MODULE *Cure*

See *ICDCS*2016: "Cure: Strong Semantics Meets High Availability and Low Latency".

EXTENDS *Naturals*, *Sequences*, *TLC*

CONSTANTS

Key, the set of keys, ranged over by $k \in Key$

Value, the set of values, ranged over by $v \in Value$

Client, the set of clients, ranged over by $c \in Client$

Partition, the set of partitions, ranged over by $p \in Partition$

Datacenter, the set of datacenters, ranged over by $d \in Datacenter$

KeySharding, the mapping from *Key* to *Partition*

ClientAttachment the mapping from *Client* to *Datacenter*

ASSUME

$\land\ KeySharding \in [Key \to Partition]$

$\land\ ClientAttachment \in [Client \to Datacenter]$

VARIABLES

At the client side:

cvc, $cvc[c]$: the vector clock of client $c \in Client$

At the server side (each for partition $p \in Partition$ in $d \in Datacenter$):

clock, $clock[p][d]$: the current clock

pvc, $pvc[p][d]$: the vector clock

css, $css[p][d]$: the stable snapshot

PMC, $PMC[p][d]$: matrix clock

store, $store[p][d]$: the *kv* store

updates, $updates[p][d]$: the buffer of updates

Client-server communication

msgs the set of messages in transit

$cVars \triangleq \langle cvc \rangle$

$sVars \triangleq \langle clock, pvc, css, PMC, store, updates \rangle$

$mVars \triangleq \langle msgs \rangle$

$vars \triangleq \langle cvc, clock, pvc, css, PMC, store, updates, msgs \rangle$

$Clock \triangleq Nat$

$VC \triangleq [Datacenter \to Clock]$ vector clock with an entry per datacenter $d \in Datacenter$

