```
──────────────────── MODULE LogSync ────────────────────
EXTENDS TLC, Naturals, Sequences

CONSTANTS Key, WatchClient, nil

 state is in-memory data
 db is the same data but on the db
 watch_info[c].chan is the receive channel for client
VARIABLES pc, current_key, db,
    state, state_seq, next_log, next_seq, wait_list, lru_keys,
    watch_pc,
    watch_keys, watch_key_pc,
    watch_info,
    watch_state, watch_local_key, watch_local_info,
    num_client_restart, num_main_restart, num_delete_state

main_vars ≜ ⟨pc, current_key, db⟩

watch_local_vars ≜ ⟨
    watch_pc, watch_state,
    watch_local_key, watch_local_info⟩

watch_key_vars ≜ ⟨watch_keys, watch_key_pc⟩

watch_remove_vars ≜ ⟨watch_info⟩

watch_vars ≜ ⟨watch_local_vars, watch_key_vars, watch_remove_vars⟩

server_vars ≜ ⟨state, state_seq, next_log, next_seq, wait_list, lru_keys⟩

aux_vars ≜ ⟨num_client_restart, num_main_restart, num_delete_state⟩

vars ≜ ⟨
    main_vars, server_vars,
    watch_local_vars, watch_key_vars, watch_remove_vars,
    aux_vars⟩


max_log_size ≜ 3

max_client_restart ≜ 1
max_main_restart  ≜ 1
max_delete_state   ≜ 2

Status ≜ {"Running", "Completed", "Gone"}

LogEntry ≜ 20 . . 30

Info ≜ [logs : Seq(LogEntry), status : Status]
```

$NullInfo \triangleq Info \cup \{nil\}$

$NullKey \triangleq Key \cup \{nil\}$

$NullLogEntry \triangleq LogEntry \cup \{nil\}$

$Event \triangleq [$
$\quad type : \{\text{“AddLog”}, \text{“Finished”}, \text{“JobGone”}\},$
$\quad key : Key,\ line : NullLogEntry]$

$NullEvent \triangleq Event \cup \{nil\}$

$Channel \triangleq [status : \{\text{“Empty”}, \text{“Ready”}, \text{“Consumed”}\},\ data : NullEvent]$

$StateSeq \triangleq 100 \,..\, 120$

$WatchState \triangleq \{\text{“Init”}, \text{“AddToWaitList”}, \text{“WaitOnChan”}, \text{“UpdateDB”}\}$

$WatchInfo \triangleq [$
$\quad chan : Channel,$
$\quad seq : [Key \rightarrow StateSeq],$
$\quad log\_index : [Key \rightarrow Nat],$
$\quad keys : \text{SUBSET}\ Key$
$]$

$TypeOK \triangleq$
$\quad \land\ pc \in \{\text{“Init”}, \text{“PushJob”}\}$
$\quad \land\ current\_key \in NullKey$
$\quad \land\ db \in [Key \rightarrow NullInfo]$

$\quad \land\ state \in [Key \rightarrow NullInfo]$
$\quad \land\ state\_seq \in [Key \rightarrow StateSeq]$
$\quad \land\ next\_log \in LogEntry$
$\quad \land\ next\_seq \in StateSeq$
$\quad \land\ wait\_list \in [Key \rightarrow \text{SUBSET}\ WatchClient]$
$\quad \land\ lru\_keys \subseteq Key$

$\quad \land\ watch\_pc \in [WatchClient \rightarrow WatchState]$
$\quad \land\ watch\_keys \in [WatchClient \rightarrow \text{SUBSET}\ Key]$
$\quad \land\ watch\_key\_pc \in [WatchClient \rightarrow \{\text{“Init”}, \text{“SetWaitList”}\}]$
$\quad \land\ watch\_info \in [WatchClient \rightarrow WatchInfo]$
$\quad \land\ watch\_state \in [WatchClient \rightarrow [Key \rightarrow NullInfo]]$
$\quad \land\ watch\_local\_key \in [WatchClient \rightarrow NullKey]$
$\quad \land\ watch\_local\_info \in [WatchClient \rightarrow NullInfo]$

