# Invertible Sketches for Network Measurement at Scale

**Patrick P. C. Lee**

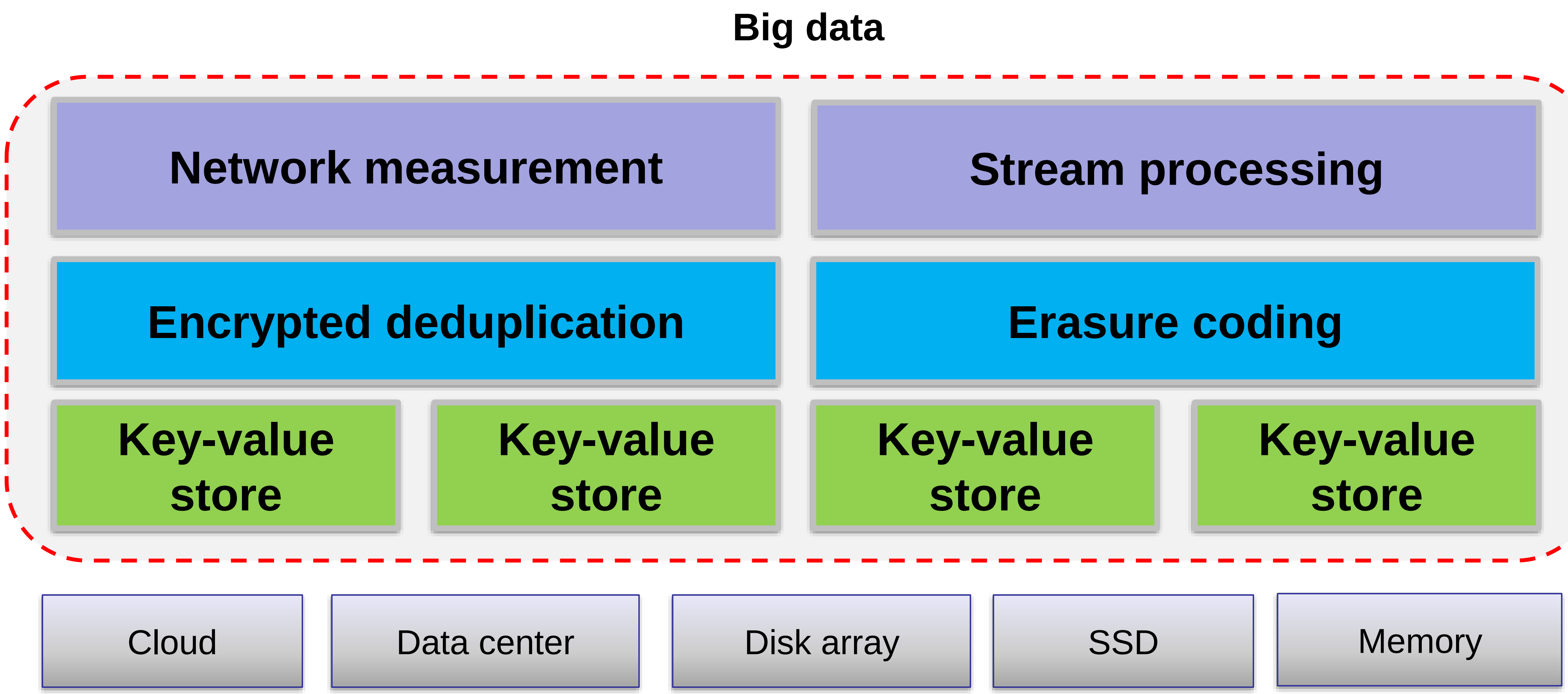
The Chinese University of Hong Kong

*Joint work with Lu Tang (Xiamen University) and Qun Huang (Peking University)*

# Applied Distributed Systems Lab (ADSLab)

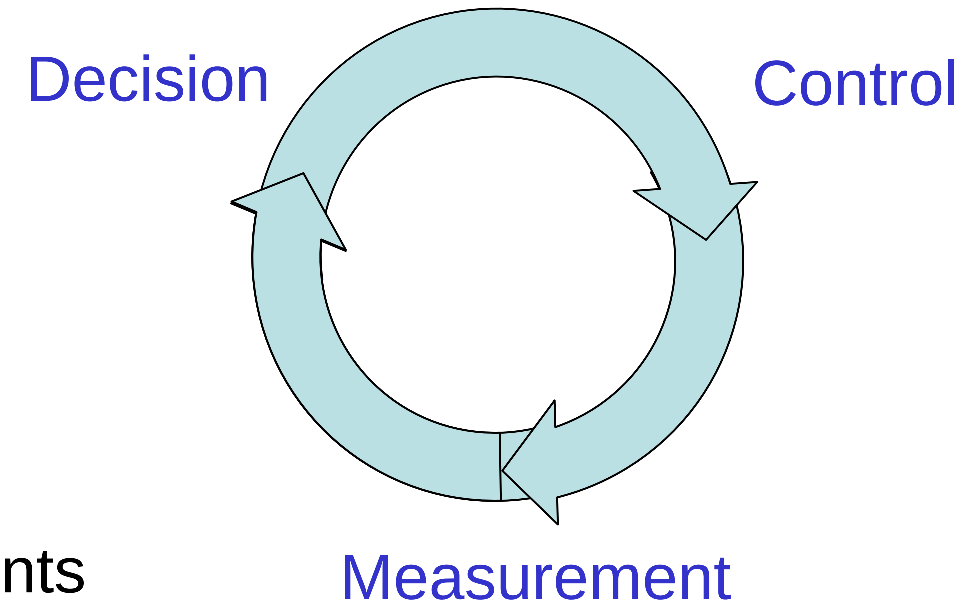
- Goal: Improve **dependability** of large-scale computer systems
  - Fault tolerance, recovery, security, and performance guarantees that need to be achieved in order to maintain the correctness and performance of a computer system
- Our approach:
  - Build prototypes, backed by experiments and theoretical analysis
  - Release open-source software
- **http://adslab.cse.cuhk.edu.hk**

