
MODULE *AggCount*

EXTENDS *TLC, Integers, FiniteSets*

CONSTANTS *Dataset, Storage, nil*

VARIABLES *replicas, pending_counters*

vars $\triangleq \langle replicas, pending_counters \rangle$

min_repl_id $\triangleq 21$

max_repl_id $\triangleq 25$

ReplicaID $\triangleq min_repl_id \dots max_repl_id$

Status $\triangleq \{ \text{"pending"}, \text{"written"} \}$

AggStatus $\triangleq \{ \text{"need_include"}, \text{"no_action"}, \text{"need_remove"} \}$

ReplicaInfo $\triangleq [ds : Dataset, status : Status, storage : Storage, agg : AggStatus]$

Replica $\triangleq [ReplicaID \rightarrow ReplicaInfo \cup \{nil\}]$

PendingKey $\triangleq Dataset \times Storage$

PendingInfo $\triangleq [count : 0 \dots 100, need_update : BOOLEAN, version : 0 \dots 500]$

TypeOK \triangleq
 $\wedge replicas \in Replica$
 $\wedge pending_counters \in [PendingKey \rightarrow PendingInfo]$

initCounter $\triangleq [count \mapsto 0, need_update \mapsto FALSE, version \mapsto 0]$

Init \triangleq
 $\wedge replicas = [id \in ReplicaID \mapsto nil]$
 $\wedge pending_counters = [k \in PendingKey \mapsto initCounter]$

addReplicaImpl(id, ds, st) \triangleq
 LET
 new_repl $\triangleq [$
 ds $\mapsto ds, status \mapsto \text{"pending"},$
 storage $\mapsto st, agg \mapsto \text{"need_include"}]$
 key $\triangleq \langle ds, st \rangle$
 old_counter $\triangleq pending_counters[key]$
 new_counter $\triangleq [old_counter \text{ EXCEPT } !.need_update = TRUE, !.version = @ + 1]$
 IN
 $\wedge replicas' = [replicas \text{ EXCEPT } ![id] = new_repl]$
 $\wedge pending_counters' = [pending_counters \text{ EXCEPT } ![key] = new_counter]$

$$\begin{aligned}
& \text{AddReplica}(id, ds, st) \triangleq \\
& \quad \wedge \text{replicas}[id] = \text{nil} \\
& \quad \wedge \text{addReplicaImpl}(id, ds, st) \\
\\
& \text{updateCounterAfterWritten}(r) \triangleq \\
& \quad \text{LET} \\
& \quad \quad k \triangleq \langle r.ds, r.storage \rangle \\
& \quad \text{IN} \\
& \quad \quad \text{pending_counters}' = [\\
& \quad \quad \quad \text{pending_counters EXCEPT } ![k] = [\\
& \quad \quad \quad \quad @ \text{ EXCEPT } !.need_update = \text{TRUE}, !.version = @ + 1 \\
& \quad \quad \quad] \\
& \quad \quad] \\
\\
& \text{computeAggStatusForWritten}(old_val) \triangleq \\
& \quad \text{IF } old_val = \text{"no_action"} \\
& \quad \quad \text{THEN "need_remove"} \\
& \quad \quad \text{ELSE "no_action"} \\
\\
& \text{doUpdateReplicaToWritten}(id) \triangleq \\
& \quad \text{LET} \\
& \quad \quad old_repl \triangleq \text{replicas}[id] \\
& \quad \quad need_remove_cond \triangleq \\
& \quad \quad \quad \vee \wedge old_repl.status = \text{"pending"} \\
& \quad \quad \quad \wedge old_repl.agg = \text{"no_action"} \\
& \quad \quad \quad \vee \wedge old_repl.status = \text{"written"} \\
& \quad \quad \quad \wedge old_repl.agg = \text{"need_remove"} \\
& \quad \quad new_agg \triangleq \\
& \quad \quad \quad \text{IF } need_remove_cond \\
& \quad \quad \quad \quad \text{THEN "need_remove"} \\
& \quad \quad \quad \quad \text{ELSE "no_action"} \\
& \quad \quad new_repl \triangleq [old_repl \text{ EXCEPT } !.status = \text{"written"}, !.agg = new_agg] \\
& \quad \text{IN} \\
& \quad \quad replicas' = [replicas \text{ EXCEPT } ![id] = new_repl] \\
\\
& \text{UpdateToWritten}(id) \triangleq \\
& \quad \wedge \text{replicas}[id] \neq \text{nil} \\
& \quad \wedge \text{replicas}[id].status \neq \text{"written"} \\
& \quad \wedge \text{doUpdateReplicaToWritten}(id) \\
& \quad \wedge \text{updateCounterAfterWritten}(\text{replicas}[id])
\end{aligned}$$

