MODULE *sflogcrash*

EXTENDS *Integers*, *Sequences*

Spec for a Write Ahead Logger (*WAL*)
WAL Logger appends data to a file called write extent (*WE*) which has max capacity of 1 *GB*.
Once write extent is full, we move it to a set of read files called read extents (RE) and create new *WE*.
Each extent/file has data in range from start to end *LSN* (logical sequence numbers).
extent: [start, end) where $start \leq end$.
means that extent contains data from [start, end) //*end* not included
$start \triangleq end$ means no data in file.
Data is appened at the end of log called tail. So, data grows from head towards tail.
Other operations performed by logger are truncation of log at head or tail,
Close and recovery of log at Open.

State machine valid steps:
1. There is only 1 *Append* request at a time.
2. After close or crash, only recovery runs.
3. Only 1 *TruncateHead* request is initiated at a time.
4. Only 1 *TruncateTail* request is initiated at a time.
5. *TruncateHead* can be initiated with *Append*.
6. No other request is served till *TruncateTail* finishes.

Todo:
1) *MetaDataFile* corruption is single point of failure.
Todo: Create *metadataFile.mdlog.tmp* file first,
then delete *metadataFile.mdlog* and rename *.mdlog.tmp* to *.mdlog*
Handle crash after each step in *Recovery*.

Variables are divided into 2 categories:
1. Variables representing on disk data structures
MAK: This reads a lot like as if *E_* are history variables (https://*arxiv.org/pdf/*1703.05121.*pdf*). It usually makes the spec mo
2. Variables representing expected values, prefixed by *E_*
This means that during recovery, we can't use *E_** variables.
*E_** variables are used in Invariants to check that disk state is correct.

VARIABLES

MAK: Like with *Vladimir*'s spec of *Floyd*'s algorithm, you should see if you really need this variable.

$PrevState$, For specicying state machine

WE, *WE* $\Rightarrow$ current file to which logger is appending.

REs, *REs* $\Rightarrow$ sequence of read only files, which became read only after they were full.

$MetadataFile$, metadata file storing metadata information of the logger.

E_LowLSN, lowest valid *LSN* of the logger
Data is stored from [*E_LowLSN*, *E_HighLSN*)

$E\_HighLSN$, next valid *LSN* of the logger. *i.e.* not including *E_HighLSN*

E_THIP, *TruncateHeadInProgress* : Is *TruncateHead* in progress ?
Truncate head means removing old data before offset *Head*

E_TTIP, *TruncateTailInProgress* : Is *TruncateTail* in progress ?

Truncate tail means removing current data from end of file.
This is sometimes needed if we want to reset the file or other false progress cases.
Note: New data is appened at tail of the log/file.

E_NWEIP, *WE* is full and Is New *WE* creation In Progress ?

New_WE, *New_WE* file while *E_NWEIP* is TRUE

$TornWrite$, Did last crash caused torn write ?

MAK: It is better to use state or action constraints (https://*tla.msr-inria.inria.fr*/tlatoolbox/doc/model/spec-optio

$MaxNum$ Variable to restrict *TLC* Model Checker (*MC*) to *MaxNum* steps.

$$
\begin{aligned}
&\mathit{TypeOK} \triangleq \\
&\quad \land \quad \mathit{WE} \in [\mathit{id} : 1 \,..\, \mathit{MaxNum},\ \mathit{start} : 1 \,..\, \mathit{MaxNum},\ \mathit{end} : 1 \,..\, \mathit{MaxNum},\ \mathit{version} : 1 \,..\, \mathit{MaxNum}] \\
&\quad \land \quad \mathit{WE}.\mathit{start} \leq \mathit{WE}.\mathit{end} \\
&\quad \land \quad \mathit{New\_WE} \in [\mathit{exist} : \{\text{TRUE, FALSE}\},\ \mathit{id} : 1 \,..\, \mathit{MaxNum},\ \mathit{start} : 1 \,..\, \mathit{MaxNum},\ \mathit{end} : 1 \,..\, \mathit{MaxNum},\ \mathit{vers} \\
&\quad \land \quad \mathit{New\_WE}.\mathit{start} \leq \mathit{New\_WE}.\mathit{end} \\
&\quad \land \quad \mathit{REs} \in \mathit{Seq}([\mathit{id} : 1 \,..\, \mathit{MaxNum},\ \mathit{start} : 1 \,..\, \mathit{MaxNum},\ \mathit{end} : 1 \,..\, \mathit{MaxNum},\ \mathit{version} : 1 \,..\, \mathit{MaxNum}]) \\
&\quad \land \quad \mathit{E\_LowLSN} \in 1 \,..\, \mathit{MaxNum} \\
&\quad \land \quad \mathit{E\_HighLSN} \in 1 \,..\, \mathit{MaxNum} \\
&\quad \land \quad \mathit{PrevState} \quad \in \{\text{“start”},\ \text{“append”},\ \text{“WE\_full\_New\_WE”},\ \text{“New\_WE\_in\_MDT”}, \\
&\qquad\qquad\qquad\qquad \text{“crash”},\ \text{“recovery”},\ \text{“close”}, \\
&\qquad\qquad\qquad\qquad \text{“truncate\_head\_p1”},\ \text{“truncate\_head”}, \\
&\qquad\qquad\qquad\qquad \text{“truncate\_tail\_p1”},\ \text{“truncate\_tail”}\}
\end{aligned}
$$

