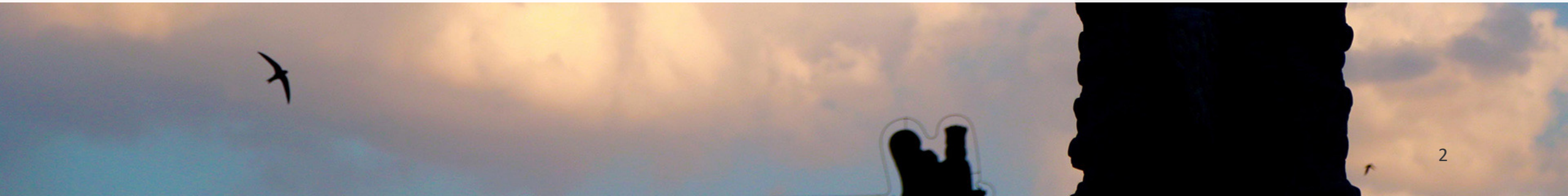
SketchPolymer: Estimate Per-item Tail Quantile Using One Sketch

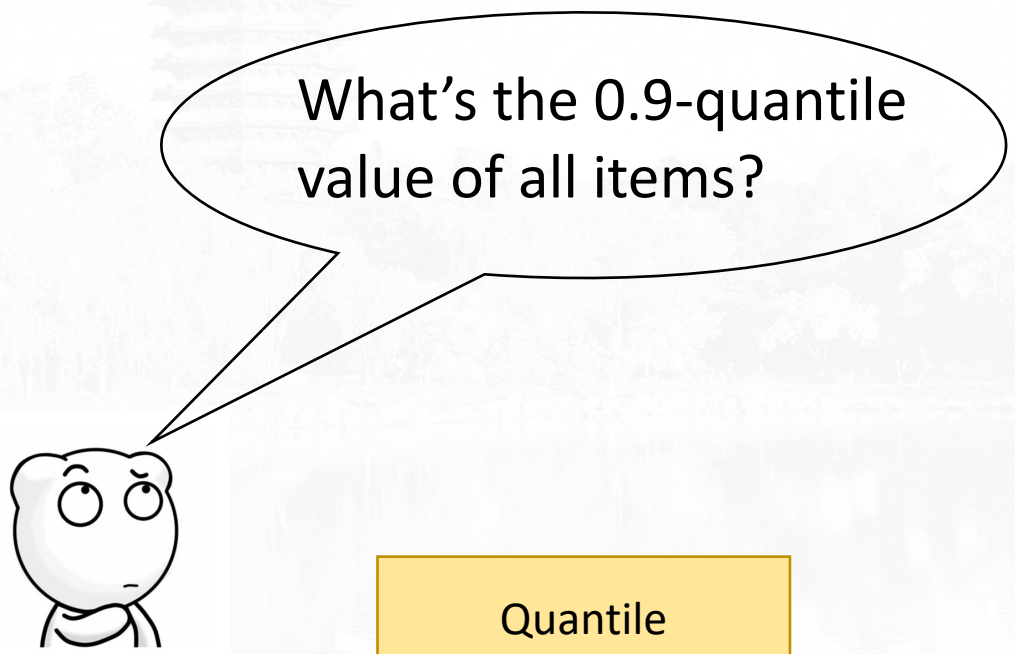
Jiarui Guo, Yisen Hong, Yuhan Wu,

Yunfei Liu, Tong Yang, Bin Cui

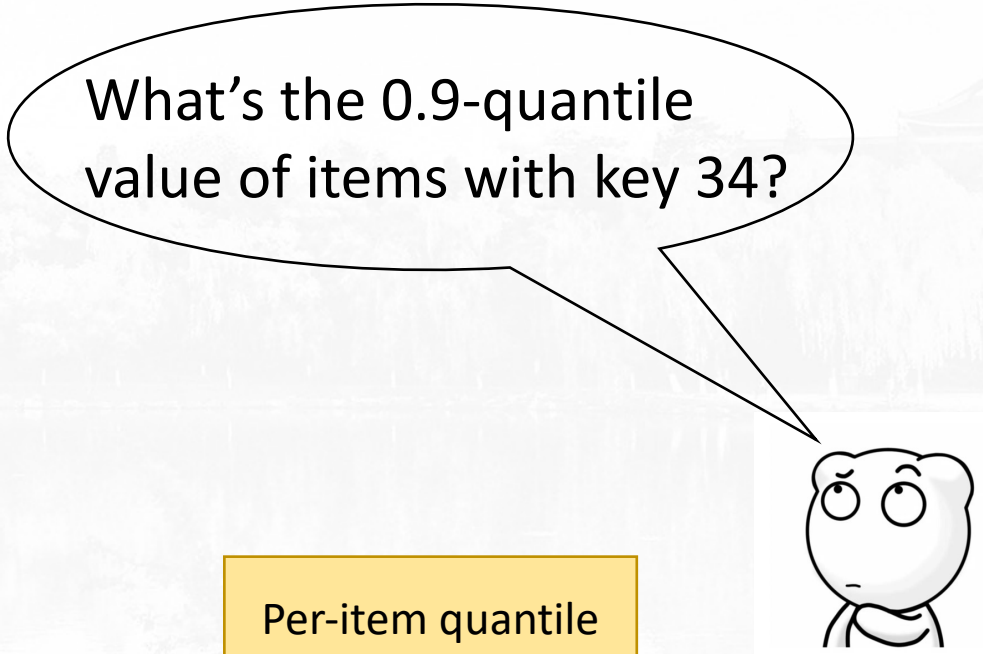
01 Part One Background



Motivation



Quantile



Per-item quantile

Motivation

- Quantile estimation algorithms
  - GK, KLL, t-digest, DDSketch……
- Per-item quantile estimation algorithms
  - ???

Motivation

- Application: Per-flow tail latency in network scenarios
  - network management
  - attack detection

