The Bloom Clock to Characterize Causality in Distributed Systems

Ajay Kshemkalyani, Anshuman Misra

University of Illinois at Chicago

ajay@uic.edu

amisra7@uic.edu

September 1, 2020

Overview

Problem Statement

Fundamental problem in distributed systems

Determining causality between pairs of events

Vector clocks solve this problem

But they do not scale well

Bloom Clock

Efficient probabilistic data structure to determine causality

Vector Clock

- In the system of vector clocks, the time domain is represented by a set of n-dimensional non-negative integer vectors.
- Each process p_i maintains a vector V_i [1..n], where V_i [i] is the local logical clock of p_i and describes the logical time progress at process p_i.
- V_i [j] represents process p_i 's latest knowledge of the local time at process p_j.

Vector Clock - Protocol

Process p_i uses the following two rules R1 and R2 to update its clock:

- R1: Before executing an event, process p_i updates its local logical time as follows:

$$V_i[i] := V_i[i] + 1$$

- R2: Each message m is piggybacked with the vector clock V_j of the sender process at sending time. On the receipt of such a message (m, V_j), process p_i executes the following sequence of actions:
  - Update its global logical time as follows:

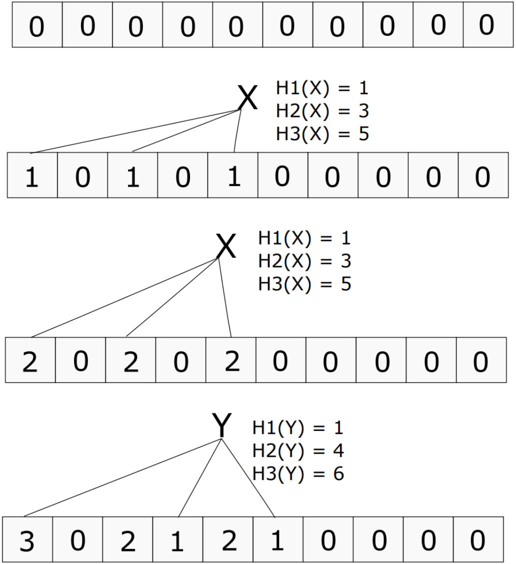
    $$1 \leq k \leq n : V_i[k] := \max(V_i[k], V_j[k])$$

  - Execute R1
  - Deliver m

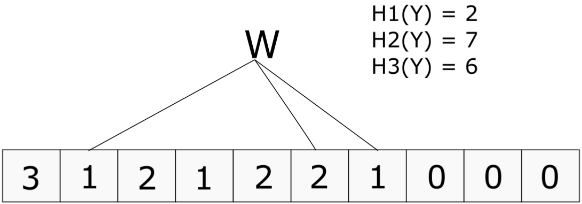
Counting Bloom Filter

- A bloom filter is a probabilistic data structure used to check whether an element is present in a set
- A counting bloom filter is a variant of bloom filter used to check the number of occurrences of an element
- Bloom filters provide a space and time efficient method of searching through a set
- Bloom filters are prone to false positives, but false negatives do not occur

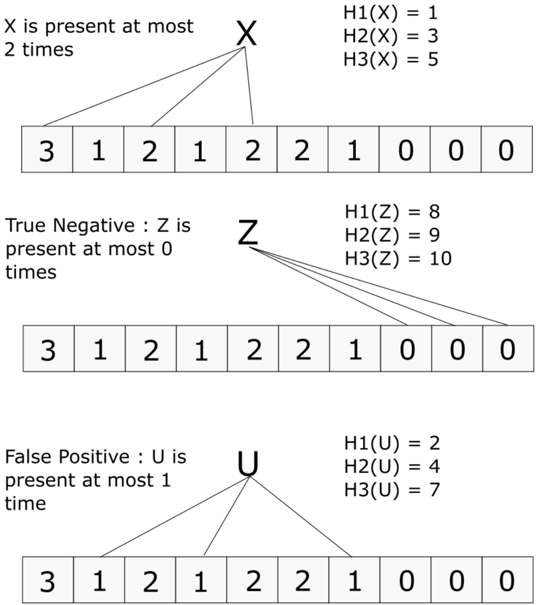
Counting Bloom Filter - Example



Counting Bloom Filter - Example (contd)



