

A Lightweight Persistence System for Network Traffic Measurements

Michael Mente 01634435

Supervisor: Oliver Michel, BSc MSc PhD



Motivation

- Reliability is one of the central requirements
- Current network traffic analysis mostly performed live
- Live traffic analysis not always sufficient
- Retrospective traffic analysis
- General purpose databases cannot handle the load

Motivation

- General purpose SQL databases: insertion rate $< 100\text{k/sec}$
 - Complex indices
 - Constraint checking
- Optimized timeseries database ingestion rate $\sim 400\text{k packets/second}$ ^[1]
 - Append only optimizations
- Traffic capture at top of rack switch
- 100m packets per second ideal ingestion rate

Methods

- Development of a prototype
 - Higher ingestion rates than traditional general-purpose databases
 - Efficient encoding methods for storage efficiency
 - Meaningful query interface
 - Fast query resolving without complex indexing methods

Methods

- Measure quantitative performance of prototype
 - Ingestion rate
 - Compression rate
 - Query resolving rate
 - Real WAN trace captured at Equinix datacenter for realistic test results*
- Qualitative analysis of queries
 - Comparison against existing solutions

* WAN link between New York City & Sao Paulo[2]

Challenges

- Fast storage access
 - Modern NVME SSDs: 1GByte/s write rate
- Packet compression
 - Data trimming & filtering
 - (header only, only IPv4 (TCP, UDP, ICMP) > 95% traffic*
 - Delta compression [3]
 - Dictionary encoding [4]

* measured from Equinix Dataset provided by Caida Organization[2]

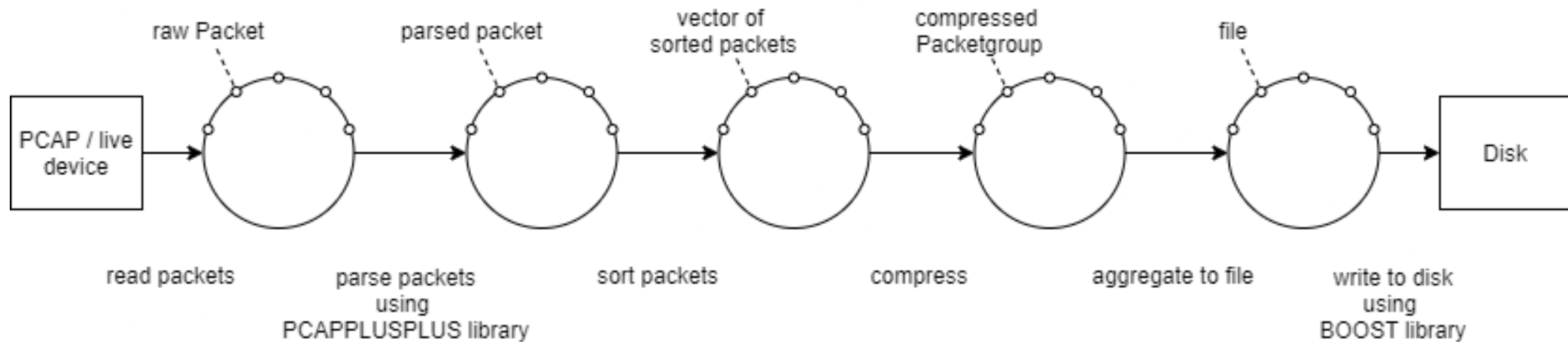
Challenges

- Querying of the data
 - Efficient data structure
 - Query performance
 - Simple parametric query language

