BelWü
Landeshochschulnetz Baden-Württemberg

Network Flow Monitoring: An Evaluation of Modern Collector Software

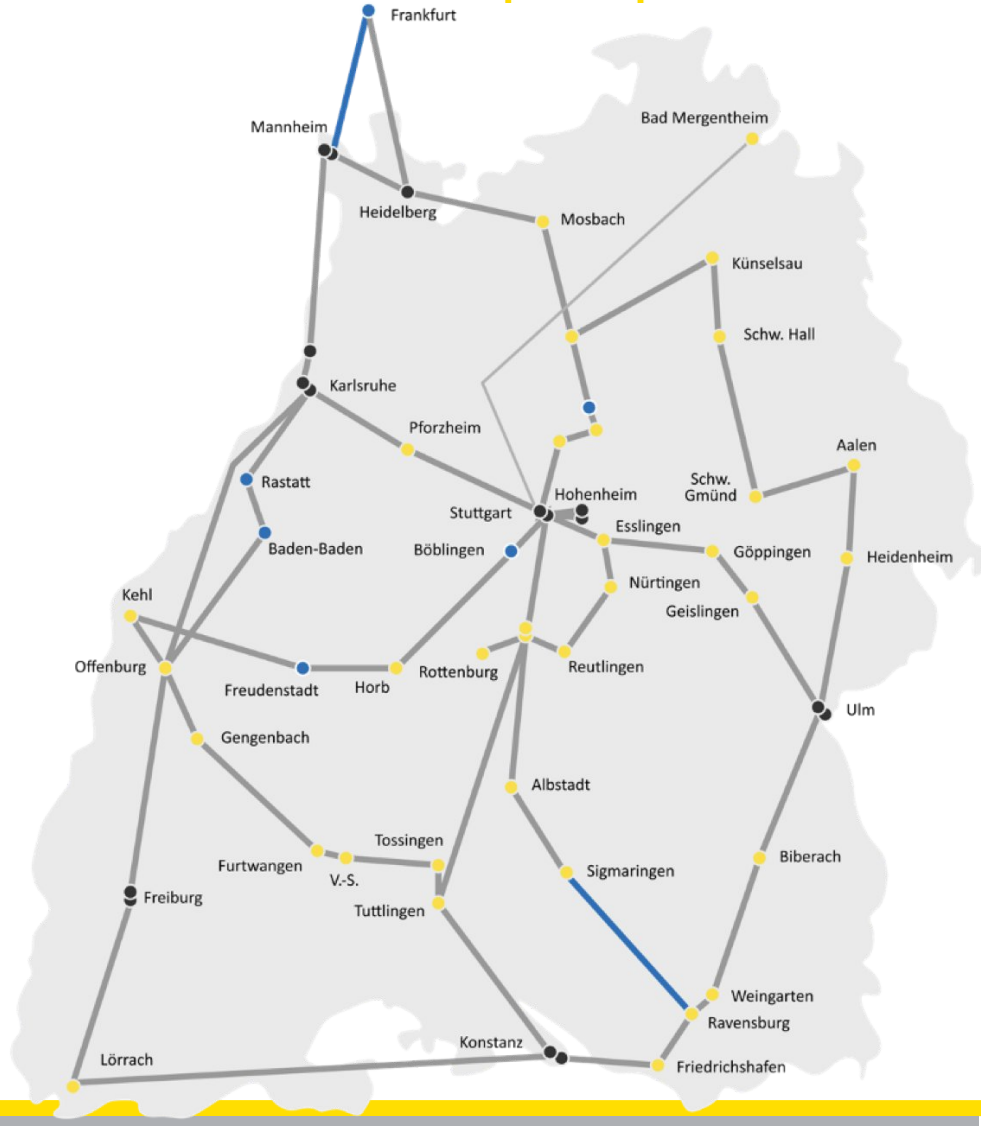
Yannick Huber*, Benjamin Steinert†§, Lukas Pietzschmann‡, Gabriel Paradzik†§, Michael Menth†

* University of Stuttgart, Belwü-Koordination, Stuttgart, Germany
† University of Tübingen, Chair of Communication Networks, Tübingen, Germany
‡ Ulm University, Institute of Distributed Systems, Ulm, Germany
§ University of Tübingen, Zentrum für Datenverarbeitung, Tübingen, Germany

bwNET

Agenda

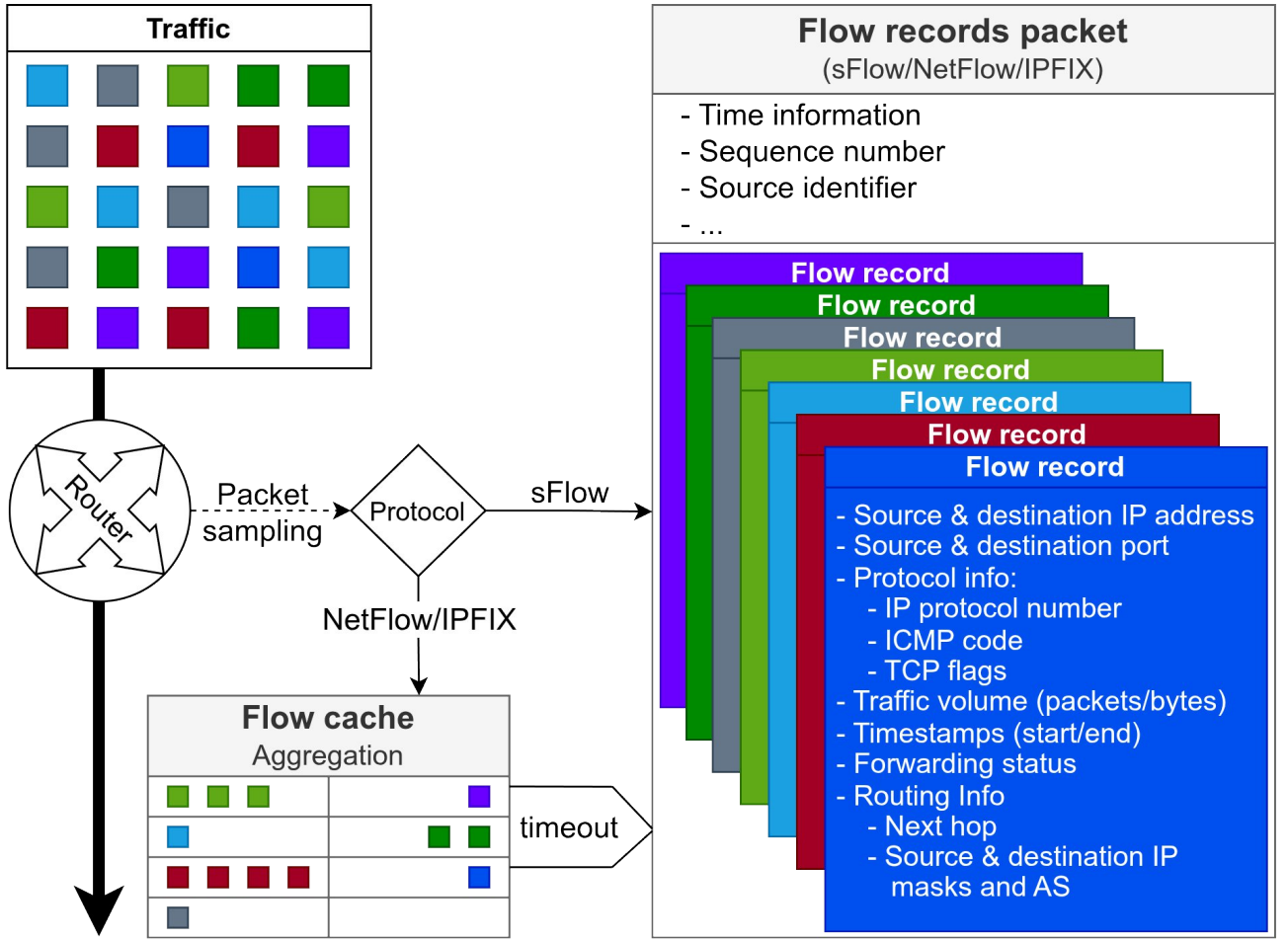
- What is network flow monitoring
- Selected flow collector software
- Qualitative comparison
- Quantitative comparison
- Additional findings