MetaDataFile ⇒ Stores *headLSN*, *tailLSN*, *tailVersionNum*, *fileNames*

When a new file is created after last *WE* fills up, it's entry is added in *metadataFile*

When Truncation happens, head and tail are updated in *metadataFile*

Recovery uses *metadataFile* for knowing list of valid files in log

headLSN corresponds to current *E_LowLSN*.

lastTailLSN corresponds to last tail truncation.

lastTailVersion is needed to know if we crashed during truncating tail or new data has been added after last tail truncation.

This is needed because we don't update metadata file on every write to log *WE* file.

$$
\begin{aligned}
&\land \mathit{MetadataFile} \in [\mathit{headLSN} : 1 \,..\, \mathit{MaxNum},\ \mathit{lastTailLSN} : 1 \,..\, \mathit{MaxNum},\ \mathit{lastTailVersion} : 1 \,..\, \mathit{MaxNum}, \\
&\qquad\qquad \mathit{cleanShutdown} : \{\text{TRUE, FALSE}\},\ \mathit{fileIds} : \mathit{Seq}(1 \,..\, \mathit{MaxNum})] \\
&\land \mathit{TornWrite} \in \text{BOOLEAN} \\
&\land \mathit{E\_THIP} \in \{\text{TRUE, FALSE}\} \\
&\land \mathit{E\_TTIP} \in \{\text{TRUE, FALSE}\} \\
&\land \mathit{E\_NWEIP} \in \{\text{TRUE, FALSE}\}
\end{aligned}
$$

MAK: This says that *MaxNum* is always 7. Why isn't this a constant?

$$
\land \mathit{MaxNum} = 7
$$

Initial state of the system.

$$
\begin{aligned}
&\mathit{Init} \triangleq \\
&\quad \land \mathit{REs} = \langle\rangle \\
&\quad \land \mathit{WE} = [\mathit{id} \mapsto 1,\ \mathit{start} \mapsto 1,\ \mathit{end} \mapsto 1,\ \mathit{version} \mapsto 1] \\
&\quad \land \mathit{New\_WE} = [\mathit{exist} \mapsto \text{FALSE},\ \mathit{id} \mapsto 1,\ \mathit{start} \mapsto 1,\ \mathit{end} \mapsto 1,\ \mathit{version} \mapsto 1]
\end{aligned}
$$

$\land\ E\_LowLSN = 1$
$\land\ E\_HighLSN = 1$
$\land\ PrevState \quad = \text{“start”}$
$\land\ MetadataFile = [headLSN \mapsto 1,\ lastTailLSN \mapsto 1,\ lastTailVersion \mapsto 1,$
$\qquad\qquad\qquad\qquad cleanShutdown \mapsto \text{FALSE},\ fileIds \mapsto \langle 1 \rangle]$
$\land\ TornWrite = \text{FALSE}$
$\land\ E\_THIP = \text{FALSE}$
$\land\ E\_TTIP = \text{FALSE}$
$\land\ E\_NWEIP = \text{FALSE}$
$\land\ MaxNum = 7$

Helper functions – begin
Don't use E_* variables in helper functions.
$GetFileIds(files) \triangleq$
$\quad [i \in 1\ ..\ Len(files) \mapsto files[i].id]$

$GetMetadataFiles \triangleq$
$\quad \text{LET}\ PresentInMetadataFiles(r) \triangleq$
$\qquad\quad \text{LET}\ SameId(r2Id) \triangleq r.id = r2Id$
$\qquad\quad \text{IN}\quad Len(SelectSeq(MetadataFile.fileIds,\ SameId)) > 0$
$\quad \text{IN}\quad SelectSeq(Append(REs,\ WE),\ PresentInMetadataFiles)$

