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"

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

$MXFID \triangleq MXFINISHES + 1$

$NotID \triangleq -1$

$NoParent \triangleq 0$

$FIRST\_ID \triangleq 1$

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

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

$NotPlace \triangleq \text{CHOOSE } v : v \notin PLACE$

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

$NotThreadID \triangleq -5050$

$EMPTY\_BLOCK \triangleq -1$

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

$NotBlockID \triangleq -1000$

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

$I\_START \triangleq -1$

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

Record Types

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

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

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

the processing unit of program instructions

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

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

$$\begin{aligned}
Activity \triangleq\ & [aid : Nat,\\
& b : BlockID,\\
& fid : IDRange]
\end{aligned}$$

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

Input Program: *Block* error used to simulate exceptions

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

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

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

$$\begin{aligned}
MasterStatus \triangleq\ & [status : \{\text{“running”}, \text{“seekAdoption”}, \text{“convertDead”}\},\\
& lastKilled : PLACE \cup \{NotPlace\}]
\end{aligned}$$

Finish Types

$$\begin{aligned}
FinishState \triangleq\ & [id : IDRange \cup \{NotID\},\\
& status : \{\text{“unused”}, \text{“waiting”}, \text{“pendingRelease”}, \text{“forgotten”}\},\\
& type : \{\text{“distroot”}, \text{“distremote”}, \text{“NA”}\},\\
& count : Nat,\\
& here : PLACE \cup \{NotPlace\},\\
& parent : PIDRange \cup \{NotID\},\ \text{used only in RESILIENT mode}\\
& root : PIDRange \cup \{NotID\},\ \text{root is the same as } id \text{ for root finishes}\\
& isGlobal : \text{BOOLEAN}\ ,\ \text{used by } P0Finish\\
& eroot : PIDRange \cup \{NotID\}\ \text{root of the enclosing finish (used in } PPoPP14 \text{ dist finish)}\\
&]
\end{aligned}$$

$$\begin{aligned}
MasterFinish \triangleq\ & [\\
& id : IDRange \cup \{NotID\},\\
& numActive : Nat,\\
& live : [PLACE \to Nat],\\
& transit : [PLACE \to [PLACE \to Nat]],
\end{aligned}$$

$liveAdopted : [PLACE \to Nat],$
$transitAdopted : [PLACE \to [PLACE \to Nat]],$
$children : \text{SUBSET } IDRange,$
$backupPlace : PLACE \cup \{NotPlace\},$
$isReleased : \text{BOOLEAN}$
]

$BackupFinish \triangleq [$
$id : IDRange \cup \{NotID\},$
$live : [PLACE \to Nat],$
$transit : [PLACE \to [PLACE \to Nat]],$
$children : \text{SUBSET } IDRange,$
$isAdopted : \text{BOOLEAN },$
$adoptedRoot : IDRange \cup \{NotID\},$
$numActive : Nat$
]

Message Types and Utilities

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

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

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

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

$$\begin{aligned} MasterLiveMessages \triangleq\ & [mid \mapsto Nat, \\ & src \mapsto PLACE, \\ & source \mapsto PLACE, \\ & target \mapsto PLACE, \end{aligned}$$

$$
\begin{array}{rl}
dst & \mapsto PLACE, \\
fid & \mapsto IDRange, \\
aid & \mapsto Nat, \\
type & \mapsto \{\text{“masterLive”}, \text{“adopterLive”}\}]
\end{array}
$$

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

$$
\begin{array}{rcl}
BackupTransit & \triangleq & [\ mid \mapsto Nat, \\
& & \quad src \mapsto PLACE, \\
& & \quad dst \mapsto PLACE, \\
& & \quad source \mapsto PLACE, \\
& & \quad target \mapsto PLACE, \\
& & \quad fid \mapsto IDRange, \\
& & \quad type \mapsto \text{“backupTransit”}]
\end{array}
$$

$$
\begin{array}{rcl}
MasterORBackupTransitDone & \triangleq & [\ mid \mapsto Nat, \\
& & \quad src \mapsto PLACE, \\
& & \quad dst \mapsto PLACE, \\
& & \quad target \mapsto PLACE, \\
& & \quad fid \mapsto IDRange, \\
& & \quad type \mapsto \{\text{“masterTransitDone”}, \text{“backupTransitDone”}\}, \\
& & \quad isAdopted \mapsto \text{BOOLEAN}, \\
& & \quad adoptedRoot \mapsto IDRange \cup \{NotID\}, \\
& & \quad adoptedFID \mapsto IDRange \cup \{NotID\}, \\
& & \quad success \mapsto \text{BOOLEAN}\ ]
\end{array}
$$