$VCInit \triangleq [d \in Datacenter \mapsto 0]$

$KVTuple \triangleq [key : Key,\ val : Value,\ vc : VC]$

$$
\begin{aligned}
Message \triangleq\ & [type : \{\text{"ReadRequest"}\},\ key : Key,\ vc : VC,\ c : Client,\ p : Partition,\ d : Datacenter] \\
\cup\ & [type : \{\text{"ReadReply"}\},\ val : Value,\ vc\ : VC,\ c : Client] \\
\cup\ & [type : \{\text{"UpdateRequest"}\},\ key : Key,\ val : Value,\ vc : VC,\ c : Client,\ p : Partition,\ d : Datacenter] \\
\cup\ & [type : \{\text{"UpdateReply"}\},\ ts : Clock,\ c : Client,\ d : Datacenter]
\end{aligned}
$$

```
TypeOK ≜
    ∧  cvc ∈ [Client → VC]
    ∧  clock ∈ [Partition → [Datacenter → Clock]]
    ∧  pvc   ∈ [Partition → [Datacenter → VC]]
    ∧  css   ∈ [Partition → [Datacenter → VC]]
    ∧  PMC ∈ [Partition → [Datacenter → [Partition → VC]]]
    ∧  store ∈ [Partition → [Datacenter → SUBSET KVTuple]]
    ∧  updates ∈ [Partition → [Datacenter → Seq(KVTuple)]]
    ∧  msgs ⊆ Message
------------------------------------------------------------------------
Init ≜
    ∧ cvc = [c ∈ Client ↦ VCInit]
    ∧ clock = [p ∈ Partition ↦ [d ∈ Datacenter ↦ 0]]
    ∧ pvc   = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
    ∧ css   = [p ∈ Partition ↦ [d ∈ Datacenter ↦ VCInit]]
    ∧ PMC = [p ∈ Partition ↦ [d ∈ Datacenter ↦ [q ∈ Partition ↦ VCInit]]]
    ∧ store = [p ∈ Partition ↦ [d ∈ Datacenter ↦ {}]]
    ∧ updates = [p ∈ Partition ↦ [d ∈ Datacenter ↦ ⟨⟩]]
    ∧ msgs = {}
------------------------------------------------------------------------
Max(a, b) ≜ IF a < b THEN b ELSE a

Send(m) ≜ msgs' = msgs ∪ {m}

Ready2Issue(c) ≜ ∀ m ∈ msgs :
    m.type ∈ {"ReadRequest", "ReadReply", "UpdateRequest", "UpdateReply"} ⇒ m.c ≠ c
------------------------------------------------------------------------
Client operations at client c ∈ Client.

Read(c, k) ≜  c ∈ Client reads from k ∈ Key
    ∧ Ready2Issue(c)
    ∧ Send([type ↦ "ReadRequest", key ↦ k, vc ↦ cvc[c],
            c ↦ c, p ↦ KeySharding[k], d ↦ ClientAttachment[c]])
    ∧ UNCHANGED ⟨cVars, sVars⟩

ReadReply(c) ≜  c ∈ Client handles the reply to its read request
    ∧ ∃ m ∈ msgs :
        ∧ m.type = "ReadReply" ∧ m.c = c   such m is unique
        ∧ cvc' = [d ∈ Datacenter ↦ Max(m.vc[d], cvc[d])]
    ∧ UNCHANGED ⟨sVars, mVars⟩

Update(c, k, v) ≜  c ∈ Client updates k ∈ Key with v ∈ Value
    ∧ Ready2Issue(c)
    ∧ Send([type ↦ "UpdateRequest", key ↦ k, val ↦ v,
            vc ↦ cvc[c], c ↦ c, p ↦ KeySharding[k], d ↦ ClientAttachment[c]])
    ∧ UNCHANGED ⟨cVars, sVars⟩
```

```
97  UpdateReply(c) ≜   c ∈ Client handles the reply to its update request
98      ∧ ∃ m ∈ msgs :
99          ∧ m.type = "UpdateReply" ∧ m.c = c   such m is unique
100         ∧ cvc′ = [cvc EXCEPT ![c][m.d] = m.ts]
101     ∧ UNCHANGED ⟨sVars, mVars⟩
102 ├──────────────────────────────────────────────────────────────────────┤
103  Server operations at partition p ∈ Partition in datacenter d ∈ Datacenter.
105 UpdateRequest(p, d) ≜
106     ∧ ∃ m ∈ msgs :
107         ∧ m.type = "UpdateRequest" ∧ m.p = p ∧ m.d = d   such m may be not unique
108         ∧ m.vc[d] ≤ clock[p][d]   waiting condition
109         ∧ pvc′ = [pvc EXCEPT ![p][d][d] = clock[p][d]]
110         ∧ css′ = [css EXCEPT ![p][d] =
111             [dc ∈ Datacenter ↦ IF dc = d THEN @[dc] ELSE Max(m.vc[dc], @[dc])]]
112         ∧ LET kv ≜ [key ↦ m.key, val ↦ m.val,
113                     vc ↦ [m.vc EXCEPT ![d] = clock[p][d]]]
114           IN  ∧ store′ = [store EXCEPT ![p][d] = @ ∪ {kv}]
115               ∧ updates′ = [updates EXCEPT ![p][d] = @ ∘ ⟨kv⟩]
116               ∧ Send([type ↦ "UpdateReply", ts ↦ clock[p][d], c ↦ m.c, d ↦ d])
117     ∧ UNCHANGED ⟨cVars, clock, PMC⟩
118 ├──────────────────────────────────────────────────────────────────────┤
119 Next ≜
120     ∨ ∃ c ∈ Client, k ∈ Key : Read(c, k)
121     ∨ ∃ c ∈ Client, k ∈ Key, v ∈ Value : Update(c, k, v)
122     ∨ ∃ c ∈ Client : ReadReply(c) ∨ UpdateReply(c)
123     ∨ ∃ p ∈ Partition, d ∈ Datacenter : UpdateRequest(p, d)
125 Spec ≜ Init ∧ □[Next]_vars
126 ├──────────────────────────────────────────────────────────────────────┤
127 └──────────────────────────────────────────────────────────────────────┘
```