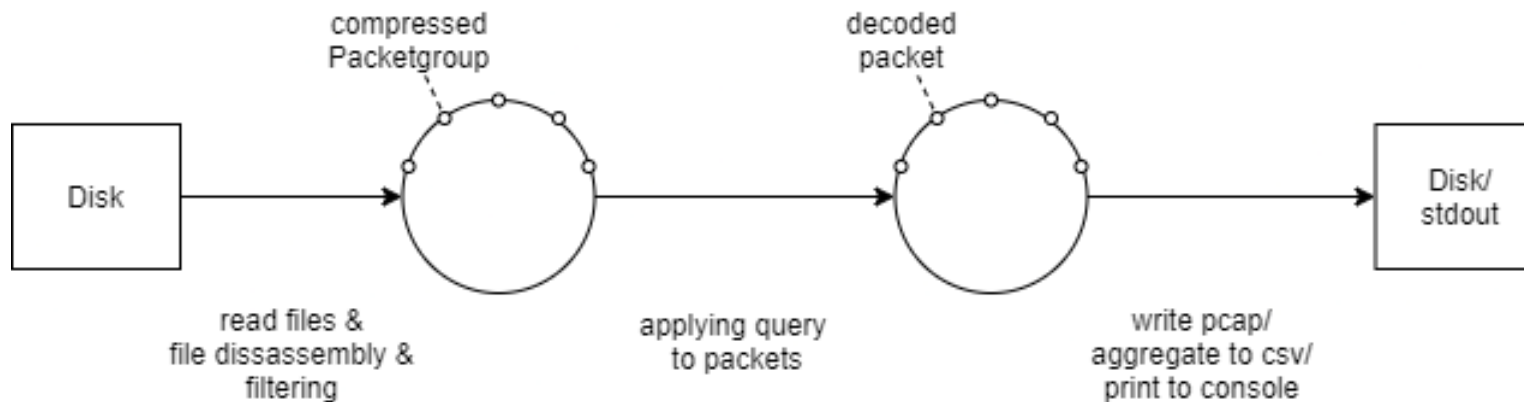
Implementation

- Two modules
 - Writer module
 - Read from network interface/converts pcap files
 - Compresses network traffic
 - Query module
 - Resolves user queries
 - Rebuilds query to pcap
 - Provides aggregation functionality

Implementation Design – Writer Module



Implementation Design – Query Module



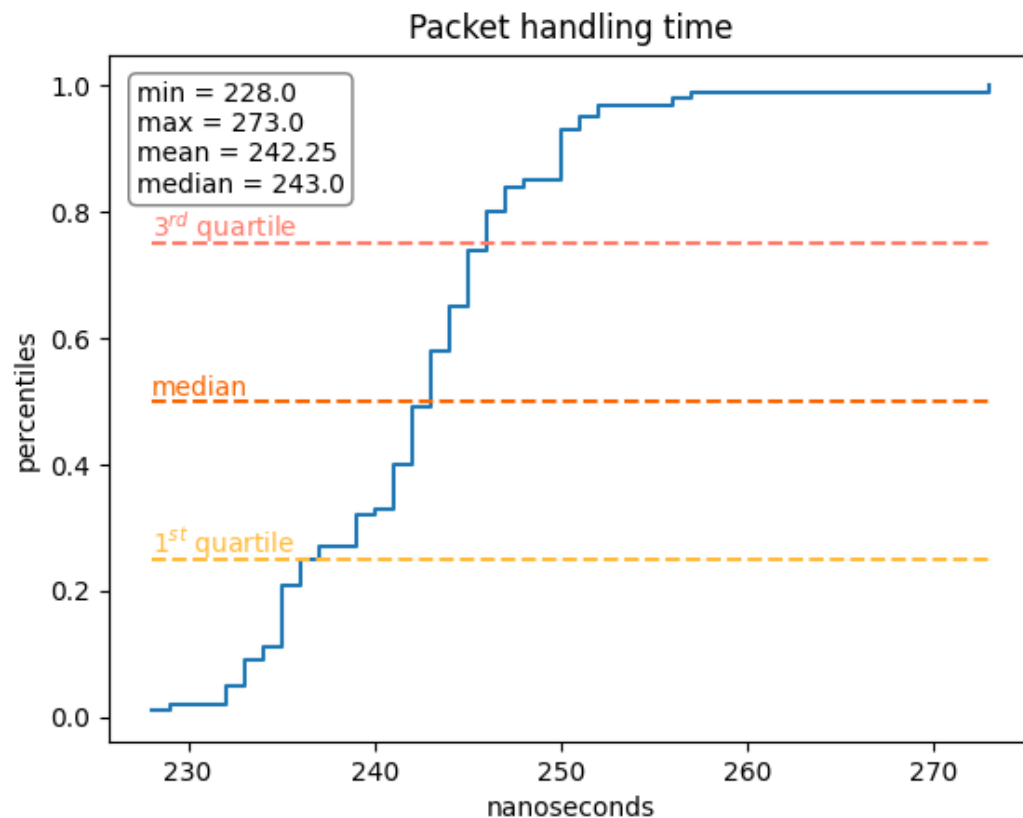
Query Language

- Similar syntax to Wireshark query language
 - *<IP tuple field> <comparison> <value> <logical operator>**
 - `ip.addr == 123.456.789.002`
 - `ip.addr != 0.0.0.0 && frame.time < Jun 15, 2021 12:00:00 || udp`
 - Internally represented by a filter tree and evaluated from right to left
 - `(ip.addr != 0.0.0.0 && (frame.time < Jun 15, 2021 12:00:00 || udp))`