$\quad \land\ num\_client\_restart \in 0 \,..\, max\_client\_restart$
$\quad \land\ num\_main\_restart \in 0 \,..\, max\_main\_restart$
$\quad \land\ num\_delete\_state \in 0 \,..\, max\_delete\_state$

```
$consumed\_chan \triangleq [status \mapsto \text{"Consumed"},\ data \mapsto nil]$

$init\_info \triangleq [$
    $chan \mapsto consumed\_chan,$
    $seq \mapsto [k \in Key \mapsto 100],$
    $log\_index \mapsto [k \in Key \mapsto 0],$
    $keys \mapsto \{\}$
$]$

$Init \triangleq$
    $\land pc = \text{"Init"}$
    $\land current\_key = nil$
    $\land db = [k \in Key \mapsto nil]$

    $\land state = [k \in Key \mapsto nil]$
    $\land state\_seq = [k \in Key \mapsto 100]$
    $\land next\_log = 20$
    $\land wait\_list = [k \in Key \mapsto \{\}]$
    $\land next\_seq\ = 100$
    $\land lru\_keys\ \ = \{\}$

    $\land watch\_pc = [c \in WatchClient \mapsto \text{"Init"}]$
    $\land watch\_keys = [c \in WatchClient \mapsto \{\}]$
    $\land watch\_key\_pc = [c \in WatchClient \mapsto \text{"Init"}]$
    $\land watch\_info = [c \in WatchClient \mapsto init\_info]$
    $\land watch\_state = [c \in WatchClient \mapsto [k \in Key \mapsto nil]]$
    $\land watch\_local\_key = [c \in WatchClient \mapsto nil]$
    $\land watch\_local\_info = [c \in WatchClient \mapsto nil]$

    $\land num\_client\_restart = 0$
    $\land num\_main\_restart = 0$
    $\land num\_delete\_state\ \ = 0$

$newJob \triangleq [logs \mapsto \langle\rangle,\ status \mapsto \text{"Running"}]$

$AddDBJob(k) \triangleq$
    $\land pc = \text{"Init"}$
    $\land db[k] = nil$
    $\land pc' = \text{"PushJob"}$
    $\land current\_key' = k$
    $\land db' = [db \text{ EXCEPT } ![k] = newJob]$
    $\land \text{UNCHANGED } server\_vars$
    $\land \text{UNCHANGED } watch\_vars$
    $\land \text{UNCHANGED } aux\_vars$
```

```
$updateStateSeq(k) \triangleq$
    $\land\ next\_seq' = next\_seq + 1$
    $\land\ state\_seq' = [state\_seq \text{ EXCEPT } ![k] = next\_seq']$

$PushJob \triangleq$
    $\land\ pc = \text{“PushJob”}$
    $\land\ pc' = \text{“Init”}$
    $\land\ current\_key' = nil$
    $\land\ state' = [state \text{ EXCEPT } ![current\_key] = db[current\_key]]$
    $\land\ lru\_keys' = lru\_keys \cup \{current\_key\}$
    $\land\ \text{UNCHANGED } \langle next\_seq, state\_seq \rangle$
    $\land\ \text{UNCHANGED } wait\_list$
    $\land\ \text{UNCHANGED } db$
    $\land\ \text{UNCHANGED } next\_log$
    $\land\ \text{UNCHANGED } watch\_vars$
    $\land\ \text{UNCHANGED } aux\_vars$

$canPushKeyToClient(k, c, old\_info) \triangleq$
    $\land\ old\_info.chan.status = \text{“Empty”}$
    $\land\ c \in wait\_list'[k]$
    $\land\ old\_info.seq[k] < state\_seq'[k]$

$pushToClientChan(k, c, old\_info) \triangleq$
  LET
      $last\_index \triangleq old\_info.log\_index[k]$
      $state\_index \triangleq Len(state'[k].logs)$

      $new\_line \triangleq state'[k].logs[last\_index + 1]$

      $add\_event \triangleq [$
          $type \mapsto \text{“AddLog”},$