$GetValidFiles(files,\ lowLSN,\ highLSN) \triangleq$
$\quad \text{LET}\ ValidFile(f) \triangleq \land f.start < f.end$
$\qquad\qquad\qquad\qquad\qquad\qquad \land \neg(\lor (f.end \leq lowLSN) \lor (f.start > highLSN))$
$\quad \text{IN}\quad SelectSeq(files,\ ValidFile)$
Helper functions – end

Append keeps appending to *WE* increasing end *LSN*.
No writes allowed while we do Truncate tail.
Writes allowed while we do Truncate head.
$AppendToFile \triangleq$
Append to file is always allowed except close or crash.
After crash/close, we first do recovery.
$\land\ PrevState \notin \{\text{“crash”},\ \text{“close”}\}$
$\land\ E\_NWEIP = \text{FALSE}$ *WE* is not full
No writes allowed while *truncate_tail* is in progress.
$\land\ E\_TTIP = \text{FALSE}$
$\land\ E\_HighLSN < MaxNum - 1$ Stop *TLC* model checker to generate more cases.
$\land\ WE' = [WE\ \text{EXCEPT}\ !.end = WE.end + 1,$
Every write needs a metadata header with version number
Next write after *TruncateTail* will append to file with new version number. Thanks TLA+
$\qquad\qquad\qquad !.version = MetadataFile.lastTailVersion]$
$\land\ E\_HighLSN' = E\_HighLSN + 1$ Ack to customer that write succeeded
$\land\ PrevState' \quad = \text{“append”}$
$\land\ \text{UNCHANGED}\ \langle E\_LowLSN,\ MaxNum,\ REs,\ MetadataFile,\ TornWrite,\ E\_THIP,\ E\_TTIP,\ E\_NWEIP,$

Action : Write extent is full - Create new Write Extent/file
1. Create a new *WE* file with right header and append data
Next Action: *NewWriteExtentAddToMetadataFile* :
Add to metadata file and move *WE* to RE in memory and $WE' = new\_WE$
These 2 steps are divided in separate Actions to simulate crash and concurrency with different actions.
If we crash before updating *Metadata* file, we ignore this write and *New_WE* file is deleted on recovery.
Todo: Allow *WriteExtentFullNewWE* to run after recovery. Add a field in file : full : {TRUE, FALSE}

$WriteExtentFullNewWE \triangleq$
$\land \lor PrevState = \text{"append"}$
$\lor \land E\_THIP = \text{TRUE}$ Truncate *Head* is allowed concurrent with writes.
$\land PrevState \neq \text{"crash"}$
$\land WE.id < MaxNum - 1$ Stop *MC* after these steps
No writes allowed while *truncate_tail* is in progress.
$\land E\_TTIP = \text{FALSE}$
$\land E\_NWEIP = \text{FALSE}$ Only *WE* full workflow at a time
$\land E\_NWEIP' = \text{TRUE}$ Stop appends to *WE*
Create new *WE*
$\land New\_WE' = [exist \mapsto \text{TRUE},\ id \mapsto WE.id + 1,\ start \mapsto WE.end,\ end \mapsto WE.end + 1,$
Next write after *TruncateTail* will append to file with new version number. Thanks TLA+
$version \mapsto MetadataFile.lastTailVersion]$
$\land PrevState' = \text{"WE\_full\_New\_WE"}$
$\land \text{UNCHANGED } \langle E\_LowLSN,\ E\_HighLSN,\ MaxNum,\ REs,\ WE,\ MetadataFile,\ TornWrite,\ E\_THIP,\ E\_$

