MODULE *Commons*

EXTENDS *Integers*, *Sequences*
VARIABLES *msgs*, *fstates*, *thrds*, *waitForMsgs*, *killed*, *seq*
CONSTANTS *PLACE*, *MXFINISHES*, *PROG_HOME*, *BACKUP*

$ROOT\_FINISH \triangleq$ "distroot"

$REMOTE\_FINISH \triangleq$ "distremote"

$MXTHREADS \triangleq 2$
$MXACTIVITIES \triangleq 20$
$MXMESSAGES \triangleq 200$

$MXFID \triangleq MXFINISHES + 1$

$NotID \triangleq -1$

$NoParent \triangleq 0$

$FIRST\_ID \triangleq 1$

$PIDRange \triangleq NoParent \,..\, MXFID$

$IDRange \triangleq FIRST\_ID \,..\, MXFID$

$NotPlace \triangleq$ CHOOSE $v : v \notin PLACE$

$ThreadID \triangleq 0 \,..\, MXTHREADS - 1$

$NotThreadID \triangleq -5050$

$EMPTY\_BLOCK \triangleq -1$

$BlockID \triangleq 0 \,..\, 25$ $NBLOCKS - 1$

$NotBlockID \triangleq -1000$

$StmtID \triangleq 0 \,..\, 5$ $MXSTMTS - 1$

$I\_START \triangleq -1$

$I\_PRE\_FIN\_ALLOC \triangleq -2$

Record Types

$Sequences \triangleq [aseq : 1 \,..\, MXACTIVITIES,\ mseq : 1 \,..\, MXMESSAGES,\ fseq : IDRange]$

Each thread has a stack, and this is the stack entry

$StackEntry \triangleq [b : BlockID,$
$\quad i : StmtID \cup \{I\_START,\ I\_PRE\_FIN\_ALLOC\},$
$\quad fid : PIDRange,$

$src : PLACE]$

the processing unit of program instructions

$Thread \triangleq [tid : ThreadID,$
$status : \{\text{“idle”}, \text{“running”}, \text{“blocked”}\},$
$blockingType : \{\text{“NA”}, \text{“FinishEnd”}, \text{“AsyncTransit”}, \text{“FinishAlloc”}, \text{“AsyncTerm”}\},$
$stack : Seq(StackEntry)]$

the activities that are pushed to scheduler's ready queue,
and will eventually be fetched by threads

$Activity \triangleq [aid : Nat,$
$b : BlockID,$
$fid : IDRange,$
$src : PLACE]$

$NotActivity \triangleq [aid \mapsto -1, b \mapsto NotBlockID, fid \mapsto NotID]$

Input Program: *Block* error used to simulate exceptions

$Block \triangleq [b : BlockID \cup \{NotBlockID\},$
$type : \{\text{“NA”}, \text{“async”}, \text{“expr”}, \text{“finish”}, \text{“error”}, \text{“kill”}\},$
$dst : PLACE \cup \{NotPlace\},$
$mxstmt : Nat,$
$stmts : [StmtID \to BlockID \cup \{EMPTY\_BLOCK, NotBlockID\}],$
$ran : \text{BOOLEAN }]$

$PlaceThread \triangleq [here : PLACE, tid : ThreadID]$

$NotPlaceThread \triangleq [\ here \mapsto NotPlace, \ \ tid \mapsto NotThreadID]$

$MasterStatus \triangleq [status : \{\text{“running”}, \text{“preConvert”}, \text{“convertToDead”}, \text{“convertFromDead”}\},$
$lastKilled : PLACE \cup \{NotPlace\}]$

Finish Types

$FinishState \triangleq [id : IDRange \cup \{NotID\},$
$status : \{\text{“unused”}, \text{“waiting”}, \text{“pendingRelease”}, \text{“forgotten”}\},$
$type : \{\text{“distroot”}, \text{“distremote”}, \text{“NA”}\},$
$count : Nat,$ local tasks count
$here : PLACE \cup \{NotPlace\},$
$parent : PIDRange \cup \{NotID\},$ used for linking finishes within the same place
$root : PIDRange \cup \{NotID\},$ root is the same as *id* for root finishes
$isGlobal : \text{BOOLEAN }$, replicated on a backup place
$eroot : PIDRange \cup \{NotID\},$ the adopter
$deny : \text{SUBSET } PLACE,$ for remote finish only
$newMaster : PLACE \cup \{NotPlace\},$
$newBackup : PLACE \cup \{NotPlace\},$
$src : PLACE \cup \{NotPlace\},$

