MODULE $U2PC\_MC$

EXTENDS $Apalache$, $U2PC$, TLC

@type: (a, b) $\Rightarrow \langle a, b \rangle$;
$Pair(A, B) \triangleq \langle A, B \rangle$

1 shard, 1-2 transactions

Checking simple commit, and conflict behaviours

$T1 \triangleq SetAsFun(\{Pair(\text{"T1"}, \{\text{"X"}\})\})$
$T1\_2 \triangleq SetAsFun(\{Pair(\text{"T1"}, \{\text{"X"}\}), Pair(\text{"T2"}, \{\text{"X"}\})\})$
$S1 \triangleq SetAsFun(\{Pair(\text{"X"}, \{\text{"X1"}, \text{"X2"}\})\})$

3 shards, 3 transactions

Checking indirect dependency loops

$T3 \triangleq SetAsFun(\{$
$Pair(\text{"T1"}, \{\text{"X"}, \text{"Y"}\}),$
$Pair(\text{"T2"}, \{\text{"Y"}, \text{"Z"}\}),$
$Pair(\text{"T3"}, \{\text{"Z"}, \text{"X"}\})\})$
$S3 \triangleq SetAsFun(\{$
$Pair(\text{"X"}, \{\text{"X1"}, \text{"X2"}\}),$
$Pair(\text{"Y"}, \{\text{"Y1"}, \text{"Y2"}\}),$
$Pair(\text{"Z"}, \{\text{"Z1"}, \text{"Z2"}\})\})$

Initial state for *Apalache* testing

$CInit \triangleq$
$\land\ Txns := T3$
$\land\ Shards := S3$

Credit to https://*github.com*/tlaplus/examples

$TransitiveClosure(R) \triangleq$
LET $S \triangleq \{r[1] : r \in R\} \cup \{r[2] : r \in R\}$
RECURSIVE $TCR(\_)$
$TCR(T) \triangleq$ IF $T = \{\}$
THEN R
ELSE LET $r \triangleq$ CHOOSE $s \in T$: TRUE
$RR \triangleq TCR(T \setminus \{r\})$
IN $RR \cup \{\langle s, t \rangle \in S \times S :$
$\langle s, r \rangle \in RR \land \langle r, t \rangle \in RR\}$
IN $TCR(S)$

$TransactionOrdering \triangleq$ LET
$F(acc, tid) \triangleq acc \cup (Range(Coordinator\_txn\_state[tid]) \times \{tid\})$
$Base \triangleq ApaFoldSet(F, \{\}, TIDs)$
IN $TransitiveClosure(Base)$

$$
\begin{array}{l}
RecoveryCommitted(S) \triangleq \\
\quad \{t \in TIDs : \\
\qquad \forall\, r \in S : \\
\qquad KeyLookup[r] \in Txns[t] \\
\qquad \Rightarrow \lor\ Replicas[r].locked \land Replicas[r].logged = t \\
\qquad\quad \lor\ Replicas[r].version = t \\
\qquad\quad \lor\ \langle t,\ Replicas[r].version\rangle \in TransactionOrdering \\
\quad \}
\end{array}
$$

Every transaction committed during recovery preserves linearisability

$$
\begin{array}{l}
Safety\_recovery \triangleq \\
\quad \forall\, S \in \text{SUBSET}\ RIDs : \\
\qquad \text{Valid recovery} \\
\quad (\forall\, k \in \text{DOMAIN}\ Shards : \exists\, r \in S : r \in Shards[k]) \\
\quad \Rightarrow Linearisability(CommittedTIDs \cup RecoveryCommitted(S))
\end{array}
$$

$$
\begin{array}{l}
RecoveryAborted(S) \triangleq \\
\quad \{t \in TIDs : \\
\qquad \exists\, r \in S : \\
\qquad \land\ KeyLookup[r] \in Txns[t] \\
\qquad \land\ \lor \neg Replicas[r].locked \\
\qquad\quad \lor\ Replicas[r].locked \land Replicas[r].logged \neq t\}
\end{array}
$$

Every committed or aborted transaction results in the same recovery decision

$$
\begin{array}{l}
Durability \triangleq \\
\quad \forall\, S \in \text{SUBSET}\ RIDs : \\
\qquad \text{Valid recovery} \\
\quad (\forall\, k \in \text{DOMAIN}\ Shards : \exists\, r \in S : r \in Shards[k]) \\
\quad \Rightarrow \\
\quad \forall\, t\ \in TIDs : \\
\quad \land\ t \in CommittedTIDs \Rightarrow t \in RecoveryCommitted(S) \\
\quad \land\ t \in AbortedTIDs \Rightarrow t \in RecoveryAborted(S)
\end{array}
$$

Since recovery stops every replica it uses, an explicit recovery check is unnecessary since that is equivalent to just checking that every possible recovery using the current state preserves the invariants.

$$
\begin{array}{l}
Invs \triangleq \\
\quad \land\ Safety\_recovery \\
\quad \land\ Durability
\end{array}
$$