Add new write extent file to *MetadataFile* and open for new appends
$NewWriteExtentAddToMetadataFile \triangleq$
$\land E\_NWEIP = \text{TRUE}$
$\land PrevState \neq \text{"crash"}$ only recovery runs after crash
$\land E\_HighLSN < MaxNum - 1$ Stop *TLC* model checker (*MC*) to generate more cases to finish *MC*.
No writes allowed while *truncate_tail* is in progress.
$\land E\_TTIP = \text{FALSE}$ How to assert that *E_TTIP* is false ?
First change *MetadataFile* on disk:
It might seem that, We could have easily done $!.fileIds = Append(MetadataFile.fileIds,\ New\_WE.id)$
but If you do that , invariant *AllMetadataFilesPresentOnDisk* will fail.
i.e. You will see that we can have file entries in *MetadataFile* which don't exist on disk.
This can be because of concurrency with *TH*. Thanks to TLA+.
Don't use *E_LowLSN*, *E_HighLSN* for *MetadataFile*:
Idea is not use Low/High *LSN* for changing disk data structures. They are
maintained parallely and used for assertion in Invariants.
$\land \text{LET } validFiles \triangleq GetValidFiles(Append(REs,\ WE),\ MetadataFile.headLSN,\ WE.end)$
$\text{IN} \quad \land MetadataFile' = [MetadataFile \text{ EXCEPT } !.fileIds = Append(GetFileIds(validFiles),\ New\_WE.id)$
Todo: Split changing RE in separate action to see what concurrency can do.
In-memory data structure change
Because of concurrency in *TH*, it is possible to get Truncation till last *WE*
while we are adding new *WE*

$\land\ REs' = GetValidFiles(validFiles,\ E\_LowLSN,\ E\_HighLSN)$
$\land\ WE' = [id \mapsto New\_WE.id,$
$start \mapsto New\_WE.start,$
$end \mapsto New\_WE.end,$
$version \mapsto New\_WE.version]$
Reset other fields
$\land\ New\_WE' = [New\_WE \text{ EXCEPT } !.exist = \text{FALSE}]$
$\land\ E\_HighLSN' = E\_HighLSN + 1$ Ack to customer that write succeeded
$\land\ PrevState' = \text{“New\_WE\_in\_MDT”}$
$\land\ E\_NWEIP' = \text{FALSE}$ allow appends to *WE* now
$\land\ \text{UNCHANGED}\ \langle E\_LowLSN,\ MaxNum,\ TornWrite,\ E\_THIP,\ E\_TTIP\rangle$

Crash: torn write : last write ignored
We can't have torn write in case of *New_WE*, as only after write is successful,
we update metadata and ack to caller.
$CrashWhileAppend \triangleq$
$\land\ PrevState = \text{“append”}$
$\land\ PrevState' = \text{“crash”}$
$\land\ E\_HighLSN' = E\_HighLSN - 1$ Simulate : we crashed before acking to customer
$\land\ TornWrite' = \text{TRUE}$
don't change metadata file as we can't do it during crash
Invariant : *CleanShutdownOnlyAfterClose* makes sure that we have clean bit set only after close
$\land\ MetadataFile' = [MetadataFile \text{ EXCEPT } !.cleanShutdown = \text{FALSE}]$
$\land\ \text{UNCHANGED}\ \langle E\_LowLSN,\ MaxNum,\ REs,\ WE,\ E\_THIP,\ E\_TTIP,\ E\_NWEIP,\ New\_WE,\ MetadataF$

Normal crash that does not cause data loss
$CrashNoDataLoss \triangleq$
$\land\ PrevState \notin \{\text{“crash”},\ \text{“close”}\}$ we can't crash after close as we aren't running
$\land\ PrevState' = \text{“crash”}$
don't change metadata file as we can't do it during crash
Invariant : *CleanShutdownOnlyAfterClose* makes sure that we have clean bit as TRUE only after close
$\land\ MetadataFile' = [MetadataFile \text{ EXCEPT } !.cleanShutdown = \text{FALSE}]$
$\land\ \text{UNCHANGED}\ \langle E\_LowLSN,\ MaxNum,\ E\_HighLSN,\ REs,\ WE,\ TornWrite,\ E\_THIP,\ E\_TTIP,\ E\_NWE$

Close the log file.
Waits for all operations to finish on log and sets the clean shutdown bit to true.
$Close \triangleq$
$\land\ PrevState \notin \{\text{“crash”},\ \text{“close”}\}$
Close waits for workflows to finish:
New_WE, *truncate_head*, *truncate_tail*
$\land\ E\_NWEIP = \text{FALSE}$
$\land\ E\_TTIP = \text{FALSE}$
$\land\ E\_THIP = \text{FALSE}$
$\land\ PrevState' = \text{“close”}$
$\land\ MetadataFile' = [MetadataFile \text{ EXCEPT } !.cleanShutdown = \text{TRUE},$
$!.lastTailLSN = WE.end]$

$\land$ UNCHANGED $\langle E\_LowLSN,\ MaxNum,\ E\_HighLSN,\ REs,\ WE,\ TornWrite,\ E\_THIP,\ E\_TTIP,\ E\_NWE$

Action: *Recovery* : It happens on Open
After crash, we can't look at value of E_* variables to know the state of the system before close/crash.
After recovery, we set cleanshutdown bit in metadatafile to false, which can only be set to TRUE during close.
We don't do anything if clean shutdown bit is set in the metadata file - fast open case.