Counting Bloom Filter - Example (contd)



Motivation behind Bloom Clocks

- Vector clocks accurately create a partial order of events in a distributed execution
- Vector clocks have a space, time and message complexity of O(n)
- Bloom clocks have a complexity of O(1)
- Bloom Clocks are prone to false positives due to the probabilistic nature of Bloom filters
- Find a set of parameters that minimize the error rate for Bloom clocks

System Model

- A distributed system is modeled as an undirected graph $(\mathcal{P}, \mathcal{L})$, where $\mathcal{P}$ is the set of processes and $\mathcal{L}$ is the set of links connecting them. Let $p = |\mathcal{P}|$.
- Between any two processes, there may be at most one logical channel over which the two processes communicate asynchronously.
- We do not assume FIFO logical channels.
- The execution of process P_i produces a sequence of events $E_i = \langle e_i^0, e_i^1, e_i^2, \cdots \rangle$, where e_i^j is the j^{th} event at process P_i.
- An event at a process can be an *internal* event, a *message send* event, or a *message receive* event.

System Model (contd.)

- Let $E = \bigcup_{i \in \mathcal{P}} \{e \,|\, e \in E_i\}$ denote the set of events in a distributed execution.
- The causal precedence relation between events, denoted as $\rightarrow$, induces an irreflexive partial order $(E, \rightarrow)$.
- Let $\downarrow e = \{f \,|\, f \in E \wedge f \rightarrow e\} \bigcup \{e\}$ denote the causal past of e.
- The vector timestamp of $\downarrow e$, $V_{\downarrow e}$ is defined as: $\forall i \in [1, p], V_{\downarrow e}[i] = V_e[i]$.
- The set of events $\downarrow e \bigcap \downarrow f$ represents the common past of e and f.
- The vector timestamp of $\downarrow e \bigcap \downarrow f$, $V_{\downarrow e \bigcap \downarrow f}$ is defined as: $\forall i \in [1, p], V_{\downarrow e \bigcap \downarrow f}[i] = \min(V_e[i], V_f[i])$.

Bloom Clock Protocol

1. Initialize $B(i) = \overline{0}$.
2. (At an internal event e_i^x):
   apply k hash functions to (i, x) and increment the corresponding k positions mapped to in $B(i)$ (local tick).
3. (At a send event e_i^x):
   apply k hash functions to (i, x) and increment the corresponding k positions mapped to in $B(i)$ (local tick). Then P_i sends the message piggybacked with $B(i)$.
4. (At a receive event e_i^x for message piggybacked with B'):
   P_i executes
   $\forall j \in [1, m], B(i)[j] = max(B(i)[j], B'[j])$ (merge);
   apply k hash functions to (i, x) and increment the corresponding k positions mapped to in $B(i)$ (local tick).
   Then deliver the message.

Causality Check with Bloom Clocks

Proposition 1

Test for $y \rightarrow z$ using Bloom clocks: if $B_z \geq B_y$ then declare $y \rightarrow z$ else declare $y \not\rightarrow z$.