# Dependable Storage Stack



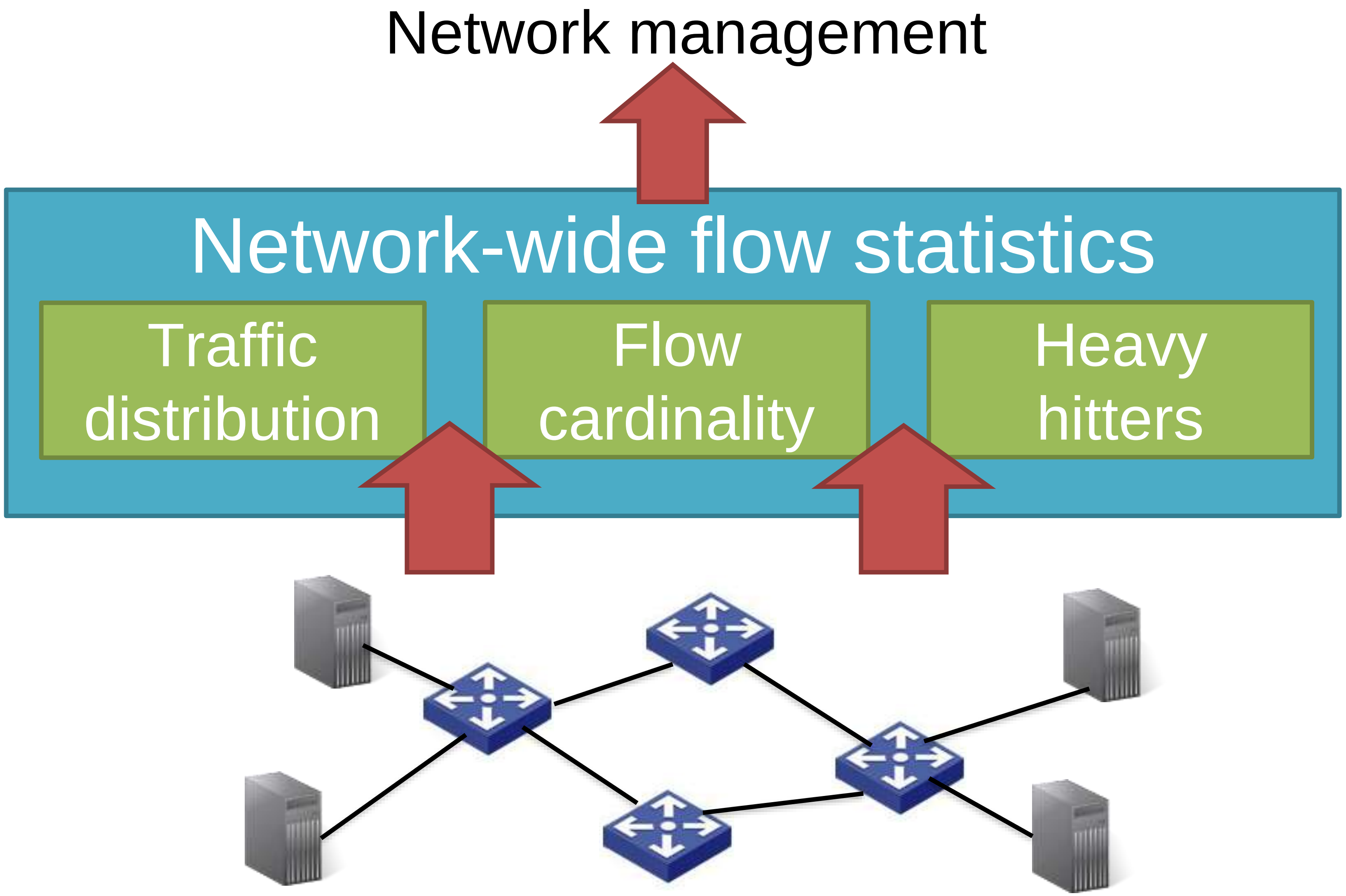
# Network Measurement

- Network measurement is critical for managing networks
  - Billing customers
  - Detecting anomalies
  - Diagnosing and fixing problems




- Difficulties for network measurement
  - Fast line rate
  - Huge volume of traffic
  - Emerging of programmable network elements

# Network Measurement at Scale



# Methodology

- Network measurement is studied in literature for decades
- We aim to address specific challenges, in the face of
  - High-speed and huge-volume traffic in large-scale networks
  - Limited measurement resources on both software and hardware
- Our approach
  - Algorithm design
    - Propose sketch-based algorithms to address the challenges
    - We focus on **invertible sketches**
      - **Invertible**: the measurement results can be readily recovered from only the sketch data structure itself → important for network forensics and distributed measurement
  - Deployment on both software and hardware

# Structure

Network Measurement

Heavy flow detection

**MV-Sketch**

INFOCOM'19
(AR: 288/1464 = 19.7%)
IEEE/ACM Trans. on Networking 2020

Superspreader detection

**SpreadSketch**

INFOCOM'20
(AR: 268/1354 = 19.8%)
IEEE/ACM Trans. on Networking, accepted 2022

# Outline

- **MV-Sketch: Heavy Flow Detection**
- SpreadSketch: Superspreader Detection

# Heavy Flow Detection

- Network traffic: a stream of packets denoted by $(\boldsymbol{x}, \boldsymbol{v_x})$ pairs
  - $x$: flow key, e.g., 5-tuple, source IP address
  - $v_x$: value, e.g., 1 for packet counting, payload bytes for size counting

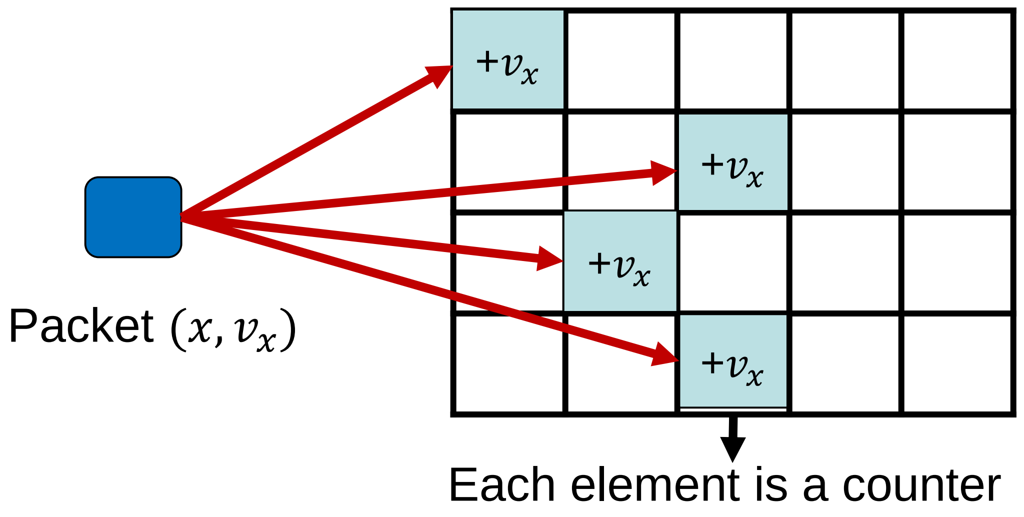
- **Heavy flows** – abnormal patterns in network traffic
  - Heavy hitters: flows with high traffic volume
  - Heavy changers: flows with high change of traffic volume
  - Detecting heavy flows in **real time** is critical for:
    - anomaly detection, load balancing, traffic engineering

# Challenges

- Fast packet processing
  - e.g. 10 Gb/s link: one packet every 67 ns
- Limited memory
  - Programmable switches: 1-2 MB per stage [Bosshart, SIGCOMM'13]
  - Servers: tens of MB of SRAM
  - Per-flow tracking is expensive
    - e.g., millions of concurrent flows per minute for 10 Gb/s link
    - Performance degrades once the working set exceeds the available software cache size

# Sketches

- **Good**:
  - High accuracy with small memory
  - Fast processing speed




- **Bad**: **Non-invertible**
  - Cannot readily return all heavy flows
  - e.g., Count-Min needs to enumerate all possible flows in entire flow key space to recover all heavy flows

# Our Contributions: MV-Sketch

- **MV-Sketch**, a fast and compact **invertible** sketch for heavy flow detection in network data streams
  - Built on **M**ajority **V**oting [Boyer and Moore, 1991]
  - **Small and static** memory usage
  - High processing speed
  - High accuracy
- Theoretical analysis on accuracy, space, and time complexity
- Experiments on real-world network traces
  - Higher accuracy; up to 3.38× throughput gain over state-of-the-arts
  - Line-rate measurement with limited resource overhead on hardware

# Problem Formulation

- Perform detection at regular time intervals called **epochs**
- Input: packet stream $(x, v_x)$
- **Heavy hitter:** all $x$ with $S(x) > \varphi$
  - $S(x)$: total traffic volume of flow $x$ in one epoch
  - $\varphi$: user-specified threshold
- **Heavy changer**: all $x$ with $D(x) > \varphi$
  - $D(x)$: total traffic change of flow $x$ across two epochs
- **Problem**: infer $S(x)$ and $D(x)$ in real-time with limited memory

