable) \text{ THEN } stable[index].store$
 $\text{ELSE } SelectSnapshot_rec(stable, cp, index + 1)$

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

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

Is node i ahead of node j
 $NotBehind(i, j) \triangleq$
 $\vee LastTerm(i) > LastTerm(j)$
 $\vee \wedge LastTerm(i) = LastTerm(j)$
 $\wedge Len(Oplog[i]) \geq Len(Oplog[j])$

$IsMajority(servers) \triangleq Cardinality(servers) * 2 > Cardinality(Server)$

Return the maximum value from a set, or undefined if the set is empty.
 $MaxVal(s) \triangleq \text{CHOOSE } x \in s : \forall y \in s : x \geq y$

commit point
 RECURSIVE $AddState(-, -, -)$
 $AddState(new, state, index) \triangleq$
 $\text{IF } index = 1 \wedge HLClt(new, state[1])$
 $\text{THEN } \langle new \rangle \circ state$ less than the first

```

ELSE IF  $index = Len(state) + 1$ 
  THEN  $state \circ \langle new \rangle$ 
ELSE IF  $HLCLt(new, state[index])$ 
  THEN  $SubSeq(state, 1, index - 1) \circ \langle new \rangle \circ SubSeq(state, index, Len(state))$ 
ELSE  $AddState(new, state, index + 1)$ 

RECURSIVE  $RemoveState(-, -, -)$ 
 $RemoveState(old, state, index) \triangleq$ 
  IF  $state[index] = old$ 
    THEN  $SubSeq(state, 1, index - 1) \circ SubSeq(state, index + 1, Len(state))$ 
  ELSE  $RemoveState(old, state, index + 1)$ 

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

clock

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

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

heartbeat
Only Primary node sends heartbeat once advance pt
 $BroadcastHeartbeat(s) \triangleq$ 
  LET  $msg \triangleq [type \mapsto \text{"heartbeat"}, s \mapsto s, aot \mapsto Ot[s],$ 
     $ct \mapsto Ct[s], cp \mapsto Cp[s], term \mapsto CurrentTerm[s]]$ 
  IN  $ServerMsg' = [x \in Server \mapsto IF\ x = s\ THEN\ ServerMsg[x]$ 
    ELSE  $Append(ServerMsg[x], msg)]$ 

Can node  $i$  sync from node  $j$ ?
 $CanSyncFrom(i, j) \triangleq$ 
   $\wedge Len(Oplog[i]) < Len(Oplog[j])$ 
   $\wedge LastTerm(i) = LogTerm(j, Len(Oplog[i]))$ 

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  $i \neq Primary \wedge len\_i < len\_j$ 
    THEN  $SubSeq(Oplog[j], len\_i + 1, len\_j)$ 
  ELSE  $\langle \rangle$ 

```

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

$$\begin{aligned}
CanRollback(i, j) \triangleq & \wedge Len(Oplog[i]) > 0 \\
& \wedge Len(Oplog[j]) > 0 \\
& \wedge CurrentTerm[i] < CurrentTerm[j] \\
& \wedge \\
& \vee Len(Oplog[i]) > Len(Oplog[j]) \\
& \vee \wedge Len(Oplog[i]) \leq Len(Oplog[j]) \\
& \wedge CurrentTerm[i] \neq LogTerm(j, Len(Oplog[i]))
\end{aligned}$$

Returns the highest common index between two divergent logs.

If there is no common index between the logs, returns 0.

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

Init Part

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

$\wedge \text{InitOt} \wedge \text{InitPt} \wedge \text{InitCp} \wedge \text{InitCalState} \wedge \text{InitInMsgc} \wedge \text{InitInMsgs}$
 $\wedge \text{InitServerMsg} \wedge \text{InitBlockedClient} \wedge \text{InitBlockedThread} \wedge \text{InitOpCount}$
 $\wedge \text{InitState} \wedge \text{InitSnap} \wedge \text{InitHistory} \wedge \text{InitCurrentTerm} \wedge \text{InitReadyToServe}$
 $\wedge \text{InitSyncSource}$

