MODULE *UpdateCluster*

EXTENDS *Integers*, *FiniteSets*

CONSTANTS
- $\_Requests$, the requests sent by the user
- $\_Workers$, the pool of workers
- $NULL$

VARIABLES
- $lastVOK$, last successfully applied version
- $toApply$, the version to apply (last request that passed the initial tests)
- $cluster$, cluster state
- $requests$, the state of all requests
- $workers$, the state of all workers
- $clusterUpdating$ damn, I used a lock ...

VARIABLES

these variables are *tla+* details
- $confOK$, are we able to get a valid conf ?
- $reqCounter$ just to keep track of the order of submissions

$vars \triangleq \langle confOK, reqCounter, lastVOK, toApply, cluster, requests, workers, clusterUpdating \rangle$

$TypeInvariants \triangleq$
- $\land\ confOK \in$ BOOLEAN won't change for a specific behavior
- $\land\ clusterUpdating \in$ BOOLEAN
- $\land\ cluster.st \in \{$
  - "idle",
  - "starting",
  - "partial",
  - "failed"
  - $\}$
- $\land \forall\, r \in \_Requests : requests[r].st \in \{$
  - "waiting", the request (req) hasn't been submitted yet
  - "submitted", req has been submitted
  - "rejected", req has been rejected (auth problem)
  - "valid" auth etc passed
  - $\}$
- $\land \forall\, w \in \_Workers : workers[w].st \in \{$
  - "waiting",
  - "starting",
  - "working"
  - $\}$

Initial State

$Init \triangleq$
$\land\ requests = [r \in \_Requests \mapsto [st \mapsto$ "waiting", $v \mapsto NULL]]$
$\land\ workers\ = [w \in \_Workers \mapsto [st \mapsto$ "waiting", $v \mapsto NULL]]$
$\land\ cluster\ = [v \mapsto 0,\ st \mapsto$ "idle"$]$
$\land\ lastVOK = 0$
$\land\ reqCounter = 0$
$\land\ toApply = 0$
$\land\ confOK \in$ BOOLEAN
$\land\ clusterUpdating =$ FALSE

Actions

$Submit(r) \triangleq$ update request received from the user
LET $newV \triangleq reqCounter + 1$IN
$\land\ requests[r].st =$ "waiting"
$\land\ reqCounter' = newV$
$\land\ requests' = [requests$ EXCEPT $![r].st =$ "submitted", $![r].v = newV]$
$\land$ UNCHANGED $\langle confOK, lastVOK, toApply, cluster, workers, clusterUpdating\rangle$

$PushToPending(r) \triangleq$ the request is pushed to queue
$\land\ requests[r].st =$ "submitted"
$\land$ IF $toApply < requests[r].v$
THEN $\land\ requests' = [requests$ EXCEPT $![r].st =$ "valid"$]$
$\land\ toApply' = requests[r].v$
$\land$ UNCHANGED $\langle confOK, reqCounter, lastVOK, cluster, workers, clusterUpdating\rangle$
ELSE $\land\ requests' = [requests$ EXCEPT $![r].st =$ "rejected"$]$
$\land$ UNCHANGED $\langle confOK, reqCounter, lastVOK, toApply, cluster, workers, clusterUpdating\rangle$

$SpawnWorker(w) \triangleq$ spawns a new worker
$\land\ workers[w].st =$ "waiting"
$\land\ toApply \neq lastVOK$
$\land\ clusterUpdating =$ FALSE
$\land\ \ \lor cluster.st =$ "idle"
$\lor cluster.st =$ "failed"
$\land\ workers' = [workers$ EXCEPT $![w].v = toApply,\ ![w].st =$ "starting"$]$
$\land\ clusterUpdating' =$ TRUE
$\land$ UNCHANGED $\langle confOK, reqCounter, lastVOK, toApply, requests, cluster\rangle$

