

```

1  |----- MODULE Cure -----|
   | See ICDCS2016: “Cure: Strong Semantics Meets High Availability and Low Latency”. |
5  | EXTENDS Naturals, FiniteSets, TLC, SequenceUtils, RelationUtils, MathUtils |
6  |-----|
7  CONSTANTS
8      Key,           the set of keys, ranged over by  $k \in Key$ 
9      Value,         the set of values, ranged over by  $v \in Value$ 
10     Client,         the set of clients, ranged over by  $c \in Client$ 
11     Partition,      the set of partitions, ranged over by  $p \in Partition$ 
12     Datacenter,     the set of datacenters, ranged over by  $d \in Datacenter$ 
13     KeySharding,     the mapping from Key to Partition
14     ClientAttachment the mapping from Client to Datacenter

16 NotVal  $\triangleq$  CHOOSE  $v : v \notin Value$ 

18 ASSUME
19      $\wedge KeySharding \in [Key \rightarrow Partition]$ 
20      $\wedge ClientAttachment \in [Client \rightarrow Datacenter]$ 
21 |-----|
22 VARIABLES
23     At the client side:
24     cvc, cvc[ $c$ ]: the vector clock of client  $c \in Client$ 
25     At the server side (each for partition  $p \in Partition$  in  $d \in Datacenter$ ):
26     clock, clock[ $p$ ][ $d$ ]: the current clock
27     pvc, pvc[ $p$ ][ $d$ ]: the vector clock
28     css, css[ $p$ ][ $d$ ]: the stable snapshot
29     store, store[ $p$ ][ $d$ ]: the kv store
30     history:
31     L, L[ $c$ ]: local history at client  $c \in Client$ 
32     communication:
33     msgs, the set of messages in transit
34     incoming incoming[ $p$ ][ $d$ ]: incoming FIFO channel for propagating updates and heartbeats

36 cVars  $\triangleq$   $\langle cvc \rangle$ 
37 sVars  $\triangleq$   $\langle clock, pvc, css, store, L \rangle$ 
38 mVars  $\triangleq$   $\langle msgs, incoming \rangle$ 
39 vars  $\triangleq$   $\langle cvc, clock, pvc, css, store, L, msgs, incoming \rangle$ 
40 |-----|
41 VC  $\triangleq$   $[Datacenter \rightarrow Nat]$  vector clock with an entry per datacenter  $d \in Datacenter$ 
42 VCInit  $\triangleq$   $[d \in Datacenter \mapsto 0]$ 
43 Merge(vc1, vc2)  $\triangleq$   $[d \in Datacenter \mapsto Max(vc1[d], vc2[d])]$ 

45 DC  $\triangleq$  Cardinality(Datacenter)
46 DCIndex  $\triangleq$  CHOOSE  $f \in [1 .. DC \rightarrow Datacenter] : Injective(f)$ 
47 LTE(vc1, vc2)  $\triangleq$  less-than-or-equal-to comparator for vector clocks
48 LET RECURSIVE LTEHelper( $-, -, -$ )

```



95  $m.type \in \{\text{"ReadRequest"}, \text{"ReadReply"}, \text{"UpdateRequest"}, \text{"UpdateReply"}\} \Rightarrow m.c \neq c$   
 97  $Read(c, k) \triangleq$   $c \in Client$  reads from  $k \in Key$   
 98  $\wedge CanIssue(c)$   
 99  $\wedge Send([type \mapsto \text{"ReadRequest"}, key \mapsto k, vc \mapsto cvc[c],$   
 100  $c \mapsto c, p \mapsto KeySharding[k], d \mapsto ClientAttachment[c]])$   
 101  $\wedge UNCHANGED \langle cVars, sVars, incoming \rangle$   
 103  $ReadReply(c) \triangleq$   $c \in Client$  handles the reply to its read request  
 104  $\wedge \exists m \in msgs :$   
 105  $\wedge m.type = \text{"ReadReply"} \wedge m.c = c$  such  $m$  is unique due to well-formedness  
 106  $\wedge cvc' = [cvc \text{ EXCEPT } ![c] = Merge(m.vc, @)]$   
 107  $\wedge msgs' = msgs \setminus \{m\}$   
 108  $\wedge UNCHANGED \langle sVars, incoming \rangle$   
 110  $Update(c, k, v) \triangleq$   $c \in Client$  updates  $k \in Key$  with  $v \in Value$   
 111  $\wedge CanIssue(c)$   
 112  $\wedge Send([type \mapsto \text{"UpdateRequest"}, key \mapsto k, val \mapsto v,$   
 113  $vc \mapsto cvc[c], c \mapsto c, p \mapsto KeySharding[k], d \mapsto ClientAttachment[c]])$   
 114  $\wedge UNCHANGED \langle cVars, sVars, incoming \rangle$   
 116  $UpdateReply(c) \triangleq$   $c \in Client$  handles the reply to its update request  
 117  $\wedge \exists m \in msgs :$   
 118  $\wedge m.type = \text{"UpdateReply"} \wedge m.c = c$  such  $m$  is unique due to well-formedness  
 119  $\wedge cvc' = [cvc \text{ EXCEPT } ![c][m.d] = m.ts]$   
 120  $\wedge msgs' = msgs \setminus \{m\}$   
 121  $\wedge UNCHANGED \langle sVars, incoming \rangle$   
 122 