$$
\begin{array}{rcl}
BackupLive & \triangleq & [\ mid \mapsto Nat, \\
& & \quad src \mapsto PLACE, \\
& & \quad dst \mapsto PLACE, \\
& & \quad source \mapsto PLACE, \\
& & \quad target \mapsto PLACE, \\
& & \quad fid \mapsto IDRange, \\
& & \quad aid \mapsto Nat, \\
& & \quad type \mapsto \text{“backupLive”}]
\end{array}
$$

$$
\begin{array}{rcl}
MasterOrBackupLiveDone & \triangleq & [\ mid \mapsto Nat, \\
& & \quad src \mapsto PLACE, \\
& & \quad dst \mapsto PLACE, \\
& & \quad target \mapsto PLACE, \\
& & \quad source \mapsto PLACE,
\end{array}
$$

$$
\begin{array}{rl}
\mathit{fid} & \mapsto \mathit{IDRange}, \\
\mathit{aid} & \mapsto \mathit{Nat}, \\
\mathit{type} & \mapsto \{\text{“masterLiveDone”}, \text{“backupLiveDone”}\}, \\
\mathit{isAdopted} & \mapsto \text{BOOLEAN}\ , \\
\mathit{adoptedRoot} & \mapsto \mathit{IDRange} \cup \{\mathit{NotID}\}, \\
\mathit{submit} & \mapsto \text{BOOLEAN}\ , \\
\mathit{success} & \mapsto \text{BOOLEAN}\ ]
\end{array}
$$

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

$$
\begin{array}{rl}
\mathit{MasterOrBackupCompletedDone} \triangleq & [\mathit{mid} \mapsto \mathit{Nat}, \\
\mathit{src} & \mapsto \mathit{PLACE}, \\
\mathit{dst} & \mapsto \mathit{PLACE}, \\
\mathit{target} & \mapsto \mathit{PLACE}, \\
\mathit{fid} & \mapsto \mathit{IDRange}, \\
\mathit{type} & \mapsto \{\text{“masterCompletedDone”}, \text{“backupCompletedDone”}\}, \\
\mathit{isAdopted} & \mapsto \text{BOOLEAN}\ , \\
\mathit{adoptedRoot} & \mapsto \mathit{IDRange} \cup \{\mathit{NotID}\}, \\
\mathit{numActive} & \mapsto \mathit{Nat}, \\
\mathit{success} & \mapsto \text{BOOLEAN}\ , \\
\mathit{finishEnd} & \mapsto \text{BOOLEAN}\ , \\
\mathit{release} & \mapsto \text{BOOLEAN}\ ]
\end{array}
$$

$$
\begin{array}{rl}
\mathit{Messages} \triangleq & \mathit{RemoteAsyncMessages} \\
& \cup\ \mathit{MasterTransitMessages} \\
& \cup\ \mathit{MasterLiveMessages} \\
& \cup\ \mathit{MasterCompletedMessages} \\
& \cup\ \mathit{BackupTransit} \\
& \cup\ \mathit{MasterORBackupTransitDone} \\
& \cup\ \mathit{BackupLive} \\
& \cup\ \mathit{MasterOrBackupLiveDone} \\
& \cup\ \mathit{BackupCompleted} \\
& \cup\ \mathit{MasterOrBackupCompletedDone} \\
& \cup\ \mathit{ReleaseFinishMessages}
\end{array}
$$

$$
\begin{array}{l}
\mathit{SendMsg}(m) \triangleq \\
\quad \mathit{msgs}' = \mathit{msgs} \cup \{m\}
\end{array}
$$

$$
\begin{array}{l}
\mathit{RecvMsg}(m) \triangleq \\
\quad \mathit{msgs}' = \mathit{msgs} \setminus \{m\}
\end{array}
$$

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

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

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

$Adopter \triangleq [here : PLACE,\ child : IDRange \cup \{NotID\},\ adopter : IDRange \cup \{NotID\}]$

$NotAdopter \triangleq [here \mapsto NotPlace,\ child \mapsto NotID,\ adopter \mapsto NotID]$

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

$NotConvTask \triangleq [here \mapsto NotPlace,\ fid \mapsto NotID,\ pl \mapsto NotPlace]$

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

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 *Wed Dec* 13 15:21:40 *AEDT* 2017 by *u5482878*
* Last modified *Tue Dec* 05 18:28:01 *AEDT* 2017 by *shamouda*
* Created *Wed Sep* 27 09:26:18 *AEST* 2017 by *u5482878*