$ApplyStart(w) \triangleq$ the cluster starts to be modified
$\land\ workers[w].st =$ "starting"
$\land$ IF $workers[w].v \geq toApply$
THEN
IF $confOK$

```
            THEN
                ∧ cluster′  = [v ↦ workers[w].v, st ↦ "partial"]
                ∧ workers′ = [workers EXCEPT ![w].st = "working"]
                ∧ UNCHANGED ⟨confOK, reqCounter, lastVOK, toApply, requests, clusterUpdating⟩
            ELSE
                ∧ clusterUpdating′ = FALSE
                ∧ workers′ = [workers EXCEPT ![w].st = "waiting", ![w].v = NULL]
                ∧ UNCHANGED ⟨confOK, reqCounter, lastVOK, toApply, cluster, requests⟩
        ELSE    a new version has been submitted, no need to apply this one
            ∧ workers′ = [workers EXCEPT ![w].st = "waiting", ![w].v = NULL]
            ∧ clusterUpdating′ = FALSE
            ∧ UNCHANGED ⟨confOK, reqCounter, lastVOK, toApply, cluster, requests⟩

RollbackVersion ≜
    to differenciate it from the original last VOK (in realworld, could be conf + timestamp)
    lastVOK + 10

ApplyFinish(w) ≜   the cluster update finishes
    ∧ workers[w].st = "working"
    ∧ clusterUpdating′ = FALSE
    ∧ ∃ ok ∈ BOOLEAN :
        IF ok ∨ workers[w].v = RollbackVersion   rollback always works
            THEN
                ∧ cluster′  = [cluster EXCEPT !.st = "idle"]
                ∧ lastVOK′ = workers[w].v
                ∧ workers′  = [workers EXCEPT ![w].st = "waiting", ![w].v = NULL]
                ∧ UNCHANGED ⟨confOK, reqCounter, toApply, requests⟩
            ELSE
                ∧ cluster′  = [cluster EXCEPT !.st = "failed"]
                ∧ workers′ = [workers EXCEPT ![w].st = "waiting", ![w].v = NULL]
                ∧ IF workers[w].v < toApply
                    THEN   a newer version has been submitted
                        ∧ UNCHANGED ⟨confOK, reqCounter, lastVOK, toApply, requests⟩
                    ELSE   let's trigger a rollback
                        ∧ toApply′ = RollbackVersion
                        ∧ UNCHANGED ⟨confOK, reqCounter, lastVOK, requests⟩


Requirements

NoConcurrentUpdate ≜
    □(Cardinality({r ∈ DOMAIN requests : requests[r].st = "working"}) < 2)

NoPartialUpdateTermination ≜   we don't want the cluster to end up in a partially update st
    ◇□(cluster.st = "idle")
```

$$EveryReqIsProcessed \triangleq$$
$$\Diamond\Box(\neg\exists\, r \in \_Requests : requests[r].st = \text{“waiting”})$$

Spec

$$Next \triangleq$$
$$\begin{array}{l} \lor \exists\, r \in \_Requests : \\ \quad \lor Submit(r) \\ \quad \lor PushToPending(r) \\ \lor \exists\, w \in \_Workers : \\ \quad \lor SpawnWorker(w) \\ \quad \lor ApplyStart(w) \\ \quad \lor ApplyFinish(w) \end{array}$$

$$\begin{array}{l} Fairness \triangleq \forall\, r \in \_Requests,\, w \in \_Workers : \\ \qquad \land \mathrm{WF}_{vars}(Submit(r)) \\ \qquad \land \mathrm{WF}_{vars}(PushToPending(r)) \\ \qquad \land \mathrm{WF}_{vars}(SpawnWorker(w)) \\ \qquad \land \mathrm{WF}_{vars}(ApplyStart(w)) \\ \qquad \land \mathrm{WF}_{vars}(ApplyFinish(w)) \end{array}$$

$$\begin{array}{l} Spec \triangleq \\ \quad \land Init \\ \quad \land \Box[Next]_{vars} \\ \quad \land Fairness \end{array}$$

THEOREM $Spec \Rightarrow \Box(TypeInvariants)$
THEOREM $Spec \Rightarrow NoPartialUpdateTermination$