          $key \mapsto k,$
          $line \mapsto new\_line]$

      $finished\_or\_gone \triangleq$
          IF $state'[k].status = \text{“Gone”}$
              THEN $\text{“JobGone”}$
              ELSE $\text{“Finished”}$

      $finish\_event \triangleq [$
          $type \mapsto finished\_or\_gone,$
          $key \mapsto k,$
          $line \mapsto nil]$

      $is\_running \triangleq state'[k].status = \text{“Running”}$
```

```
        $add\_log\_cond \triangleq last\_index < state\_index$

        $update\_seq\_cond \triangleq$
            IF $last\_index \geq state\_index$
                THEN TRUE
                ELSE IF $last\_index + 1 \geq state\_index \land is\_running$
                    THEN TRUE
                    ELSE FALSE

        $new\_event \triangleq$
            IF $add\_log\_cond$
                THEN $add\_event$
                ELSE $finish\_event$

        $new\_chan \triangleq [status \mapsto \text{“Ready”},\ data \mapsto new\_event]$

        $new\_log\_index \triangleq [old\_info.log\_index$ EXCEPT $![k] = last\_index + 1]$

        $new\_seq \triangleq [$
            $old\_info.seq$ EXCEPT $![k] =$
                IF $update\_seq\_cond$
                    THEN $state\_seq'[k]$
                    ELSE $old\_info.seq[k]$
            $]$

        $new\_info \triangleq [$
            $chan \mapsto new\_chan,$
            $seq \mapsto new\_seq,$
            $log\_index \mapsto new\_log\_index,$
            $keys \mapsto old\_info.keys]$
    IN
        $watch\_info' = [watch\_info$ EXCEPT $![c] = new\_info]$

$pushToClientOrDoNothing(c,\ old\_info) \triangleq$
    LET
        $doNothing \triangleq$
            $\land \forall k \in Key : \neg canPushKeyToClient(k,\ c,\ old\_info)$
            $\land watch\_info' = [watch\_info$ EXCEPT $![c] = old\_info]$
    IN
    $\lor \exists k \in Key :$
        $\land canPushKeyToClient(k,\ c,\ old\_info)$
        $\land pushToClientChan(k,\ c,\ old\_info)$
    $\lor doNothing$

$pushKeyOrDoNothing(k) \triangleq$
    LET
```

```
        $doPush \triangleq$
            $\exists\, c \in WatchClient :$
                $\land\ canPushKeyToClient(k,\ c,\ watch\_info[c])$
                $\land\ pushToClientChan(k,\ c,\ watch\_info[c])$

        $doNothing \triangleq$
            $\land\ \forall\, c \in WatchClient : \neg canPushKeyToClient(k,\ c,\ watch\_info[c])$
            $\land$ UNCHANGED $watch\_info$
    IN
        $doPush \lor doNothing$

$ProduceLog(k) \triangleq$
    $\land\ state[k] \neq nil$
    $\land\ state[k].status =$ "Running"
    $\land\ Len(state[k].logs) < max\_log\_size$

    $\land\ next\_log' = next\_log + 1$
    $\land\ state' = [state$ EXCEPT $![k].logs = Append(@,\ next\_log')]$

    $\land\ updateStateSeq(k)$

    $\land$ UNCHANGED $wait\_list$
    $\land\ pushKeyOrDoNothing(k)$

    $\land$ UNCHANGED $lru\_keys$
    $\land$ UNCHANGED $main\_vars$
    $\land$ UNCHANGED $watch\_local\_vars$
    $\land$ UNCHANGED $watch\_key\_vars$