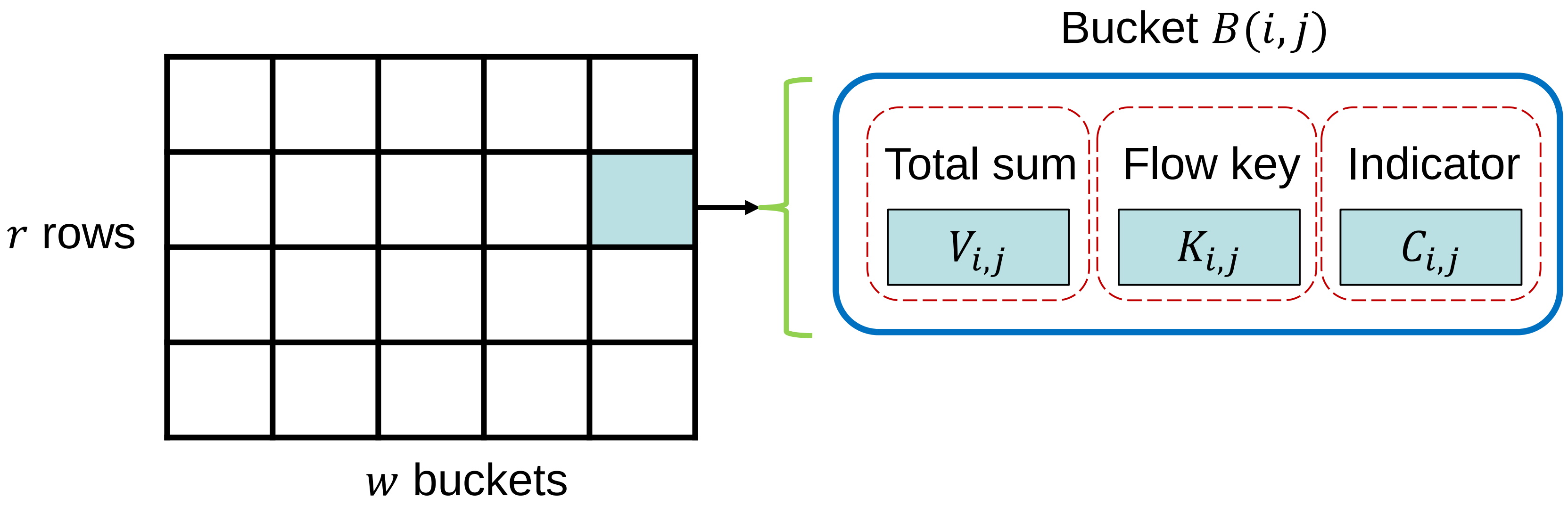
# Design

- **Key observation**: small number of large flows dominate
  - e.g., 9% of flows account for 90% of traffic [Fang, 1999]

- **Idea**:
  - A heavy flow has more traffic than all other flows in the same bucket with high probability
  - Track a candidate heavy flow in each bucket via **majority voting (MJRTY)** [Boyer and Moore, 1991]
  - **Theorem**: MJRTY ensures that the true majority vote (with over half of total vote counts) must be tracked

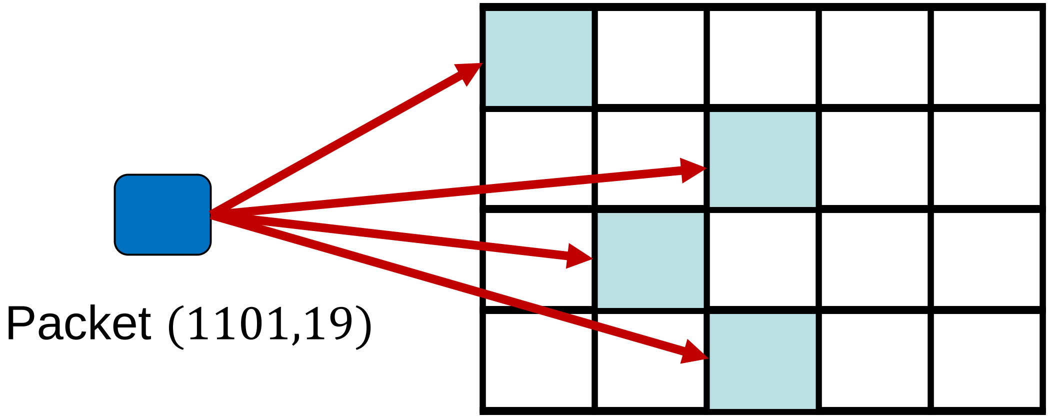
# Design

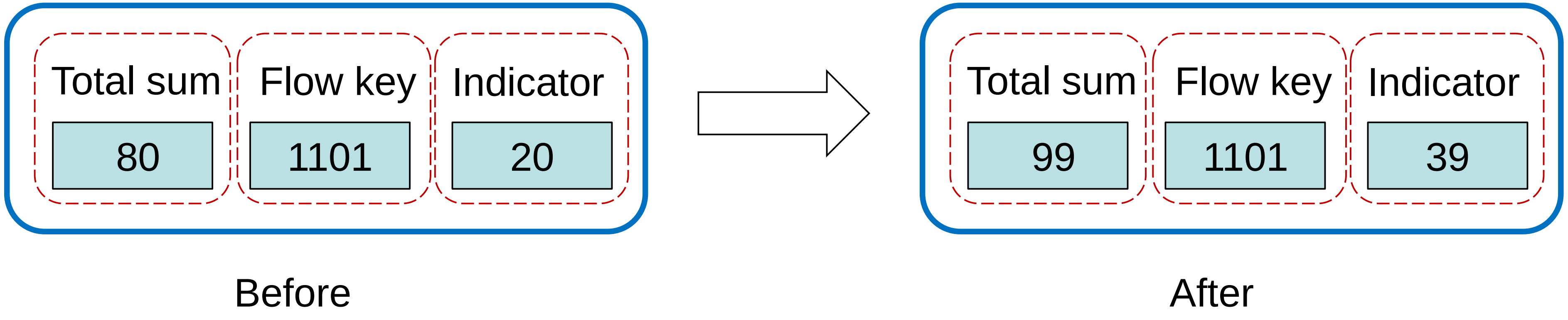
- **Data structure**: $r \times w$ table of buckets



# Update

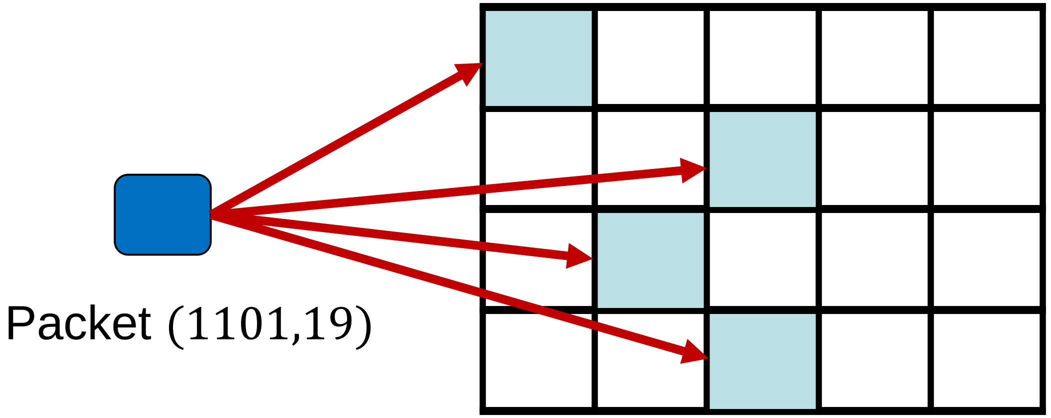
- Insert a packet $(x, v_x)$ into the sketch
  - Map $x$ to one bucket per row
  - Increment $V$ with $v_x$
  - Compare $x$ with $K$
    - **Case1**: $K = x$, increment $C$ with $v_x$

