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

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

Initial state for *Apalache* testing

$$
\begin{aligned}
&CInit \triangleq\\
&\quad \land Txns := T3\\
&\quad \land Shards := S3
\end{aligned}
$$

Credit to *github.com*/tlaplus/examples

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

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

$$
\begin{aligned}
&RecoveryCommitted(S) \triangleq\\
&\quad \{t \in TIDs :
\end{aligned}
$$

$\forall r \in S :$
$KeyLookup[r] \in Txns[t]$
$\Rightarrow \lor Replicas[r].locked \land Replicas[r].logged = t$
$\lor Replicas[r].version = t$
$\lor \langle t, Replicas[r].version \rangle \in TransactionOrdering$
$\}$

Every transaction committed during recovery preserves linearisability

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

$RecoveryAborted(S) \triangleq$
$\{t \in TIDs :$
$\exists r \in S :$
$\land KeyLookup[r] \in Txns[t]$
$\land \lor \neg Replicas[r].locked$
$\lor Replicas[r].locked \land Replicas[r].logged \neq t\}$

Every committed or aborted transaction results in the same recovery decision

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

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.

$Invs \triangleq$
$\land Safety\_recovery$
$\land Durability$