    $\land$ UNCHANGED $aux\_vars$


$FinishJob(k) \triangleq$
    $\land\ state[k] \neq nil$
    $\land\ state[k].status =$ "Running"
    $\land\ state' = [state$ EXCEPT $![k].status =$ "Completed"$]$
    $\land\ updateStateSeq(k)$

    $\land$ UNCHANGED $wait\_list$
    $\land\ pushKeyOrDoNothing(k)$

    $\land$ UNCHANGED $lru\_keys$
    $\land$ UNCHANGED $next\_log$
    $\land$ UNCHANGED $main\_vars$
    $\land$ UNCHANGED $watch\_local\_vars$
    $\land$ UNCHANGED $watch\_key\_vars$
    $\land$ UNCHANGED $aux\_vars$
```

```
$NewWatchChan(c) \triangleq$
    LET
        $new\_chan \triangleq [status \mapsto \text{"Empty"},\ data \mapsto nil]$

        $new\_info \triangleq [watch\_info[c]\ \text{EXCEPT}\ !.chan = new\_chan]$
    IN
        $\land\ watch\_pc[c] = \text{"Init"}$
        $\land\ watch\_pc' = [watch\_pc\ \text{EXCEPT}\ ![c] = \text{"WaitOnChan"}]$

        $\land$ UNCHANGED $server\_vars$
        $\land\ pushToClientOrDoNothing(c,\ new\_info)$

        $\land$ UNCHANGED $\langle watch\_keys,\ watch\_key\_pc\rangle$
        $\land$ UNCHANGED $\langle watch\_state,\ watch\_local\_key,\ watch\_local\_info\rangle$
        $\land$ UNCHANGED $main\_vars$
        $\land$ UNCHANGED $aux\_vars$


$active\_keys \triangleq$
    LET
        $db\_set \triangleq \{k \in Key : db[k] \neq nil \land db[k].status = \text{"Running"}\}$
    IN
        $db\_set \setminus \{current\_key\}$


$clearWatchStateKeyNotInSet(c,\ set) \triangleq$
    $[k \in Key \mapsto$ IF $k \in set$ THEN $watch\_state[c][k]$ ELSE $nil]$

$UpdateWatchKeys(c) \triangleq$
    $\land\ watch\_key\_pc[c] = \text{"Init"}$
    $\land\ watch\_keys[c] \neq active\_keys$
    $\land\ watch\_key\_pc' = [watch\_key\_pc\ \text{EXCEPT}\ ![c] = \text{"SetWaitList"}]$

    $\land\ watch\_keys' = [watch\_keys\ \text{EXCEPT}\ ![c] = active\_keys]$
    $\land\ watch\_state' = [watch\_state\ \text{EXCEPT}$
            $![c] = clearWatchStateKeyNotInSet(c,\ active\_keys)]$

    $\land$ UNCHANGED $watch\_remove\_vars$
    $\land$ UNCHANGED $\langle watch\_pc\rangle$
    $\land$ UNCHANGED $\langle watch\_local\_key,\ watch\_local\_info\rangle$
    $\land$ UNCHANGED $main\_vars$
    $\land$ UNCHANGED $server\_vars$
    $\land$ UNCHANGED $aux\_vars$


$updateLRUKeys(c) \triangleq$
    LET
        $removed \triangleq watch\_info'[c].keys$
        $added \triangleq \{k \in watch\_info[c].keys : wait\_list'[k] = \{\}\}$
    IN
        $lru\_keys' = (lru\_keys \cup added) \setminus removed$

$updateServerWaitList(c) \triangleq$
    LET
        $old\_set(k) \triangleq wait\_list[k]$
        $new\_set(k) \triangleq$
            IF $k \in watch\_keys[c]$
                THEN $old\_set(k) \cup \{c\}$
                ELSE $old\_set(k) \setminus \{c\}$
    IN
        $wait\_list' = [k \in Key \mapsto new\_set(k)]$

$createPlaceHolderStateForWaitList \triangleq$
    LET
        $in\_wait\_list(k) \triangleq wait\_list'[k] \neq \{\}$

        $keysWithNilState \triangleq$