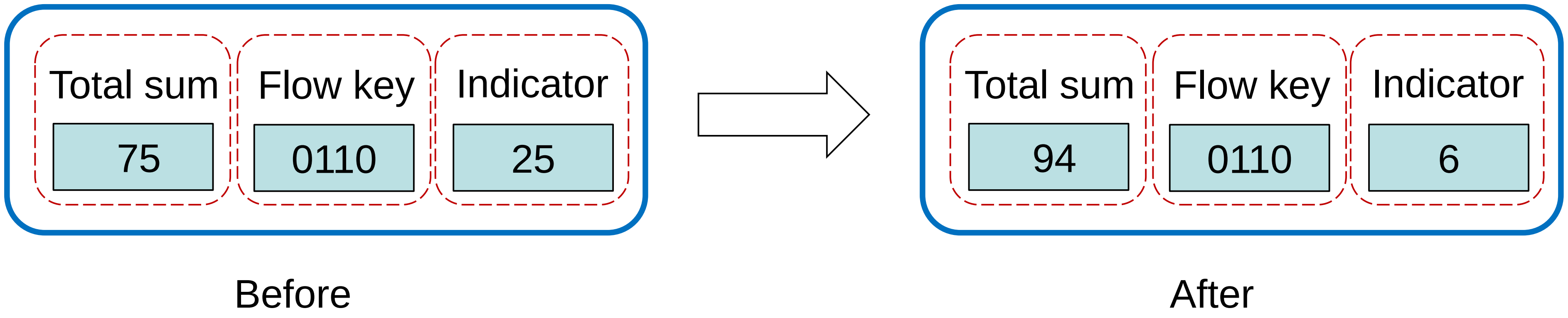


# Update

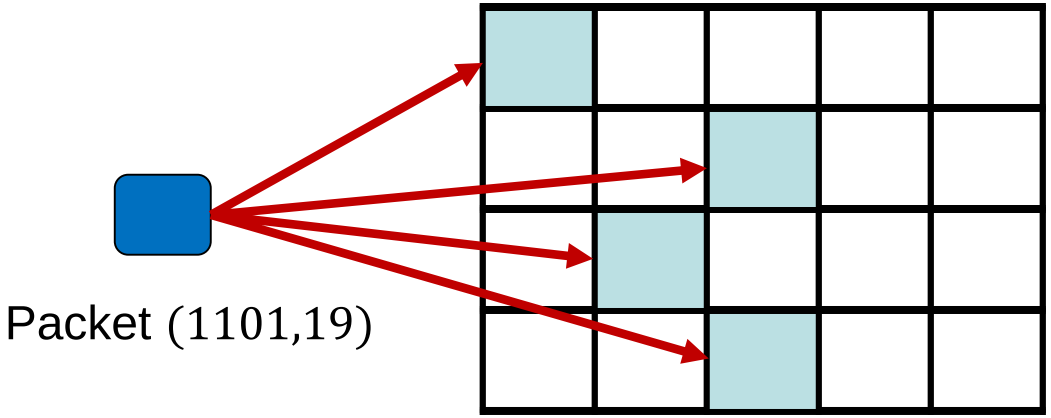
- Insert a packet $(x, v_x)$ into the sketch
  - Map $x$ to one bucket per row
  - Increment $V$ with $v_x$
  - Compare $x$ with $K$
    - **Case1**: $K = x$, increment $C$ with $v_x$
    - **Case2**: $K \neq x$, decrement $C$ with $v_x$

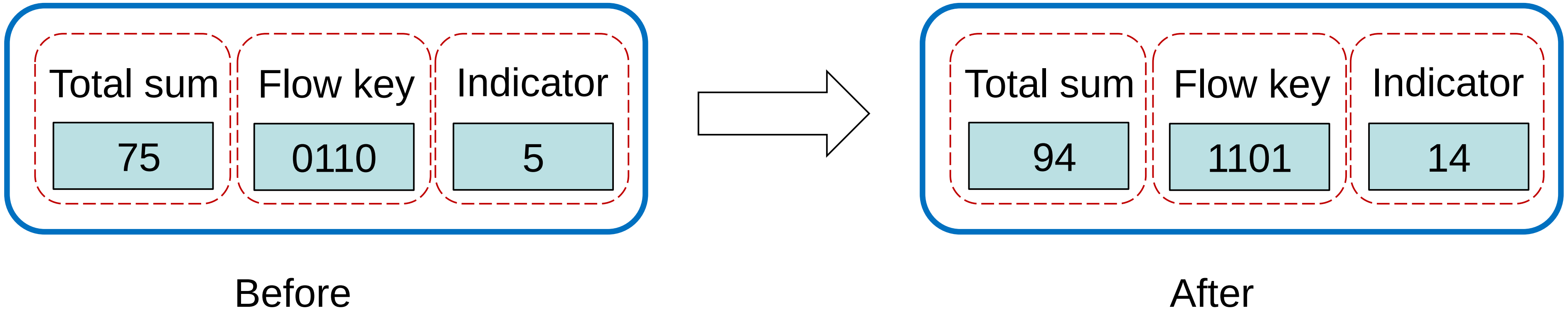


# Update

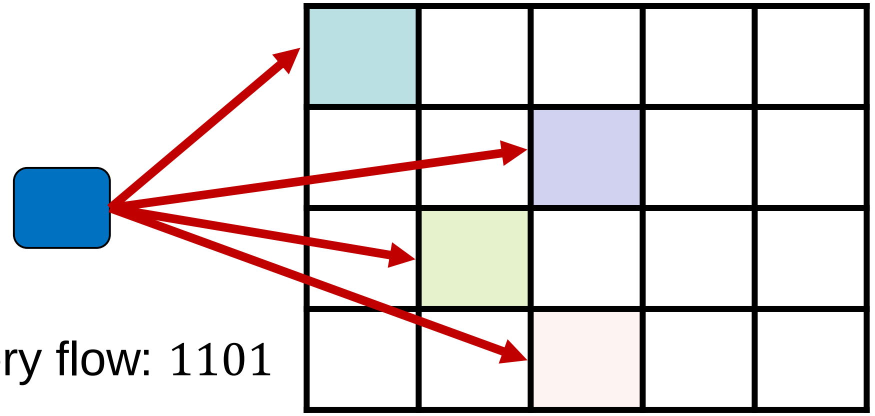
- Insert a packet $(x, v_x)$ into the sketch
  - Map $x$ to one bucket per row
  - Increment $V$ with $v_x$
  - Compare $x$ with $K$
    - **Case1**: $K = x$, increment $C$ with $v_x$
    - **Case2**: $K \neq x$, decrement $C$ with $v_x$
      - **if $C < 0$, copy $x$ to $K$, $C = abs(C)$**



# Query

- Returns the estimated sum of a given flow



Query flow: 1101

| Total sum | Flow key | Indicator |
|---|---|---|
| 90 | 1101 | 10 |
| 120 | 0011 | 6 |
| 100 | 1101 | 2 |
| 210 | 0101 | 90 |