What is Network Flow Monitoring?

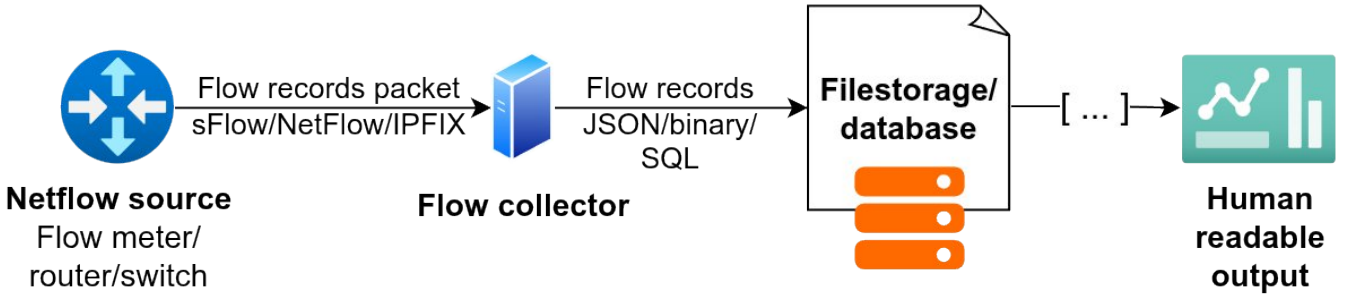
What is network flow monitoring?

- Per-flow data
  - 5-tuple: src/dst IP, src/dst port, protocol
  - Counter, timestamps, metadata
- Use cases
  - traffic analysis, billing, capacity planning, anomaly detection, forensics
- Multiple protocols
  - sFlow: multi-vendor, samples packets, no flow aggregation
  - NetFlow: original Cisco solution
  - IPFIX: IETF standardized flow format, extensible, template-based

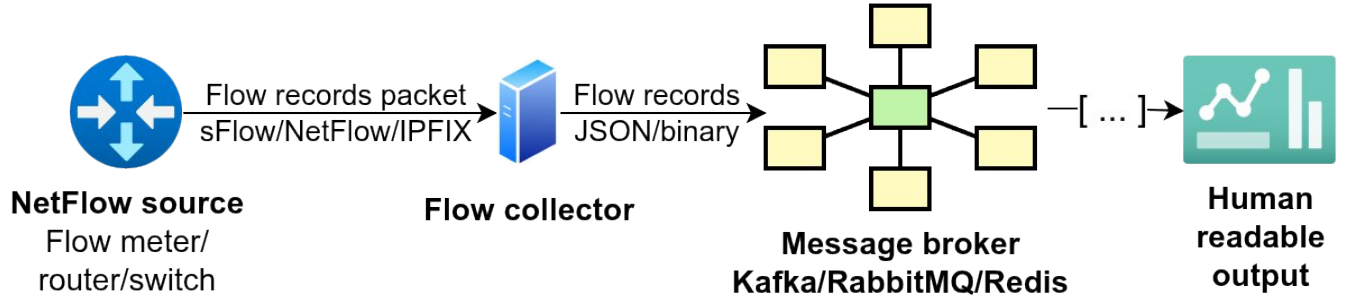


Flow Monitoring Deployments

- Historically export to local file system
  - Flow collector-specific formats
  - Collector ecosystem with analysis tools