$Recovery \triangleq$
$\land \lor PrevState = \text{“crash”}$
$\lor PrevState = \text{“close”}$
$\land$ IF $MetadataFile.cleanShutdown$
THEN $\land REs' = REs$
$\land WE' = WE$
$\land MetadataFile' = [MetadataFile$ EXCEPT $!.cleanShutdown = \text{FALSE}]$
Otherwise we crashed.
Read metadata file and check WE/REs file to rebuild the state.
ELSE LET $allFiles \triangleq GetMetadataFiles$
$lowLSN \triangleq MetadataFile.headLSN$
last file in metadatafile is supposed to be WE
Todo : add a flag in metadata file for which file is WE.
$lastWE \triangleq$ LET $lastWEId \triangleq MetadataFile.fileIds[Len(MetadataFile.fileIds)]$
$SameId(r) \triangleq r.id = lastWEId$
IN $Head(SelectSeq(allFiles,\ SameId))$
$highLSN$: Thanks TLA+
case : Crash during $append - TornWrite$: Last LSN in write file.
case : Crash during $TruncateTail\ phase1$
$LastTailLSN$: if version of WE is $<$ $metadataFile$'s version
We can't append while $TruncateTail$ is going on,
so we can't have both cases occuring at same time together.
$highLSN \triangleq$ LET $lastValidWrite \triangleq lastWE.end$
IN IF $TornWrite$
THEN $lastValidWrite - 1$
ELSE IF $lastWE.version < MetadataFile.lastTailVersion$
THEN $MetadataFile.lastTailLSN$
ELSE $lastValidWrite$
$goodExtents \triangleq GetValidFiles(allFiles,\ lowLSN,\ highLSN)$
$cleanState \triangleq Len(goodExtents) = 0$
IN IF $cleanState$ if we don't have any data
THEN $\land REs' = \langle\rangle$
$\land WE' = [id \mapsto 1,\ start \mapsto lowLSN,\ end \mapsto highLSN,$
$version \mapsto MetadataFile.lastTailVersion]$
Not setting clean bit to false, as it is expected to be false because we crashed.
$\land MetadataFile' = [MetadataFile$ EXCEPT $!.fileIds = \langle 1\rangle]$
ELSE $\land REs' = SubSeq(goodExtents,\ 1,\ Len(goodExtents) - 1)$
$\land WE' = [goodExtents[Len(goodExtents)]$

```
                                                EXCEPT !.end = highLSN,
                                                       !.version = MetadataFile.lastTailVersion]
                                    ∧ MetadataFile' = [MetadataFile EXCEPT !.fileIds = GetFileIds(goodExtents)]
    Reset variables correctly so that appends can work.
    ∧ PrevState' = "recovery"
    ∧ E_THIP' = FALSE
    ∧ E_TTIP' = FALSE
    ∧ E_NWEIP' = FALSE
    Delete New_WE file if it exists – crash happened before updating MetadataFile
    ∧ New_WE' = [New_WE EXCEPT !.exist = FALSE]
    ∧ TornWrite' = FALSE
    ∧ UNCHANGED ⟨E_LowLSN, MaxNum, E_HighLSN⟩

TruncateHead (TH):
Remove old data from the log.
ASSUMPTIONS:
1. There is only 1 TH call at a time - but because of 2 phases, there can be multiple TH in phase2.
2. 1 TH and 1 append can happen concurrently.
3. No TT when TH starts.
We broke truncate head in 2 phases to simulate a crash in between 2 stages.
Other states like appends can happen in between 2 phases.
Phase1 : Update metadata file first.
TruncateHeadP1 ≜
    ∧ PrevState ∉ {"crash", "close"}
    truncate_head waits for new_WE workflow to finish.
    Todo: This is possibly bad as even starting the TruncateHead is waiting.
        It is not very bad because New_WE workflow should finish fast and is also rare.
    ∧ E_NWEIP = FALSE
    ∧ E_TTIP = FALSE No truncate tail in progress.
    ∧ E_LowLSN < E_HighLSN
    ∧ PrevState' = "truncate_head_p1"
    ∧ E_LowLSN' = E_LowLSN + 1
    WE is never removed from MetadataFile in case of TH
    as we need at least 1 file in logger at all time.
    ∧ LET newREs ≜ LET nonTruncatedRE(re) ≜ re.end > E_LowLSN'
                   IN SelectSeq(REs, nonTruncatedRE)
      IN MetadataFile' = [MetadataFile EXCEPT
                                       !.headLSN = E_LowLSN',
                                       !.lastTailLSN = E_HighLSN,
                                       !.fileIds = GetFileIds(Append(newREs, WE))]
    ∧ E_THIP' = TRUE
    ∧ UNCHANGED ⟨E_HighLSN, MaxNum, REs, WE, TornWrite, E_TTIP, E_NWEIP, New_WE⟩

Delete/Zero out RE files in 2nd phase of TruncateHead
TruncateHeadP2 ≜
```

