```
──────────────────────── MODULE AggCount ────────────────────────
EXTENDS TLC, Integers, FiniteSets

CONSTANTS Dataset, Storage, nil

VARIABLES replicas, pending_counters

vars ≜ ⟨replicas, pending_counters⟩

min_repl_id ≜ 21
max_repl_id ≜ 25

ReplicaID ≜ min_repl_id . . max_repl_id

Status ≜ {"pending", "written"}

AggStatus ≜ {"need_include", "no_action", "need_remove"}

ReplicaInfo ≜ [ds : Dataset, status : Status, storage : Storage, agg : AggStatus]

Replica ≜ [ReplicaID → ReplicaInfo ∪ {nil}]

PendingKey ≜ Dataset × Storage

PendingInfo ≜ [count : 0 . . 100, need_update : BOOLEAN , version : 0 . . 500]

TypeOK ≜
    ∧  replicas ∈ Replica
    ∧  pending_counters ∈ [PendingKey → PendingInfo]

initCounter ≜ [count ↦ 0, need_update ↦ FALSE, version ↦ 0]

Init ≜
    ∧ replicas = [id ∈ ReplicaID ↦ nil]
    ∧ pending_counters = [k ∈ PendingKey ↦ initCounter]

addReplicaImpl(id, ds, st) ≜
   LET
       new_repl ≜ [
           ds ↦ ds, status ↦ "pending",
           storage ↦ st, agg ↦ "need_include"]
       key ≜ ⟨ds, st⟩
       old_counter  ≜ pending_counters[key]
       new_counter ≜ [old_counter EXCEPT !.need_update = TRUE, !.version = @ + 1]
   IN
       ∧ replicas′ = [replicas EXCEPT ![id] = new_repl]
       ∧ pending_counters′ = [pending_counters EXCEPT ![key] = new_counter]
```

```
$AddReplica(id, ds, st) \triangleq$
    $\land replicas[id] = nil$
    $\land addReplicaImpl(id, ds, st)$

$updateCounterAfterWritten(r) \triangleq$
   LET
       $k \triangleq \langle r.ds, r.storage \rangle$
   IN
       $pending\_counters' = [$
           $pending\_counters$ EXCEPT $![k] = [$
               $@$ EXCEPT $!.need\_update = $ TRUE, $!.version = @ + 1$
           $]$
       $]$

$computeAggStatusForWritten(old\_val) \triangleq$
   IF $old\_val = $ "no_action"
       THEN "need_remove"
       ELSE "no_action"

$doUpdateReplicaToWritten(id) \triangleq$
   LET
       $old\_repl \triangleq replicas[id]$

       $need\_remove\_cond \triangleq$
           $\lor \land old\_repl.status = $ "pending"
             $\land old\_repl.agg = $ "no_action"
           $\lor \land old\_repl.status = $ "written"
             $\land old\_repl.agg = $ "need_remove"

       $new\_agg \triangleq$
           IF $need\_remove\_cond$
               THEN "need_remove"
               ELSE "no_action"

       $new\_repl \triangleq [old\_repl$ EXCEPT $!.status = $ "written", $!.agg = new\_agg]$
   IN
       $replicas' = [replicas$ EXCEPT $![id] = new\_repl]$

$UpdateToWritten(id) \triangleq$
    $\land replicas[id] \neq nil$
    $\land replicas[id].status \neq $ "written"
    $\land doUpdateReplicaToWritten(id)$
    $\land updateCounterAfterWritten(replicas[id])$
```

```
$replicaHasKey(id, k) \triangleq$
    $\land replicas[id] \neq nil$
    $\land replicas[id].ds = k[1]$
    $\land replicas[id].storage = k[2]$

$getPendingReplicas(k) \triangleq$
   LET
      $selectCond(id) \triangleq$
         $\land replicaHasKey(id, k)$
         $\land replicas[id].status =$ "pending"
   IN
      $\{id \in ReplicaID : selectCond(id)\}$

$setAggToNoAction(k) \triangleq$
   LET
      $new\_fn(id) \triangleq$
         IF $replicaHasKey(id, k)$
            THEN $[replicas[id]$ EXCEPT $!.agg =$ "no_action"$]$
            ELSE $replicas[id]$ unchanged
   IN
      $replicas' = [id \in ReplicaID \mapsto new\_fn(id)]$

$doUpdatePendingCounter(k) \triangleq$
   LET
      $pending\_repls \triangleq getPendingReplicas(k)$
      $num \triangleq Cardinality(pending\_repls)$

      $old\_counter \triangleq pending\_counters[k]$
      $new\_counter \triangleq [old\_counter$ EXCEPT $!.count = num, !.need\_update =$ FALSE$]$
   IN
      $\land pending\_counters' = [pending\_counters$ EXCEPT $![k] = new\_counter]$
      $\land setAggToNoAction(k)$

$UpdatePendingCounter(k) \triangleq$
    $\land pending\_counters[k].need\_update =$ TRUE
    $\land doUpdatePendingCounter(k)$

$TerminateCond \triangleq$
    $\land \forall id \in ReplicaID :$
       $\land replicas[id] \neq nil$
       $\land replicas[id].agg =$ "no_action"
    $\land \forall key \in PendingKey : pending\_counters[key].need\_update =$ FALSE

$Terminated \triangleq$
    $\land TerminateCond$
    $\land$ UNCHANGED $vars$

$Next \triangleq$
    $\lor \exists\, id \in ReplicaID,\ ds \in Dataset,\ st \in Storage :$
        $\lor AddReplica(id,\ ds,\ st)$  $TODO$ Add Written $Replica$
    $\lor \exists\, id \in ReplicaID :$
        $UpdateToWritten(id)$
    $\lor \exists\, k \in PendingKey :$
        $UpdatePendingCounter(k)$
    $\lor Terminated$

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

$FairSpec \triangleq Spec \land \mathrm{WF}_{vars}(Next)$

$allPendingReplicas(k) \triangleq$
    LET
        $checkCond(id) \triangleq$
            $\land replicaHasKey(id,\ k)$
            $\land replicas[id].status =$ “pending”
        $S \triangleq \{id \in ReplicaID : checkCond(id)\}$
    IN
        $Cardinality(S)$

$numPendingByCounter(k) \triangleq$
    LET
        $isPending(id) \triangleq$
            $\land replicaHasKey(id,\ k)$
            $\land replicas[id].agg =$ “need_include”

        $isNonPending(id) \triangleq$
            $\land replicaHasKey(id,\ k)$
            $\land replicas[id].agg =$ “need_remove”

        $S1 \triangleq \{id \in ReplicaID : isPending(id)\}$
        $S2 \triangleq \{id \in ReplicaID : isNonPending(id)\}$
    IN
        $Cardinality(S1) + pending\_counters[k].count - Cardinality(S2)$

$Inv \triangleq$
    $\land \forall\, k \in PendingKey :$
        $allPendingReplicas(k) = numPendingByCounter(k)$

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