$received : [PLACE \to Nat]$
$]$

$$
\begin{aligned}
&MasterFinish \triangleq [\\
&\quad id : IDRange \cup \{NotID\},\\
&\quad src : PLACE \cup \{NotPlace\},\\
&\quad home : PLACE \cup \{NotPlace\},\\
&\quad numActive : Nat,\\
&\quad transit : [PLACE \to [PLACE \to Nat]],\\
&\quad adoptedChildren : \text{SUBSET } IDRange,\\
&\quad newBackup : PLACE \cup \{NotPlace\},\\
&\quad isAdopted : \text{BOOLEAN},\\
&\quad adopterPlace : PLACE \cup \{NotPlace\},\\
&\quad isReleased : \text{BOOLEAN},\\
&\quad \_src : PLACE \cup \{NotPlace\},\\
&\quad \_home : PLACE \cup \{NotPlace\},\\
&\quad \_numActive : Nat,\\
&\quad \_transit : [PLACE \to [PLACE \to Nat]],\\
&\quad \_adoptedChildren : \text{SUBSET } IDRange,\\
&\quad \_newBackup : PLACE \cup \{NotPlace\},\\
&\quad \_isAdopted : \text{BOOLEAN},\\
&\quad \_isReleased : \text{BOOLEAN},\\
&\quad \_adopterPlace : PLACE \cup \{NotPlace\}\\
&]
\end{aligned}
$$

$$
\begin{aligned}
&BackupFinish \triangleq [\\
&\quad id : IDRange \cup \{NotID\},\\
&\quad src : PLACE \cup \{NotPlace\},\\
&\quad home : PLACE \cup \{NotPlace\},\\
&\quad numActive : Nat,\\
&\quad transit : [PLACE \to [PLACE \to Nat]],\\
&\quad adoptedChildren : \text{SUBSET } IDRange,\\
&\quad newMaster : PLACE \cup \{NotPlace\},\\
&\quad isAdopted : \text{BOOLEAN},\\
&\quad adopterPlace : PLACE \cup \{NotPlace\},\\
&\quad isReleased : \text{BOOLEAN},\\
&\quad \_src : PLACE \cup \{NotPlace\},\\
&\quad \_home : PLACE \cup \{NotPlace\},\\
&\quad \_numActive : Nat,\\
&\quad \_transit : [PLACE \to [PLACE \to Nat]],\\
&\quad \_adoptedChildren : \text{SUBSET } IDRange,\\
&\quad \_newMaster : PLACE \cup \{NotPlace\},\\
&\quad \_isAdopted : \text{BOOLEAN},\\
&\quad \_isReleased : \text{BOOLEAN},\\
&\quad \_adopterPlace : PLACE \cup \{NotPlace\}
\end{aligned}
$$

]

Message Types and Utilities

$$NotMessage \triangleq [fid \mapsto NotID,\ src \mapsto NotPlace]$$

$$\begin{aligned} RemoteAsyncMessages \triangleq\ & [mid : Nat,\\ & src : PLACE,\\ & dst : PLACE,\\ & type : \{\text{“async”}\},\\ & b : BlockID,\\ & fid : IDRange] \end{aligned}$$

$$\begin{aligned} ReleaseFinishMessages \triangleq\ & [mid : Nat,\\ & src : PLACE,\\ & dst : PLACE,\\ & fid : IDRange,\\ & type : \{\text{“releaseFinish”}\}] \end{aligned}$$

$$\begin{aligned} MasterTransitMessages \triangleq\ & [mid \mapsto Nat,\\ & src \mapsto PLACE,\\ & dst \mapsto PLACE,\\ & target \mapsto PLACE,\\ & fid \mapsto IDRange,\\ & taskFID \mapsto IDRange,\\ & finishSrc \mapsto PLACE,\\ & type \mapsto \text{“masterTransit”}] \end{aligned}$$

$$\begin{aligned} MasterCompletedMessages \triangleq\ & [mid \mapsto Nat,\\ & src \mapsto PLACE,\\ & dst \mapsto PLACE,\\ & source \mapsto PLACE,\\ & target \mapsto PLACE,\\ & fid \mapsto IDRange,\\ & taskFID \mapsto IDRange,\\ & finishEnd \mapsto \text{BOOLEAN},\\ & type \mapsto \text{“masterCompleted”}] \end{aligned}$$

$$\begin{aligned} BackupGetNewMaster \triangleq\ & [\ mid \mapsto Nat,\\ & src \mapsto PLACE,\\ & dst \mapsto PLACE,\\ & fid \mapsto IDRange,\\ & source \mapsto PLACE,\\ & target \mapsto PLACE,\\ & actionType \mapsto \{\text{“transit”}, \text{“completed”}\},\\ & finishEnd \mapsto \text{BOOLEAN},\\ & finishSrc \mapsto PLACE, \end{aligned}$$