Next State Actions

Replication Protocol: possible actions

snapshot periodically

$\text{Snapshot} \triangleq$

$\wedge \text{ReadyToServe} > 0$

$\wedge \exists s \in \text{Server} :$

$\text{SnapshotTable}' = [\text{SnapshotTable} \text{ EXCEPT } ![s] =$
 $\text{Append}(@, [ot \mapsto \text{Ot}[s], \text{store} \mapsto \text{Store}[s]])]$
 create a new snapshot

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

$\text{Stepdown} \triangleq$

$\wedge \text{ReadyToServe} > 0$

$\wedge \exists s \in \text{Primary} :$

$\wedge \text{Primary}' = \text{Primary} \setminus \{s\}$

$\wedge \text{Secondary}' = \text{Secondary} \cup \{s\}$

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

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

$\text{ElectPrimary}(i, \text{majorNodes}) \triangleq$

$\wedge \text{ReadyToServe} > 0$

$\wedge \forall j \in \text{majorNodes} : \wedge \text{NotBehind}(i, j)$

$\wedge \text{CurrentTerm}[i] \geq \text{CurrentTerm}[j]$

$\wedge \text{IsMajority}(\text{majorNodes})$

voted nodes for i cannot be primary anymore

$\wedge \text{Primary}' = \text{LET } \text{possiblePrimary} \triangleq \text{Primary} \setminus \text{majorNodes}$
 $\text{IN } \text{possiblePrimary} \cup \{i\}$

add voted nodes into secondaries

$\wedge \text{Secondary}' = \text{LET } \text{possibleSecondary} \triangleq \text{Secondary} \cup \text{majorNodes}$
 $\text{IN } \text{possibleSecondary} \setminus \{i\}$

$\wedge \text{CurrentTerm}' = [\text{index} \in \text{Server} \mapsto \text{IF } \text{index} \in (\text{majorNodes} \cup \{i\})$
 $\text{THEN } \text{CurrentTerm}[i] + 1$

ELSE $CurrentTerm[index]$

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

$TurnOnReadyToServe \triangleq$

$\wedge ReadyToServe = 0$

$\wedge \exists s \in Primary :$

$\wedge CurrentTerm' = [CurrentTerm \text{ EXCEPT } ![s] = CurrentTerm[s] + 1]$

$\wedge ReadyToServe' = ReadyToServe + 1$

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

$AdvanceCp \triangleq$

$\wedge ReadyToServe > 0$

$\wedge Cp' = [Cp \text{ EXCEPT } ![Primary] = CalState[Cardinality(Server) \div 2 + 1]]$

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

$ServerTakeHeartbeat \triangleq$

$\wedge ReadyToServe > 0$

$\wedge \exists s \in Server :$

$\wedge Len(ServerMsg[s]) \neq 0$ message channel is not empty

$\wedge ServerMsg[s].type = \text{"heartbeat"}$

$\wedge Ct' = [Ct \text{ EXCEPT } ![s] = HLCMax(Ct[s], ServerMsg[s][1].ct)]$

$\wedge State' =$

LET $SubHbState \triangleq State[s]$

$hb \triangleq [SubHbState \text{ EXCEPT } ![ServerMsg[s][1].s] =$

$ServerMsg[s][1].aot]$

IN $[State \text{ EXCEPT } ![s] = hb]$

$\wedge CalState' = \text{LET } newcal \triangleq$

IF $s \in Primary$ primary node: update CalState

THEN $AdvanceState(ServerMsg[s][1].aot,$

$State[s][ServerMsg[s][1].s], CalState)$

ELSE $CalState$ IN $newcal$

$\wedge Cp' = \text{LET } newcp \triangleq$

primary node: compute new mcp

IF $s \in Primary$ THEN $CalState'[Cardinality(Server) \div 2 + 1]$

secondary node: update mcp

ELSE IF $\neg HLClt(ServerMsg[s][1].cp, Cp[s])$

$\wedge \neg HLClt(Ot[s], ServerMsg[s][1].cp)$

THEN $ServerMsg[s][1].cp$

ELSE $Cp[s]$

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

NTPPrimary

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

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

Replication

Idea: replicate canSyncFrom log term

SyncSource[s].SyncSource UpdatePosition

UpdatePosition action type updatePosition

Replicate oplog from node j to node i , and update related structures accordingly