$\land \lor PrevState = $ "truncate_head_p1"
$\quad \lor \land E\_THIP = \text{TRUE}$
$\quad\quad \land PrevState \neq$ "crash" After crash, only recovery runs
$\land PrevState' = $ "truncate_head"
$\land REs' = \text{LET } nonTruncatedRE(re) \triangleq re.end > E\_LowLSN$
$\quad\quad \text{IN } SelectSeq(REs, nonTruncatedRE)$
$\land E\_THIP' = \text{FALSE}$
$\land \text{UNCHANGED } \langle E\_LowLSN, E\_HighLSN, MaxNum, WE, MetadataFile, TornWrite, E\_TTIP, E\_NWEI$

TruncateTailP1 :
Remove data from tail of the log.
ASSUMPTIONS:
1. No *TruncateTail* (*TT*) while append is called.
2. No *Append*/*TruncateHead* (*TH*) while *TT* is going on.

*Phase*1 : Update metadata file first.
We broke truncate tail in 2 phases to simulate a crash in between 2 stages.
Update metadata file first:
If we crash after updating metadata file, we can truncate tail of *WE* on recovery.
Other valid states like appends can't run between 2 phases.

$TruncateTailP1 \triangleq$
$\land PrevState \notin \{$"crash", "close"$\}$
No append, truncate head going on at this time
$\land E\_NWEIP = \text{FALSE}$
$\land E\_THIP = \text{FALSE}$
$\land E\_TTIP = \text{FALSE}$ Only one truncate tail allowed at a time.
$\land E\_LowLSN < E\_HighLSN$
$\land MetadataFile.lastTailVersion < MaxNum - 1$ Restrict *MC* to finite states.
$\land PrevState' = $ "truncate_tail_p1"
$\land E\_HighLSN' = E\_HighLSN - 1$
$\land E\_TTIP' = \text{TRUE}$
In *TruncateTail* − update *tailLsn*, version, *fileIds* in *MetadataFile*
$\land \text{LET } validExtents \triangleq \text{IF } WE.start < WE.end$ *WE* has data
$\quad \text{THEN } Append(REs, [WE \text{ EXCEPT } !.end = WE.end - 1])$
$\quad \text{ELSE LET } lastRE \triangleq REs[Len(REs)]$
$\quad\quad \text{IN } Append(SubSeq(REs, 1, Len(REs) - 1), [lastRE \text{ EXCEPT } !.end = last$
$\quad \text{IN } MetadataFile' = [MetadataFile \text{ EXCEPT } !.lastTailLSN = E\_HighLSN',$
$\quad\quad !.lastTailVersion = MetadataFile.lastTailVersion + 1,$
$\quad\quad !.fileIds = GetFileIds(validExtents)]$
$\land \text{UNCHANGED } \langle E\_LowLSN, WE, REs, MaxNum, TornWrite, E\_THIP, E\_NWEIP, New\_WE \rangle$

Now, actual delete file/zero *WE* file's tail in Phase 2.