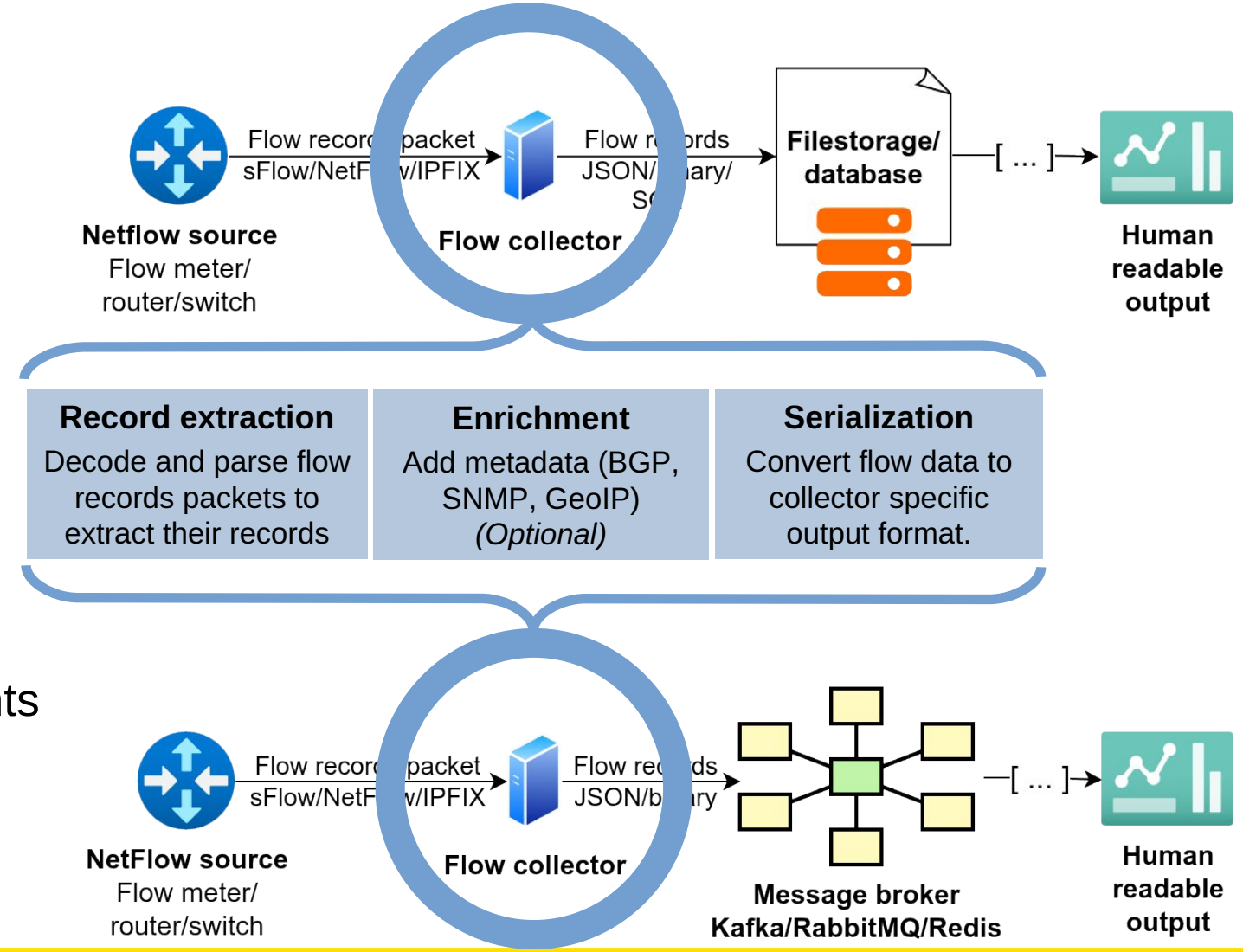


- Export to message broker
  - Allows decentralized deployments
  - Improves scalability



Flow Monitoring Deployments

- Historically export to local file system
  - Flow collector-specific formats
  - Collector ecosystem with analysis tools
- Export to message broker
  - Allows decentralized deployments
  - Improves scalability



Flow Collector Selection

- Selection criteria
  - Self-hosted
  - Open source
  - Maintained
- Evaluation of
  - Features
  - Processing capacity

	nfdump	SiLK	pmacct	IPFIXcol2	GoFlow v2	Akvorado
Initial release	2004	2003	2003	2014 (v1) 2018 (v2)	2018 (v1) 2021 (v2)	2022
License	BSD	GNU GPL 2.0-style	GPL	BSD 3-Clause, GPL 2	BSD 3-Clause	AGPL-3.0
Language	C	C	C	C++	Go	Go
Maintained by	Peter Haag Private person	CERT NetSA Carnegie Mellon University	Paolo Lucente Private person	CESNET Czech NREN	Louis P. Private person	Free French ISP
Focus	Historic data analyzation	Historic data analyzation	Network monitoring toolkit	Modular plugin-based flow processing	Flow processing pipeline entry point	All-in-one pipeline

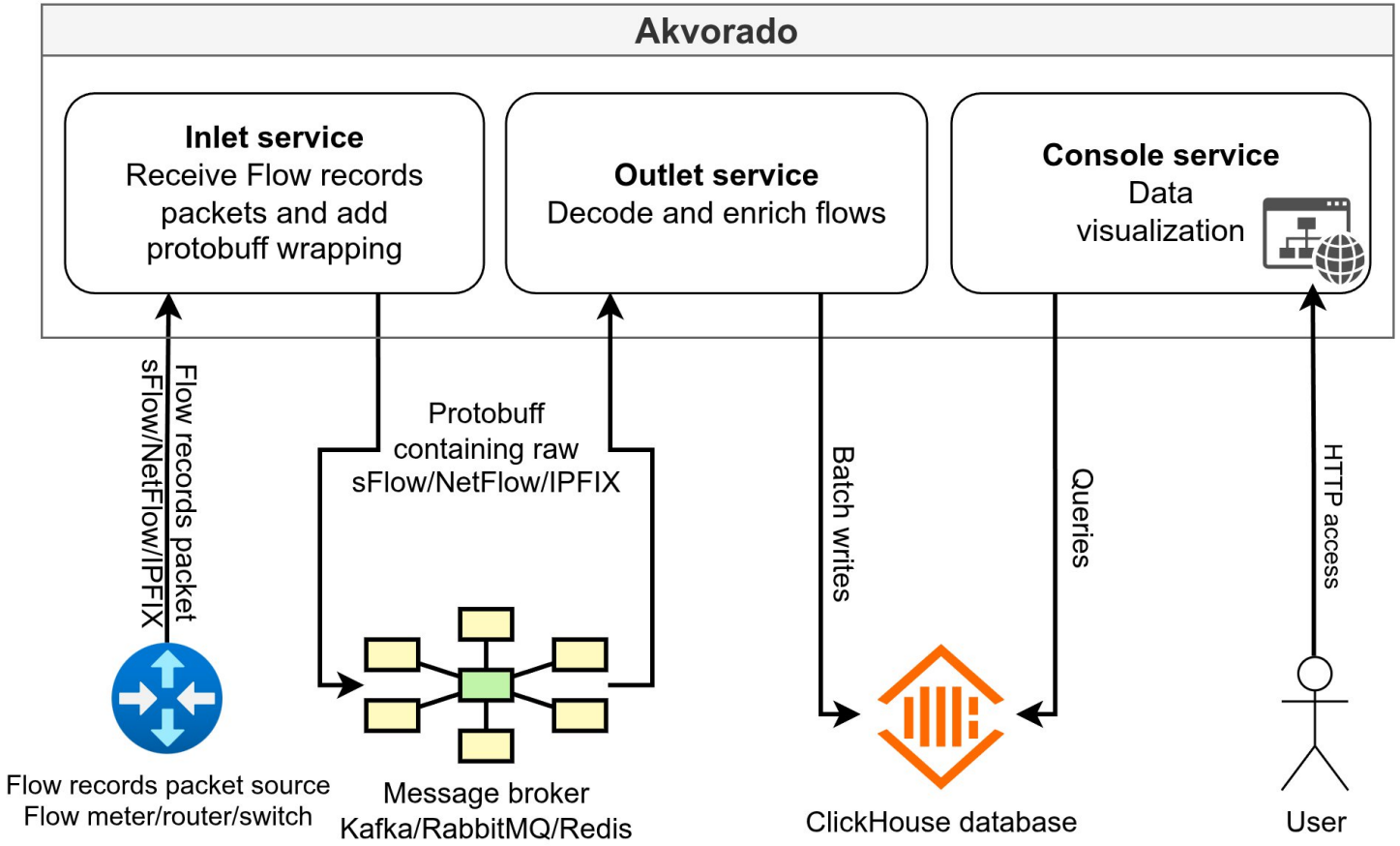
About pmacct

- pmacct flow collectors
  - nfacctd: NetFlow/IPFIX collector
  - sfacctd: sFlow collector