Aggregation Operations

- Performed on filtered packets
 - Aggregation interval can be specified in microsecond resolution
 - Possible operations are: sum, min, max, mean, count, count distinct
 - Can be applied to every field of the IP tuple (src addr, src port, dst address, dst port, protocol, length)
 - e.g. used bandwidth can be measured by aggregation *sum of length* every second

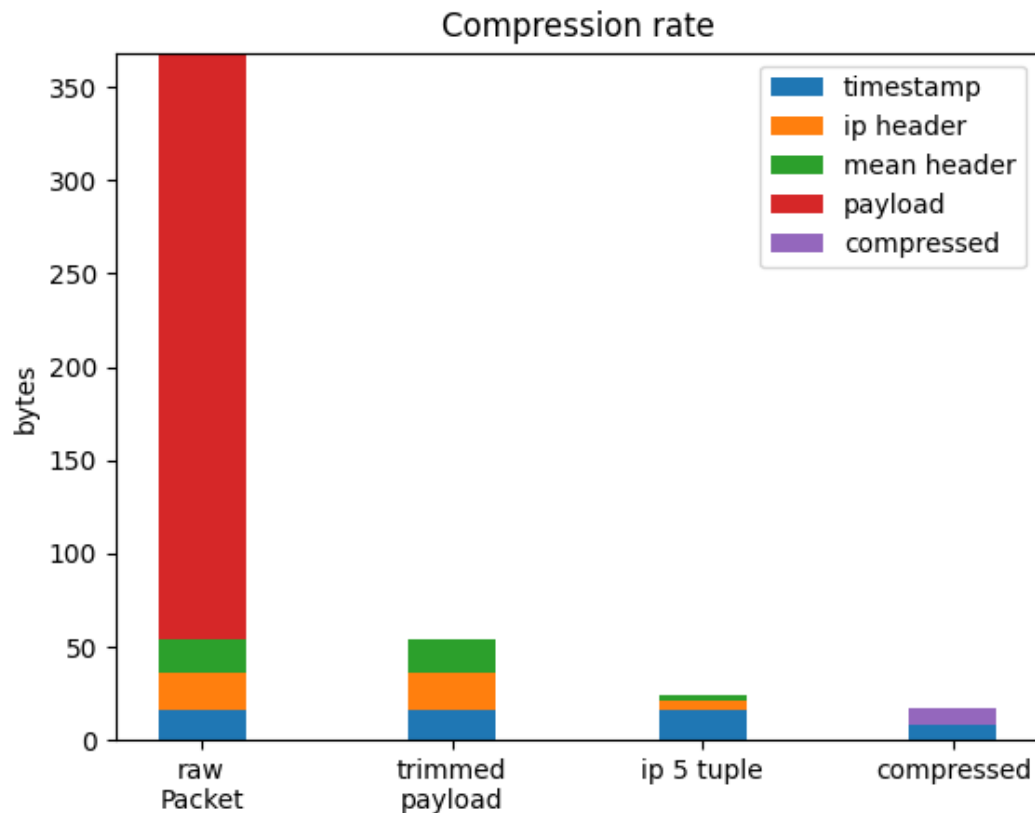
Implementation Evaluation – Writer Performance



- 100m packets/s -> handling time 10ns
- 240ns per packet on own machine
- measured* performance of 4.1m packets/s
- measured on 8c@4.2 GHz processor in VM
- Increased performance by 30% since last presentation