$TruncateTailP2 \triangleq$
$\land PrevState = $ "truncate_tail_p1"
$\land PrevState' = $ "truncate_tail"
$\land \text{IF } WE.start < WE.end$ *WE* has data

THEN $\land\ WE' = [WE \text{ EXCEPT } !.end = WE.end - 1]$
$\land\ RES' = RES$
ELSE $\land\ WE' = \text{LET } lastRE \triangleq RES[Len(RES)]$
IN $[lastRE \text{ EXCEPT } !.end = lastRE.end - 1]$
$\land\ RES' = SubSeq(RES, 1, Len(RES) - 1)$
$\land\ E\_TTIP' = \text{FALSE}$
$\land$ UNCHANGED $\langle E\_LowLSN, E\_HighLSN, MaxNum, MetadataFile, TornWrite, E\_THIP, E\_NWEIP, N$

$Next \triangleq$
$\lor\ AppendToFile$
$\lor\ WriteExtentFullNewWE$
$\lor\ NewWriteExtentAddToMetadataFile$
$\lor\ CrashWhileAppend$
$\lor\ CrashNoDataLoss$
$\lor\ Close$
$\lor\ Recovery$
$\lor\ TruncateHeadP1$
$\lor\ TruncateHeadP2$
$\lor\ TruncateTailP1$
$\lor\ TruncateTailP2$
Not modelling Data Loss.
I am not sure, if we should just fail to open if we find we lost data
so that we build from new replica.
$\lor\ CrashDataLost$

Invariants:

Invariant 1: *NoDataLoss*
All read only extents have non missing *LSN*
$RES[1].start < RES[1].end \triangleq RES[2].start < RES[2].end \triangleq RES[3].start < \ \ldots$
write extent has latest data
$E\_HighLSN \triangleq WE.end \geq WE.start \triangleq RES[last].end$

[*dangling_extent*] [*lowLSN*, *highLSN* − valid range] [*dangling_extent*]
$OrderedExtent(ex1, ex2, highLSN) \triangleq$
$\land\ ex1.start < ex1.end$
$\land\ ex1.end = ex2.start$
$\land\ ex1.end \leq highLSN$

$ValidReadOnlyExtents \triangleq$
$\land\ \forall\, i \in 1 \,..\, Len(RES) - 1 :\ \land\ OrderedExtent(RES[i], RES[i+1], E\_HighLSN)$
$\land\ RES[i].end < E\_HighLSN$
$\land$ IF $Len(RES) > 0$
THEN $OrderedExtent(RES[Len(RES)], WE, E\_HighLSN)$
ELSE $1 = 1$

$ValidWriteExtent \triangleq$

$\land\ WE.start \leq WE.end$
$\land\ WE.end = E\_HighLSN$

$MetadataExtentsCoverDataRange \triangleq$
LET $allFiles \triangleq GetMetadataFiles$
$firstFile \triangleq allFiles[1]$
$lastFile \triangleq allFiles[Len(allFiles)]$
IN $\land\ firstFile.start \leq E\_LowLSN$
$\land\ lastFile.end \geq E\_HighLSN$

$NoDataLoss \triangleq$
Not valid state during crash or *truncate_tail_phase*1
$\lor\ PrevState \in \{$"crash", "truncate_tail_p1"$\}$
$\lor\ E\_TTIP =$ TRUE *TruncateTail* in progress
$\lor\ \land\ ValidReadOnlyExtents$
$\land\ ValidWriteExtent$
$\land\ MetadataExtentsCoverDataRange$
$\land\ E\_LowLSN \leq E\_HighLSN$

Invariant 2: No dangling files on disk
No file/extent present on disk which are not required.
$NotDanglingExtent(ex,\ lowLSN,\ highLSN) \triangleq$
$\neg(\ \lor\ ex.start \geq highLSN$
$\lor\ ex.end \leq lowLSN)$

$NoDanglingExtents \triangleq$
$\lor\ PrevState =$ "crash"
TH/TT is in progress - so some files are dangling.
$\lor\ E\_THIP =$ TRUE
$\lor\ E\_TTIP =$ TRUE
$\lor\ \land \forall\, i \in 1\,..\,Len(REs) : NotDanglingExtent(REs[i],\ E\_LowLSN,\ E\_HighLSN)$
$\land\ \lor\ WE.start\ \ = WE.end$ *WE* is empty
$\lor\ E\_LowLSN = E\_HighLSN$ There is no data in log
If there is some data, *WE* should be valid
$\lor\ NotDanglingExtent(WE,\ E\_LowLSN,\ E\_HighLSN)$

Invariant 3 : Correctness of *MetadataFile*:
1. *FileIds* should be in increasing order
2. *HeadLSN* should be same as Expected *E_LowLSN*

$IsFileIdPresent(fileIds,\ id) \triangleq$
LET $SameId(fid) \triangleq fid = id$
IN $Len(SelectSeq(fileIds,\ SameId)) = 1$

