HW2 Solutions: CS425 FA16

1. (Solution and Grading by: Xiaocheng Yuan)
   The program terminates on 2022.
   2016 hashes to bit 0
   2017 hashes to bit 2
   2018 hashes to bits 4 6 10 18
   2019 hashes to bits 6 12 20 30
   2020 hashes to bits 4 8 20
   2021 hashes to bits 2 10 22 30
   2022 hashes to bits 10 12 22 30

2. (Solution and Grading by: Si Liu)
   a. Last writer wins:
      (i) less storage - a new value overwrites the value stored
      (ii) easier update logic - only timestamp-based, no need to use causal context
      (iii) better performance - in the cases where keys are rewritten very often and quickly,
          usually by one client, and new values do not necessarily depend on the old values
   Vector clocks:
      (i) preventing data loss - allowing concurrent updates returned to a client and letting the
          client resolve the conflict
      (ii) good for reasoning about siblings - when a use case requires to reason about
          differing values produced by concurrent updates, e.g., whether one value is a direct descendant
          of the other
      (iii) no dependence on clock synchronization - each entity simply uses its own vector
          clock
   b. Both answers are accepted. This TA prefers vector clocks. One reason is that to support
      distributed transactions in a key/value store, we highly rely on the metadata attached to each
      transaction to resolve conflicts (e.g., to satisfy the user-specific consistency such as causality
      we need metadata to determine the dependence between different values), and those metadata
      usually include vector clocks.

3. (Solution and Grading by: Xiaocheng Yuan)
   The false positive rate for a bloom filter is: \((1 - e^{-\frac{kn}{m}})^k\)
   Where \(k\) = # of hash functions, \(n\) = # of items inserted, \(m\) = # of bits.
   The false positive rates for a 1024-bit bloom filter with 4 hash functions with 5 and 100
   items inserted:
   \(\left(1 - e^{\frac{4 \times 5}{1024}}\right)^4 \approx 1.3995 \times 10^{-7}\)
   \(\left(1 - e^{\frac{4 \times 100}{1024}}\right)^4 \approx 0.0109\)
The false positive rates for Orlando’s solution are:

\[
(1 - e^{-\frac{4 \times 5}{512}})^4 \times (1 - e^{-\frac{4 \times 5}{512}})^4 \approx 4.6392 \times 10^{-12}
\]

\[
(1 - e^{-\frac{4 \times 100}{512}})^4 \times (1 - e^{-\frac{4 \times 100}{512}})^4 \approx 0.0075
\]

because false positive only occurs when the item is a false positive in both filters.

For both cases, Orlando’s solution performs better in terms of false positive rate. However please note that when \( n \) becomes larger (the breakeven point is when \( n \) is \( \sim 130 \)), the original solution will start dominating Orlando’s solution (\( \text{FP(Orlando)} - \text{FP(Original)} \) gets as big as \( \sim 0.3 \)).

4. (Solution and Grading by: Si Liu)
   a. sequential consistency: requires that all the operations appear to be executed in some sequential order that is consistent with the order seen at individual processes.
      linearizability: when the order above also preserves the global (i.e., wall-clock) ordering of operations
      causal consistency: requires that the effects are observed only after their causes - reads will not see a write unless its dependencies are also seen.
   b.
5. (Solution and Grading by: Shiv Verma) [Edits made by Shiv on 10/18]

The main problem is that the RM might have a clock drift (with respect to the outside world) and will force the entire cluster to drift away for the outside time. This means there will be an issue with external synchronization.

The way to definitively solve the issue is to bring in a pre-synchronized GPS/atomic clock and use that as the master for synchronization (either via NTP or Cristian’s as described in the question). GPS/atomic clocks were mentioned in class. They do not violate the “no internet” rule in the question.

Alternatively, the problem can be mitigated by using something like Berkeley’s algorithm which involves averaging the time across the cluster (and thereby probably cancelling out most of the clock drift).

The alternative solution, even though it doesn’t entirely solve the problem, also gets full credit.

6. (Solution and Grading by: Shiv Verma)

\[
\text{maxError} = \frac{(RTT - \text{min1} - \text{min2})}{2} \quad \text{(where \text{min1} and \text{min2} are the latencies from client to server and back respectively)}
\]

\[
\Rightarrow \text{maxError} = \frac{(1010 - 13.7 - 20 - 230)}{2} \text{ micro-seconds} = 373.15 \text{ micro seconds}.
\]

7. (Solution and Grading by: Hongwei Wang)
8. (Solution and Grading by: Hongwei Wang)

P0

P1

P2

P3

9. (Solution and Grading by: Sili Hui)
10. (Solution and Grading by: Bo)

a. Yes
b. Yes

Explanation: Any positive integer (fixed or random) preserves causality, since this guarantees timestamps only monotonically increase along a causal path. Thus two events that are causally related will have timestamps that obey that ordering.