Advanced Topics in Communication Networks

Programming Network Data Planes

Laurent Vanbever

nsg.ee.ethz.ch

ETH Zürich

Oct 25 2018
Two weeks ago on

Advanced Topics in Communication Networks
A bloom filter is a streaming algorithm answering specific questions approximately.
A bloom filter is a streaming algorithm answering specific questions approximately.

Is X in the stream?
What is in the stream?

Invertible Bloom Filter
A bloom filter is a streaming algorithm answering specific questions approximately.

Is X in the stream? What is in the stream? Invertible Bloom Filter

What about other questions?
Is a certain flow in the stream?
Bloom Filter

What flows are in the stream?
Invertible Bloom Filter, HyperLogLog Sketch, ...

**How frequently does an flow appear?**
Count Sketch, CountMin Sketch, ...

What are the most frequent elements?
Count/CountMin + Heap, ...

How many flows belong to a certain subnet?
SketchLearn \textsuperscript{SIGCOMM 18}
In the worst case, an algorithm providing **exact frequencies** requires **linear space**.

\[
\text{Data Stream} \quad \begin{align*}
  n \text{ elements in total} & \quad \rightarrow \quad n \text{ distinct elements} \quad \text{(in the worst case)} \\
  & \quad \rightarrow \quad n \text{ counters} \text{ required? :(}
\end{align*}
\]
Probabilistic datastructures can help again!

Bloom Filters
quickly “filter” only those elements that might be in the set
Save space by allowing false positives.

Sketches
provide a approximate frequencies of elements in a data stream.
Save space by allowing mis-counting.
A **CountMin sketch** uses the same principles as a counting bloom filter, but is **designed** to have **provable L1 error bounds** for frequency queries.
A **CountMin sketch** uses the same principles as a counting bloom filter, but is **designed** to have **provable L1 error bounds** for frequency queries.
The estimation error exceeds $\varepsilon \| x \|_1$ with a probability smaller than $\delta$. 

\[ \Pr \left[ \hat{x}_i - x_i \geq \varepsilon \| x \|_1 \right] \leq \delta \]

**Relative to L1 norm**

The estimated frequency $\hat{x}_i$ minus the true frequency $x_i$ is greater than or equal to $\varepsilon$ times the L1 norm of $x$, with a probability smaller than $\delta$. 

The estimation error exceeds $\varepsilon \| x \|_1$ with a probability smaller than $\delta$. 

*estimated frequency* 
*true frequency* 
*sum of frequencies*
A **CountMin sketch** uses the same principles as a counting bloom filter, but is **designed** to have **provable L1 error bounds** for frequency queries.
A **CountMin** Sketch uses multiple arrays and hashes.

d arrays
(one hash function per array)

w indices per array
(range of hashes)

w · d counters
(total size)
A CountMin sketch uses the same principles as a counting bloom filter, but is designed to have provable L1 error bounds for frequency queries.

**CountMin sketch recipe**

Choose \( d = \lceil \ln \frac{1}{\delta} \rceil, \ w = \lceil \frac{e}{\varepsilon} \rceil \)

Then \( \hat{x}_i - x_i \geq \varepsilon \|x\|_1 \) with a probability less than \( \delta \)
A **CountMin sketch** uses the same principles as a counting bloom filter, but is **designed** to have **provable L1 error bounds** for frequency queries.

→ only one design out of many!
A **Count sketch** uses the same principles as a counting bloom filter, but is **designed** to have **provable L2 error bounds** for frequency queries.
The Count sketch uses **additional hashing** to give **L2 error bounds**, but requires more **resources**.

**CountMin sketch recipe**

Choose \( d = \lceil \ln \frac{1}{\delta} \rceil, \ w = \left\lfloor \frac{e}{\epsilon} \right\rfloor \)

Then \( \hat{x}_i - x_i \geq \epsilon \|x\|_1 \) with a probability less than \( \delta \)

**Count sketch recipe**

Choose \( d = \lceil \ln \frac{1}{\delta} \rceil, \ w = \left\lfloor \frac{e}{\epsilon^2} \right\rfloor \)

Then \( \hat{x}_i - x_i \geq \epsilon \|x\|_2 \) with a probability less than \( \delta \)
Sketches are the new black

OpenSketch
NSDI ‘13

UnivMon
SIGCOMM ‘16

SketchLearn
SIGCOMM ‘18

One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon

Zaowei Liu, Artemiy Maniuk, Gregory Varghese, Yaya Seier, Vladimir Bavarian, John A. Stankovic, and Neera Joshi

ABSTRACT
Network management requires accurate estimates of network flow metrics, such as bandwidth utilization, for fine-grained traffic control. While network management tools can provide access to these metrics, they can be expensive and impractical to deploy in large networks. In this paper, we propose UnivMon, a network monitoring platform that uses sketch-based monitoring techniques to estimate network flow metrics. We evaluate UnivMon using real-world data from a large-scale network and show that it provides accurate estimates of network flow metrics without incurring the cost of deploying traditional network management tools.

CCS Concepts
Network and distributed computing → Network monitoring, Network measurement.

Keywords

SketchLearn: Relieving User Burdens in Approximate Measurement with Automated Statistical Inference

Qun Huang, Patrick C. C. Lee, and Yangang Bao

ABSTRACT
Network measurements are indispensable to network management and applications. However, they are often costly and impractical to deploy in large-scale networks. In this paper, we propose SketchLearn, a system that uses machine learning techniques to automatically learn and improve network measurements. We evaluate SketchLearn using real-world data from a large-scale network and show that it provides accurate estimates of network flow metrics without incurring the cost of deploying traditional network management tools.
Today we’ll talk about: important questions, how ‘sketches’ answer them, limitations of ‘sketches’, and my master thesis :)
Sketches compute statistical summaries, favoring elements with high frequency.

Let $\varepsilon = 0.01$, $\|x\|_1 = 10000$ (⇒ $\varepsilon \cdot \|x\|_1 = 100$)

Assume two flows $x_a$, $x_b$,

with $\|x_a\|_1 = 1000$, $\|x_b\|_1 = 50$

Error relative to stream size: 1%

flow size: $x_a$: 10%, $x_b$: 200%
Other Problems a Sketch can’t handle

causality  patterns  rare things
This week on
Advanced Topics in Communication Networks
We will look at one example of a P4 hardware switch along with two examples of P4-enabled applications.

- P4 hardware
- NetCache (load-balancing cache for key-value store) [SOSP'17]
- NetChain (consistent, fault-tolerant key-value store) [NSDI'18]

+ Albert Gran's Master Thesis presentation on "Making Scheduling Programmable"
See https://adv-net.ethz.ch for follow-up slides.
Advanced Topics in Communication Networks
Programming Network Data Planes

Laurent Vanbever

nsg.ee.ethz.ch

ETH Zürich
Oct 25 2018