
MODULE *AggCount*

EXTENDS *TLC, Integers, FiniteSets*

CONSTANTS *Dataset, Storage, nil*

VARIABLES *replicas, pending_counters*

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

$min_repl_id \triangleq 21$

$max_repl_id \triangleq 24$

$ReplicaID \triangleq min_repl_id .. max_repl_id$

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

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

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

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

$PendingKey \triangleq Dataset \times Storage$

$PendingInfo \triangleq [count : 0 .. 100, need_update : BOOLEAN]$

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

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

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

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

$AddPendingReplica(id, ds, st) \triangleq$
 $\wedge replicas[id] = nil$

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

$$replicas' = [replicas \text{ EXCEPT } ![id] = new_repl]$$

$$\begin{aligned} UpdateToWritten(id) &\triangleq \\ &\wedge replicas[id] \neq nil \\ &\wedge replicas[id].status \neq \text{"written"} \\ &\wedge doUpdateReplicaToWritten(id) \\ &\wedge updateCounterAfterWritten(replicas[id]) \end{aligned}$$

$$\begin{aligned} replicaHasKey(id, k) &\triangleq \\ &\wedge replicas[id] \neq nil \\ &\wedge replicas[id].ds = k[1] \\ &\wedge replicas[id].storage = k[2] \end{aligned}$$

$$\begin{aligned} getPendingReplicas(k) &\triangleq \\ \text{LET} & \\ &selectCond(id) \triangleq \\ &\quad \wedge replicaHasKey(id, k) \\ &\quad \wedge replicas[id].status = \text{"pending"} \\ \text{IN} & \\ &\{id \in ReplicaID : selectCond(id)\} \end{aligned}$$

$$\begin{aligned} setAggToNoAction(k) &\triangleq \\ \text{LET} & \\ &new_fn(id) \triangleq \\ &\quad \text{IF } replicaHasKey(id, k) \\ &\quad \quad \text{THEN } [replicas[id] \text{ EXCEPT } !.agg = \text{"no_action"}] \\ &\quad \quad \text{ELSE } replicas[id] \text{ unchanged} \\ \text{IN} & \\ &replicas' = [id \in ReplicaID \mapsto new_fn(id)] \end{aligned}$$

$$\begin{aligned} doUpdatePendingCounter(k) &\triangleq \\ \text{LET} & \\ &pending_repls \triangleq getPendingReplicas(k) \\ &num \triangleq Cardinality(pending_repls) \\ &old_counter \triangleq pending_counters[k] \\ &new_counter \triangleq [old_counter \text{ EXCEPT } !.count = num, !.need_update = \text{FALSE}] \\ \text{IN} & \\ &\wedge pending_counters' = [pending_counters \text{ EXCEPT } ![k] = new_counter] \\ &\wedge setAggToNoAction(k) \end{aligned}$$

$$\begin{aligned} UpdatePendingCounter(k) &\triangleq \\ &\wedge pending_counters[k].need_update = \text{TRUE} \\ &\wedge doUpdatePendingCounter(k) \end{aligned}$$

$$\begin{aligned}
\textit{TerminateCond} &\triangleq \\
&\wedge \forall id \in \textit{ReplicaID} : \\
&\quad \wedge \textit{replicas}[id] \neq \textit{nil} \\
&\quad \wedge \textit{replicas}[id].\textit{agg} = \text{"no_action"} \\
&\wedge \forall key \in \textit{PendingKey} : \textit{pending_counters}[key].\textit{need_update} = \text{FALSE}
\end{aligned}$$

$$\begin{aligned}
\textit{Terminated} &\triangleq \\
&\wedge \textit{TerminateCond} \\
&\wedge \text{UNCHANGED } \textit{vars}
\end{aligned}$$

$$\begin{aligned}
\textit{Next} &\triangleq \\
&\vee \exists id \in \textit{ReplicaID}, ds \in \textit{Dataset}, st \in \textit{Storage} : \\
&\quad \vee \textit{AddPendingReplica}(id, ds, st) \\
&\quad \vee \textit{AddWrittenReplica}(id, ds, st) \\
&\vee \exists id \in \textit{ReplicaID} : \\
&\quad \textit{UpdateToWritten}(id) \\
&\vee \exists k \in \textit{PendingKey} : \\
&\quad \textit{UpdatePendingCounter}(k) \\
&\vee \textit{Terminated}
\end{aligned}$$

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

$$\textit{FairSpec} \triangleq \textit{Spec} \wedge \text{WF}_{\textit{vars}}(\textit{Next})$$

$$\textit{AlwaysTerminate} \triangleq \Diamond \textit{TerminateCond}$$

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

$$\begin{aligned}
\textit{numPendingByCounter}(k) &\triangleq \\
&\text{LET} \\
&\quad \textit{isPending}(id) \triangleq \\
&\quad \quad \wedge \textit{replicaHasKey}(id, k) \\
&\quad \quad \wedge \textit{replicas}[id].\textit{agg} = \text{"need_include"} \\
&\quad \textit{isNonPending}(id) \triangleq \\
&\quad \quad \wedge \textit{replicaHasKey}(id, k) \\
&\quad \quad \wedge \textit{replicas}[id].\textit{agg} = \text{"need_remove"}
\end{aligned}$$

$$\begin{aligned}
S1 &\triangleq \{id \in ReplicaID : isPending(id)\} \\
S2 &\triangleq \{id \in ReplicaID : isNonPending(id)\} \\
IN & \quad Cardinality(S1) + pending_counters[k].count - Cardinality(S2) \\
Inv &\triangleq \\
&\quad \wedge \forall k \in PendingKey : \\
&\quad \quad allPendingReplicas(k) = numPendingByCounter(k) \\
Sym &\triangleq Permutations(Dataset) \cup Permutations(Storage)
\end{aligned}$$