$est_1 = \frac{90+10}{2} = 50$

$est_2 = \frac{120-6}{2} = 57$

$est_3 = \frac{100+2}{2} = 51$

$est_4 = \frac{210-90}{2} = 60$

$$est(1101) = \mathbf{Min}\ (50, 57, 51, 60) = 50$$

# Identify Heavy Hitters

- **Idea**: consider keys tracked by buckets
  - Enumerate all buckets

$$\text{Bucket } B(i,j) \xrightarrow{V_{i,j} > threshold\,?} \text{Get the sum estimation of } K_{i,j}$$

  - Report $K_{i,j}$ as a heavy hitter if its estimation exceeds threshold

# Identify Heavy Changers

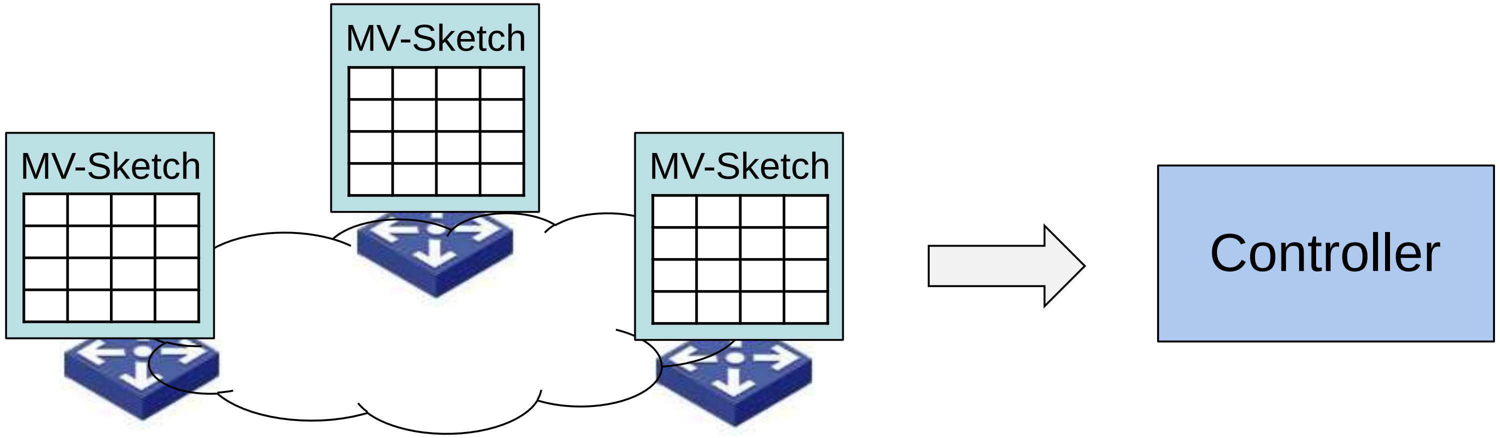
- **Idea**: use estimated maximum change
- Get upper and lower bounds of $x$ in one sketch:
  - Upper bound $\mathrm{U}(x)$ : estimated sum of $x$
  - Lower bound $L(x)$: check each hashed bucket $B(i, j)$
    - $L_{i,j}(x) = \begin{cases} C_{i,j}, & if\ K_{i,j} = x \\ 0\ , & if\ K_{i,j} \neq x \end{cases}$, $L(x) = \mathrm{Max}\{L_{i,j}(x)\}$
- Estimate maximum change: $\widehat{D}(x) = \max\{|U^1(x) - L^2(x)|, |L^1(x) - U^2(x)|\}$

MV-Sketch at 1st epoch → $U^1(x)$ and $L^1(x)$

MV-Sketch at 2nd epoch → $U^2(x)$ and $L^2(x)$

# Extension

- **Architecture**: $q > 1$ monitoring nodes and a centralized controller




- MV-Sketch supports both scalable and network-wide detection
  - Scalable detection: improve the performance and scalability by performing heavy flow detection on multiple packet streams in parallel
  - Network-wide detection: provide an accurate network-wide measurement view as if all traffic were measured in one big detector

# Theoretical Analysis

## ➢ On accuracy

- Bounded estimate errors
- Small false negative rate (almost zero false negatives in our evaluation)

## ➢ On complexity

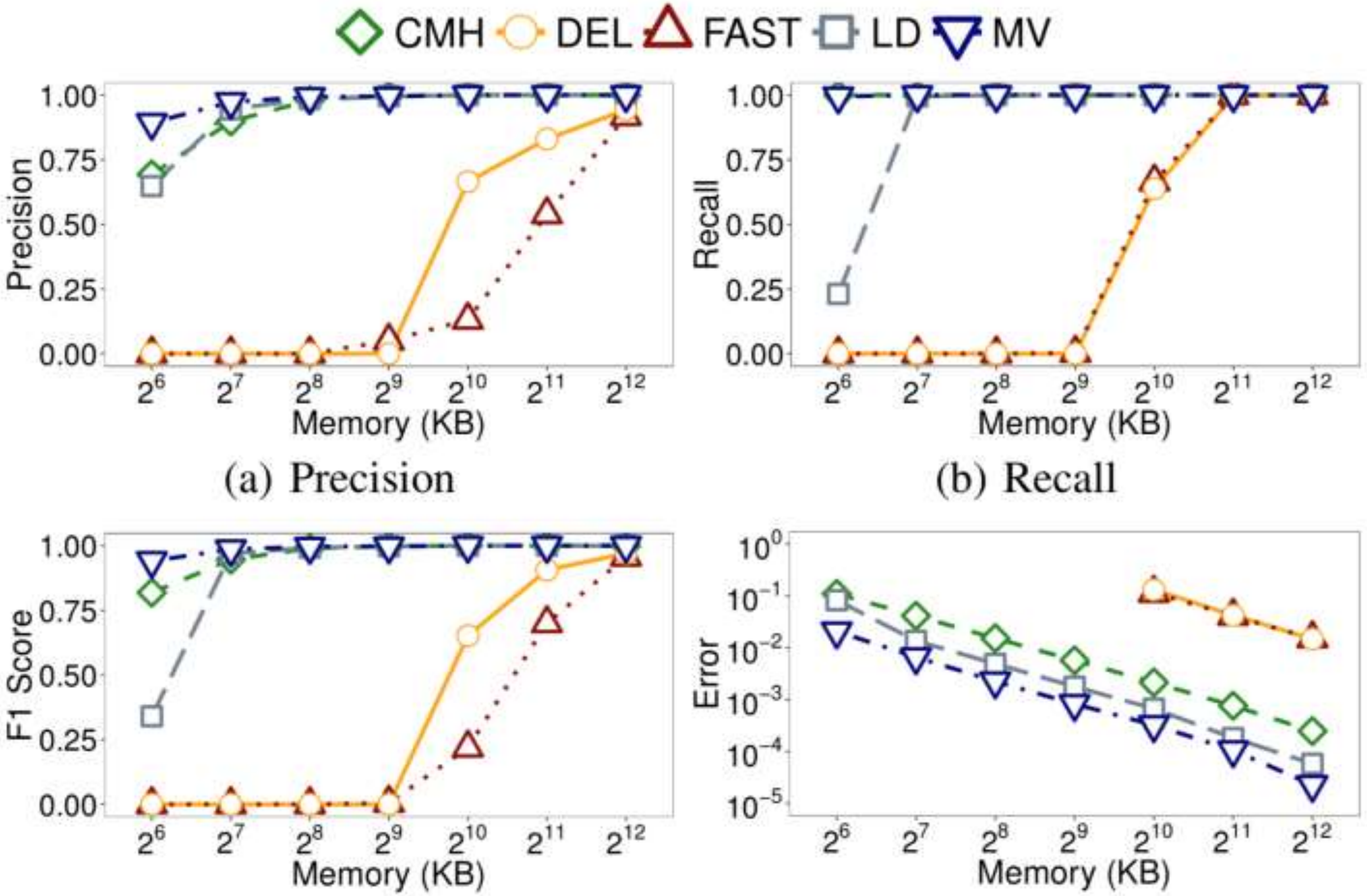
- Let $r = log\frac{1}{\delta}$, $w = \frac{2}{\varepsilon}$, $r$ and $w$ are numbers of rows and columns resp.
- Space complexity: $\mathrm{O}\,(\mathrm{r} \times w \times \log n)$, $n$ is flow key space
- Per-packet update time complexity: $\mathrm{O}(r)$
    - Better than existing invertible sketches