- pmacct is a network monitoring toolkit
  - Not restricted to flow monitoring
    - Interface packet capture
    - Streaming telemetry
    - BGP monitoring

About Akvorado

- All-in-one solution
  - Collection
  - Enrichment
  - Visualization
- Evaluation of Inlet service
  - Not a traditional collector
  - Missing decoding
  - New scaling approach



Based on https://demo.akvorado.net/assets/docs/design.svg

Qualitative Comparison

Function Overview

Category	nfdump	SiLK	pmacct	IPFIXcol2	GoFlow v2	Akvorado
sFlow	✓	✓	✓	✗	✓	✓
NetFlow	✓	✓	✓	✓	✓	✓
IPFIX	✓	✓	✓	✓	✓	✓
Enrichment						
Anonymization						
Telemetry endpoint						
File output						
Kafka output						
Additional output targets						

Function Overview

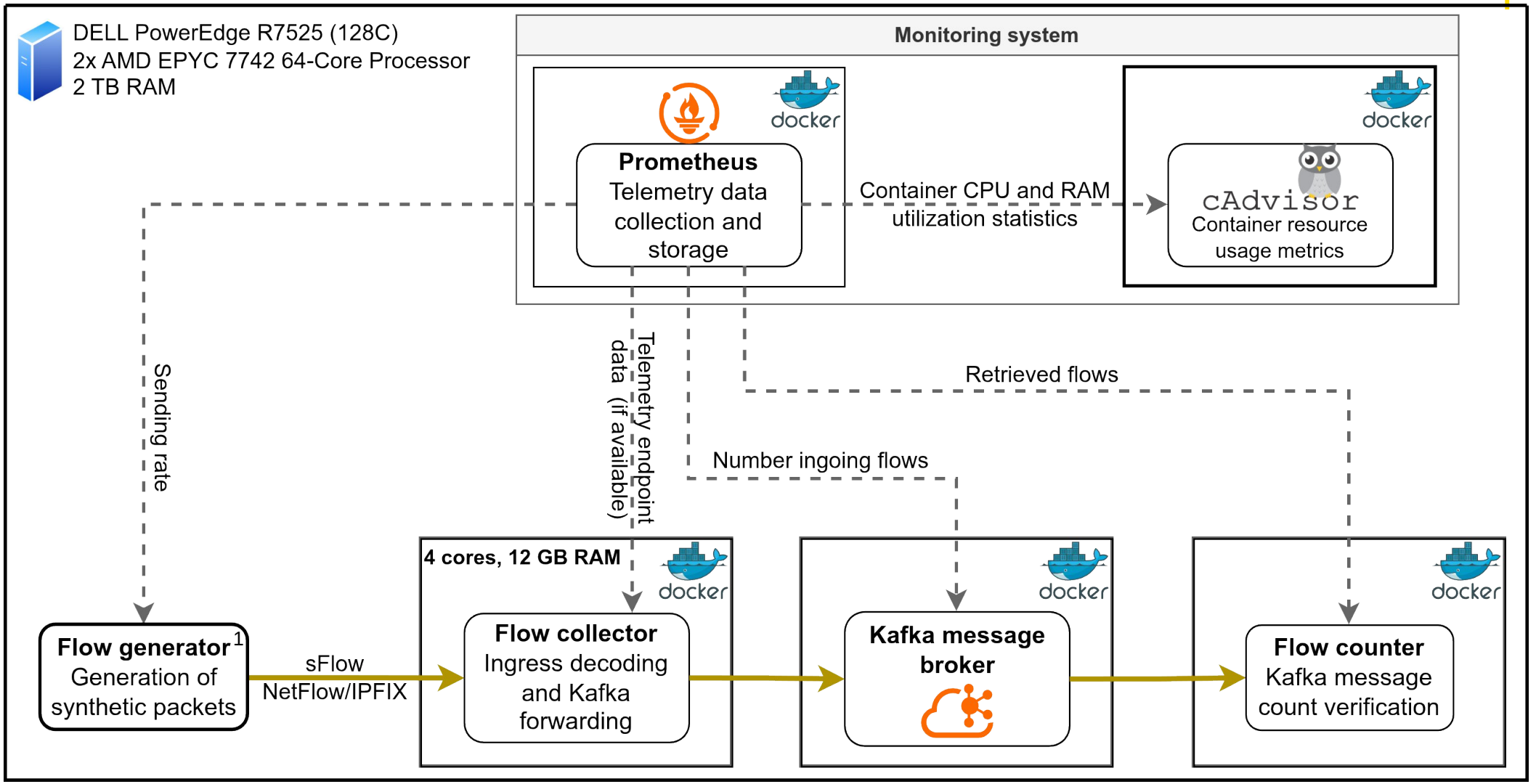
Category	nfdump	SiLK	pmacct	IPFIXcol2	GoFlow v2	Akvorado
sFlow	✓	✓	✓	✗	✓	✓
NetFlow	✓	✓	✓	✓	✓	✓
IPFIX	✓	✓	✓	✓	✓	✓
Enrichment	✓	✓	✓	✗	External enricher	✓
Anonymization	✓	✗	✗	✓	✗	✗
Telemetry endpoint	Unix socket HTTP endpoint via nfexporter	✗	✗	✗	✓	✓
File output						
Kafka output						
Additional output targets						