$Replicate(i, j) \triangleq$
 $\wedge ReadyToServe > 0$
 $\wedge CanSyncFrom(i, j)$ i can sync from j only need not to rollback
 $\wedge i \in Secondary$
 $\wedge ReplicateOplog(i, j) \neq \langle \rangle$
 $\wedge Oplog' = [Oplog \text{ EXCEPT } ![i] = @ \circ ReplicateOplog(i, j)]$
 $\wedge Store' = [Store \text{ EXCEPT } ![i] = Store[j]]$
 $\wedge Ct' = [Ct \text{ EXCEPT } ![i] = HLCMax(Ct[i], Ct[j])]$ update $Ct[i]$
 $\wedge Ot' = [Ot \text{ EXCEPT } ![i] = HLCMax(Ot[i], Ot[j])]$ update $Ot[i]$
 $\wedge Cp' = [Cp \text{ EXCEPT } ![i] = HLCMax(Cp[i], Cp[j])]$ update $Cp[i]$
 $\wedge CurrentTerm' = [CurrentTerm \text{ EXCEPT } ![i] = Max(CurrentTerm[i], CurrentTerm[j])]$ update $CurrentTerm$
 $\wedge State' =$
 $\quad LET \ SubHbState \triangleq State[i]$
 $\quad \quad hb \triangleq [SubHbState \text{ EXCEPT } ![j] = Ot[j]]$
 $\quad IN \ [State \text{ EXCEPT } ![i] = hb]$ update j 's state i knows
 $\wedge LET \ msg \triangleq [type \mapsto \text{"update_position"}, s \mapsto i, aot \mapsto Ot'[i], ct \mapsto Ct'[i], cp \mapsto Cp'[i]]$
 $\quad IN \ ServerMsg' = [ServerMsg \text{ EXCEPT } ![j] = Append(ServerMsg[j], msg)]$
 $\wedge SyncSource' = [SyncSource \text{ EXCEPT } ![i] = j]$
 $\wedge UNCHANGED \langle Primary, Secondary, InMsgc, InMsgs, BlockedClient,$
 $BlockedThread, OpCount, Pt, CalState, SnapshotTable,$
 $History, ReadyToServe \rangle$

Rollback i 's oplog and recover it to j 's state
Recover to j 's state immediately to prevent internal client request
 $RollbackAndRecover(i, j) \triangleq$
 $\wedge ReadyToServe > 0$
 $\wedge i \in Secondary$
 $\wedge CanRollback(i, j)$
 $\wedge LET\ cmp \triangleq RollbackCommonPoint(i, j)\ IN$
 $\quad LET\ commonLog \triangleq SubSeq(Oplog[i], 1, cmp)$
 $\quad \quad appendLog \triangleq SubSeq(Oplog[j], cmp + 1, Len(Oplog[j]))$
 $\quad IN\ Oplog' = [Oplog\ EXCEPT\ ![i] = commonLog \circ appendLog]$
 $\wedge CurrentTerm' = [CurrentTerm\ EXCEPT\ ![i] = Max(CurrentTerm[i], CurrentTerm[j])] \quad \text{update } CurrentTerm$
 $\wedge Store' = [Store\ EXCEPT\ ![i] = Store[j]]$
 $\wedge Ct' = [Ct\ EXCEPT\ ![i] = HLCMax(Ct[i], Ct[j])] \quad \text{update } Ct[i]$
 $\wedge Ot' = [Ot\ EXCEPT\ ![i] = HLCMax(Ot[i], Ot[j])] \quad \text{update } Ot[i]$
 $\wedge Cp' = [Cp\ EXCEPT\ ![i] = HLCMax(Cp[i], Cp[j])] \quad \text{update } Cp[i]$
 $\wedge State' =$
 $\quad LET\ SubHbState \triangleq State[i]$
 $\quad \quad hb \triangleq [SubHbState\ EXCEPT\ ![j] = Ot[j]]$
 $\quad \quad IN\ [State\ EXCEPT\ ![i] = hb] \quad \text{update } j\text{'s state } i\text{ knows}$
 $\wedge LET\ msg \triangleq [type \mapsto \text{"update_position"}, s \mapsto i, aot \mapsto Ot'[i], ct \mapsto Ct'[i], cp \mapsto Cp'[i]]$
 $\quad IN\ ServerMsg' = [ServerMsg\ EXCEPT\ ![j] = Append(ServerMsg[j], msg)]$
 $\wedge SyncSource' = [SyncSource\ EXCEPT\ ![i] = j]$
 $\wedge UNCHANGED \langle Primary, Secondary, InMsgs, InMsgs, BlockedClient,$
 $\quad BlockedThread, OpCount, Pt, CalState, SnapshotTable,$
 $\quad History, ReadyToServe \rangle$