            $\{k \in Key : in\_wait\_list(k) \land state[k] = nil\}$

        $new\_state\_fn(k) \triangleq$
            IF $k \in keysWithNilState$
                THEN $[logs \mapsto \langle\rangle, status \mapsto$ "Gone"$]$
                ELSE $state[k]$

        $new\_seq\_fn(k) \triangleq$
            IF $k \in keysWithNilState$
                THEN $next\_seq'$
                ELSE $state\_seq[k]$

        $update\_state \triangleq$
            $\land next\_seq' = next\_seq + 1$
            $\land state' = [k \in Key \mapsto new\_state\_fn(k)]$
            $\land state\_seq' = [k \in Key \mapsto new\_seq\_fn(k)]$

        $do\_nothing \triangleq$
            UNCHANGED $\langle state, next\_seq, state\_seq\rangle$
    IN
        IF $keysWithNilState \neq \{\}$
            THEN $update\_state$
            ELSE $do\_nothing$

$removeSeqLogIndexNotInWaitList(c) \triangleq$
    LET
        $old\_info \triangleq watch\_info[c]$

        $in\_list(k) \triangleq wait\_list'[k] \neq \{\}$
        new_seq ≜
            [k ∈ Key ↦ IF in_list(k) THEN old_info.seq[k] ELSE 100]

        new_log_index ≜
            [k ∈ Key ↦ IF in_list(k) THEN old_info.log_index[k] ELSE 0]
    IN
        [old_info EXCEPT
            !.seq = new_seq,
            !.log_index = new_log_index,
            !.keys = watch_keys[c]]

AddToWaitList(c) ≜
    ∧ watch_key_pc[c] = "SetWaitList"
    ∧ watch_key_pc' = [watch_key_pc EXCEPT ![c] = "Init"]
    ∧ updateServerWaitList(c)

    ∧ createPlaceHolderStateForWaitList
    ∧ LET
            new_info ≜ removeSeqLogIndexNotInWaitList(c)
        IN
            pushToClientOrDoNothing(c, new_info)
    ∧ updateLRUKeys(c)

    ∧ UNCHANGED ⟨watch_keys⟩
    ∧ UNCHANGED watch_local_vars
    ∧ UNCHANGED main_vars
    ∧ UNCHANGED next_log
    ∧ UNCHANGED aux_vars


updateStateFromChan(c) ≜
    LET
        chan ≜ watch_info[c].chan
        k ≜ chan.data.key
        type ≜ chan.data.type
        log_line ≜ chan.data.line

        old_state ≜ watch_state[c][k]

        old_logs ≜
            IF old_state = nil
                THEN ⟨⟩
                ELSE old_state.logs

        new_state ≜
            [logs ↦ Append(old_logs, log_line), status ↦ "Running"]

        do_add_log ≜
```

```
            $\land$ $watch\_state' = [$
                  $watch\_state$ EXCEPT $![c][k] = new\_state]$
            $\land$ UNCHANGED $\langle watch\_local\_key, watch\_local\_info \rangle$
            $\land$ $watch\_pc' = [watch\_pc$ EXCEPT $![c] =$ "Init"$]$

        $new\_status \triangleq$
            IF $type =$ "JobGone"
                THEN "Gone"
                ELSE "Completed"

        $do\_complete \triangleq$
            $\land$ $watch\_state' = [$
                $watch\_state$ EXCEPT
                    $![c][k] = [logs \mapsto old\_logs, status \mapsto new\_status]]$

            $\land$ $watch\_local\_key' = [watch\_local\_key$ EXCEPT $![c] = k]$

            $\land$ $watch\_local\_info' = [$
                $watch\_local\_info$ EXCEPT $![c] = watch\_state'[c][k]]$

