Zero-CPU Collection with Direct Telemetry Access

Jonatan Langlet  
Queen Mary University of London

Ran Ben-Basat  
University College London

Sivaramakrishnan Ramanathan  
University of Southern California

Gabriele Oliaro  
Harvard University

Michael Mitzenmacher  
Harvard University

Minlan Yu  
Harvard University

Gianni Antichi  
Queen Mary University of London
Telemetry overview

- High load on collection
  - **Millions** of events per switch each second
- Extract even more?
  - Path tracing
  - Congestions
  - Flow sizes

Collection grows very expensive

CPU-based Collection

- Benchmarked two collectors
  - Broken into I/O and Storage cycles
- Packet I/O is not the issue
- The CPU really struggles with telemetry storage

Design goals

- We don’t want the CPU to do storage work
  - No centralized data organization
  - No centralized collision handling

The main point is to allow efficient centralized queryability

- We want it **scalable**, with **low overheads**
  - **Very** low on-switch statefulness.
  - Minimal inter-switch collaboration
  - Fully in the data plane
So what do we do?

Telemetry Data

A packet specifies a location for a value
That is exactly where the value will be written

This is what **RDMA** is designed for!

No centralized pre-processing!

O(100Mops)
Key-value design

All telemetry data has a unique identifier: its Key

- E.g., Per-flow per-switch
  - Path tracing
  - INT postcarding

[flowID & S1] = val

Data

Zero-CPU Collection with Direct Telemetry Access
Global hash functions

Loc: $h(\text{Key})$

RDMA-WRITE

READ

Loc: $h(\text{Key})$
There is a problem!

- Hashes can collide!
  - Overwritten data
- But we want:
  - Near-stateless
  - No inter-switch collaboration
  - Shared global memory
Dedicated memory?

- Key collisions within each switch?
- Stateful solutions not feasible
  - Amount of memory
  - Increased complexity
- Efficient queryability
  - Which switch?
  - Where?
Built-in redundancies

1) Checksumming
2) Plurality voting
Overview

Zero-CPU Collection with Direct Telemetry Access
Overview

Zero-CPU Collection with Direct Telemetry Access
Overview

Zero-CPU Collection with Direct Telemetry Access
Overview

Zero-CPU Collection with Direct Telemetry Access
Overview
Overview

Zero-CPU Collection with Direct Telemetry Access
Overview
Overview

Zero-CPU Collection with Direct Telemetry Access
Overview

Zero-CPU Collection with Direct Telemetry Access
Overview

Zero-CPU Collection with Direct Telemetry Access
Overview

Zero-CPU Collection with Direct Telemetry Access
Future?

- **Generalize** beyond key-value queries
  - E.g., aggregated network states
- How to **query the unknown**?
  - Iterate over data
- **Immediate** controller reactivity?
  - Real-time controller response to events
- **Tailored RDMA** for telemetry?
  - New primitives
    - E.g., multi-write for redundancy
  - Remove the need for a super-reliable network

**Vision:** a transport protocol for telemetry data