Tunable Protocol: Server Actions

Server Get

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

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

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

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

CurrentTerm, ReadyToServe, SyncSource

ServerGetReply_linearizable_sleep \triangleq
 $\wedge \text{ReadyToServe} > 0$
 $\wedge \exists s \in \text{Server} :$
 $\wedge s = \text{Primary}$
 $\wedge \text{Len}(\text{InMsgs}[s]) \neq 0$
 $\wedge \text{InMsgs}[s][1].\text{op} = \text{"get"}$
 $\wedge \text{InMsgs}[s][1].\text{rc} = \text{"linea"}$ 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
 $\wedge \text{CalState}' = \text{AdvanceState}(\text{Ot}'[Primary], \text{Ot}[Primary], \text{CalState})$
 $\wedge \text{InMsgs}' = [\text{InMsgs} \text{ EXCEPT } ![s] = \text{Tail}(@)]$
 $\wedge \text{BlockedThread}' = [\text{BlockedThread} \text{ EXCEPT } ![\text{InMsgs}[s][1].c] =$
 $[type \mapsto \text{"read_linea"}, ot \mapsto Ct'[s], s \mapsto s, k$
 $\mapsto \text{InMsgs}[s][1].k, v \mapsto \text{Store}[s][\text{InMsgs}[s][1].k]]]$
 $\text{add the user thread to BlockedThread}[c]$
 $\wedge \text{UNCHANGED } \langle \text{Primary}, \text{Secondary}, \text{Store}, \text{InMsgc}, \text{ServerMsg}, \text{BlockedClient},$
 $\text{OpCount}, \text{Pt}, \text{Cp}, \text{SnapshotTable}, \text{History},$
 $\text{CurrentTerm}, \text{ReadyToServe}, \text{SyncSource} \rangle$

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

Server Put
serveroplog

$$\begin{aligned}
& \text{IN } [State \text{ EXCEPT } ![s] = hb] \text{ update primary state} \\
& \wedge CalState' = AdvanceState(Ot'[s], Ot[s], CalState) \\
& \wedge BlockedThread' = [BlockedThread \text{ EXCEPT } ![InMsgs[s][1].c] = [type \mapsto \text{"write_num"}, \\
& \quad ot \mapsto Ot'[s], s \mapsto s, numnode \mapsto InMsgs[s][1].num, \\
& \quad k \mapsto InMsgs[s][1].k, v \mapsto InMsgs[s][1].v]] \\
& \quad \text{add the user } thHistory \text{ to } BlockedThread[c] \\
& \wedge InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Tail(@)] \\
& \wedge \text{UNCHANGED } \langle Primary, Secondary, InMsgc, ServerMsg, BlockedClient, \\
& \quad OpCount, Pt, Cp, SnapshotTable, \\
& \quad History, CurrentTerm, ReadyToServe, SyncSource \rangle
\end{aligned}$$

$ServerPutReply_number_wake \triangleq$

$$\begin{aligned}
& \wedge ReadyToServe > 0 \\
& \wedge \exists c \in Client : \\
& \quad \wedge BlockedThread[c] \neq Nil \\
& \quad \wedge BlockedThread[c].type = \text{"write_num"} \\
& \quad \wedge \neg HCLt(CalState[Cardinality(Server) - BlockedThread[c].numnode + 1], \\
& \quad \quad BlockedThread[c].ot) \quad CalState[s][n - num + 1] \geq target \quad ot \\
& \quad \wedge InMsgc' = [InMsgc \text{ EXCEPT } ![c] = Append(@, [op \mapsto \text{"put"}, ct \\
& \quad \mapsto Ct[Primary], ot \mapsto BlockedThread[c].ot, k \mapsto BlockedThread[c].k, v \mapsto BlockedThread[c].v])] \\
& \quad \wedge BlockedThread' = [BlockedThread \text{ EXCEPT } ![c] = Nil] \\
& \quad \quad \text{remove blocked state} \\
& \wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs, \\
& \quad ServerMsg, BlockedClient, OpCount, Pt, Cp, \\
& \quad CalState, State, SnapshotTable, History, \\
& \quad CurrentTerm, ReadyToServe, SyncSource \rangle
\end{aligned}$$

