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
Monitoring Traffic Statistics

Network management

Network-wide flow statistics

Traffic distribution
Flow cardinality
Heavy hitters
Sketch: A Promising Solution

- Sketch: a family of randomized algorithms
  - Key idea: project high-dimensional data into small subspace

- Subspace reflects mathematical properties
  - Strong theoretical error bounds when querying for statistics

Sketch: Randomized projection

Input data → Randomized projection → Subspace → Statistics

High-dimensional data

Small subspace:
low computation & communication overheads
Example: Count-Min Sketch

- **Count flow packets**

- **Update with a packet**
  - Hash flow id to one counter per row
  - Increment each selected counter

- **Query a flow**
  - Hash the flow to multiple counters
  - Take the minimum counter as estimated packet count

- **Theoretical guarantees**
  - Allocate rows and counters each row
  - The error for a flow is at most with probability at least
Our Focus

- **Sketch-based measurement atop software switches**
Limitation of Sketches

Basic sketches

Lack of generality

Limited query

More structures

Complicated sketches

![Graph showing CPU cycles and throughput](image)
Our Contributions

SketchVisor: Sketch-based Measurement System for Software Packet Processing

- **Performance**
  - Catch up with underlying packet forwarding speed

- **Resource efficiency**
  - Consume only limited resources

- **Accuracy**
  - Preserve high accuracy of sketches

- **Generality**
  - Support multiple sketch-based algorithms

- **Simplicity**
  - Automatically mitigate performance burdens of sketches without manual tuning
Architecture: Double-Path Design

Control plane

Merge two paths
- Recover lost information
- Transparent to users

Data plane

User-defined sketches
- High accuracy
- (Relatively) slower

Fast path
- High speed
- (Relatively) less accurate
- General for multiple sketches

Network-wide sketch
Network-wide merge & recovery
Global normal path
Global fast path

Switches

Local normal path
Sketch 1
Sketch 2
Sketch 3
Sketch 4

Local fast path
Fast path algorithm

Buffer

Packets
Forwarding

To control plane

Fast path

Control plane

Data plane
Key Questions

- Data plane: how to design the fast path algorithm?

- Control plane: how to merge the normal path and fast path?
Intuitions

➢ Consider sketches which map flow byte counts into counters
  • Other sketches (e.g., Bloom Filter) can be converted

Each large flow has significant impact
Each small flow has limited impact
Aggregated impact of small flows is significant
Fast Path Algorithm

**Ideal algorithm**
Infeasible with limited resources

- Per-flow byte count of large flows
- Aggregated byte count of small flows

**Our practical algorithm**

- (Approximate) per-flow byte count of large flows
- (Approximate) aggregated byte count of small flows

**How**

Byte of small flows = total byte – byte of large flows
Approximate Tracking of Large Flows

- A small hash table
  - “Guess” and kick out potentially small flows when table is full
  - Each flow has three counters

<table>
<thead>
<tr>
<th>Flow ID</th>
<th>Counter 1</th>
<th>Counter 2</th>
<th>Counter 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Flow 2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Flow 3</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Performance and Accuracy

- Theoretical analysis shows:
  - All large flows are tracked
  - Amortized $O(1)$ processing time per packet
  - Bounded errors

- Compared to Misra Gries top-k algorithm
Key Questions

➢ Data plane: how to design a fast path algorithm?

➢ Control plane: how to merge the normal path and fast path?
Control Plane: Challenge

- Input insufficient to form network-wide sketches

**Global normal path**

Input 1: Incomplete sketch with missing values

**Global fast path**

Input 2: Approximate large flows in fast path

Input 3: Total byte counts in fast path

Expected output: Network-wide sketch

Network-wide recovery

- Flow ID | Counter 1 | Counter 2 | Counter 3
- Flow 1  | 4         | 0         | 1
- Flow 2  | 1         | 0         | 1
- Flow 3  | 2         | 0         | 1

Total byte count
The recovery process can be expressed as

\[ T = N + sk(x + y) \]

- Expected output sketch (unknown)
- Sketch in global normal path (known)
- Large flows in fast path (unknown)
- Small flows in fast path (unknown)
Matrix Interpolation Problem

Based on theoretical analysis and microbenchmarks

\[ T = N + \text{sk}(x + y) \]
Matrix Interpolation Problem

- Based on theoretical analysis and microbenchmarks

\[ T = N + sk(x + y) \]

- Expected output sketch (unknown)
- Sketch in global normal path (known)
- Large flows in fast path (unknown)
- Small flows in fast path (unknown)

(low-rank structure)
Matrix Interpolation Problem

- Based on theoretical analysis and microbenchmarks

\[ T = N + sk(x + y) \]

- Expected output sketch (unknown)
- Sketch in global normal path (known)
- Large flows in fast path (unknown)
- Small flows in fast path (unknown)
- (low-rank structure)
- (1. sparse vector)
- (2. each flow is bounded)
Matrix Interpolation Problem

- Based on theoretical analysis and microbenchmarks

Expected output sketch (unknown)

Sketch in global normal path (known)

\[ T = N + sk(x + y) \]

Large flows in fast path (unknown)

Small flows in fast path (unknown) (small and close values)

(1. sparse vector)
(2. each flow is bounded)

(low-rank structure)
Matrix Interpolation Problem

- Based on theoretical analysis and microbenchmarks

\[ T = N + sk(x + y) \]

- Expected output sketch (unknown)
- Sketch in global normal path (known)
- Total traffic is known
- Large flows in fast path (unknown)
- Small flows in fast path (unknown) (small and close values)

- (low-rank structure)
- (1. sparse vector)
- (2. each flow is bounded)
Recovery Approach

Existing Information

\[ T = N + sk(x+y) \]

- \( x \) is sparse
- Flows in \( x \) are bounded
- \( T \) has low-rank structure
- Values in \( y \) are small and close
- Total traffic of \( x \) and \( y \) is known

Compressive sensing framework

Optimization problem (encode existing information)

Solve optimization problem

An estimated network-wide sketch
Evaluation
Evaluation Setup

- Prototype based on OpenVSwitch
- Environments
  - Testbed: 8 OVS switches connected by one 10Gbps hardware switch
  - In-memory simulation: 1 – 128 simulation processes
- Workloads: CAIDA

Measurement tasks

Heavy hitter detection  Heavy changer detection  Superspreader detection
DDoS detection          Cardinality estimation   Entropy estimation
                        Flow distribution estimation
Throughput

- Compared with two data plane approaches
  - NoFastPath: use only Normal Path to process all traffic
  - MGFastPath: use Misra-Gries Algorithm to track large flows in Fast Path
- Achieve ~10 Gbps in testbed (single CPU core)
- Achieve ~20 Gbps in simulation (single CPU core)
Accuracy

- Compare with four recovery approaches
  - Ideal: an oracle to recover the perfect sketch
  - NR: no recovery at all
  - LR: only use lower estimate of large flows in Fast Path
  - UR: only use upper estimate of large flows in Fast Path

- SketchVisor matches the ideal approach
Network-wide Results

- Recover sketch from 1-128 hosts
- Accuracy improved as number of hosts increases

- Work for both byte-based tasks (heavy hitter detection) and connection-based tasks (cardinality estimation)
Conclusion

- SketchVisor: high-performance system for sketch algorithms
- Double-path architecture design
  - Slower and accurate sketch channel (normal path)
  - Fast and less accurate channel (fast path)
- Fast path algorithm in data plane
  - General and high performance
- Recovery in control plane
  - Achieve high accuracy using compressive sensing
- Implementation and evaluation
  - OpenVSwitch based implementation
  - Trace-driven experiments