$$
\begin{array}{rl}
taskFID & \mapsto IDRange, \\
type & \mapsto \text{“backupCreateNewMaster”}]
\end{array}
$$

$$
\begin{array}{rcll}
BackupGetNewMasterDone & \triangleq & [& mid \mapsto Nat, \\
 & & & src \mapsto PLACE, \\
 & & & dst \mapsto PLACE, \\
 & & & source \mapsto PLACE, \\
 & & & target \mapsto PLACE, \\
 & & & fid \mapsto IDRange, \\
 & & & newMaster \mapsto PLACE, \\
 & & & actionType \mapsto \{\text{“transit”}, \text{“completed”}\}, \\
 & & & finishEnd \mapsto \text{BOOLEAN}, \\
 & & & finishSrc \mapsto PLACE, \\
 & & & taskFID \mapsto IDRange, \\
 & & & type \mapsto \text{“backupCreateNewMasterDone”}]
\end{array}
$$

$$
\begin{array}{rcll}
MasterTransitDone & \triangleq & [& mid \mapsto Nat, \\
 & & & src \mapsto PLACE, \\
 & & & dst \mapsto PLACE, \\
 & & & target \mapsto PLACE, \\
 & & & fid \mapsto IDRange, \\
 & & & taskFID \mapsto IDRange, \\
 & & & finishSrc \mapsto PLACE, \\
 & & & type \mapsto \text{“masterTransitDone”}, \\
 & & & submit \mapsto \text{BOOLEAN}, \\
 & & & success \mapsto \text{BOOLEAN}, \\
 & & & backupPlace \mapsto PLACE]
\end{array}
$$

$$
\begin{array}{rcll}
BackupTransit & \triangleq & [& mid \mapsto Nat, \\
 & & & src \mapsto PLACE, \\
 & & & dst \mapsto PLACE, \\
 & & & target \mapsto PLACE, \\
 & & & fid \mapsto IDRange, \\
 & & & finishSrc \mapsto PLACE, \\
 & & & knownMaster \mapsto PLACE, \\
 & & & taskFID \mapsto IDRange, \\
 & & & type \mapsto \text{“backupTransit”}]
\end{array}
$$

$$
\begin{array}{rcll}
BackupTransitDone & \triangleq & [& mid \mapsto Nat, \\
 & & & src \mapsto PLACE, \\
 & & & dst \mapsto PLACE, \\
 & & & target \mapsto PLACE, \\
 & & & fid \mapsto IDRange, \\
 & & & finishSrc \mapsto PLACE, \\
 & & & type \mapsto \text{“backupTransitDone”}, \\
 & & & success \mapsto \text{BOOLEAN}\ ]
\end{array}
$$

$$
\begin{aligned}
BackupCompleted \triangleq\ & [mid \mapsto Nat,\\
& \ src \mapsto PLACE,\\
& \ dst \mapsto PLACE,\\
& source \mapsto PLACE,\\
& target \mapsto PLACE,\\
& \ fid \mapsto IDRange,\\
& knownMaster \mapsto PLACE,\\
& taskFID \mapsto IDRange,\\
& type \mapsto \text{“backupCompleted”},\\
& finishEnd \mapsto \text{BOOLEAN }]
\end{aligned}
$$

$$
\begin{aligned}
MasterCompletedDone \triangleq\ & [mid \mapsto Nat,\\
& \ src \mapsto PLACE,\\
& \ dst \mapsto PLACE,\\
& source \mapsto PLACE,\\
& target \mapsto PLACE,\\
& \ fid \mapsto IDRange,\\
& taskFID \mapsto IDRange,\\
& type \mapsto \text{“masterCompletedDone”},\\
& success \mapsto \text{BOOLEAN },\\
& backupPlace \mapsto PLACE]
\end{aligned}
$$

$$
\begin{aligned}
BackupCompletedDone \triangleq\ & [mid \mapsto Nat,\\
& \ src \mapsto PLACE,\\
& \ dst \mapsto PLACE,\\
& source \mapsto PLACE,\\
& target \mapsto PLACE,\\
& \ fid \mapsto IDRange,\\
& type \mapsto \text{“backupCompletedDone”},\\
& success \mapsto \text{BOOLEAN }]
\end{aligned}
$$

$$
\begin{aligned}
Messages \triangleq\ & RemoteAsyncMessages\\
& \cup MasterTransitMessages\\
& \cup MasterCompletedMessages\\
& \cup BackupTransit\\
& \cup MasterTransitDone\\
& \cup BackupTransitDone\\
& \cup BackupCompleted\\
& \cup MasterCompletedDone\\
& \cup BackupCompletedDone\\
& \cup ReleaseFinishMessages\\
& \cup BackupGetNewMaster\\
& \cup BackupGetNewMasterDone
\end{aligned}
$$

$$
\begin{aligned}
&SendMsg(m) \triangleq\\
&\quad msgs' = msgs \cup \{m\}
\end{aligned}
$$

