MODULE *Cure*

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

EXTENDS *Naturals*, *FiniteSets*, *TLC*, *SequenceUtils*, *RelationUtils*, *MathUtils*

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*

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

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

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

history:

L, *L*[*c*]: local history at client $c \in Client$

communication:

msgs, the set of messages in transit

incoming *incoming*[*p*][*d*]: incoming *FIFO* channel for propagating updates and heartbeats

$cVars \triangleq \langle cvc \rangle$

$sVars \triangleq \langle clock,\ pvc,\ css,\ store,\ L \rangle$

$mVars \triangleq \langle msgs,\ incoming \rangle$

$vars \triangleq \langle cvc,\ clock,\ pvc,\ css,\ store,\ L,\ msgs,\ incoming \rangle$

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

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

$Merge(vc1,\ vc2) \triangleq [d \in Datacenter \mapsto Max(vc1[d],\ vc2[d])]$

$DC \triangleq Cardinality(Datacenter)$

$DCIndex \triangleq$ CHOOSE $f \in [1\ ..\ DC \to Datacenter] : Injective(f)$

$LTE(vc1,\ vc2) \triangleq$ less-than-or-equal-to comparator for vector clocks

LET RECURSIVE $LTEHelper(\_,\ \_,\ \_)$

$$
\begin{array}{l}
\quad LTEHelper(vc1h,\ vc2h,\ index) \triangleq \\
\qquad \text{IF } index > DC \text{ THEN TRUE } EQ \\
\qquad \text{ELSE LET } d \triangleq DCIndex[index] \\
\qquad\qquad \text{IN CASE } vc1h[d] < vc2h[d] \to \text{TRUE} \quad LT \\
\qquad\qquad\qquad \square \quad vc1h[d] > vc2h[d] \to \text{FALSE} \quad GT \\
\qquad\qquad\qquad \square \quad \text{OTHER} \to LTEHelper(vc1h,\ vc2h,\ index + 1) \\
\text{IN} \quad LTEHelper(vc1,\ vc2,\ 1)
\end{array}
$$

$$
\begin{array}{l}
KVTuple \triangleq [key : Key,\ val : Value \cup \{NotVal\},\ vc : VC] \\
OpTuple \triangleq [type : \{\text{“R”},\ \text{“W”}\},\ kv : KVTuple,\ c : Client,\ cnt : Nat]
\end{array}
$$

$$
\begin{array}{ll}
Message \triangleq & \\
 & [type : \{\text{“ReadRequest”}\},\ key : Key,\ vc : VC,\ c : Client,\ p : Partition,\ d : Datacenter] \\
\cup & [type : \{\text{“ReadReply”}\},\ val : Value \cup \{NotVal\},\ vc : VC,\ c : Client] \\
\cup & [type : \{\text{“UpdateRequest”}\},\ key : Key,\ val : Value,\ vc : VC,\ c : Client,\ p : Partition,\ d : Datacent \\
\cup & [type : \{\text{“UpdateReply”}\},\ ts : Nat,\ c : Client,\ d : Datacenter] \\
\cup & [type : \{\text{“Replicate”}\},\ d : Datacenter,\ kv : KVTuple] \\
\cup & [type : \{\text{“Heartbeat”}\},\ d : Datacenter,\ ts : Nat]
\end{array}
$$

$$
\begin{array}{l}
Send(m) \triangleq msgs' = msgs \cup \{m\} \\
SendAndDelete(sm,\ dm) \triangleq msgs' = (msgs \cup \{sm\}) \setminus \{dm\}
\end{array}
$$

$$
\begin{array}{ll}
TypeOK \triangleq & \\
\quad \wedge & cvc \in [Client \to VC] \\
\quad \wedge & clock \in [Partition \to [Datacenter \to Nat]] \\
\quad \wedge & pvc \in [Partition \to [Datacenter \to VC]] \\
\quad \wedge & css \in [Partition \to [Datacenter \to VC]] \\
\quad \wedge & store \in [Partition \to [Datacenter \to \text{SUBSET } KVTuple]] \\
\quad \wedge & msgs \subseteq Message \\
\quad \wedge & incoming \in [Partition \to [Datacenter \to Seq(Message)]] \\
\quad \wedge & L \in [Client \to Seq(OpTuple)]
\end{array}
$$

