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 use 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

request actions
$Submit(r) \triangleq$
update request received
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
$\land\ requests[r].st =$ "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 $\langle confOK, reqCounter, lastVOK, toApply, cluster, workers, clusterUpdating\rangle$

worker action
$SpawnWorker(w) \triangleq$
spawns a new worker
$\land\ toApply \neq lastVOK$
$\land\ workers[w].st =$ "waiting"
$\land\ clusterUpdating =$ 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

$\land\ workers[w].st = \text{“starting”}$

$\land$ IF $workers[w].v \geq toApply$

THEN

IF $confOK$

THEN

$\land\ cluster' = [v \mapsto workers[w].v,\ st \mapsto \text{“partial”}]$

$\land\ workers' = [workers \text{ EXCEPT } ![w].st = \text{“working”}]$

$\land$ UNCHANGED $\langle confOK, reqCounter, lastVOK, toApply, requests, clusterUpdating\rangle$

ELSE

$\land\ clusterUpdating' = \text{FALSE}$

$\land\ workers' = [workers \text{ EXCEPT } ![w].st = \text{“waiting”},\ ![w].v = NULL]$

$\land$ UNCHANGED $\langle confOK, reqCounter, lastVOK, toApply, cluster, requests\rangle$

ELSE a new version has been submitted, no need to apply this one

$\land\ workers' = [workers \text{ EXCEPT } ![w].st = \text{“waiting”},\ ![w].v = NULL]$

$\land\ clusterUpdating' = \text{FALSE}$

$\land$ UNCHANGED $\langle confOK, reqCounter, lastVOK, toApply, cluster, requests\rangle$

$RollbackVersion \triangleq$

to differenciate it from the original last VOK (in realworld, could be $conf +$ timestamp)

$lastVOK + 10$

$ApplyFinish(w) \triangleq$

the cluster update finishes

$\land\ workers[w].st = \text{“working”}$

$\land\ clusterUpdating' = \text{FALSE}$

$\land\ \exists\, ok \in \text{BOOLEAN} :$

IF $ok \lor workers[w].v = RollbackVersion$ rollback always works

THEN

$\land\ cluster' = [cluster \text{ EXCEPT } !.st = \text{“idle”}]$

$\land\ lastVOK' = workers[w].v$

$\land\ workers' = [workers \text{ EXCEPT } ![w].st = \text{“waiting”},\ ![w].v = NULL]$

$\land$ UNCHANGED $\langle confOK, reqCounter, toApply, requests\rangle$

ELSE

$\land\ cluster' = [cluster \text{ EXCEPT } !.st = \text{“failed”}]$

$\land\ workers' = [workers \text{ EXCEPT } ![w].st = \text{“waiting”},\ ![w].v = NULL]$

$\land$ IF $workers[w].v < toApply$

THEN a newer version has been submitted

$\land$ UNCHANGED $\langle confOK, reqCounter, lastVOK, toApply, requests\rangle$

ELSE let's trigger a rollback

$\land\ toApply' = RollbackVersion$

$\land$ UNCHANGED $\langle confOK, reqCounter, lastVOK, requests\rangle$

Spec

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

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

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

Expectations

$$
NoConcurrentUpdate \triangleq
\quad \Box(Cardinality(\{w \in \text{DOMAIN } workers : workers[w].st = \text{“working”}\}) < 2)
$$

$$
NoPartialUpdateTermination \triangleq
$$

we don't want the cluster to end up in a partially update state

$$
\quad \Diamond\Box(cluster.st = \text{“idle”})
$$

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

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