Solution: SketchPolymer

- Approximate stream algorithms can solve this problem
  - Small: placed on cache
  - Fast: updated within O(1) time complexity
  - Accurate: 32.67 times better than state-of-the-arts

02 Part Two Algorithm

SketchPolymer Algorithm

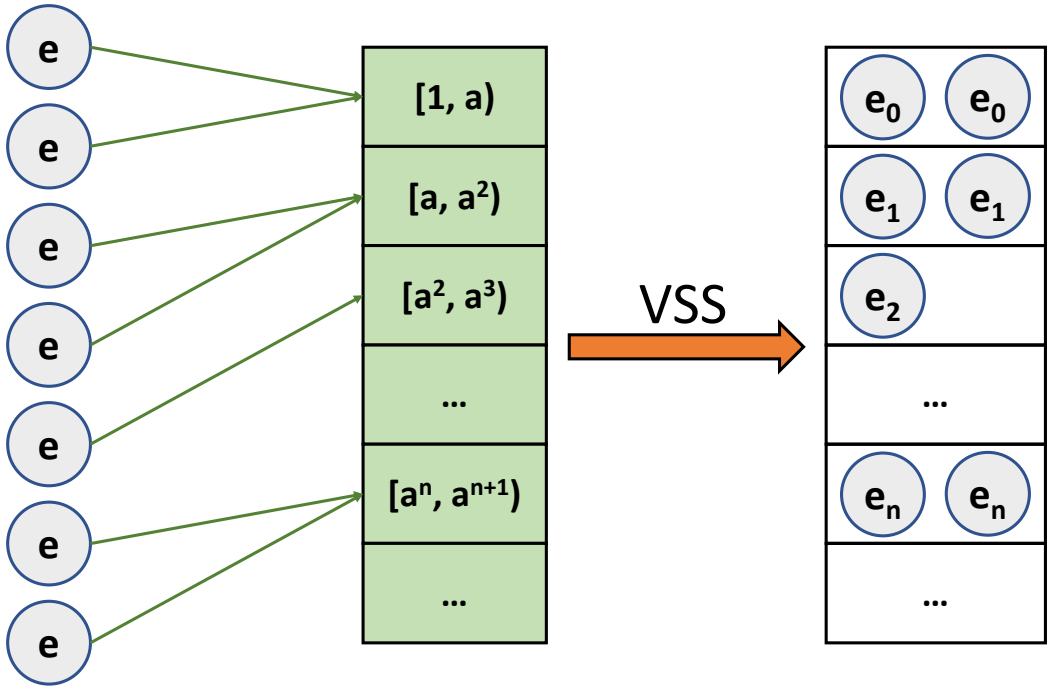
- Polymer: substance made from combinations of small simple molecules
  - SketchPolymer: Polymer of sketches
- Data structure: 4 stages
  - Filter Stage, Polymer Stage, Splitting Stage, Verification Filter
- Key techniques:
  - Early Filtration
  - Value Splitting and Sharing (VSS)

