SketchVisor

Robust Network Measurement for Software Packet Processing

Qun Huang, Xin Jin, Patrick P. C. Lee, Runhui Li, Lu Tang, Yi-Chao Chen, Gong Zhang
Huawei, Johns Hopkins University, The Chinese University Hong Kong
Presentation Outline

Context

- Network Measurement
- Sketches

Paper

- Design of SketchVisor
  - Data Plane
  - Control Plane
- Evaluation
- Conclusion
Defining Measurement: Traffic Statistics

For each epoch, we want to detect or measure one or several of the following:

- Heavy Hitter
- Heavy Changer
- DDoS
- Superspreader
- Cardinality
- Flow size distribution
- Entropy
Defining Measurement: Problem Statement

- Controller
- Switches
- endhosts
- routing
- measurements
- packets
Measurement Example: Heavy Hitter

Heavy Hitter

**A flow** whose byte count exceeds a threshold (during an epoch).
Measurement Example: Heavy Hitter

Heavy Hitter

A flow whose byte count exceeds a threshold (during an epoch).

Result: a list of flows (5 tuples)

Need to collect: byte count per flow
Measurement Example: Heavy Hitter

Heavy Hitter

A flow whose byte count exceeds a threshold (during an epoch).

Result: a list of flows (5 tuples)

Need to collect: byte count per flow

Naive algorithm:

A table with all flows and corresponding byte counts. When new packet arrives, add byte count to the associated flow.
Measurement Qualities Tradeoff

- Naive Counting Table
  - very accurate
  - bad performance
  - high memory usage

- Packet Sampling
  - better performance
  - much less accuracy
Measurement Qualities Tradeoff

- Naive Counting Table
  - very accurate
  - bad performance
  - high memory usage

- Packet Sampling
  - better performance
  - much less accuracy

- Sketches
  - less memory usage
  - slightly less accurate
  - until now: bad performance under stress
# Count-Min Sketch for HH: Hashmap

A Count-Min Sketch is a probabilistic data structure used for approximating the frequency of elements in a data set. It uses multiple hash functions to distribute the elements into a smaller data structure, allowing for fast approximation of the frequency of elements. The sketch is constructed by counting the number of times each element is hashed to each slot in the data structure.

## Hash Functions

The hash functions used in the sketch are denoted as $h_0, h_1, h_2, h_3$. Each function maps elements to a hash range, which is the number of slots in the data structure. The hash range $w$ is defined as $w = \lceil \frac{m}{k} \rceil$, where $m$ is the size of the data set and $k$ is the number of hash functions.

### Hash Range

The hash range is divided into $w$ slots, each corresponding to a hash function. The elements are hashed to these slots, and the count in each slot is incremented.

### Count-Min Sketch Table

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>0</th>
<th>22</th>
<th>12</th>
<th>14</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$h_1$</td>
<td>32</td>
<td>32</td>
<td>12</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>$h_2$</td>
<td>22</td>
<td>12</td>
<td>28</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>$h_3$</td>
<td>14</td>
<td>18</td>
<td>3</td>
<td>12</td>
<td>36</td>
</tr>
</tbody>
</table>
Count-Min Sketch for HH: Adding a Packet

id: 5-tuple
size: 8

<table>
<thead>
<tr>
<th>hash functions</th>
<th>h0</th>
<th>h1</th>
<th>h2</th>
<th>h3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>32</td>
<td>22</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>32</td>
<td>12</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>+8</td>
<td>12</td>
<td>+8</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>28</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>18</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+8</td>
<td>36</td>
</tr>
<tr>
<td>hash range w</td>
<td>0</td>
<td>32</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>12</td>
<td>24</td>
<td>36</td>
</tr>
</tbody>
</table>

James Dermelj, 9.3.2018
What is the bytecount of 5-tuple $x$?

<table>
<thead>
<tr>
<th>hashfunctions</th>
<th>h0</th>
<th>h1</th>
<th>h2</th>
<th>h3</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash range</td>
<td>0</td>
<td>30</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>32</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>12</td>
<td>28</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>18</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>
Count-Min Sketch for HH: Querying

What is the bytecount of 5-tuple $x$?

<table>
<thead>
<tr>
<th>hash functions</th>
<th>h0</th>
<th>h1</th>
<th>h2</th>
<th>h3</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash range $w$</td>
<td>0</td>
<td>30</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>32</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>12</td>
<td>28</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>18</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>
Conclusion on Sketches

- Sketches use a small amount of memory
- Query approximates the solution by identifying boundaries
- Relatively good performance due to few operations
- Variety of different Sketches can be used simultaneously

BUT

- Under high traffic: Sampling often still necessary, which reduces accuracy.
SketchVisor: **Robust** Network Measurement for Software Packet Processing
Data Plane: Overview

- use several sketches to accurately measure statistics
- if the buffer for the normal path is full the fast path is used instead
Data Plane: Fast Path

Idea: Only track the most influential flows (highest bytecounts). Drop all other flows.

*Fast path algorithm* (section 4.2), key points:

- estimates byte counts for the top-k flows
- stores additional data for defining bounds
- estimation error upper bound for any flow:
  \[ O(V / k) \]
  \( V \): sum of the byte counts of all packets in fast path
Control Plane: Network-Wide Recovery

- Control Plane collects the measurements from all switches

The goal is to recover a total sketch $T$, as if the fast path was tracked by a sketch.

(b) SketchVisor control plane.
Control Plane: Compressive Sensing

Approximations for compressive sensing:
- $T$ is a low rank matrix
- $x$ and $sk(x)$ are sparse
- $y$ and $sk(y)$ are of small noise

LENS objective function:

$$\text{minimize: } \alpha \|T\|_* + \beta \|x\|_1 + \frac{1}{2\gamma} \|y\|_F^2$$

Solving it with the Alternating Direction Method
Experiments

- testbed
  9 hosts, 10Gb NIC
  does not show scalability

- in-memory
  simulating the whole network from memory
Results - Throughput

and when using parallelization: 2 cores are sufficient to achieve above 40Gbps throughput (in-memory test)
Results - Accuracy

(f) HC Relative error

(e) HC Precision

(d) HC Recall
Results - further

- accuracy increases with the number of hosts
- accuracy increases with fast path size (should increase linearly with number of flows)
- kick out in fast path algorithm very computationally expensive, but rather rare
Conclusion

- Sketches still are a very effective and efficient way of measuring network statistics.
- In comparison to similar methods, SketchVisor exceeds in every aspect (at least in the paper)
- Is easily applicable on existing sketch-solutions without needing specialized hardware.

- Potential problem: Reversibility: which flow contributed to which flow by how much? (not addressed in paper)
Conclusion on the paper

- Well structured: identify problem, pose solution, elaborate and demonstrate
- Detailed description of theory
- Very detailed description of experiments and results. This probably makes replication easy.

- No suggestions of further work
Thank You!

and now: discussion