$$
\begin{array}{l}
Init \triangleq \\
\quad \wedge cvc = [c \in Client \mapsto VCInit] \\
\quad \wedge clock = [p \in Partition \mapsto [d \in Datacenter \mapsto 0]] \\
\quad \wedge pvc = [p \in Partition \mapsto [d \in Datacenter \mapsto VCInit]] \\
\quad \wedge css = [p \in Partition \mapsto [d \in Datacenter \mapsto VCInit]] \\
\quad \wedge store = [p \in Partition \mapsto [d \in Datacenter \mapsto \\
\qquad\qquad [key : \{k \in Key : KeySharding[k] = p\},\ val : \{NotVal\},\ vc : \{VCInit\}]]] \\
\quad \wedge msgs = \{\} \\
\quad \wedge incoming = [p \in Partition \mapsto [d \in Datacenter \mapsto \langle\rangle]] \\
\quad \wedge L = [c \in Client \mapsto \langle\rangle]
\end{array}
$$

Client operations at client $c \in Client$.

$CanIssue(c) \triangleq \forall\, m \in msgs :$ to ensure well-formedness of clients

$m.type \in \{\text{“ReadRequest”}, \text{“ReadReply”}, \text{“UpdateRequest”}, \text{“UpdateReply”}\} \Rightarrow m.c \neq c$

$Read(c, k) \triangleq$ $c \in Client$ reads from $k \in Key$
$\land\ CanIssue(c)$
$\land\ Send([type \mapsto \text{“ReadRequest”}, key \mapsto k, vc \mapsto cvc[c],$
$c \mapsto c, p \mapsto KeySharding[k], d \mapsto ClientAttachment[c]])$
$\land$ UNCHANGED $\langle cVars, sVars, incoming\rangle$

$ReadReply(c) \triangleq$ $c \in Client$ handles the reply to its read request
$\land \exists\, m \in msgs :$
$\land\ m.type = \text{“ReadReply”} \land m.c = c$ such m is unique due to well-formedness
$\land\ cvc' = [cvc$ EXCEPT $![c] = Merge(m.vc, @)]$
$\land\ msgs' = msgs \setminus \{m\}$
$\land$ UNCHANGED $\langle sVars, incoming\rangle$

$Update(c, k, v) \triangleq$ $c \in Client$ updates $k \in Key$ with $v \in Value$
$\land\ CanIssue(c)$
$\land\ Send([type \mapsto \text{“UpdateRequest”}, key \mapsto k, val \mapsto v,$
$vc \mapsto cvc[c], c \mapsto c, p \mapsto KeySharding[k], d \mapsto ClientAttachment[c]])$
$\land$ UNCHANGED $\langle cVars, sVars, incoming\rangle$

$UpdateReply(c) \triangleq$ $c \in Client$ handles the reply to its update request
$\land \exists\, m \in msgs :$
$\land\ m.type = \text{“UpdateReply”} \land m.c = c$ such m is unique due to well-formedness
$\land\ cvc' = [cvc$ EXCEPT $![c][m.d] = m.ts]$
$\land\ msgs' = msgs \setminus \{m\}$
$\land$ UNCHANGED $\langle sVars, incoming\rangle$

Server operations at partition $p \in Partition$ in datacenter $d \in Datacenter$.