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

$ServerPutReply_majority_sleep \triangleq$

$$\begin{aligned}
& \wedge ReadyToServe > 0 \\
& \wedge \exists s \in Primary : \\
& \quad \wedge Len(InMsgs[s]) \neq 0 \\
& \quad \wedge InMsgs[s][1].op = \text{"put"} \\
& \quad \wedge InMsgs[s][1].wc = \text{"major"} \\
& \quad \wedge Tick(s) \\
& \quad \wedge Ot' = [Ot \text{ EXCEPT } ![s] = Ct'[s]] \\
& \quad \wedge Store' = [Store \text{ EXCEPT } ![s][InMsgs[s][1].k] = InMsgs[s][1].v] \\
& \quad \wedge Oplog' = \\
& \quad \quad \text{LET } entry \triangleq [k \mapsto InMsgs[s][1].k, v \mapsto InMsgs[s][1].v, ot \mapsto Ot'[s], term \mapsto CurrentTerm[s]] \\
& \quad \quad \quad newLog \triangleq Append(Oplog[s], entry) \\
& \quad \quad \text{IN } [Oplog \text{ EXCEPT } ![s] = newLog] \\
& \wedge State' = \\
& \quad \text{LET } SubHbState \triangleq State[s]
\end{aligned}$$

$$\begin{aligned}
&hb \triangleq [SubHbState \text{ EXCEPT } ![s] = Ot'[s]] \\
&\text{IN } [State \text{ EXCEPT } ![s] = hb] \text{ update primary state} \\
&\wedge CalState' = AdvanceState(Ot'[s], Ot[s], CalState) \\
&\wedge BlockedThread' = [BlockedThread \text{ EXCEPT } ![InMsgs[s][1].c] = [type \mapsto \text{"write_major"}, ot \\
&\quad \mapsto Ot'[s], s \mapsto s, k \mapsto InMsgs[s][1].k, v \mapsto InMsgs[s][1].v]] \\
&\wedge InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Tail(@)] \\
&\wedge \text{UNCHANGED } \langle Primary, Secondary, InMsgc, ServerMsg, BlockedClient, OpCount, \\
&\quad Pt, Cp, SnapshotTable, History, CurrentTerm, ReadyToServe, SyncSource \rangle
\end{aligned}$$

$$\begin{aligned}
&ServerPutReply_majority_wake \triangleq \\
&\wedge ReadyToServe > 0 \\
&\wedge \exists c \in Client : \\
&\quad \wedge BlockedThread[c] \neq Nil \\
&\quad \wedge BlockedThread[c].type = \text{"write_major"} \\
&\quad \wedge \neg HLClt(Cp[Primary], BlockedThread[c].ot) \\
&\quad \wedge InMsgc' = [InMsgc \text{ EXCEPT } ![c] = \\
&\quad \quad Append(@, [op \mapsto \text{"put"}, ct \mapsto Ct[BlockedThread[c].s], ot \mapsto BlockedThread[c].ot, \\
&\quad \quad k \mapsto BlockedThread[c].k, v \mapsto BlockedThread[c].v])] \\
&\quad \wedge BlockedThread' = [BlockedThread \text{ EXCEPT } ![c] = Nil] \\
&\wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgs, ServerMsg, \\
&\quad BlockedClient, OpCount, Pt, Cp, CalState, State, SnapshotTable, \\
&\quad History, CurrentTerm, ReadyToServe, SyncSource \rangle
\end{aligned}$$