Early Filtration

- Quantile estimation for infrequent items:
  - Hard & inaccurate
  - Solution: filter infrequent items
- SketchPolymer uses Filter Stage to filter infrequent items
- Filter Stage:
  - Query Filter Stage before insertion
  - Items with frequency exceeding the threshold enter the following stage

VSS

- Split all positive numbers by logarithm
- Quantile estimation ⇒ frequency estimation



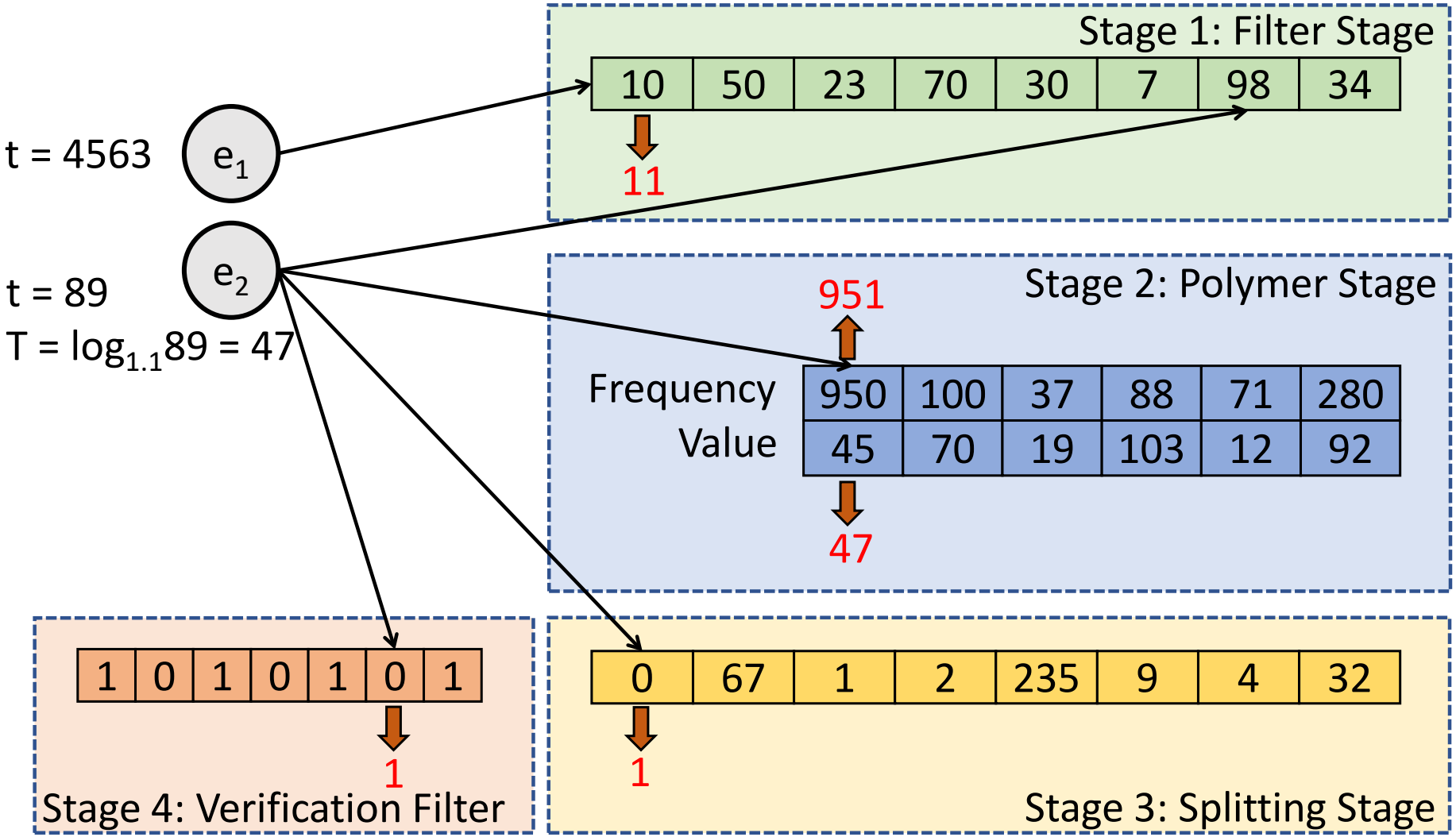
SketchPolymer Operations

- Polymer Stage and Splitting Stage: based on CMSketch
  - Polymer Stage records frequency & max log value
  - Splitting Stage records frequency after VSS