$ReadRequest(p, d) \triangleq$ handle a “ReadRequest”
$\land \exists\, m \in msgs :$
$\land\ m.type = \text{“ReadRequest”} \land m.p = p \land m.d = d$
$\land\ css' = [css$ EXCEPT $![p][d] = Merge(m.vc, @)]$
$\land$ LET $kvs \triangleq \{kv \in store[p][d] :$
$\land\ kv.key = m.key$
$\land \forall\, dc \in Datacenter \setminus \{d\} : kv.vc[dc] \leq css'[p][d][dc]\}$
$lkv \triangleq$ CHOOSE $kv \in kvs : \forall\, akv \in kvs : LTE(akv.vc, kv.vc)$
IN $\land\ SendAndDelete([type \mapsto \text{“ReadReply”}, val \mapsto lkv.val, vc \mapsto lkv.vc, c \mapsto m.c], m)$
$\land\ L' = [L$ EXCEPT $![m.c] = Append(@, [type \mapsto \text{“R”}, kv \mapsto lkv, c \mapsto m.c, cnt \mapsto Len(@) + 1$
$\land$ UNCHANGED $\langle cVars, clock, pvc, store, incoming\rangle$

$UpdateRequest(p, d) \triangleq$ handle a “UpdateRequest”
$\land \exists\, m \in msgs :$
$\land\ m.type = \text{“UpdateRequest”} \land m.p = p \land m.d = d$
$\land\ m.vc[d] < clock[p][d]$ waiting condition; (“$\leq$” strengthed to “$<$”)
$\land\ css' = [css$ EXCEPT $![p][d] = Merge(m.vc, @)]$

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

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

$Heartbeat(p,\ d) \triangleq$ handle a "Heartbeat"
$\land\ incoming[p][d] \neq \langle\rangle$
$\land$ LET $m \triangleq Head(incoming[p][d])$
IN $\land\ m.type =$ "Heartbeat"
$\land\ pvc' = [pvc$ EXCEPT $![p][d][m.d] = m.ts]$
$\land\ incoming' = [incoming$ EXCEPT $![p][d] = Tail(@)]$
$\land$ UNCHANGED $\langle cVars,\ cvc,\ clock,\ css,\ store,\ L,\ msgs\rangle$

Clock management at partition $p \in Partition$ in datacenter $d \in Datacenter$
$Tick(p,\ d) \triangleq$ $clock[p][d]$ ticks
$\land\ clock' = [clock$ EXCEPT $![p][d] = @ + 1]$
$\land\ pvc' = [pvc$ EXCEPT $![p][d][d] = clock'[p][d]]$
$\land\ incoming' = [incoming$ EXCEPT $![p] = [dc \in Datacenter \mapsto$
IF $dc = d$ THEN $@[dc]$ ELSE $Append(@[dc], [type \mapsto$ "Heartbeat", $d \mapsto d,\ ts \mapsto pvc'[p][d][d]])]]$
$\land$ UNCHANGED $\langle cVars,\ cvc,\ css,\ store,\ L,\ msgs\rangle$

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

$Next \triangleq$
$\lor \exists\, c \in Client,\ k \in Key : Read(c,\ k)$
$\lor \exists\, c \in Client,\ k \in Key,\ v \in Value : Update(c,\ k,\ v)$
$\lor \exists\, c \in Client : ReadReply(c) \lor UpdateReply(c)$
$\lor \exists\, p \in Partition,\ d \in Datacenter :$
$\lor\ ReadRequest(p,\ d)$
$\lor\ UpdateRequest(p,\ d)$

$\lor\ Replicate(p,\ d)$
$\lor\ Heartbeat(p,\ d)$
$\lor\ Tick(p,\ d)$
$\lor\ UpdateCSS(p,\ d)$

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

$Valid(s) \triangleq$ Is s a valid serialization?
LET RECURSIVE $ValidHelper(\_,\ \_)$
$ValidHelper(seq,\ kvs) \triangleq$
IF $seq = \langle\rangle$ THEN TRUE
ELSE LET $op \triangleq Head(seq)$
IN IF $op.type =$ "W" overwritten
THEN $ValidHelper(Tail(seq),\ op.kv.key :> op.kv.vc\ @@\ kvs)$
ELSE $\land\ op.kv.vc = kvs[op.kv.key]$
$\land\ ValidHelper(Tail(seq),\ kvs)$
IN $ValidHelper(s,\ [k \in Key \mapsto VCInit])$ with initial values

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

THEOREM $Spec \Rightarrow \Box CM$
