

CSE 4/586: Project 1

Murat Demirbas

[2016-10-27 Thu]

1 Replicated Storage with Client-Side Routing

There are $N=5$ storage nodes and a client that is writing to the storage nodes. At most $FAILNUM$ number of storage nodes can fail, and failed nodes can recover anytime with their storage intact. In order to ensure persistence, the client writes to a subset of the up nodes, denoted as the write quorum ($WriteQ$). In order to ensure consistency of the writes, before performing the write the client reads from a subset of the up nodes, denoted as the read quorum ($ReadQ$), and learns the highest versioned write completed, increments the version number, and performs the write to the $WriteQ$ nodes.

This protocol corresponds to DynamoDB and Voldemort key-value storage protocols with some simplifications.

For simplicity we assume the client keeps writing to only one item, so we ignore modeling of the key part of the key-value pair item, and hashing of the key to the storage system to figure out which nodes form the $WriteQ$ and $ReadQ$. Our $WriteQ$ and $ReadQ$ selection will consist of the lowest id storage nodes that are up (currently not failed).

1.1 Write a PlusCal program to represent this algorithm.

Use the below template as your starting point, and fill in the redacted parts. Use the toolkit to translate your code to TLA+ and model-check for correctness.

Replicated storage protocol with clientside routing

EXTENDS *Integers, Sequences, FiniteSets, TLC*

CONSTANTS $N, C, STOP, ReadQ, WriteQ, FAILNUM$

ASSUME $N = 5 \wedge C = 1 \wedge STOP < 10 \wedge 1 \leq ReadQ \wedge ReadQ \leq 3$

$\wedge 1 \leq WriteQ \wedge WriteQ \leq 3 \wedge 0 \leq FAILNUM \wedge FAILNUM \leq 2$

$Nodes \triangleq 1 \dots N$

$Clients \triangleq N + 1 \dots N + C$ *** should give different *ID* space to Client

--algorithm *voldemort*

```
{
  variable FailNum = FAILNUM,
    up = [n ∈ Nodes ↦ TRUE], \*Initially all nodes are up
    db = [n ∈ Nodes ↦ {[ver ↦ 0, val ↦ 0]}];
  \* All nodes have database, wherein [ver = 0, val = 0] stored for the item

  define
  {
    UpNodes  $\triangleq$  {
    ReturnReadQ  $\triangleq$ 
    ReturnWriteQ  $\triangleq$  CHOOSE  $i \in \text{SUBSET}(UpNodes) : \text{Cardinality}(i) = WriteQ$ 
      \* CHOOSE deterministically returns lowest ID nodes that satisfy the requirement
    }

  fair process ( c ∈ Clients )
  variable cntr = 0, hver = 0, Q = {};
  {
    CL: while ( cntr ≤ STOP ) {
      cntr := cntr + 1;
      \* get the highest version number from read Quorum
      \* write val = cntr to writeQuorum with higher version number
    }
  }
}
```

```

fair process ( n ∈ Nodes )
{
  NODE: while ( TRUE ) {
    if ( FailNum > 0 ∧ up[self] = TRUE ) { \* Storage node can fail
      [redacted]
    }
    else if ( up[self] = FALSE ) { \* Or recover
      [redacted]
    }
  }
}

```

1.2 Model-check safety properties with TLA+

- Write an invariant to capture the single-copy consistency property of the storage protocol. Single-copy consistency means the $N=5$ storage nodes appear to outside as if it is a single virtual unfailable node, even though upto $FAILNUM$ of physical storage nodes can fail. More specifically, the highest version number result returned by a read from $ReadQ$ of storage nodes should match the item stored by the most recent write operation on the system.
 - We have a single client of the storage system, and for the sake of simplicity it only implements writes to the system. But you can still check for single-copy consistency is satisfied without having to implement the read operation. You can also use a simplified/shortcut check for the single-copy consistency leveraging the way the client stores items to the storage system.
 - Single-copy consistency property should be satisfied when $FAILNUM=0$, i.e., when no node is allowed to fail. The property will fail to be satisfied for certain combinations $ReadQ$ & $WriteQ$ when $FAILNUM>0$.
- Write in the comments section, after the "=====" line, your findings/observations about the relation between $ReadQ$, $WriteQ$, and $FAILNUM$ that ensures the protocol satisfies the single-copy consistency property.

2 Submission

Your TLA+ file should be named *voldemort.tla* . Your model's name should be the default name *Model_1* (do not name your model file differently).

Generate a pdf print of your TLA+ program using the "Produce Pdf version" from the TLA+ menu. (This will get included in your submission as it is created under the ".toolbox" directory.)

Now create a zip file from the ".tla" file and the corresponding ".toolbox" directory. **Name the zipfile as: proj1.zip**

Not following these directions will cause you to lose points.

You will use the submit command (*submit_cse486* or *submit_cse586* respectively) to submit your work. The submit command instructions are here: <https://wiki.cse.buffalo.edu/services/content/submit-script>