Function Overview

Category	nfdump	SiLK	pmacct	IPFIXcol2	GoFlow v2	Akvorado
sFlow	✓	✓	✓	✗	✓	✓
NetFlow	✓	✓	✓	✓	✓	✓
IPFIX	✓	✓	✓	✓	✓	✓
Enrichment	✓	✓	✓	✗	External enricher	✓
Anonymization	✓	✗	✗	✓	✗	✗
Telemetry endpoint	Unix socket HTTP endpoint via nfexporter	✗	✗	✗	✓	✓
File output	Custom format, JSON and CSV via export	Custom format, CSV via rwcut, IPFIX via rwsilk2ipfix	JSON, AVRO, CSV	JSON, FDS, IPFIX, nfdump format	JSON, protobuf	✗
Kafka output	✗	✗	JSON, AVRO	JSON	JSON, protobuf	Protobuf of raw flow records packets
Additional output targets	✗	✗	DB, RabbitMQ	ClickHouse	✗	Visualizations, ClickHouse

Quantitative Comparison

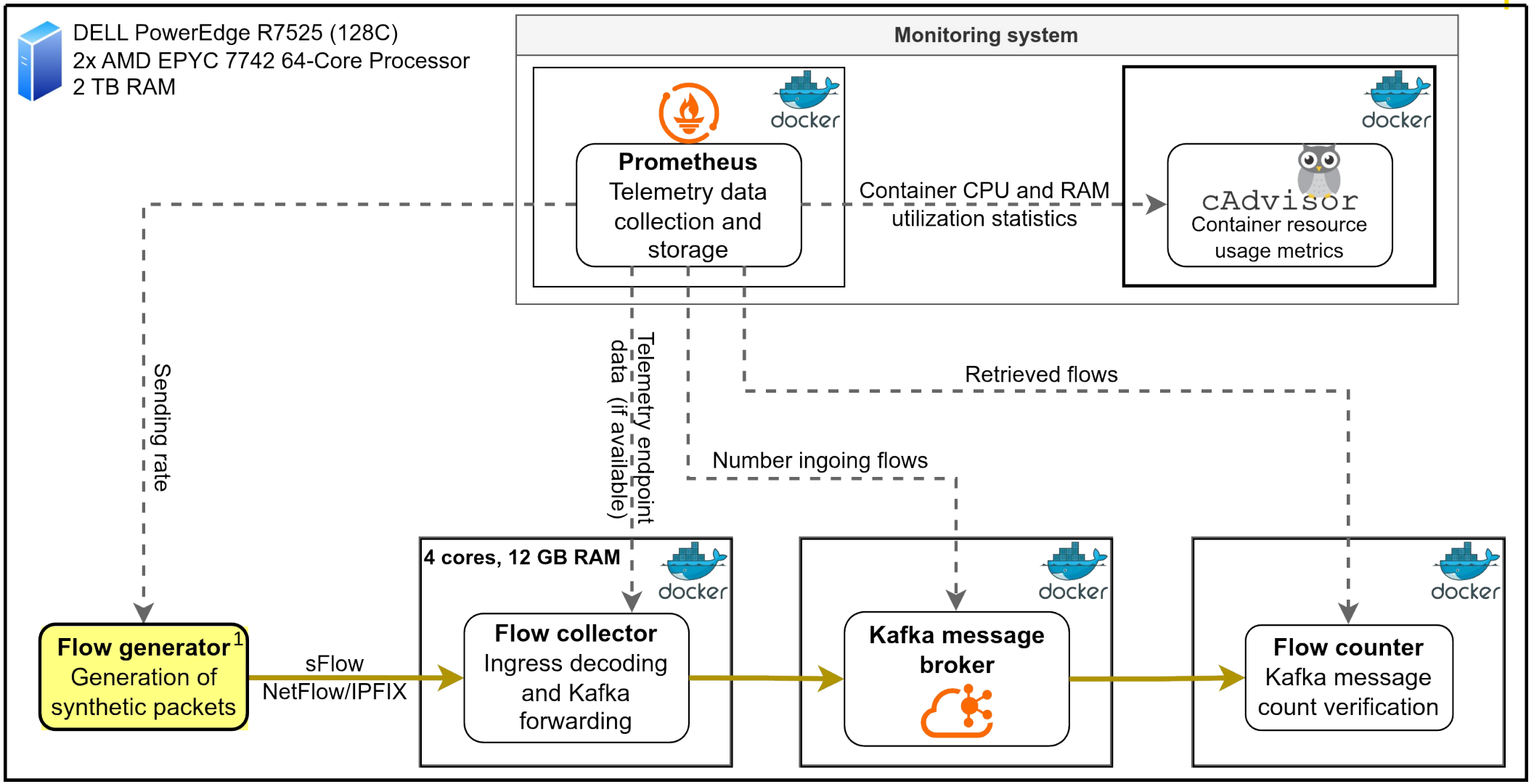
Measurement Testbed



1: Generator source code