Tunable Protocol: Client Actions

Client Get

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

$$\begin{aligned}
&ClientGetRequest_local_secondary \triangleq \\
&\wedge ReadyToServe > 0 \\
&\wedge \exists k \in Key, c \in Client \setminus BlockedClient, s \in Secondary : \\
&\quad \wedge InMsgs' = [InMsgs \text{ EXCEPT } ![s] = Append(@, \\
&\quad \quad [op \mapsto \text{"get"}, c \mapsto c, rc \mapsto \text{"local"}, k \mapsto k, ct \mapsto Ct[c], ot \mapsto Ot[c]])] \\
&\quad \wedge BlockedClient' = BlockedClient \cup \{c\} \\
&\wedge \text{UNCHANGED } \langle Primary, Secondary, Oplog, Store, Ct, Ot, InMsgc, ServerMsg,
\end{aligned}$$

BlockedThread, OpCount, Pt, Cp, CalState, State, SnapshotTable, History, CurrentTerm, ReadyToServe, SyncSource

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

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

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

Client Put

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

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

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

$\text{ClientPutRequest_number} \triangleq$

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

$\text{ClientPutRequest_majority} \triangleq$

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

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

$\text{ClientGetResponse} \triangleq$

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

CurrentTerm, ReadyToServe, SyncSource)

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

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

Action Wrapper

ClientGetRequest_local $\triangleq \vee \text{ClientGetRequest_local_primary}$
 $\vee \text{ClientGetRequest_local_secondary}$
ClientGetRequest_majority $\triangleq \vee \text{ClientGetRequest_majority_primary}$
 $\vee \text{ClientGetRequest_majority_secondary}$

all possible client get actions

ClientGetRequest $\triangleq \vee \text{ClientGetRequest_local}$
 $\vee \text{ClientGetRequest_majority}$
 $\vee \text{ClientGetRequest_linearizable}$

all possible client put actions

ClientPutRequest $\triangleq \vee \text{ClientPutRequest_zero}$
 $\vee \text{ClientPutRequest_number}$
 $\vee \text{ClientPutRequest_majority}$

all possible server get actions

ServerGetReply $\triangleq \vee \text{ServerGetReply_local_sleep}$
 $\vee \text{ServerGetReply_local_wake}$
 $\vee \text{ServerGetReply_majority_sleep}$
 $\vee \text{ServerGetReply_majority_wake}$

$$\begin{aligned} & \vee \text{ServerGetReply_linearizable_sleep} \\ & \vee \text{ServerGetReply_linearizable_wake} \end{aligned}$$

$$\begin{aligned} & \text{all possible server put actions} \\ \text{ServerPutReply} & \triangleq \vee \text{ServerPutReply_zero} \\ & \vee \text{ServerPutReply_number_sleep} \\ & \vee \text{ServerPutReply_majority_sleep} \\ & \vee \text{ServerPutReply_number_wake} \\ & \vee \text{ServerPutReply_majority_wake} \end{aligned}$$

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

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

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

$$\begin{aligned} & \text{Next state for all configurations} \\ \text{Next} & \triangleq \vee \text{ClientGetRequest} \vee \text{ClientPutRequest} \\ & \vee \text{ClientGetResponse} \vee \text{ClientPutResponse} \\ & \vee \text{ServerGetReply} \vee \text{ServerPutReply} \\ & \vee \text{ReplicateAction} \\ & \vee \text{AdvancePt} \\ & \vee \text{ServerTakeHeartbeat} \\ & \vee \text{ServerTakeUpdatePosition} \\ & \vee \text{Snapshot} \\ & \vee \text{Stepdown} \\ & \vee \text{RollbackOplogAction} \\ & \vee \text{TurnOnReadyToServe} \\ & \vee \text{ElectPrimaryAction} \end{aligned}$$

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

$$\begin{aligned} \text{Next_Except_ClientRequest} & \triangleq \vee \text{ClientGetResponse} \\ & \vee \text{ClientPutResponse} \\ & \vee \text{ServerGetReply} \\ & \vee \text{ServerPutReply} \\ & \vee \text{ReplicateAction} \\ & \vee \text{AdvancePt} \\ & \vee \text{ServerTakeHeartbeat} \\ & \vee \text{ServerTakeUpdatePosition} \\ & \vee \text{Snapshot} \\ & \vee \text{Stepdown} \\ & \vee \text{RollbackOplogAction} \\ & \vee \text{TurnOnReadyToServe} \end{aligned}$$