Prediction Scenarios

1. True Positive
   - $y \rightarrow z$ and $B_z \geq B_y$
2. False Negative
   - $y \rightarrow z$ and $B_z \ngeq B_y$
3. True Negative
   - $y \not\rightarrow z$ and $B_z \ngeq B_y$
4. False Positive
   - $y \not\rightarrow z$ and $B_z \geq B_y$

Prediction Scenarios - Observations

Observation 1

The probability of a False negative occurring is 0

Observation 2

Given that the prediction is negative, the probability of true negative occurring is 1

Bloom Clock - Performance Metrics

- The probability of a false positive, pr_{fp} = $pr(y \not\to z$ and $B_z \geq B_y)$
- The probability of a positive, pr_p = $pr(B_z \geq B_y)$.
- We approximate the probability that $\exists i \mid V_y[i] > V_z[i]$ as the probability that $\exists i \mid B_y[i] > B_z[i]$, which equals $1 - pr_p$.
- $pr(y \not\to z)$ = $1 - pr_p$.
- $pr_{fp} = pr(y \not\to z) \cdot pr(B_z \geq B_y) = (1 - pr_p) \cdot pr_p$.
- Given a positive outcome, the probability that it is false, $pr_{pf} = 1 - pr_p$
- The probability of a true positive, $pr_{tp} = pr_p \cdot pr_p = pr_p^2$
- The probability of a true negative, $pr_{tn} = 1 \cdot (1 - pr_p) = 1 - pr_p$.

Performance Metrics - A Difference in Perspective

- The probabilities pr_{pf} and pr_{fp} are functions of pr_p.
- Redefine pr_p as a step function, $pr_{\delta(p)}$:
  - $pr_{\delta(p)} = 1$ when $B_z \geq B_y$
  - $pr_{\delta(p)} = 0$ when $B_z \not\geq B_y$
- pr_{fp} becomes $(1 - pr_p) \cdot pr_{\delta(p)}$
- pr_{pf} remains $1 - pr_p$ and evaluates to pr_{fp}.
- pr_{tp} becomes $pr_p \cdot pr_{\delta(p)}$
- pr_{tn} becomes $1 - pr_{\delta(p)}$.

Computing pr_p with Bloom Clocks

The probability of getting exactly l successes in n independent Bernoulli trials with probability of success p is given by:

$$b(l,n,p) = \binom{n}{l}\, p^l\, (1-p)^{(n-l)}$$

Utilizing the binomial distribution to compute pr_p we get:

$$\widehat{pr_p}(k, m, B_y, B_z) = \prod_{i=1}^{m}(1 - \sum_{l=0}^{B_y[i]-1} b(l, B_z^{sum}, 1/m))$$

Computing Metrics on an Execution Slice

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \; Precision = \frac{TP}{TP + FP},$$
$$Recall = \frac{TP}{TP + FN}, \; fpr = \frac{FP}{FP + TN}$$

Recall is always 1 with Bloom clocks

Computing Metrics on an Execution Slice (contd)

$$1 - \widehat{Acc} = \frac{\sum_{x,x'} (1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'} 1}$$

$$\begin{aligned} 1 - \widehat{Prec} &= \frac{\sum_{x,x'} (1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'} (1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x') + pr_p(x,x') \cdot pr_{\delta(p)}(x,x')} \\ &= \frac{\sum_{x,x'} (1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'} pr_{\delta(p)}(x,x')} \end{aligned}$$

$$\begin{aligned} \widehat{fpr} &= \frac{\sum_{x,x'} (1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'} (1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x') + (1 - pr_{\delta(p)}(x,x'))} \\ &= \frac{\sum_{x,x'} (1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'} 1 - pr_p(x,x') \cdot pr_{\delta(p)}(x,x')} \end{aligned}$$

Analysis and Discussion

1. pr_p, the probability of a positive, is low if z is close to y and this probability increases as z goes further in the future of y.
2. pr_{pf}, the probability that a positive is false, decreases as z goes further in the future of y.
3. pr_{fp}, the probability of a false positive, which is the product of pr_p and pr_{pf}, is lower than the above two probabilities. It will likely reach a maximum of 0.25 and then decrease.

Future Work

1. Experiment with a large number of processes ($>$1000) and compute false positive rate, precision and accuracy of bloom clocks.
2. Determine optimal values of parameters (m,k) in relation to the number of processes.
3. Ascertain applications where vector clocks can be replaced by bloom clocks.