# Software Evaluation Setup

- Traces:
  - CAIDA16 1-hour trace, we focus on the first five minutes
  - Epoch length: 1 minute
  - Each epoch: ~29M packets, ~1M flows on average
- Approach:
  - Compare with Count-Min-Heap, LD-Sketch, Deltoid, Fast Sketch
  - Flow key: 64 bit (source/destination address pairs)
- Metric:
  - **Accuracy**: precision, recall, relative error
  - **Speed**: throughput (pkts/s)

# Evaluation on Software - Accuracy

- **Heavy hitter detection**



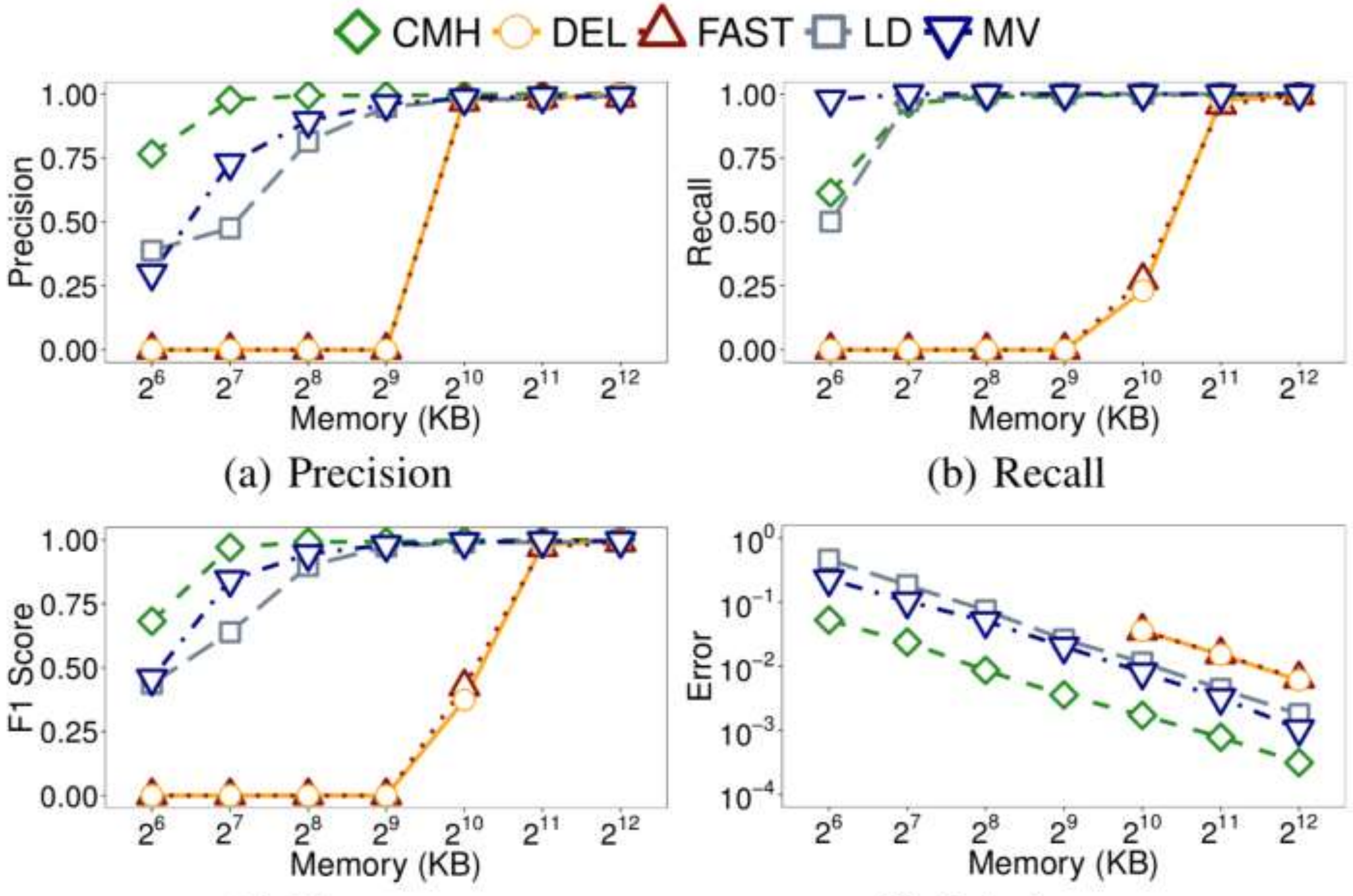

(a) Precision (b) Recall

(c) F1 score (d) Relative error

- MV-Sketch has highest accuracy
- Reduce relative error by 55% and 87% over LD-Sketch and Count-Min-Heap, resp.