$$\begin{aligned}
& replicaHasKey(id, k) \triangleq \\
& \quad \wedge replicas[id] \neq nil \\
& \quad \wedge replicas[id].ds = k[1] \\
& \quad \wedge replicas[id].storage = k[2] \\
\\
& getPendingReplicas(k) \triangleq \\
& \quad LET \\
& \quad \quad selectCond(id) \triangleq \\
& \quad \quad \quad \wedge replicaHasKey(id, k) \\
& \quad \quad \quad \wedge replicas[id].status = \text{"pending"} \\
& \quad IN \\
& \quad \quad \{id \in ReplicaID : selectCond(id)\} \\
\\
& setAggToNoAction(k) \triangleq \\
& \quad LET \\
& \quad \quad new_fn(id) \triangleq \\
& \quad \quad \quad IF replicaHasKey(id, k) \\
& \quad \quad \quad \quad THEN [replicas[id] EXCEPT !.agg = \text{"no_action"}] \\
& \quad \quad \quad \quad ELSE replicas[id] \text{ unchanged} \\
& \quad IN \\
& \quad \quad replicas' = [id \in ReplicaID \mapsto new_fn(id)] \\
\\
& doUpdatePendingCounter(k) \triangleq \\
& \quad LET \\
& \quad \quad pending_repls \triangleq getPendingReplicas(k) \\
& \quad \quad num \triangleq Cardinality(pending_repls) \\
& \quad \quad old_counter \triangleq pending_counters[k] \\
& \quad \quad new_counter \triangleq [old_counter EXCEPT !.count = num, !.need_update = FALSE] \\
& \quad IN \\
& \quad \quad \wedge pending_counters' = [pending_counters EXCEPT ![k] = new_counter] \\
& \quad \quad \wedge setAggToNoAction(k) \\
\\
& UpdatePendingCounter(k) \triangleq \\
& \quad \wedge pending_counters[k].need_update = TRUE \\
& \quad \wedge doUpdatePendingCounter(k) \\
\\
& TerminateCond \triangleq \\
& \quad \wedge \forall id \in ReplicaID : \\
& \quad \quad \wedge replicas[id] \neq nil \\
& \quad \quad \wedge replicas[id].agg = \text{"no_action"} \\
& \quad \wedge \forall key \in PendingKey : pending_counters[key].need_update = FALSE \\
\\
& Terminated \triangleq \\
& \quad \wedge TerminateCond
\end{aligned}$$

$\wedge \text{UNCHANGED } vars$

$Next \triangleq$
 $\vee \exists id \in ReplicaID, ds \in Dataset, st \in Storage :$
 $\quad \vee AddReplica(id, ds, st) \text{ } \textcolor{gray}{TODO \text{ Add Written Replica}}$
 $\vee \exists id \in ReplicaID :$
 $\quad UpdateToWritten(id)$
 $\vee \exists k \in PendingKey :$
 $\quad UpdatePendingCounter(k)$
 $\vee Terminated$

$allPendingReplicas(k) \triangleq$
 LET
 $\quad checkCond(id) \triangleq$
 $\quad \quad \wedge replicaHasKey(id, k)$
 $\quad \quad \wedge replicas[id].status = \text{"pending"}$
 $\quad S \triangleq \{id \in ReplicaID : checkCond(id)\}$
 IN
 $\quad Cardinality(S)$

$numPendingByCounter(k) \triangleq$
 LET
 $\quad isPending(id) \triangleq$
 $\quad \quad \wedge replicaHasKey(id, k)$
 $\quad \quad \wedge replicas[id].agg = \text{"need_include"}$
 $\quad isNonPending(id) \triangleq$
 $\quad \quad \wedge replicaHasKey(id, k)$
 $\quad \quad \wedge replicas[id].agg = \text{"need_remove"}$
 $\quad S1 \triangleq \{id \in ReplicaID : isPending(id)\}$
 $\quad S2 \triangleq \{id \in ReplicaID : isNonPending(id)\}$
 IN
 $\quad Cardinality(S1) + pending_counters[k].count - Cardinality(S2)$

$Inv \triangleq$
 $\quad \wedge \forall k \in PendingKey :$
 $\quad \quad allPendingReplicas(k) = numPendingByCounter(k)$

$Sym \triangleq Permutations(Dataset) \cup Permutations(Storage)$