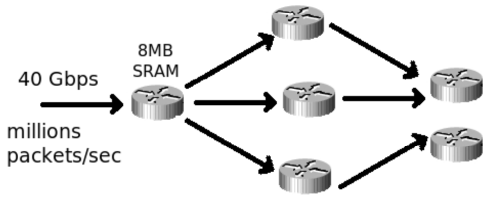
Lecture 2: Streaming Algorithms for Counting Distinct Elements

Ashwin Lall

20th August, 2008

Streaming Algorithms



Anomaly detection
- DoS, DDoS, worm
- link down

Traffic accounting

Quality of Service

Traffic engineering

Streaming Algorithms

Streaming algorithms have the following properties:

1. items in the stream are presented sequentially
2. single pass over the data
3. limited (sublinear) space in which to operate
4. updates per item must be very fast

Notation

- A *stream* is an ordered tuple over the alphabet

$$\{1, 2, 3, \ldots, n\}.$$

  - source or destination address ($n = 2^{32}$)
  - source or destination port ($n = 64\text{K}$)
  - or any combination of flow header fields ($n \approx 2^{100}$)

- Frequency of ith item is m_i, and the length of the stream is

$$m = \sum_{i=1}^{n} m_i.$$

- The stream is represented by $(a_1, a_2, a_3, \ldots, a_m)$, where each $a_t \in \{1, 2, 3, \ldots, n\}$.

Some Sample Streaming Problems

- Estimating frequency moments (e.g., F_2 is a measure of the “skewness”)

$$F_k = \sum_{i=1}^{n} m_i^k$$

- Estimating entropy (Useful metric for anomaly detection)

$$H = -\sum_{i=1}^{n} \frac{m_i}{m} \log\left(\frac{m_i}{m}\right)$$

- Counting the number of distinct elements:

$$D = ||\{i \mid m_i \neq 0\}||$$

Definition

(A, B, B, A, C, A, E, A, C, C, C, B, B)

Number of distinct elements = 4

Motivation

What does it mean when

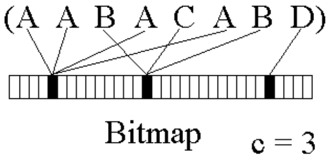
1. the number of destination ports spikes?
   - Port scan
2. the number of source addresses spikes?
   - Flash crowd
3. the number of source addresses to a fixed destination address spikes?
   - DDoS attack

First-attempt Solutions

- Maintain a record for every distinct flow
  - Needs $O(n)$ memory or at least $O(m)$
- Sample (e.g., 1 in 100 packets) - Cisco's NetFlow
  - Hohn and Veitch show that sampling can give inaccurate results
- Use a lossy set maintenance data structure (Bloom filter)
  - Still requires $O(m)$ space

Bitmap Algorithm

- Initialize a large bitmap (binary array) of size B with all zeroes
- Choose a hash function $h : \{1, \ldots, n\} \to \{1, \ldots, B\}$
- For each flow label $f \in \{1, \ldots, n\}$, compute $h(f)$ and mark that position in the bitmap with a 1.
- Afterwards, count the number of positions in the bitmap with a 1 and call it c
- Estimate number of distinct items by $B \ln \left(\frac{B}{B-c}\right)$.



Brief Proof

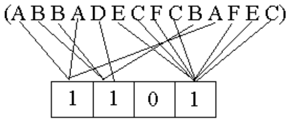
Consider the probability of a position in the bitmap being zero at the end: $(1 - 1/B)^D \approx e^{-D/B}$, where D is the number of distinct items.

The expected number of zeroes is B times this (since the bitmap is of size B), so we have: $B - c \approx B * e^{-D/B}$.

Solving for D, we get $D = B \ln\left(\frac{B}{B-c}\right)$.

This turns out to be a maximum likelihood estimator for the number of distinct elements.

Bitmap Example



Estimated number of distinct elements = $4 * \ln(4/1) = 5.55 \approx 6$

The Coupon Collector Problem

A coupon collector wants to collect all of C different types of coupons.

Every time he gets a coupon, he gets one of the C with replacement.

Question: How many coupons will he need to draw before he has one of every type?

The Coupon Collector Problem - Solution

We look at the number of coupons that he must pick to get the ith distinct coupon:

1. To get the first coupon, he only needs one try.
2. To get the second distinct coupon (any of $C - 1$ of C) he needs expected $\frac{C}{C-1}$ tries.
   $\ldots$
3. To get the ith distinct coupon (any of $C - i + 1$ of C) he needs expected $\frac{C}{C-i+1}$ tries.

Hence, expected number of tries is
$1 + \frac{C}{C-1} + \frac{C}{C-2} + \ldots \frac{C}{1} = C * (1 + 1/2 + 1/3 + \ldots + 1/C) \approx C \ln C$.

Multi-resolution Bitmaps

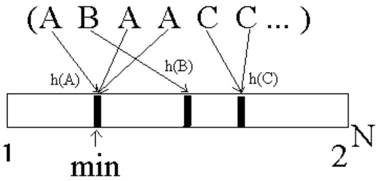
A *virtual* bitmap is one where the flows are first sampled before being entered into the bitmap. This cuts down the amount of memory by about the sampling factor.

However, from the Coupon Collector's problem, we know that $B \ln B$ distinct flows will (in expectation) fill the bitmap entirely with 1's.

If we are unsure how large the number of distinct items can get beforehand, we can use bitmaps of various sizes and approximate the value from the one with the most appropriate size.

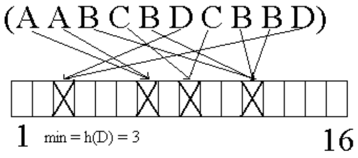
Min-hash Algorithm

- Use a hash function $h : \{1, \dots, n\} \to \{1, \dots, 2^N\}$
- For each flow label $f \in \{1, \dots, n\}$, compute $h(f)$ and retain the minimal value
- Approximate the number of distinct items by $2^N / \min_f h(f) - 1$



Only need $N = O(\log(n))$ bits to maintain minimum.

Min-hash Example



Number of distinct elements $\approx 16/3 - 1 = 4.33$