            $\land$ $watch\_pc' = [watch\_pc$ EXCEPT $![c] =$ "UpdateDB"$]$

        $do\_nothing \triangleq$
            $\land$ $watch\_pc' = [watch\_pc$ EXCEPT $![c] =$ "Init"$]$
            $\land$ UNCHANGED $watch\_state$
            $\land$ UNCHANGED $\langle watch\_local\_key, watch\_local\_info \rangle$
    IN
        IF $k \in watch\_keys[c]$
            THEN IF $type =$ "AddLog"
                THEN $do\_add\_log$
                ELSE $do\_complete$
            ELSE $do\_nothing$

$ConsumeWatchChan(c) \triangleq$
    $\land$ $watch\_pc[c] =$ "WaitOnChan"
    $\land$ $watch\_info[c].chan.status =$ "Ready"

    $\land$ $watch\_info' = [$
            $watch\_info$ EXCEPT
                $![c].chan.status =$ "Consumed",
                $![c].chan.data = nil]$

    $\land$ $updateStateFromChan(c)$

    $\land$ UNCHANGED $\langle watch\_keys, watch\_key\_pc \rangle$
    $\land$ UNCHANGED $main\_vars$
    $\land$ UNCHANGED $server\_vars$
    $\land$ UNCHANGED $aux\_vars$
```

```
$UpdateDB(c) \triangleq$
    LET
        $k \triangleq watch\_local\_key[c]$
        $info \triangleq watch\_local\_info[c]$
    IN
        $\land\ watch\_pc[c] = \text{“UpdateDB”}$
        $\land\ watch\_pc' = [watch\_pc \text{ EXCEPT } ![c] = \text{“Init”}]$
        $\land\ db' = [db \text{ EXCEPT } ![k] = info]$
        $\land\ watch\_local\_key' = [watch\_local\_key \text{ EXCEPT } ![c] = nil]$
        $\land\ watch\_local\_info' = [watch\_local\_info \text{ EXCEPT } ![c] = nil]$
        $\land$ UNCHANGED $watch\_remove\_vars$
        $\land$ UNCHANGED $\langle watch\_keys, watch\_key\_pc, watch\_state \rangle$
        $\land$ UNCHANGED $server\_vars$
        $\land$ UNCHANGED $\langle pc, current\_key \rangle$
        $\land$ UNCHANGED $aux\_vars$

$removeClientFromWaitList(k, c) \triangleq$
    $wait\_list[k] \setminus \{c\}$

$ClientRestart(c) \triangleq$
    $\land\ num\_client\_restart < max\_client\_restart$
    $\land\ num\_client\_restart' = num\_client\_restart + 1$
    $\land\ watch\_info' = [watch\_info \text{ EXCEPT } ![c] = init\_info]$
    $\land\ watch\_keys' = [watch\_keys \text{ EXCEPT } ![c] = \{\}]$

    $\land\ watch\_local\_key' = [watch\_local\_key \text{ EXCEPT } ![c] = nil]$
    $\land\ watch\_local\_info' = [watch\_local\_info \text{ EXCEPT } ![c] = nil]$

    $\land\ watch\_state' = [watch\_state \text{ EXCEPT } ![c] = [k \in Key \mapsto nil]]$
    $\land\ watch\_pc' = [watch\_pc \text{ EXCEPT } ![c] = \text{“Init”}]$
    $\land\ wait\_list' = [k \in Key \mapsto removeClientFromWaitList(k, c)]$
    $\land\ watch\_key\_pc' = [watch\_key\_pc \text{ EXCEPT } ![c] = \text{“Init”}]$
    $\land\ updateLRUKeys(c)$

    $\land$ UNCHANGED $\langle state, state\_seq, next\_log, next\_seq \rangle$
    $\land$ UNCHANGED $main\_vars$
    $\land$ UNCHANGED $\langle num\_main\_restart, num\_delete\_state \rangle$

$MainRestart \triangleq$
    $\land\ num\_main\_restart < max\_main\_restart$