# Evaluation on Software - Accuracy

- **Heavy changer detection**

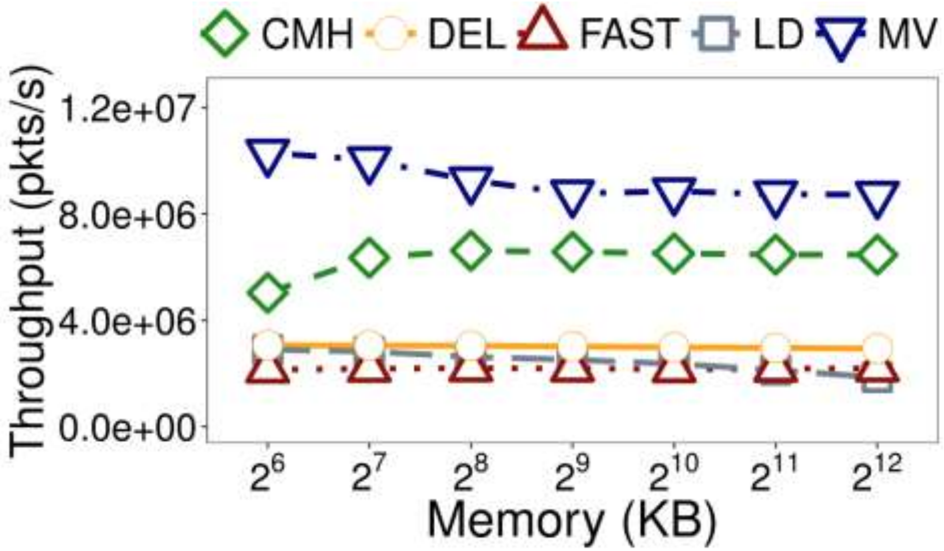


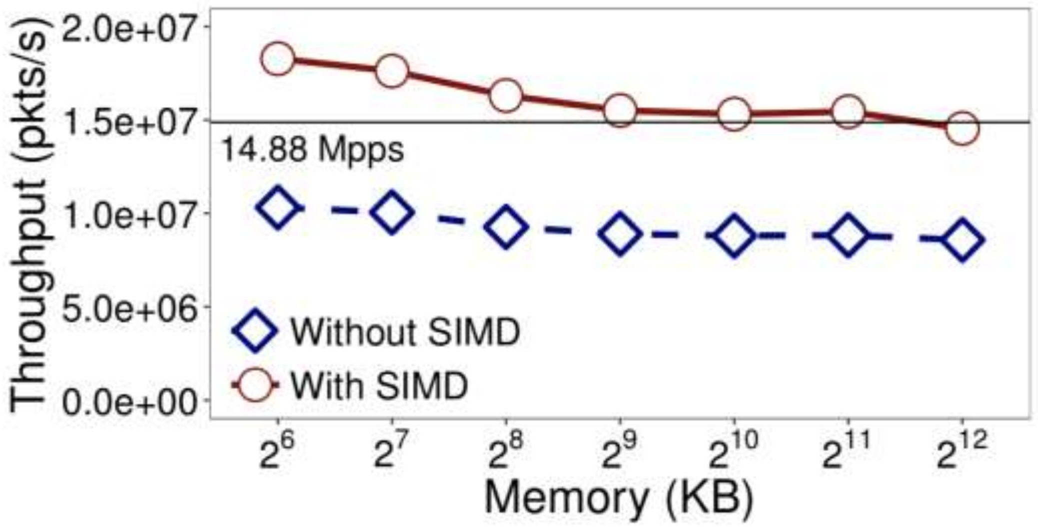

(a) Precision (b) Recall

(c) F1 score (d) Relative error

- MV-Sketch has recall 1 in all cases except 64KB
- MV-Sketch has lower precision than CMH for small memory

# Evaluation on Software - Speed




- MV-Sketch achieves up to 3.38X throughput gain
- With SIMD, MV-Sketch's throughput further improves by 75%

# Hardware Evaluation Setup

- Testbed:
  - Two servers that each has two 12-core 2.2GHz CPUs, 32 GB RAM, and a 40 Gbps NIC
  - A Barefoot Tofino switch with 32 100 Gb ports
- Approach:
  - Compare with PRECISION, which is designed for heavy hitter detection in programmable switches
- Metric:
  - **Resource usage**: memory and computation resource usage
  - **Speed**: throughput (pkts/s)

# Evaluation on Hardware

➢ Resource usage

Switch resource usage (percentages in brackets are fractions of total resource usage)

| | MVFULL | MVSC | MVPC | PRECISION |
|---|---|---|---|---|
| SRAM (KiB) | 144 (0.94%) | 80 (0.52%) | 80 (0.52%) | 192 (1.25%) |
| No. stages | 4 (33.33%) | 2 (16.67%) | 1 (1.33%) | 8 (66.67%) |
| No. actions | 10 | 5 | 3 | 15 |
| No. ALUs | 3 (6.25%) | 2 (4.17%) | 2 (4.17%) | 6 (12.5%) |
| PHV (bytes) | 133 (17.32%) | 108 (14.06%) | 102 (13.28%) | 137 (17.84%) |

- All the MV-Sketch implementations achieve less resource usage

➢ Throughput

- Both MV-Sketch and PRECISION achieve line-rate measurement

# Summary

- MV-Sketch, an invertible sketch that enables fast and accurate heavy flow detection in network data streams
- Contributions:
  - Propose a new sketch design for invertible sketches
    - High accuracy with small and static memory
    - Fast processing speed
  - Extensions to distributed heavy flow detection
  - Extensive experiments on real-world traces
- Source code:
  - **http://adslab.cse.cuhk.edu.hk/software/mvsketch**

# Outline

- **MV-Sketch: Heavy Flow Detection**
- SpreadSketch: Superspreader Detection

# Superspreader Detection

- Network traffic: a stream of packets denoted by $(x, y)$ pairs
  - $x$: one or more source fields in the packet header
    - e.g. $x$ = $(SrcIP)$ or $(SrcIP, SrcPort)$
  - $y$: one or more destination fields in the header
- **Fan-out** of $x$: $S(x) = \#(\text{distinct } y)$ $x$ connects to
- **Superspreaders**: sources with large fan-outs
  - Same definition applies to destinations
- Detecting superspreaders in real time is critical to find
  - DDoS attacks, port scanning, hot-spots

# Our Contributions: SpreadSketch

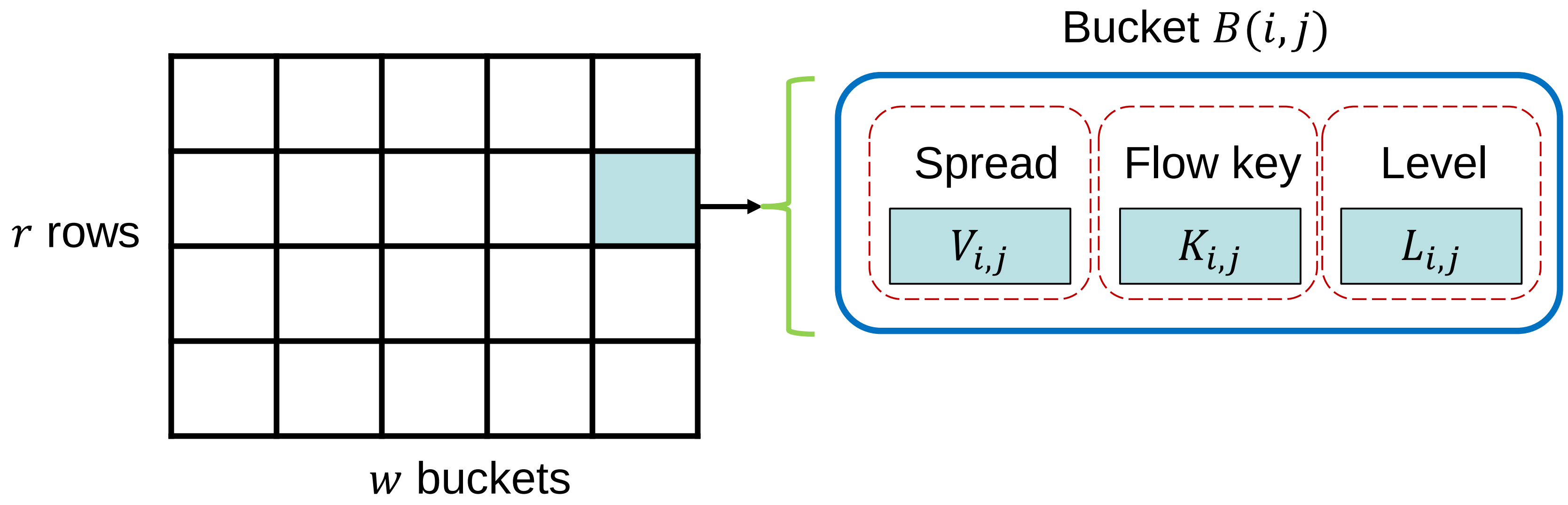
- **SpreadSketch**, a fast and invertible sketch for network-wide superspreader detection in network data streams
  - **Fast** and **invertible**
    - High processing speed, fast recovery of superspreaders
  - **Compact**: small and static memory usage
  - **Network-wide**: network-wide view of superspreaders
- Theoretical analysis on accuracy, space, and time complexity
- Extensive experiments on real-world network traces
  - Higher accuracy and performance over state-of-the-art sketches
  - Feasibility on a Barefoot Tofino switch with resource efficiency