* all performance measurements were conducted at least 100 times iteratively

Implementation Evaluation – Compression

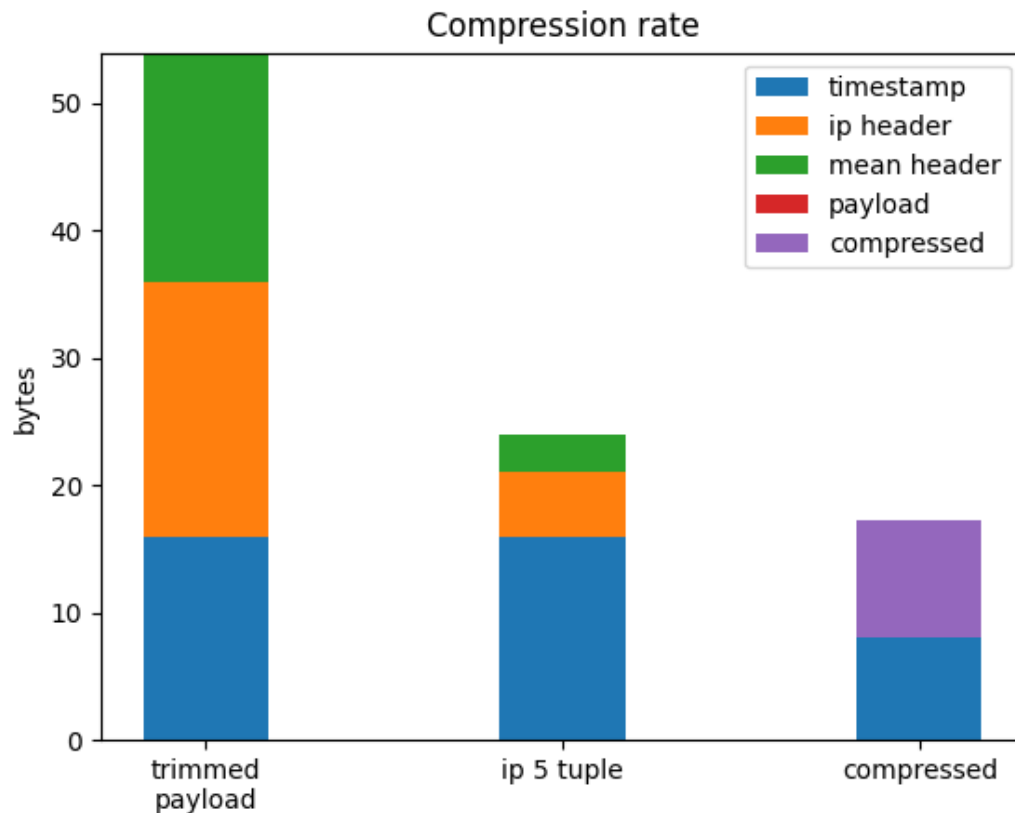


- Original : 354.45 bytes per packet*
- Trimmed : 37.96 bytes
- IP tuple + timestamp : 31 bytes
- Compressed : 17.25 bytes

- Total Compression Ratio: 20:1

* measured from Equinix Dataset provided by Caida Organization[2]