    $\land\ num\_main\_restart' = num\_main\_restart + 1$
    $\land\ current\_key' = nil$
    $\land\ pc' = \text{“Init”}$
    $\land$ UNCHANGED $db$
    $\land$ UNCHANGED $\langle num\_client\_restart, num\_delete\_state \rangle$
    $\land$ UNCHANGED $server\_vars$
    $\land$ UNCHANGED $watch\_vars$

$DeleteRandomKeyInState(k) \triangleq$
    $\land num\_delete\_state < max\_delete\_state$
    $\land num\_delete\_state' = num\_delete\_state + 1$
    $\land state[k] \neq nil$
    $\land k \in lru\_keys$

    $\land state' = [state$ EXCEPT $![k] = nil]$
    $\land state\_seq' = [state\_seq$ EXCEPT $![k] = 100]$
    $\land wait\_list' = [wait\_list$ EXCEPT $![k] = \{\}]$
    $\land lru\_keys' = lru\_keys \setminus \{k\}$

    $\land$ UNCHANGED $\langle next\_log, next\_seq\rangle$
    $\land$ UNCHANGED $\langle num\_client\_restart, num\_main\_restart\rangle$
    $\land$ UNCHANGED $main\_vars$
    $\land$ UNCHANGED $watch\_local\_vars$
    $\land$ UNCHANGED $watch\_key\_vars$
    $\land$ UNCHANGED $watch\_remove\_vars$

$statusIsFinished(st) \triangleq$
    $\lor st =$ "Completed"
    $\lor st =$ "Gone"

$serverWatchClientKeys(c) \triangleq \{k \in Key : c \in wait\_list[k]\}$

$TerminateCond \triangleq$
    $\land \forall k \in Key : db[k] \neq nil \land statusIsFinished(db[k].status)$
    $\land \forall k \in Key : state[k] \neq nil \Rightarrow statusIsFinished(state[k].status)$
    $\land \forall c \in WatchClient :$
        $\land watch\_pc[c] =$ "WaitOnChan"
        $\land watch\_keys[c] = active\_keys$
        $\land watch\_keys[c] = serverWatchClientKeys(c)$
        $\land watch\_info[c].chan.status =$ "Empty"

$Terminated \triangleq$
    $\land TerminateCond$
    $\land$ UNCHANGED $vars$

$Next \triangleq$
    $\lor \exists k \in Key :$
        $\lor AddDBJob(k)$
        $\lor ProduceLog(k)$
        $\lor FinishJob(k)$
        $\lor\ DeleteRandomKeyInState(k)$
    $\lor\ PushJob$

    $\lor\ \exists\, c \in WatchClient :$
        $\lor\ NewWatchChan(c)$
        $\lor\ UpdateWatchKeys(c)$
        $\lor\ AddToWaitList(c)$
        $\lor\ ConsumeWatchChan(c)$
        $\lor\ UpdateDB(c)$
        $\lor\ ClientRestart(c)$

    $\lor\ MainRestart$

    $\lor\ Terminated$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$FairSpec \triangleq Spec \land \mathrm{WF}_{vars}(Next)$

$AlwaysTerminate \triangleq \Diamond TerminateCond$

$AllJobsMustBeFinished \triangleq$
    $TerminateCond \Rightarrow$
        $\forall\, k \in Key : db[k] \neq nil \land statusIsFinished(db[k].status)$

$infoEqual(db\_val,\ state\_val) \triangleq$
    $\land\ db\_val.status \in \{$"Completed", "Gone"$\}$
    $\land\ state\_val.status \in \{$"Completed", "Gone"$\}$

    $\land\ state\_val.status =$ "Completed" $\Rightarrow db\_val.logs = state\_val.logs$
    $\land\ state\_val.status =$ "Completed" $\Rightarrow db\_val.status =$ "Completed"

$DBShouldSameAsMem \triangleq$
    $TerminateCond \Rightarrow$
        $\forall\, k \in Key : state[k] \neq nil \Rightarrow infoEqual(db[k],\ state[k])$