---

  
 123 Server operations at partition  $p \in Partition$  in datacenter  $d \in Datacenter$ .  
 125  $ReadRequest(p, d) \triangleq$  handle a "ReadRequest"  
 126  $\wedge \exists m \in msgs :$   
 127  $\wedge m.type = \text{"ReadRequest"} \wedge m.p = p \wedge m.d = d$   
 128  $\wedge css' = [css \text{ EXCEPT } ![p][d] = Merge(m.vc, @)]$   
 129  $\wedge LET \ kvs \triangleq \{kv \in store[p][d] :$   
 130  $\wedge kv.key = m.key$   
 131  $\wedge \forall dc \in Datacenter \setminus \{d\} : kv.vc[dc] \leq css'[p][d][dc]\}$   
 132  $\quad lkv \triangleq \text{CHOOSE } kv \in kvs : \forall akv \in kvs : LTE(akv.vc, kv.vc)$   
 133  $\quad IN \wedge SendAndDelete([type \mapsto \text{"ReadReply"}, val \mapsto lkv.val, vc \mapsto lkv.vc, c \mapsto m.c], m)$   
 134  $\wedge L' = [L \text{ EXCEPT } ![m.c] = Append(@, [type \mapsto \text{"R"}, kv \mapsto lkv, c \mapsto m.c, cnt \mapsto Len(@) + 1])]$   
 135  $\wedge UNCHANGED \langle cVars, clock, pvc, store, incoming \rangle$   
 137  $UpdateRequest(p, d) \triangleq$  handle a "UpdateRequest"  
 138  $\wedge \exists m \in msgs :$   
 139  $\wedge m.type = \text{"UpdateRequest"} \wedge m.p = p \wedge m.d = d$   
 140  $\wedge m.vc[d] < clock[p][d]$  waiting condition; (" $\leq$ " strengthened to " $<$ ")  
 141  $\wedge css' = [css \text{ EXCEPT } ![p][d] = Merge(m.vc, @)]$

```

142       $\wedge$  LET  $kv \triangleq [key \mapsto m.key, val \mapsto m.val,$ 
143           $vc \mapsto [m.vc \text{ EXCEPT } ![d] = clock[p][d]]]$ 
144      IN  $\wedge store' = [store \text{ EXCEPT } ![p][d] = @ \cup \{kv\}]$ 
145           $\wedge SendAndDelete([type \mapsto \text{"UpdateReply"}, ts \mapsto clock[p][d], c \mapsto m.c, d \mapsto d], m)$ 
146           $\wedge incoming' = [incoming \text{ EXCEPT } ![p] = [dc \in Datacenter \mapsto$ 
147              IF  $dc = d$  THEN  $@[dc]$  ELSE  $Append(@[dc], [type \mapsto \text{"Replicate"}, d \mapsto d, kv \mapsto kv])]$ 
148               $\wedge L' = [L \text{ EXCEPT } ![m.c] = Append(@, [type \mapsto \text{"W"}, kv \mapsto kv, c \mapsto m.c, cnt \mapsto Len(@) + 1])]$ 
149           $\wedge$  UNCHANGED  $\langle cVars, clock, pvc \rangle$ 

151 Replicate( $p, d$ )  $\triangleq$  handle a "Replicate"
152      $\wedge incoming[p][d] \neq \langle \rangle$ 
153      $\wedge$  LET  $m \triangleq Head(incoming[p][d])$ 
154     IN  $\wedge m.type = \text{"Replicate"}$ 
155          $\wedge store' = [store \text{ EXCEPT } ![p][d] = @ \cup \{m.kv\}]$ 
156          $\wedge pvc' = [pvc \text{ EXCEPT } ![p][d][m.d] = m.kv.vc[m.d]]$ 
157          $\wedge incoming' = [incoming \text{ EXCEPT } ![p][d] = Tail(@)]$ 
158      $\wedge$  UNCHANGED  $\langle cVars, cvc, clock, css, L, msgs \rangle$ 

160 Heartbeat( $p, d$ )  $\triangleq$  handle a "Heartbeat"
161      $\wedge incoming[p][d] \neq \langle \rangle$ 
162      $\wedge$  LET  $m \triangleq Head(incoming[p][d])$ 
163     IN  $\wedge m.type = \text{"Heartbeat"}$ 
164          $\wedge pvc' = [pvc \text{ EXCEPT } ![p][d][m.d] = m.ts]$ 
165          $\wedge incoming' = [incoming \text{ EXCEPT } ![p][d] = Tail(@)]$ 
166      $\wedge$  UNCHANGED  $\langle cVars, cvc, clock, css, store, L, msgs \rangle$ 

167 |-----|
168 Clock management at partition  $p \in Partition$  in datacenter  $d \in Datacenter$ 
169 Tick( $p, d$ )  $\triangleq$   $clock[p][d]$  ticks
170      $\wedge clock' = [clock \text{ EXCEPT } ![p][d] = @ + 1]$ 
171      $\wedge pvc' = [pvc \text{ EXCEPT } ![p][d][d] = clock'[p][d]]$ 
172      $\wedge incoming' = [incoming \text{ EXCEPT } ![p] = [dc \in Datacenter \mapsto$ 
173         IF  $dc = d$  THEN  $@[dc]$  ELSE  $Append(@[dc], [type \mapsto \text{"Heartbeat"}, d \mapsto d, ts \mapsto pvc'[p][d][d]])]$ 
174      $\wedge$  UNCHANGED  $\langle cVars, cvc, css, store, L, msgs \rangle$ 

176 UpdateCSS( $p, d$ )  $\triangleq$  update  $css[p][d]$ 
177      $\wedge css' = [css \text{ EXCEPT } ![p][d] =$ 
178          $[dc \in Datacenter \mapsto SetMin(\{pvc[pp][d][dc] : pp \in Partition\})]$ 
179      $\wedge$  UNCHANGED  $\langle cVars, mVars, clock, pvc, store, L \rangle$ 

180 |-----|
181 Next  $\triangleq$ 
182      $\vee \exists c \in Client, k \in Key : Read(c, k)$ 
183      $\vee \exists c \in Client, k \in Key, v \in Value : Update(c, k, v)$ 
184      $\vee \exists c \in Client : ReadReply(c) \vee UpdateReply(c)$ 
185      $\vee \exists p \in Partition, d \in Datacenter :$ 
186          $\vee ReadRequest(p, d)$ 
187          $\vee UpdateRequest(p, d)$ 