# Design – Main Idea

- Track the source in each bucket that dominates the bucket's spread
- Find the source with highest spread by tracking highest level value
- Replace integer counters in sketch with distinct counters
    - Enable distinct counting in sketch using multiresolution bitmap [Estan, IMC'03]
    - Apply bitwise-AND operation across bitmaps for count estimation

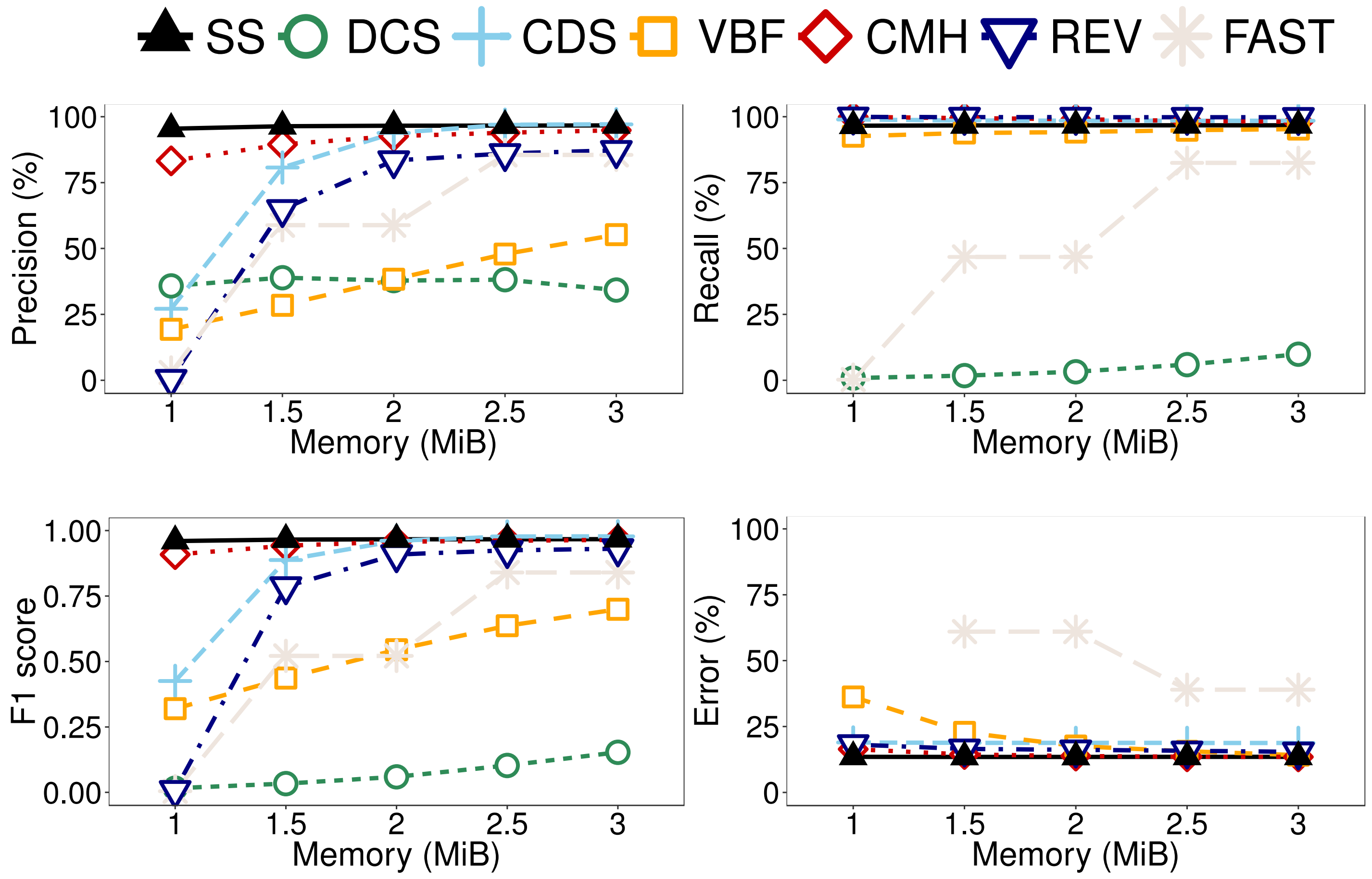
# Design – Data Structure

➢ **Data structure**: $r \times w$ table of buckets



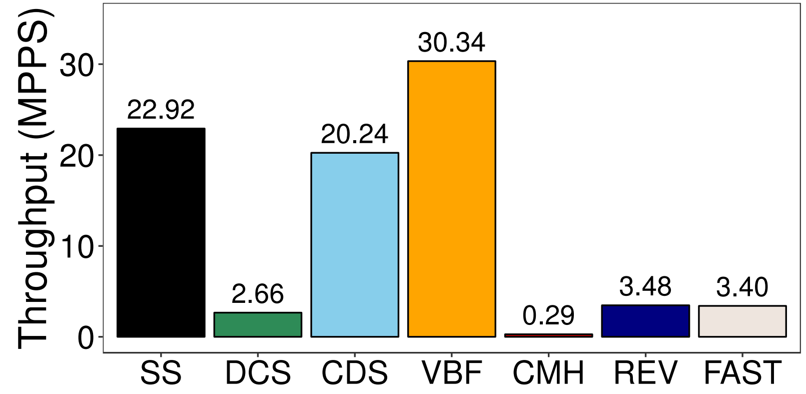

- $V_{i,j}$ : Distinct counter to track the total spread in the bucket
- $K_{i,j}$ : the candidate source key with the highest level in the bucket
- $L_{i,j}$, the maximum level seen in the bucket

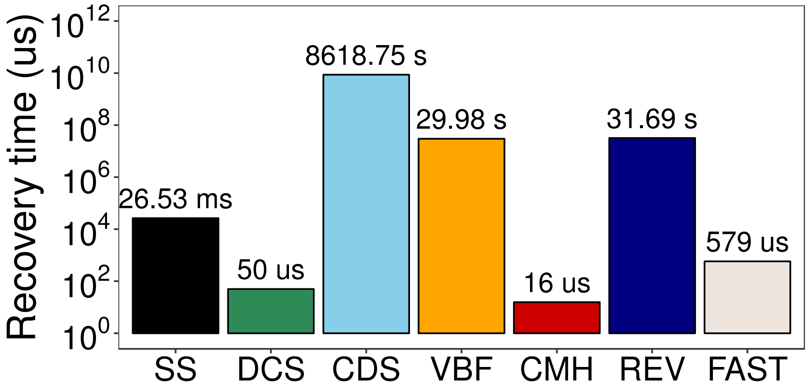
# Result – Accuracy CAIDA19




- SpreadSketch is more robust and accurate compared with state-of-the-art sketches
- Similar observations on CAIDA18 and CAIDA16 traces

# Result – Speed




- SpreadSketch (SS) achieves throughput more than 22 MPPS
  - it is easily catch up with10 Gbs line speed

- SpreadSketch recovers superspreaders within few milliseconds

Overall, SpreadSketch achieves both high update and recovery speed

# Summary

- ➢ SpreadSketch, an invertible sketch that enables fast and accurate network-wide superspreader detections in network data streams
- ➢ Contributions:
  - Propose a new invertible sketch design to detect superspreaders
    - High accuracy and robust on real-word traces
    - Fast processing and recovery speed
    - Feasibility on commodity hardware switches
  - Detailed theoretical analysis on both accuracy and complexities
  - Extensive experiments on real-world traces
- ➢ Source code: http://adslab.cse.cuhk.edu.hk/software/spreadsketch/

香港中文大學
THE CHINESE UNIVERSITY OF HONG KONG

# Thank you!
# &
# Questions?