$RecvMsg(m) \triangleq$
$msgs' = msgs \setminus \{m\}$

$ReplaceMsg(toRemove,\ toAdd) \triangleq$
$msgs' = (msgs \setminus \{toRemove\}) \cup \{toAdd\}$

$ReplaceMsgSet(toRemove,\ toAddSet) \triangleq$
$msgs' = (msgs \setminus \{toRemove\}) \cup toAddSet$

Predicates to extract the finish *id* from messages and *fstates*

$ExtractFIDFromMSG(src,\ dst,\ type) \triangleq$
LET $mset \triangleq \{m \in msgs : \land m.src = src$
$\land m.dst = dst$
$\land m.type = type$
$\land m.fid \in IDRange$
$\}$
IN IF $mset = \{\}$ THEN $NotID$
ELSE $(\text{CHOOSE } x \in mset : \text{TRUE}).fid$

$FindIncomingMSG(here,\ type) \triangleq$
LET $mset \triangleq \{m \in msgs : \land m.dst = here$
$\land m.type = type$
$\land m.dst \notin killed$
$\}$
IN IF $mset = \{\}$ THEN $NotMessage$
ELSE CHOOSE $x \in mset$: TRUE

$FindMSG(type) \triangleq$
LET $mset \triangleq \{m \in msgs : \land m.type = type$
$\land m.dst \notin killed$
$\}$
IN IF $mset = \{\}$ THEN $NotMessage$
ELSE CHOOSE $x \in mset$: TRUE

$GetActiveFID(type,\ here,\ pid) \triangleq$
LET $mset \triangleq \{id \in IDRange : \land fstates[id].here = here$
$\land fstates[id].root = pid$
$\land fstates[id].type = type$
$\land fstates[id].status =$ "waiting"
$\}$
IN IF $mset = \{\}$ THEN $NotID$
ELSE $(\text{CHOOSE } x \in mset : \text{TRUE})$

$GetFinishHome(fid) \triangleq$
IF $fid = NoParent$ THEN $PROG\_HOME$ ELSE $fstates[fid].here$

$GetEnclosingRoot(parent,\ me) \triangleq$
IF $parent = NoParent$ THEN $NoParent$ ELSE $fstates[parent].root$

Predicate to extract thread ids with a specific status

$FindThread(here,\ status) \triangleq$
LET $tset \triangleq \{t \in ThreadID : thrds[here][t].status = status\}$
IN IF $tset = \{\}$ THEN $NotThreadID$
ELSE CHOOSE $x \in tset$: TRUE

$FindThread2(here,\ statusSet) \triangleq$
LET $tset \triangleq \{t \in ThreadID : thrds[here][t].status \in statusSet\}$
IN IF $tset = \{\}$ THEN $NotThreadID$
ELSE CHOOSE $x \in tset$: TRUE

Resilient Store Types and Utilities

$ConvTask \triangleq [here : PLACE,\ fid : IDRange \cup \{NotID\},$
$to\_pl : PLACE \cup \{NotPlace\},\ from\_pl : PLACE \cup \{NotPlace\}]$

$NotConvTask \triangleq [here \mapsto NotPlace,\ fid \mapsto NotID,$
$to\_pl \mapsto NotPlace,\ from\_pl \mapsto NotPlace]$

$GetBackup(p) \triangleq BACKUP[p]$

Utilities to increment sequences used to give unique ids to finish (*fseq*) messages (*mseq*), and activities (*aseq*)

$IncrFSEQ \triangleq$
$seq' = [aseq \mapsto seq.aseq,\ fseq \mapsto seq.fseq + 1,\ mseq \mapsto seq.mseq]$

$IncrMSEQ(c) \triangleq$
$seq' = [aseq \mapsto seq.aseq,\ fseq \mapsto seq.fseq,\ mseq \mapsto seq.mseq + c]$

$IncrASEQ \triangleq$
$seq' = [aseq \mapsto seq.aseq + 1,\ fseq \mapsto seq.fseq,\ mseq \mapsto seq.mseq]$

$IncrAll \triangleq$
$seq' = [aseq \mapsto seq.aseq + 1,\ fseq \mapsto seq.fseq + 1,\ mseq \mapsto seq.mseq + 1]$

---

* Modification History
* Last modified *Fri Dec* 15 16:53:36 *AEDT* 2017 by *u5482878*
* Last modified *Wed Dec* 13 22:56:06 *AEDT* 2017 by *shamouda*
* Created *Wed Sep* 27 09:26:18 *AEST* 2017 by *u5482878*