$\vee \text{ElectPrimaryAction}$

$$\begin{aligned}
\text{ClientRequest}_1 &\triangleq \vee \text{ClientPutRequest_majority} \\
&\quad \vee \text{ClientGetRequest_local_primary} \\
\text{ClientRequest}_2 &\triangleq \vee \text{ClientPutRequest_majority} \\
&\quad \vee \text{ClientGetRequest_local_secondary} \\
\text{ClientRequest}_3 &\triangleq \vee \text{ClientPutRequest_majority} \\
&\quad \vee \text{ClientGetRequest_majority_primary} \\
\text{ClientRequest}_4 &\triangleq \vee \text{ClientPutRequest_majority} \\
&\quad \vee \text{ClientGetRequest_majority_secondary} \\
\text{ClientRequest}_5 &\triangleq \vee \text{ClientPutRequest_majority} \\
&\quad \vee \text{ClientGetRequest_linearizable} \\
\text{ClientRequest}_6 &\triangleq \vee \text{ClientPutRequest_number} \\
&\quad \vee \text{ClientGetRequest_local_primary} \\
\text{ClientRequest}_7 &\triangleq \vee \text{ClientPutRequest_number} \\
&\quad \vee \text{ClientGetRequest_local_secondary} \\
\text{ClientRequest}_8 &\triangleq \vee \text{ClientPutRequest_number} \\
&\quad \vee \text{ClientGetRequest_majority_primary} \\
\text{ClientRequest}_9 &\triangleq \vee \text{ClientPutRequest_number} \\
&\quad \vee \text{ClientGetRequest_majority_secondary} \\
\text{ClientRequest}_{10} &\triangleq \vee \text{ClientPutRequest_number} \\
&\quad \vee \text{ClientGetRequest_linearizable} \\
\text{Next1} &\triangleq \vee \text{Next_Except_ClientRequest} \\
&\quad \vee \text{ClientRequest}_1 \\
\text{Next2} &\triangleq \vee \text{Next_Except_ClientRequest} \\
&\quad \vee \text{ClientRequest}_2 \\
\text{Next3} &\triangleq \vee \text{Next_Except_ClientRequest} \\
&\quad \vee \text{ClientRequest}_3 \\
\text{Next4} &\triangleq \vee \text{Next_Except_ClientRequest} \\
&\quad \vee \text{ClientRequest}_4 \\
\text{Next5} &\triangleq \vee \text{Next_Except_ClientRequest} \\
&\quad \vee \text{ClientRequest}_5 \\
\text{Next6} &\triangleq \vee \text{Next_Except_ClientRequest} \\
&\quad \vee \text{ClientRequest}_6
\end{aligned}$$

$$Next7 \triangleq \vee Next_Except_ClientRequestet \\ \vee ClientRequestet_7$$

$$Next8 \triangleq \vee Next_Except_ClientRequestet \\ \vee ClientRequestet_8$$

$$Next9 \triangleq \vee Next_Except_ClientRequestet \\ \vee ClientRequestet_9$$

$$Next10 \triangleq \vee Next_Except_ClientRequestet \\ \vee ClientRequestet_10$$

$$Spec1 \triangleq Init \wedge \Box [Next1]_{vars}$$

$$Spec2 \triangleq Init \wedge \Box [Next2]_{vars}$$

$$Spec3 \triangleq Init \wedge \Box [Next3]_{vars}$$

$$Spec4 \triangleq Init \wedge \Box [Next4]_{vars}$$

$$Spec5 \triangleq Init \wedge \Box [Next5]_{vars}$$

$$Spec6 \triangleq Init \wedge \Box [Next6]_{vars}$$

$$Spec7 \triangleq Init \wedge \Box [Next7]_{vars}$$

$$Spec8 \triangleq Init \wedge \Box [Next8]_{vars}$$

$$Spec9 \triangleq Init \wedge \Box [Next9]_{vars}$$

$$Spec10 \triangleq Init \wedge \Box [Next10]_{vars}$$

Causal Specifications

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

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

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

$$WriteFollowRead \triangleq \forall c \in Client : \forall i, j \in \text{DOMAIN } History[c] : \\ \wedge i < j \\ \wedge History[c][i].op = \text{"get"} \\ \wedge History[c][j].op = \text{"put"}$$

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

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