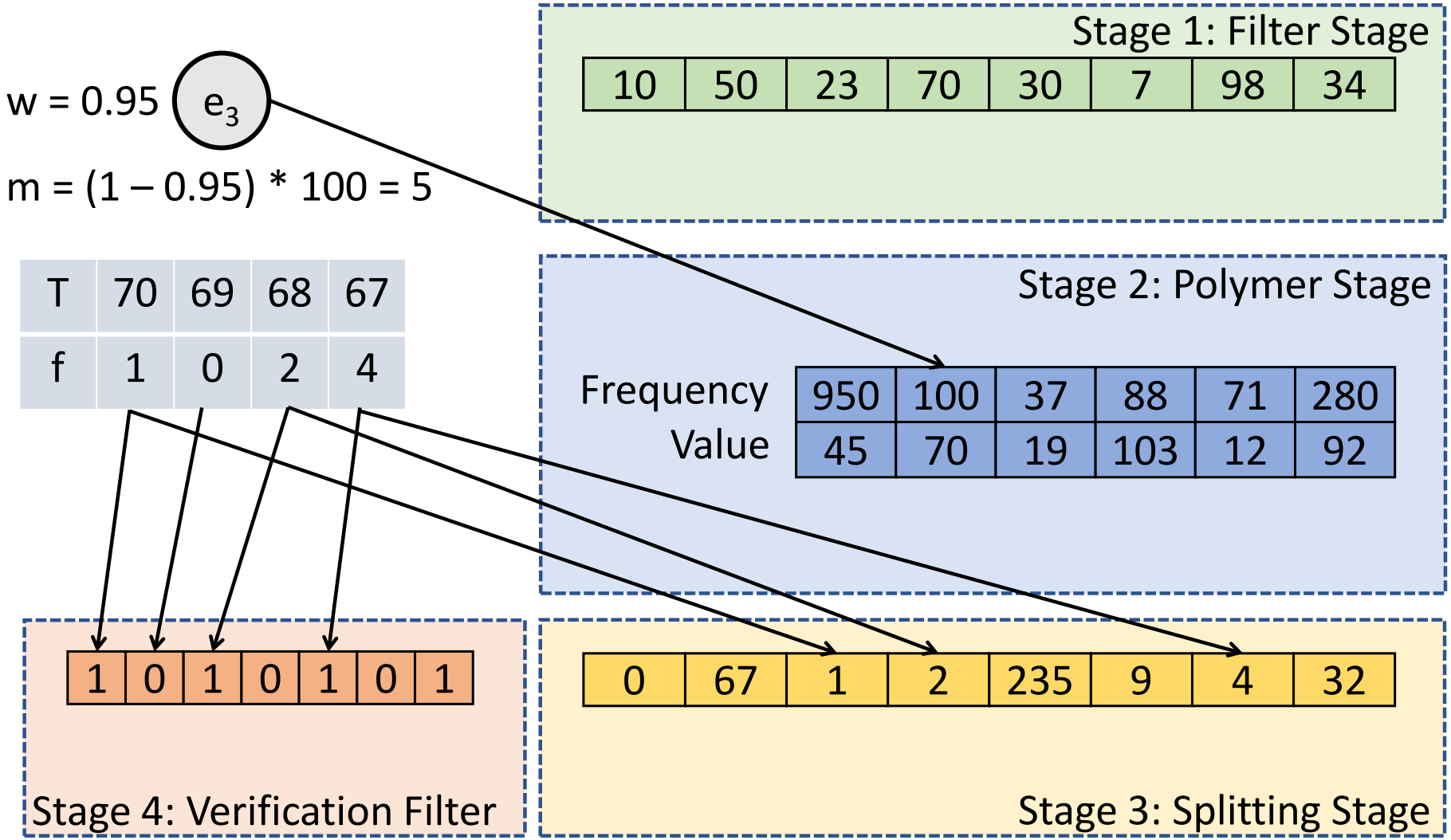
Two Optimizations

- Memory Optimization: Counter Truncation
  - Using 8-bit counters in Splitting Stage
- Accuracy Optimization: Overestimation Avoidance
  - Using Verification Filter

SketchPolymer Algorithm



SketchPolymer Algorithm