$DBShouldSameAsMemWhenNoRestart \triangleq$
    LET
        $cond \triangleq$
            $\land\ TerminateCond$
            $\land\ num\_main\_restart = 0$
            $\land\ num\_delete\_state \ \ = 0$
    IN
        $cond \Rightarrow \forall\, k \in Key : state[k] = db[k] \land db[k].status =$ "Completed"
```

```
$StateAlwaysMatchWaitList \triangleq$
    $\forall k \in Key :$
        $wait\_list[k] \neq \{\} \Rightarrow state[k] \neq nil$

$StateAlwaysMatchSeq \triangleq$
    $\forall k \in Key :$
        $state[k] = nil \Rightarrow state\_seq[k] = 100$

$WatchKeysMatchWatchState \triangleq$
    $\forall c \in WatchClient, k \in Key :$
        $\neg(k \in watch\_keys[c]) \Rightarrow watch\_state[c][k] = nil$

$WatchListMatchSeqAndLogIndex \triangleq$
    $\forall c \in WatchClient, k \in Key :$
        $\neg(c \in wait\_list[k]) \Rightarrow$
            $\land\ watch\_info[c].seq[k] = 100$
            $\land\ watch\_info[c].log\_index[k] = 0$

$LRUKeysMatchWaitList \triangleq$
    $lru\_keys = \{k \in Key : state[k] \neq nil \land wait\_list[k] = \{\}\}$

$InfoKeysMatchSeq \triangleq$
    $\forall c \in WatchClient, k \in Key :$
        $\land\ watch\_info[c].seq[k] > 100 \Rightarrow k \in watch\_info[c].keys$
        $\land\ watch\_info[c].log\_index[k] > 0 \Rightarrow k \in watch\_info[c].keys$

$channelInitByClient(c) \triangleq$
     $\land\ watch\_info[c].chan.status = \text{"Consumed"}$
     $\land\ watch\_info[c].chan.data = nil$

$channelInit \triangleq \forall c \in WatchClient : channelInitByClient(c)$

$channelNextByClient(c) \triangleq$
    LET
        $old\_chan \triangleq watch\_info[c].chan$
        $new\_chan \triangleq watch\_info'[c].chan$

        $empty\_to\_ready \triangleq$
            $\land\ old\_chan.status = \text{"Empty"}$
            $\land\ new\_chan.status = \text{"Ready"}$
            $\land\ new\_chan.data \neq nil$

        $consumed\_to\_empty \triangleq$
            $\land\ old\_chan.status = \text{"Consumed"}$
            $\land\ new\_chan.status = \text{"Empty"}$
            $\land\ new\_chan.data = nil$

        $consumed\_to\_ready \triangleq$
            $\land\ old\_chan.status\ \ = \text{"Consumed"}$
            $\land\ new\_chan.status = \text{"Ready"}$
            $\land\ new\_chan.data \neq nil$

        $to\_consumed \triangleq$
            $\land\ \lor\ old\_chan.status = \text{"Ready"}$
               $\lor\ old\_chan.status = \text{"Empty"}$
            $\land\ new\_chan.status = \text{"Consumed"}$
            $\land\ new\_chan.data = nil$
    IN
        $\lor\ empty\_to\_ready$
        $\lor\ consumed\_to\_empty$
        $\lor\ consumed\_to\_ready$
        $\lor\ to\_consumed$
        $\lor\ new\_chan = old\_chan$  UNCHANGED

$channelNextActions \triangleq \exists\, c \in WatchClient : channelNextByClient(c)$

$ChannelSpec \triangleq$
    $channelInit \land \Box[channelNextActions]_{watch\_info}$

$Sym \triangleq Permutations(Key)$
```