MODULE *kcpStorage*

Goal: Modeling interaction of controllers/users to ensure data safety as data and workloads are created, deleted, used, and migrated across clusters

****

Processes:

- Controllers

- Users

Constants:

- Clusters - Set of locations where things can be deployed.

- *Namepaces* - Set of kcp namespaces. For now, skip this and only model a single *NS*. Extend the model once we have cross *NS* interaction

- *PVCs* - Set of *PVCs*. Since they don't interact w/ each other, we'll only model one

- *SyncStates* - "nil", "Empty", "Sync"

State:

- *ns* - record mapping Clusters to *SyncStates*. This represents the state/A:"" label on the namespace

- *pvc* - record mapping Clusters to *SyncStates*, representing the state/A label on the *PVC*.

****

EXTENDS *FiniteSets*, *Naturals*, *Sequences*, *TLC*

CONSTANTS
CLUSTERS, The set of clusters that workloads can be assigned to.
NC, *PVCC*, *U* Model values

ASSUME
$Cardinality(CLUSTERS) > 0$

$SyncStates \triangleq \{\text{"nil"}, \text{"Empty"}, \text{"Sync"}\}$

--fair algorithm *kcpStorage*
variables
Namespaces start unassigned to any cluster
$ns = [c \in CLUSTERS \mapsto \text{"nil"}]$,
$pvc = [c \in CLUSTERS \mapsto \text{"nil"}]$

Process representing the user's actions
process *User* = *U*
begin
Start:
skip;
end process;

```
61   The kcp-level Namespace controller
62 process NamespaceController = NC
63 variables
64 begin
65     Start:
66       assert(PrintT(⟨ns, pvc⟩));
67     either   Assign a NS to a cluster
68         await ∀ c ∈ CLUSTERS : ns[c] = "nil" ;   Only assign if not on any cluster
69         with c ∈ CLUSTERS do
70             ns[c] := "Sync" ;
71         end with ;
72     or   Remove the NS from a cluster
73         with c ∈ {c ∈ CLUSTERS : ns[c] = "Sync"} do
74             ns[c] := "nil" ;
75         end with ;
76     end either ;
77     goto Start  ;
78 end process ;

80   kcp-level PVC controller
81 process PVCController = PVCC
82 begin
83     Start:
84     pvc := ns ;   Set the PVC's state to match the NS state
85     goto Start ;
86 end process ;

88   The syncer process running on each workload cluster
89 process Syncer ∈ CLUSTERS
90 variables
91     pvc_state = "nil"   Starts not synced here
92 begin
93     Start:
94       Update our local state to match desired
95     pvc_state := ns[self] ;
96     goto Start ;
97 end process ;

99 end algorithm  ;
100  BEGIN TRANSLATION (chksum(pcal) = "c54d5efb" ∧ chksum(tla) = "2e992517")
101  Label Start of process NamespaceController at line 67 col 5 changed to Start_
102  Label Start of process PVCController at line 84 col 5 changed to Start_P
103 VARIABLES ns, pvc, pc, pvc_state

105 vars ≜ ⟨ns, pvc, pc, pvc_state⟩
```

```
ProcSet ≜ {NC} ∪ {PVCC} ∪ (CLUSTERS)

Init ≜ Global variables
       ∧ ns = [c ∈ CLUSTERS ↦ "nil"]
       ∧ pvc = [c ∈ CLUSTERS ↦ "nil"]
       Process Syncer
       ∧ pvc_state = [self ∈ CLUSTERS ↦ "nil"]
       ∧ pc = [self ∈ ProcSet ↦ CASE self = NC → "Start_"
                                  □ self = PVCC → "Start_P"
                                  □ self ∈ CLUSTERS → "Start"]

Start_ ≜ ∧ pc[NC] = "Start_"
         ∧ ∨ ∧ ∀ c ∈ CLUSTERS : ns[c] = "nil"
             ∧ ∃ c ∈ CLUSTERS :
                 ns' = [ns EXCEPT ![c] = "Sync"]
           ∨ ∧ ∃ c ∈ {c ∈ CLUSTERS : ns[c] = "Sync"} :
                 ns' = [ns EXCEPT ![c] = "nil"]
         ∧ pc' = [pc EXCEPT ![NC] = "Start_"]
         ∧ UNCHANGED ⟨pvc, pvc_state⟩

NamespaceController ≜ Start_

Start_P ≜ ∧ pc[PVCC] = "Start_P"
          ∧ pvc' = ns
          ∧ pc' = [pc EXCEPT ![PVCC] = "Start_P"]
          ∧ UNCHANGED ⟨ns, pvc_state⟩

PVCController ≜ Start_P

Start(self) ≜ ∧ pc[self] = "Start"
              ∧ pvc_state' = [pvc_state EXCEPT ![self] = ns[self]]
              ∧ pc' = [pc EXCEPT ![self] = "Start"]
              ∧ UNCHANGED ⟨ns, pvc⟩

Syncer(self) ≜ Start(self)

Allow infinite stuttering to prevent deadlock on termination.
Terminating ≜ ∧ ∀ self ∈ ProcSet : pc[self] = "Done"
              ∧ UNCHANGED vars

Next ≜ NamespaceController ∨ PVCController
       ∨ (∃ self ∈ CLUSTERS : Syncer(self))
       ∨ Terminating

Spec ≜ ∧ Init ∧ □[Next]_vars
       ∧ WF_vars(Next)

Termination ≜ ◇(∀ self ∈ ProcSet : pc[self] = "Done")
```

END TRANSLATION

All variables are of the correct type

$TypeOK \triangleq$

ns has a mapping for exactly every cluster

$\land$ DOMAIN $ns = CLUSTERS$

All cluster sync states are valid values

$\land \forall\, c \in$ DOMAIN $ns : ns[c] \in SyncStates$

A *Namespace* is assigned to at most 1 cluster

$Inv\_NSAtMostOneCluster \triangleq Cardinality(\{\forall\, c \in CLUSTERS : ns[c] = \text{“Sync”}\}) \leq 1$

A *PVC* is assigned to at most 1 cluster

$Inv\_PVCAtMostOneCluster \triangleq Cardinality(\{\forall\, c \in CLUSTERS : pvc[c] = \text{“Sync”}\}) \leq 1$

At most one cluster thinks they can use the *PVC*

$Inv\_UsableByAtMostOne \triangleq Cardinality(\{c \in CLUSTERS : pvc\_state[c] = \text{“Sync”}\}) \leq 1$

Statements that must be true in ALL states

$Invariants \triangleq$

$\land\ TypeOK$

$\land\ Inv\_NSAtMostOneCluster$

$\land\ Inv\_PVCAtMostOneCluster$

$\land\ Inv\_UsableByAtMostOne$

---

* Modification History

* Last modified *Fri Sep* 16 16:07:04 *EDT* 2022 by *jstrunk*

* Created *Fri Sep* 16 09:16:27 *EDT* 2022 by *jstrunk*