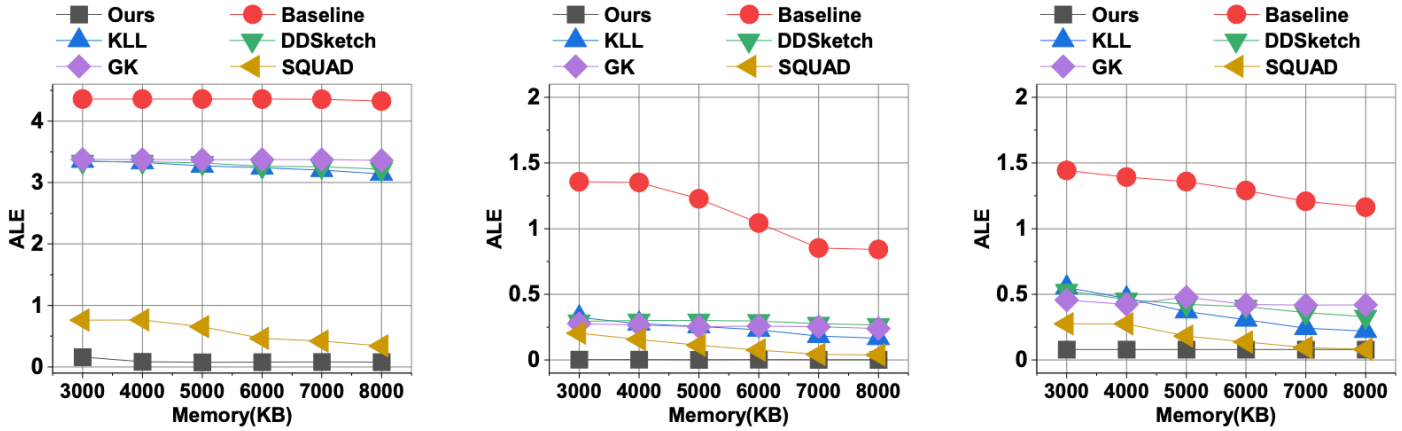
03 Part Three Mathematics

Theoretical Analysis

- Error bound
  - If the Polymer Stage and the Splitting Stage both use d hash functions, then the error of SketchPolymer is at most ε w.p. $1 - O(\varepsilon^{-d})$.
- Time complexity
  - $O(1)$ insertion time complexity