https://codeberg.org/BelWue/go-nflow-stresstest

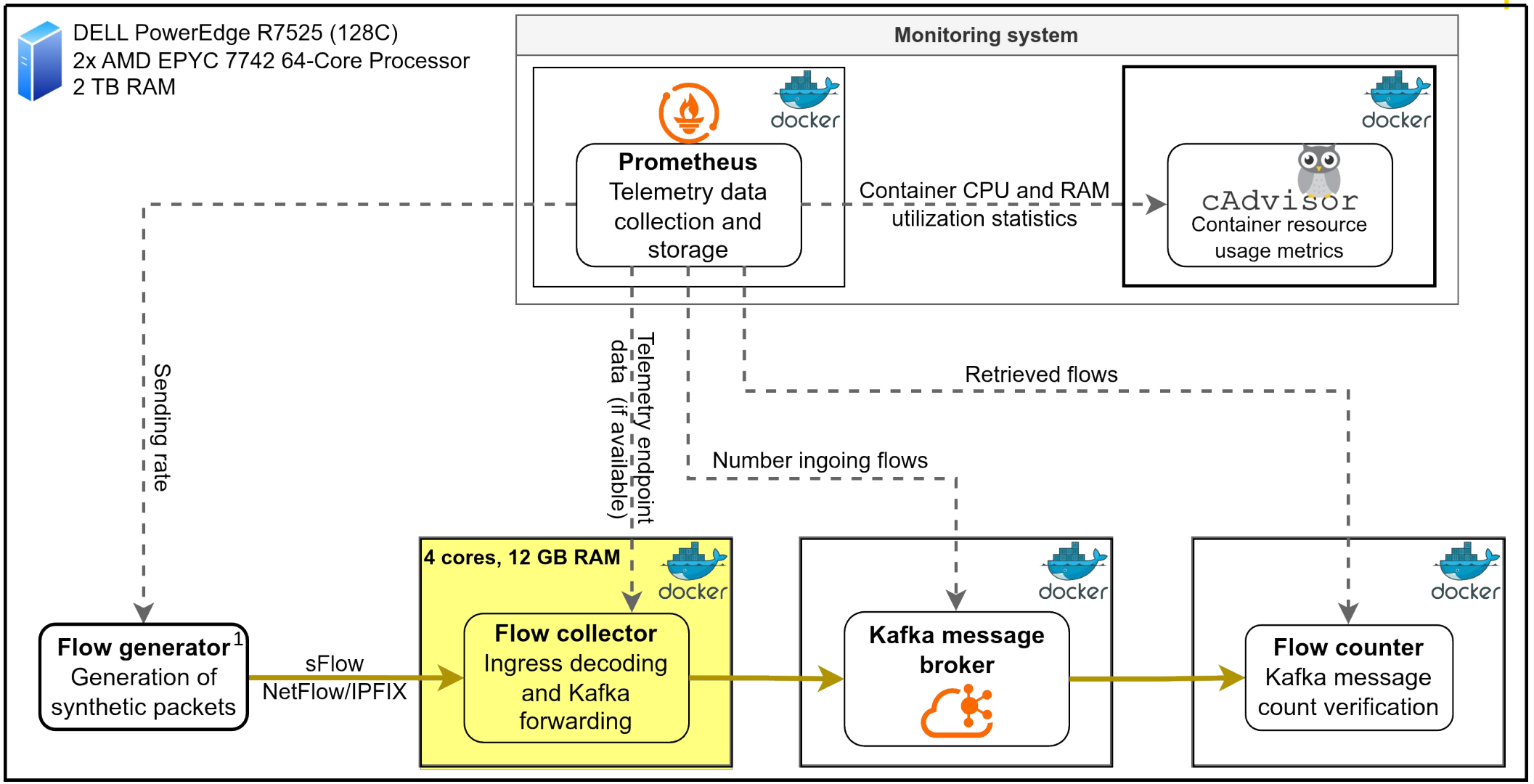
Measurement Testbed



1: Generator source code

https://codeberg.org/BelWue/go-nflow-stresstest

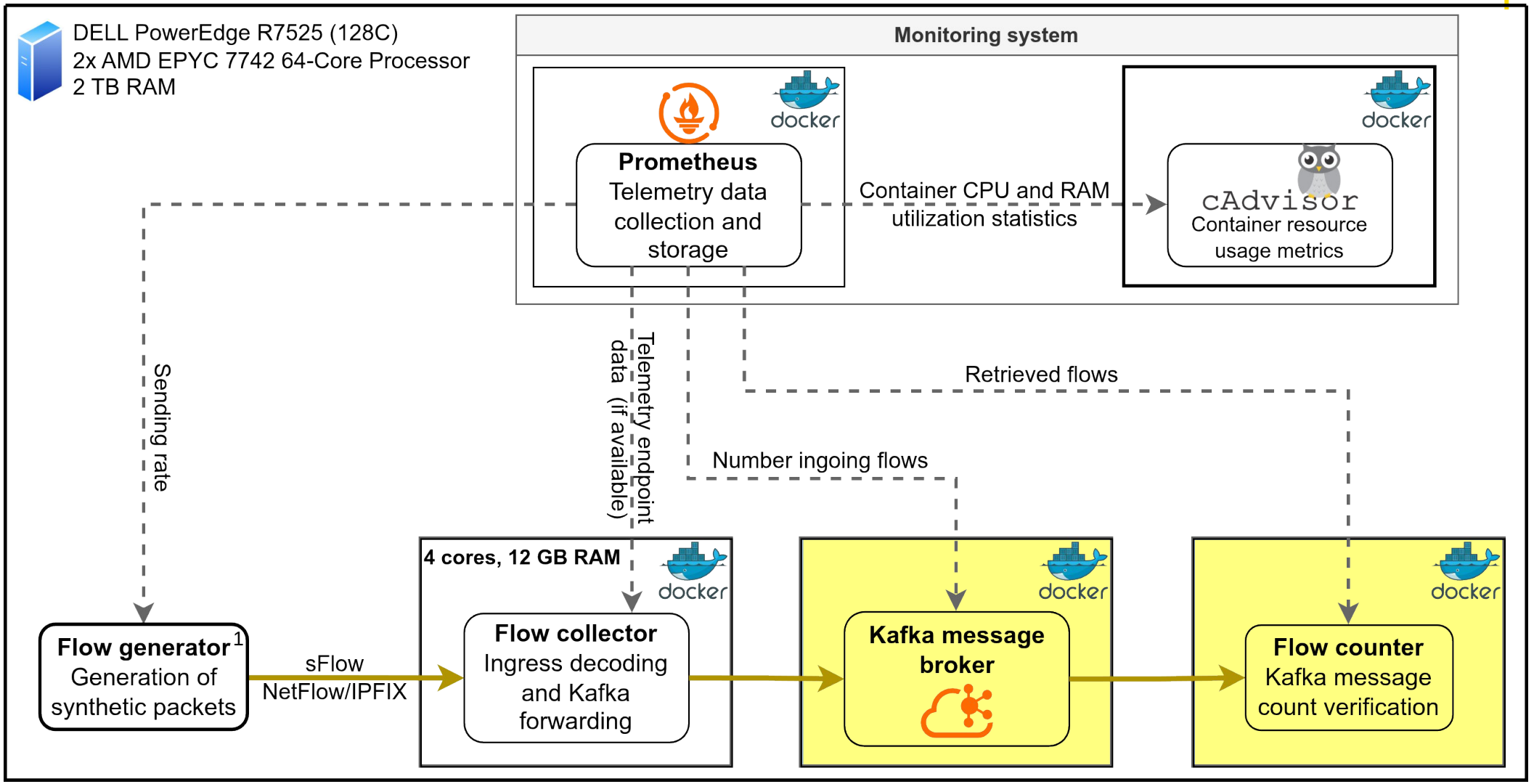
Measurement Testbed



1: Generator source code

https://codeberg.org/BelWue/go-nflow-stresstest

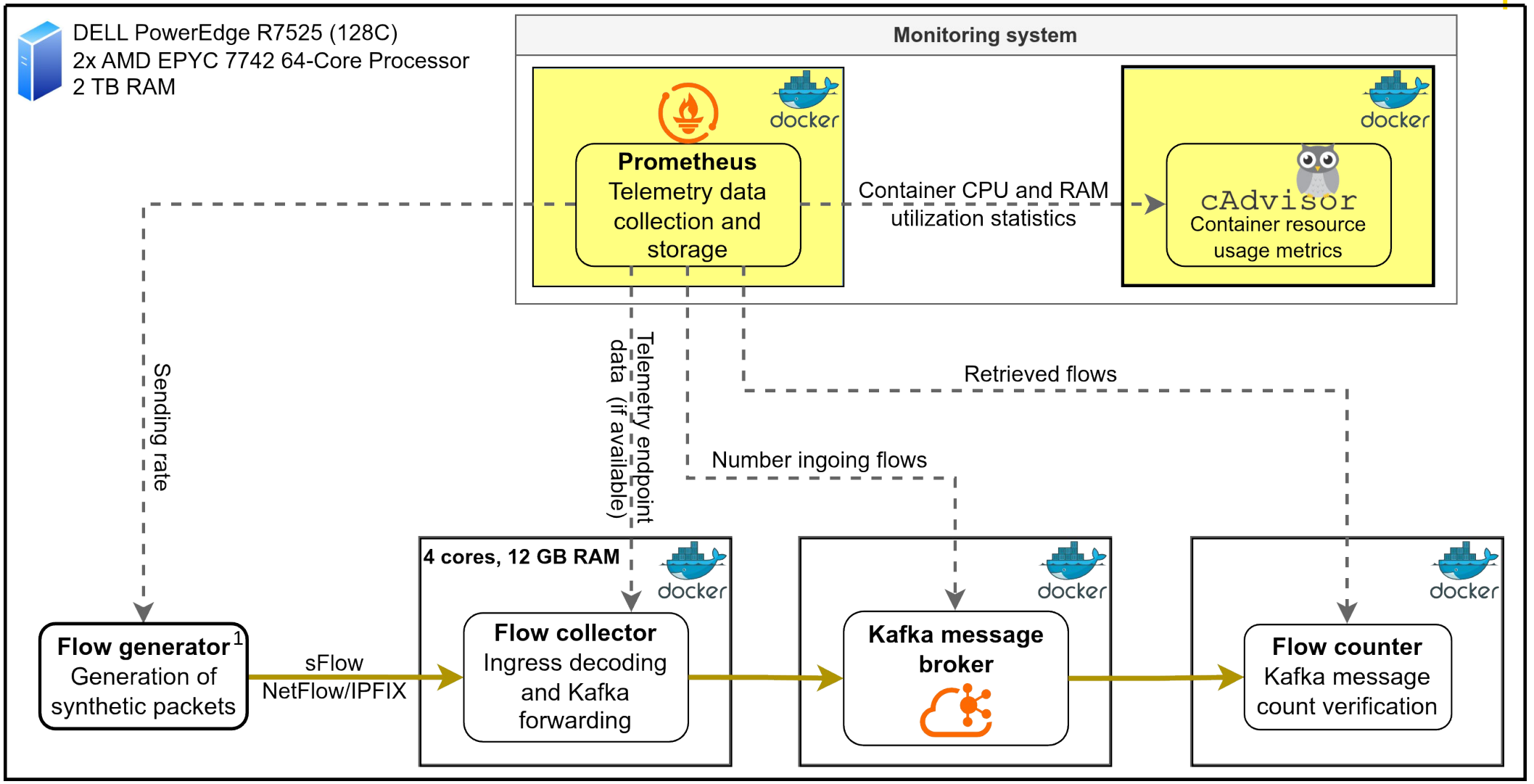
Measurement Testbed



1: Generator source code

https://codeberg.org/BelWue/go-nflow-stresstest

Measurement Testbed

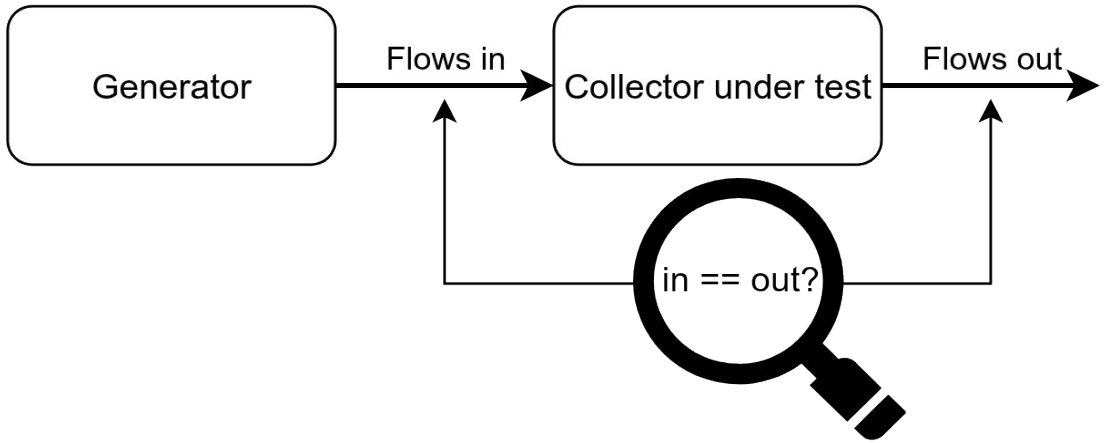


1: Generator source code

https://codeberg.org/BelWue/go-nflow-stresstest

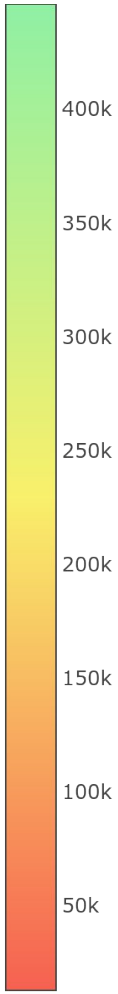
Measurement Approach

- Find sustainable processing rate
  - Create synthetic data with 20 flows per flow records packet
  - While in == out
    - Increase packet rate
    - in = flows sent to collector
    - out = flows received from collector
- Difficulties
  - File output
  - Flow records packets buffered by collector



Comparison: Kafka Output (flows/s)

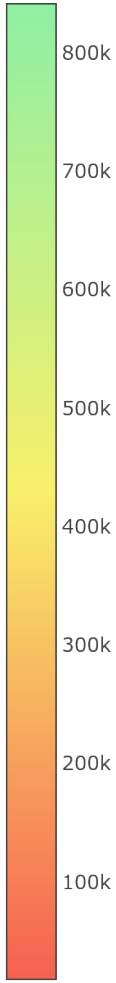
	nfdump	SiLK	pmacct	ipfixcol2	GoFlow (JSON)	GoFlow (protobuf)	Akvorado*
sFlow	not supported	not supported	186k	not supported	13.9k	81.2k	540.8k
NetFlow v5	not supported	not supported	204.7k	375.3k	14.4k	86.3k	510.8k
NetFlow v9	not supported	not supported	208.6k	423.9k	12.9k	79k	538.5k
IPFIX	not supported	not supported	205.1k	446.3k	13.1k	77.3k	548.5k



