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

Qun Huang, Patrick P. C. Lee, Yungang Bao
Typical Approximate Measurement

Our goal: identify and relieve user burdens for approximate measurement
Errors need to be specified:

- **bias**: an answer deviates true answer by $\epsilon$
- **failure probability**: fail to produce small-error answer with a probability $\delta$
Large-threshold configurations fail to work for small-threshold queries
No guideline for sufficiently small threshold
• Vary across management operations and traffic characteristics
User Burden 3

1: Hard to specify
Expected errors

2: Hard to know in advance
Query thresholds

Network traffic

Measurement Algorithm

Measurement Results

Query

Network statistics

Domain knowledge

3: Hard to follow theory to tune

Domain knowledge is not always available
- Theories usually show worst-case results
- Configuration for worst case not practically efficient
Algorithm works on a particular flow-key
- Flow-key definition must be fixed in deployment
User Burden 5

1: Hard to specify
Expected errors

2: Hard to know in advance
Query thresholds

3: Hard to follow theory to tune
Domain knowledge

4: Hard to redefine flow-keys
Network traffic

5: Hard to quantify actual errors
Measurement Results

Query

Infeasible to track errors for a particular flow
- Configuration only tells worst-case and overall errors
All User Burdens

1: Hard to specify
Expected errors

2: Hard to know in advance
Query thresholds

3: Hard to follow theory to tune
Domain knowledge

4: Hard to redefine flow-keys
Network traffic

5: Hard to quantify actual errors
Query results

Measurement Algorithm

Configuration

Measurement Results
Our Work

SketchLearn: Sketch-based measurement system with limited user burdens

- Relieve five user burdens
- Performance
  - Catch up with underlying packet forwarding speed
- Memory efficiency
  - Consume only limited memory
- Accuracy
  - Preserve high accuracy of sketches
- Generality
  - One design and one configuration for multiple tasks
  - Deployable in both software and hardware
Root Cause: Resource Conflicts

- Previous work: **perfect** configuration to eliminate conflicts
  - Determined by many factors
High-Level Idea

- Not pursue perfect configurations to mitigate resource conflicts
  - Hard to identify right trade-offs
- Characterize resource conflicts in an “imperfect” configuration

- Expected errors
- Query parameters
- Domain knowledge
- Flow definitions

Data structure with imperfect configuration

Network queries

Statistical Properties

Conflict model

Not necessary
How to Realize?

Data structure with imperfect configuration

Network queries

Statistical Properties

Conflict model
Design Data Structure

Data structure with imperfect configuration

Statistical Properties

Conflict model

Network queries

13
Multi-level Sketch [Cormode, ToN 05]

- **L**: # of bits considered
- Data structure: **L+1** levels (from 0 to L), each of which is a counter matrix
- Level 0 is always updated
- Level k is updated iff k-th bit in a flowkey is 1
- All levels share **same** hash functions
Statistical Properties of Resource Conflicts
Properties

- Does a flow update level \( k \)?
  - Depends on inherent distribution of flow keys

- Does a flow update bucket \((i, j)\)?
  - Depends on hash functions of sketch

- A theory should characterize the above two factors
Main Theorem: if no large flows, $R_{i,j}[k]$ follows Gaussian distribution with mean $p[k]$
Build Conflict Model

Multi-level Sketch

Main Theorem

Conflict model

Network queries
Statistical Model Inference

Goals
- Extract all large flows
- Guarantee remaining flows in sketches are small
- Estimate Gaussian distributions for each level

Challenges
- No guidelines to distinguish large and small flows
Self-Adaptive Inference Algorithm

Estimate Gaussian distributions

Multi-Level Sketch

Remove extracted large flows (Flow extracted)

Use smaller threshold (No flow extracted)

Imperfect Gaussian Distribution

Extract large flows

Large Flows
Self-Adaptive Inference Algorithm

- Estimate Gaussian distributions
- Remove extracted large flows (Flow extracted)
- Use smaller threshold (No flow extracted)
- Extract large flows
Intuition: a large flow
- (i) results in extremely (large or small) $R_{i,j}[k]$, or
- (ii) at least deviates $R_{i,j}[k]$ from its expectation $p[k]$ significantly

Key idea: examine $R_{i,j}[k]$ and its difference from $p[k]$, then
- Determine $k$-th bit of a large flow (assuming it exists)
- Estimate frequency
- Associate flow confidence

More details in paper!
 Guarantee

- **Theorem**: \( w \) is sketch width, flows exceeding \( 1/w \) of total must be extracted
  - Empirical results: even flows that are smaller than \( 1/w \) can also be extracted!

64KB: flows above 0.6% of total traffic can be extracted (by theorem)

Practice: >99% flows exceeding 0.3% are also extracted with 64KB
How to Perform Network-wide Queries?

Multi-level Sketch

Self-adaptive Model Inference

Main Theorem

Decoupled Components
- Extracted Large Flows
- Residual Sketch

Conflict Characteristics
- Flow Confidence
- Gaussian Distributions

Network queries
Supported Network Queries

- Per-flow byte count
- Heavy hitter detection
- Heavy changer detection
- Cardinality estimation
- Flow size distribution estimation
- Entropy estimation
Extended Query Model

- Allow query for arbitrary flowkeys
  - Only use corresponding levels

- Estimate actual query errors
  - Use attached errors for each large flow
  - Use Gaussian distributions
Putting It Together

Multi-level Sketch

Main Theorem

Self-adaptive Model Inference

Decoupled Components

Conflict Characteristics

Extracted Large Flows

Flow Confidence

Residual Sketch

Gaussian Distributions

Network-wide Query Runtime
(Slight) User Burdens

- Users now just need to configure the multi-level sketch

- Expected errors
- Query parameters
- Domain knowledge
- Flow definitions

Multi-level Sketch

Columns: based on minimum flow or memory budget

- One row suffices

Example:
400 KB memory
Require flows exceeding 0.1% → 1000 columns
Implementation

- Challenge: updating L+1 levels is time consuming

- Solution: parallel updating
  - L+1 levels are independent

- Software
  - Based on OpenVSwitch + DPDK
  - Parallelism with SIMD

- Hardware
  - Based on P4 programmable switches
  - Parallelism with P4 pipeline stages
Evaluation

Platforms
• Software: OpenVSwitch + DPDK
• Hardware: Tofino switch
• Large-scale simulation

Traces
• Caida 2017
• Data center traffic (IMC 2010)
Fitting Theorem

- Guaranteed boundary of flow extraction

Flows that are guaranteed to be extracted

Flows that are not guaranteed (but are likely) to be extracted
Arbitrary Flow Keys

- Query for three flow keys

Heavy hitter detection

Traffic size distribution
More Experiments

- Resource consumption
- Generality for various measurement tasks
- Efficiency of attached query errors
- Network-wide measurement
Conclusion

- Analyze 5 user burdens in existing approximate measurement

- SketchLearn framework
  - Multi-level data structure design
  - Theory: counters follow Gaussian distributions when no large flows
  - Self-adaptive model inference algorithm
  - Extended query models

- Prototype and evaluations

- Source code available at: https://github.com/huangqundl/SketchLearn
Limitations and Future Work

- Less pipeline consumptions
- Quantify Gaussian distributions and convergence rate
- More applications