04 *Part Four* *Experiments*

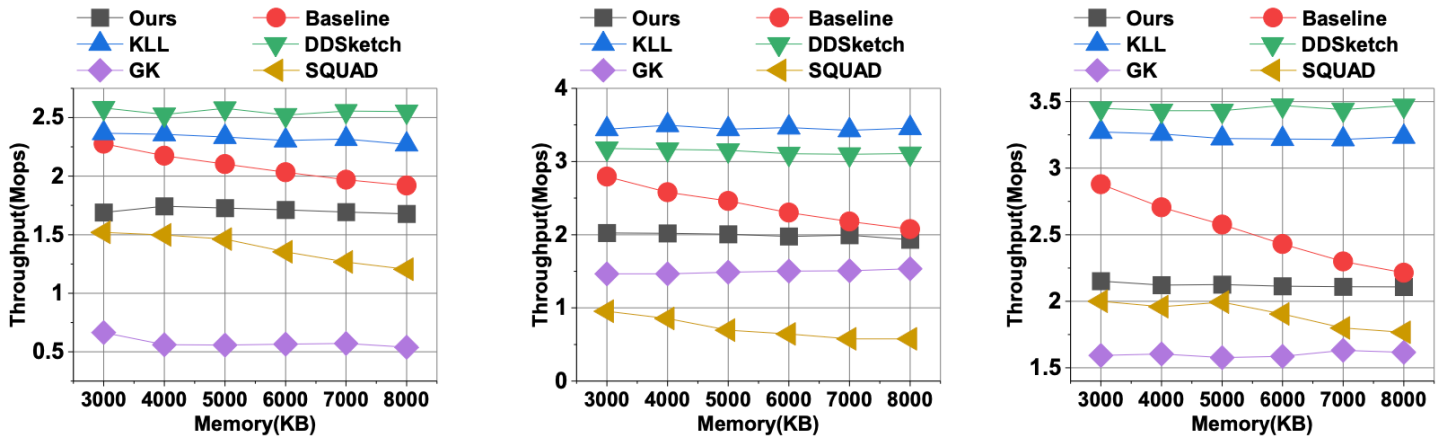
Experimental Results



(a) CAIDA (b) Seattle (c) Web

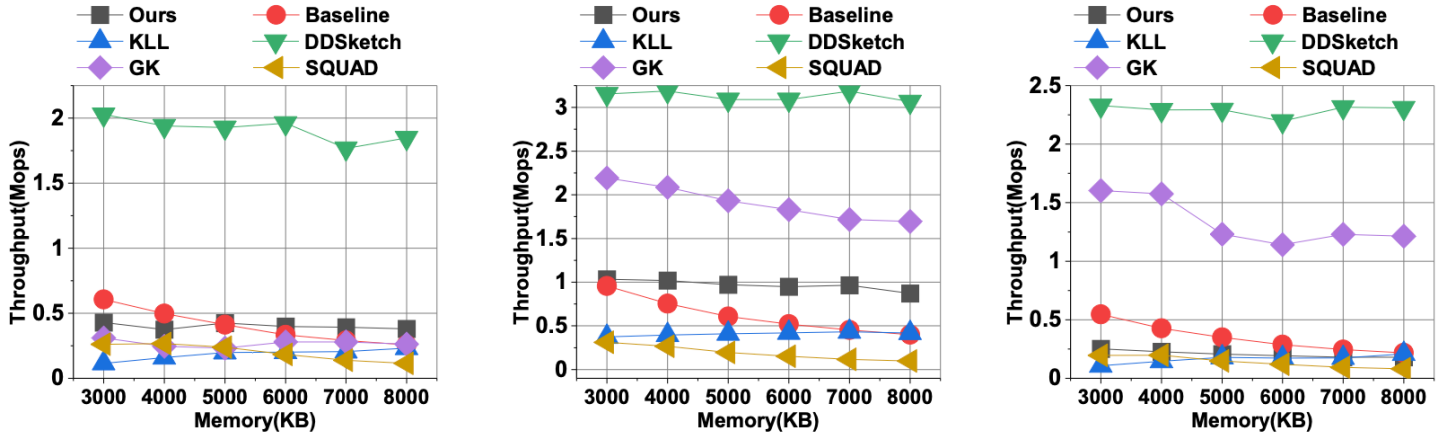
Figure 9: ALE on Different Datasets

Experimental Results



(a) CAIDA (b) Seattle (c) Web

Figure 10: Insertion Throughput on Different Datasets



(a) CAIDA (b) Seattle (c) Web

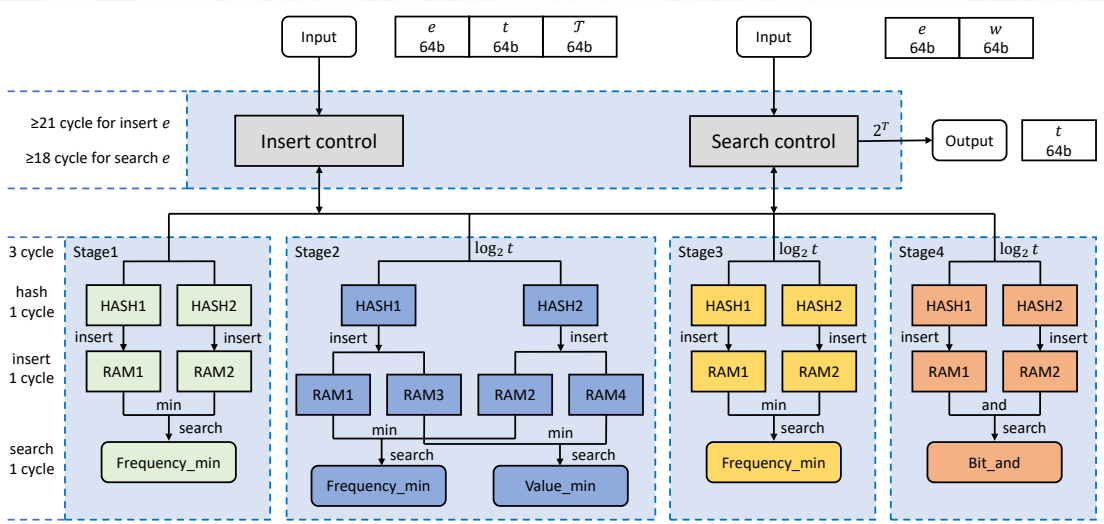
Figure 11: Query Throughput on Different Datasets

SketchPolymer on Hardware Platforms

Table 2: Hardware Resources Used by SketchPolymer

Resource	Usage	Percentage
Hash bits	149	5.97%
Exact Xbar	54	7.03%
Ternary Xbar	8	2.02%
Stateful ALU	5	20.83%
SRAM	19	3.96%
TCAM	2	1.39%
Map RAM	17	5.9%

P4 Implementation



FPGA Implementation

05 Part Five
Summary

Summary

- We design a novel sketch to estimate per-item tail quantile.
- We provide mathematical analysis for SketchPolymer.
- Experimental results show that SketchPolymer outperforms existing algorithms in terms of error and speed.
- We implement SketchPolymer on P4 switches and FPGA.

Thank You!

Source code: https://github.com/SketchPolymer/SketchPolymer-code

Jiarui Guo
Peking University, China
Email: ntguojiarui@pku.edu.cn
Homepage: https://ntguojiarui.github.io/