*Forwarding raw flows to Kafka

Comparison: File Output (flows/s)

	nfdump	SiLK	pmacct	ipfixcol2 (FDS)	GoFlow* (JSON)	GoFlow* (protobuf)	Akvorado
sFlow	426.5k	39.4k	162.6k	not supported	17.7k	125k	not supported
NetFlow v5	412.9k	217.4k	160.6k	422.5k	18k	145.5k	not supported
NetFlow v9	423.3k	94.9k	161.7k	425.4k	18.7k	122.7k	not supported
IPFIX	841.7k	53.6k	164.2k	679.6k	18.8k	120.2k	not supported



*Measured via collector metrics

Main Findings

Input format

- Performance differences for some collectors
  - Ipfixcol2 & nfdump: better IPFIX performance
  - SiLK: better NetFlow v5 performance
  - SiLK & pmacct: worse sFlow performance

Output format

- Crucial impact on performance
  - GoFlow2: inefficient JSON serialization
  - Custom binary formats more efficient for file output

Output target (Kafka/file)

- File output faster for most collectors
  - Not adjusted for buffer sizes
  - pmacct: better performance with Kafka

Scaling

- Message brokers support parallel writes
  - Enables horizontal scaling
  - Simplifies redundant architecture deployment

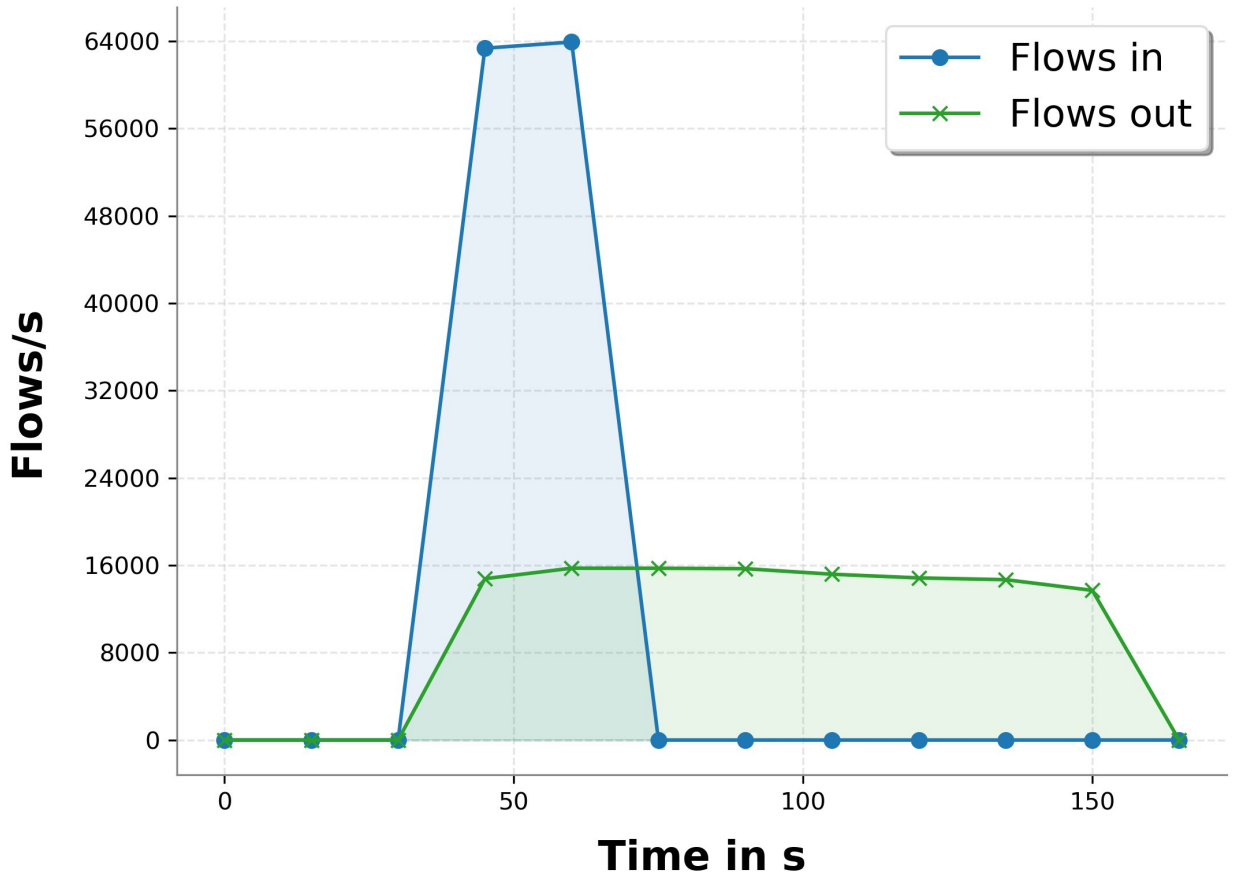
Open Questions

- Custom fields
  - IPFIX allows to add custom data fields
  - Not covered by the study: could impact performance
- Performance improves with additional resources
  - Study done with 4 CPU cores and 12 GB RAM
  - Collectors could scale differently with more resources
- Different flows per flow records packet
  - All tests were done with 20 flows per packet
  - Effect of decapsulation overhead increases with fewer flows per flow records packet
- Collector ecosystems
  - Analysis tool chain compatibility and performance

Additional Findings

Input Buffer for Flow Collector

- Avoids packet loss during provisional overload
  - Packet processing delayed instead of packets being dropping
  - Configurable buffer size
  - Unlimited buffer size can cause swapping



Drop Detection in Production

- Detection in testbed
  - Known input
  - Usage of telemetry endpoints when available
- Challenges in production environments
  - Unknown number of incoming flow records packet
  - Missing telemetry endpoints for multiple collectors
- Alternative drop detection approaches
  - Indirect monitoring via CPU, memory, and interface metrics
  - Ingestion of canary flows
    - Known amount of identifiable flow records packets
    - Loss of canary flows as drop indicator

Conclusion

- Studied multiple open-source flow collectors
- Collector limits relevant for flow monitoring correctness
  - Drops are difficult to monitor
- Study provides guidance for flow collector selection and dimensioning
  - Performance benchmarks and feature comparison
  - Load testing tool [1]
- No single best collector solely based on performance
  - Variations in feature support, output formats, and tool chains
  - Scaling and redundancy considerations (Message broker setup)

[1]

Questions?

huber@belwue.de

BelWü
Landeshochschulnetz Baden-Württemberg