```

---

```

188       $\vee \text{Replicate}(p, d)$ 
189       $\vee \text{Heartbeat}(p, d)$ 
190       $\vee \text{Tick}(p, d)$ 
191       $\vee \text{UpdateCSS}(p, d)$ 

193    $\text{Spec} \triangleq \text{Init} \wedge \Box[\text{Next}]_{\text{vars}}$ 
194 |-----|
195    $\text{Valid}(s) \triangleq$  Is s a valid serialization?
196     LET RECURSIVE  $\text{ValidHelper}(-, -)$ 
197        $\text{ValidHelper}(\text{seq}, \text{kvs}) \triangleq$ 
198         IF  $\text{seq} = \langle \rangle$  THEN TRUE
199         ELSE LET  $op \triangleq \text{Head}(\text{seq})$ 
200           IN IF  $op.\text{type} = \text{"W"}$  overwritten
201             THEN  $\text{ValidHelper}(\text{Tail}(\text{seq}), op.kv.key :> op.kv.vc @@ \text{kvs})$ 
202             ELSE  $\wedge op.kv.vc = \text{kvs}[op.kv.key]$ 
203                 $\wedge \text{ValidHelper}(\text{Tail}(\text{seq}), \text{kvs})$ 
204     IN  $\text{ValidHelper}(s, [k \in \text{Key} \mapsto \text{VCInit}])$  with initial values

206    $\text{CM} \triangleq$  causal memory consistency model; see Ahamad@DC'1995
207     LET  $ops \triangleq \text{UNION } \{\text{Range}(L[c] : c \in \text{Client})\}$ 
208         $rops \triangleq \{op \in ops : op.type = \text{"R"}\}$ 
209         $wops \triangleq \{op \in ops : op.type = \text{"W"}\}$ 
210         $so \triangleq \text{UNION } \{\text{SeqToRel}(L[c] : c \in \text{Client})\}$  session order
211         $rf \triangleq \{\langle w, r \rangle \in wops \times rops : w.kv.key = r.kv.key \wedge w.kv.vc = r.kv.vc\}$ 
212         $co \triangleq \text{TC}(so \cup rf)$  causality order
213     IN  $\forall c \in \text{Client} :$ 
214         $\exists sc \in \text{PermutationsOf}(L[c] \circ \text{SetToSeq}(wops)) :$ 
215           $\wedge \text{Valid}(sc)$ 
216           $\wedge \text{Respect}(sc, co)$ 

218 THEOREM  $\text{Spec} \Rightarrow \Box \text{CM}$ 
219 |-----|

```