MODULE *APS_Mempool*

EXTENDS *Naturals*, *FiniteSets*

Figure-D artifact: *APS* scheduler + *mempool* support definitions.

APS_Scheduler

$$
\begin{aligned}
&APSConfigType(maxBatch,\ maxPipeline,\ maxTimeout) \triangleq \\
&\quad [\\
&\qquad batchSize : 1 \,..\, maxBatch, \\
&\qquad pipelineDepth : 1 \,..\, maxPipeline, \\
&\qquad timeout : 1 \,..\, maxTimeout \\
&\quad]
\end{aligned}
$$

$$
\begin{aligned}
&ConfigSatisfiesNetwork(cfg,\ networkCondition) \triangleq \\
&\quad \text{IF } networkCondition = \text{“Unstable”} \\
&\quad \text{THEN } \land\ cfg.timeout \geq 2 \\
&\qquad\qquad \land\ cfg.batchSize \leq 2 \\
&\qquad\qquad \land\ cfg.pipelineDepth \leq 2 \\
&\quad \text{ELSE } \land\ cfg.timeout \geq 1 \\
&\qquad\qquad \land\ cfg.batchSize \geq 1 \\
&\qquad\qquad \land\ cfg.pipelineDepth \geq 1
\end{aligned}
$$

$$
\begin{aligned}
&LatencyScore(cfg,\ networkCondition) \triangleq \\
&\quad \text{IF } networkCondition = \text{“Unstable”} \\
&\quad \text{THEN } (2 * cfg.timeout) + cfg.batchSize + cfg.pipelineDepth \\
&\quad \text{ELSE } cfg.timeout + cfg.batchSize
\end{aligned}
$$

$$
ThroughputScore(cfg) \triangleq cfg.batchSize * cfg.pipelineDepth
$$

$$
\begin{aligned}
&PerformanceScore(cfg,\ networkCondition) \triangleq \\
&\quad (2 * LatencyScore(cfg,\ networkCondition)) - ThroughputScore(cfg)
\end{aligned}
$$

$$
\begin{aligned}
&ChooseBetterConfig(current,\ candidate,\ networkCondition) \triangleq \\
&\quad \text{IF } PerformanceScore(candidate,\ networkCondition) \\
&\qquad \leq PerformanceScore(current,\ networkCondition) \\
&\quad \text{THEN } candidate \\
&\quad \text{ELSE } current
\end{aligned}
$$

$$
\begin{aligned}
&RefineTimeout(timeout,\ networkCondition,\ maxTimeout) \triangleq \\
&\quad \text{IF } networkCondition = \text{“Unstable”} \\
&\quad \text{THEN IF } timeout < maxTimeout \text{ THEN } timeout + 1 \text{ ELSE } maxTimeout \\
&\quad \text{ELSE } 1
\end{aligned}
$$

$$
\begin{aligned}
&AdvanceSchedulerState(state) \triangleq \\
&\quad \text{IF } state = \text{“Monitor”} \text{ THEN “Sample”} \\
&\quad \text{ELSE IF } state = \text{“Sample”} \text{ THEN “Estimate”} \\
&\quad \text{ELSE IF } state = \text{“Estimate”} \text{ THEN “Explore”}
\end{aligned}
$$

```
    ELSE IF state = "Explore" THEN "Deploy"
    ELSE "Monitor"
```

Mempool

```
PriorityClass(tx, age, hotTx, warmTx, agingThreshold) ≜
    IF tx ∈ hotTx ∨ age[tx] ≥ agingThreshold THEN "High"
     ELSE IF tx ∈ warmTx THEN "Mid"
     ELSE "Low"

PriorityBucket(pool, age, hotTx, warmTx, agingThreshold, class) ≜
    {tx ∈ pool :
       PriorityClass(tx, age, hotTx, warmTx, agingThreshold) = class}

PriorityFront(pool, age, hotTx, warmTx, agingThreshold) ≜
    LET hi ≜
            PriorityBucket(
                pool,
                age,
                hotTx,
                warmTx,
                agingThreshold,
                "High"
            )
        mid ≜
             PriorityBucket(
                 pool,
                 age,
                 hotTx,
                 warmTx,
                 agingThreshold,
                 "Mid"
             )
        low ≜
            PriorityBucket(
                pool,
                age,
                hotTx,
                warmTx,
                agingThreshold,
                "Low"
            )
    IN  IF hi ≠ {} THEN hi ELSE IF mid ≠ {} THEN mid ELSE low

SelectBatch(pool, age, hotTx, warmTx, agingThreshold, limit) ≜
    CHOOSE chosen ∈ SUBSET pool :
```

$\land \quad chosen \neq \{\}$
$\land \quad Cardinality(chosen) \leq limit$
$\land \quad chosen \subseteq$
$\quad\quad PriorityFront(pool,\ age,\ hotTx,\ warmTx,\ agingThreshold)$

$RecoverTransactions(abstractSet,\ validatedPool) \triangleq$
$\quad abstractSet \cap validatedPool$

$AgeMapBump(age,\ tx,\ maxAge) \triangleq$
$\quad [age \text{ EXCEPT } ![tx] = \text{IF } age[tx] < maxAge \text{ THEN } age[tx] + 1 \text{ ELSE } maxAge]$

$ResetAges(age,\ txs) \triangleq$
$\quad [t \in \text{DOMAIN } age \mapsto \text{IF } t \in txs \text{ THEN } 0 \text{ ELSE } age[t]]$