$AllMetadataFilesPresentOnDisk \triangleq$
LET $allFiles \triangleq Append(REs,\ WE)$
$allFileIds \triangleq [i \in 1\,..\,Len(allFiles) \mapsto allFiles[i].id]$
IN $\land \forall\, i \in 1\,..\,Len(MetadataFile.fileIds) :$

$IsFileIdPresent(allFileIds,\ MetadataFile.fileIds[i])$

$\land$ IF $New\_WE.exist$

if New_WE is present, it should not be in RE, WE and mentioned in $MetadataFile$

$i.e\ New\_WE$ is transient file

THEN $\land \neg IsFileIdPresent(allFileIds,\ New\_WE.id)$

$\land \neg IsFileIdPresent(MetadataFile.fileIds,\ New\_WE.id)$

ELSE $1 = 1$

$MetadataFileCorrect \triangleq$

$\land$ No missing file $-$ *files* in increasing order

$\forall\, i \in 1\,..\,Len(MetadataFile.fileIds) - 1 :$

$MetadataFile.fileIds[i] < MetadataFile.fileIds[i+1]$

$\land\ MetadataFile.headLSN = E\_LowLSN$

$\land\ AllMetadataFilesPresentOnDisk$ even during crash

$\land$ IF $MetadataFile.cleanShutdown$

THEN $MetadataFile.lastTailLSN = E\_HighLSN$

Todo: What should still be correct in clean shutdown case ?

ELSE $1 = 1$

Invariant 4: Valid version number

$CorrectVersionNumber \triangleq$

IF $MetadataFile.lastTailLSN < E\_HighLSN$ some write finished after last TT

THEN $WE.version = MetadataFile.lastTailVersion$

Multiple TT can happen one after another increasing version no. Thanks TLA+

ELSE $WE.version \leq MetadataFile.lastTailVersion$

If we have clean shutdown state - except after close, we are in bad state.

$CleanShutdownOnlyAfterClose \triangleq$

$\neg(\ \land\ PrevState \neq$ "close"

$\land\ MetadataFile.cleanShutdown = \text{TRUE}$

)

Change below value to see different steps taken for particular test run.

$LSNSteps \triangleq$

$E\_HighLSN < MaxNum$

Spec Ends

MAK: Do you check that the algorithm makes progress? Remember, a system that does nothing doesn't violate any safety pro

Todo:

Spec after this is *WIP* and not used.

Invariants that should fail - Signifies that we have handled these cases.

TruncateTail is not called on empty WE for truncating data upto *REs*

Todo: This is not failing - This case is not handled.

Using E_TTIP is not correct as that means that TT has already begin.

$TruncateTailCalledOnEmptyWE \triangleq$
$\neg(\ \wedge E\_TTIP = \text{TRUE}$
$\quad \wedge WE.start = WE.end$
$\ )$

$TruncateHeadCalledOnEmptyWE \triangleq$
$\neg(\ \wedge E\_THIP = \text{TRUE}$
$\quad \wedge WE.start = WE.end$
$\ )$

Crash: Not modelling case of data lost.
we lost all data after some *LSN*
Todo: need to model - data lost in between *E_LowLSN* and *E_HighLSN*
In that case, we will Fail replica in real world
and rebuild from another source.

$MaxOf2(a,\ b) \triangleq$
IF $a < b$
THEN b
ELSE a

$CrashDataLost \triangleq$
$\wedge\ PrevState' = \text{“crash”}$
$\wedge\ E\_HighLSN' = \text{IF } E\_HighLSN > (MaxNum \div 2)$
$\text{THEN } MaxOf2(E\_LowLSN,\ MaxNum \div 2)$
$\text{ELSE IF } E\_HighLSN > 3$
$\text{THEN } MaxOf2(E\_LowLSN,\ 3)$
$\text{ELSE } MaxOf2(E\_LowLSN,\ 1)$
$\wedge\ MetadataFile' = [MetadataFile \text{ EXCEPT } !.cleanShutdown = \text{FALSE}]$
$\wedge\ \text{UNCHANGED } \langle E\_LowLSN,\ MaxNum,\ REs,\ WE,\ TornWrite\rangle$

* Modification History
* Last modified *Tue Nov* 17 09:44:38 *PST* 2020 by *markus*
* Last modified *Tue Nov* 17 09:22:58 *PST* 2020 by *markus*
* Last modified *Mon Nov* 16 16:32:52 *PST* 2020 by *asnegi*
* Created *Wed Oct* 28 17:55:29 *PDT* 2020 by *asnegi*