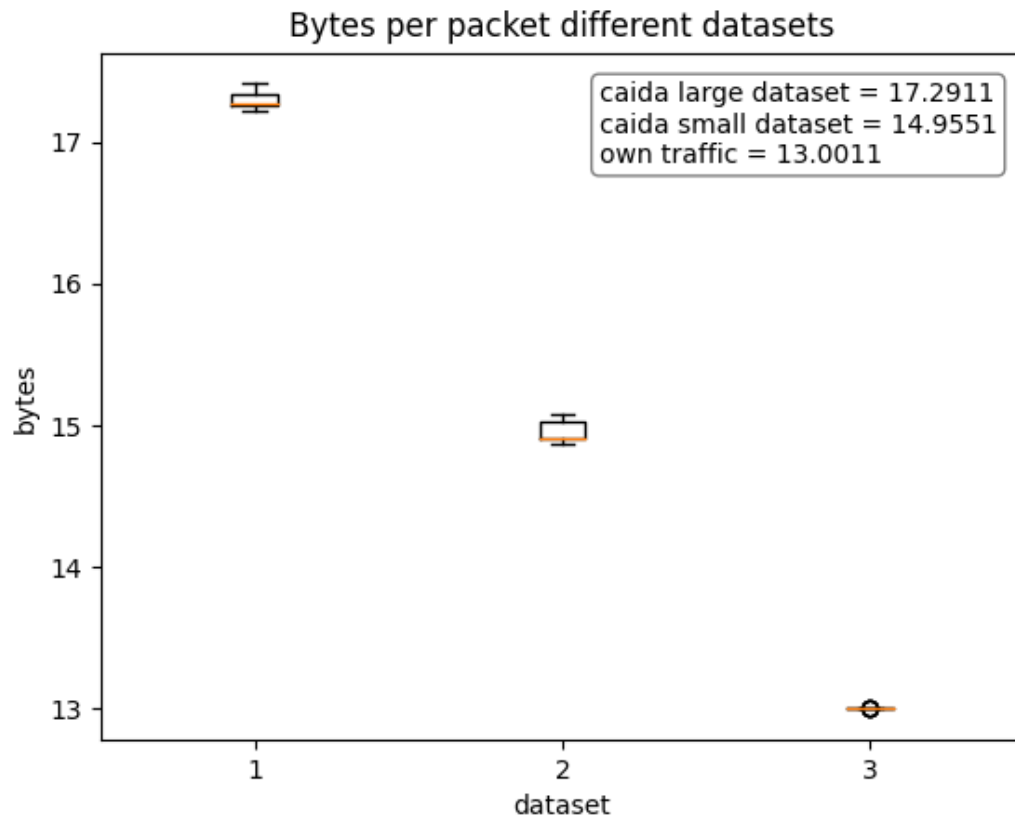
Implementation Evaluation – Compression



- Original : 354.45 bytes per packet*
- Trimmed : 37.96 bytes
- IP tuple + timestamp : 31 bytes
- Compressed : 17.25 bytes
- Effective Compression Ratio: 1.8:1

* measured from Equinix Dataset provided by Caida Organization[2]

Implementation Evaluation – Compression



- Compression ratio strongly depends on traffic type
- The longer the conversation the better
- In ideal conditions compression ratio up to 2.92 : 1*

* measured from own traffic capture

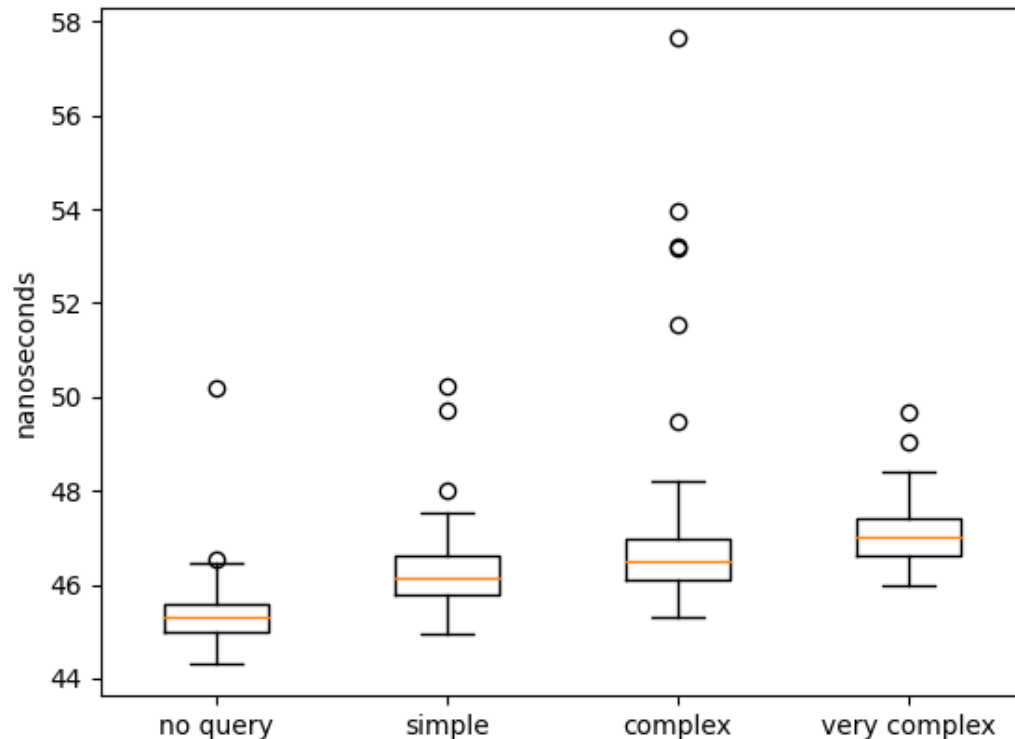
Implementation Evaluation – Compression

- 100 Gbit Link -> 1.375 GByte/s just in Headers (11% of total traffic*)
- Compression ratio must be above 1.375 : 1 to prevent storage bottleneck
- Lowest measured ratio: 1.8 : 1
- Estimated in datacenter scenario: >2:1

* measured from Equinix Dataset provided by Caida Organization[2]

Implementation Evaluation – Query Resolving

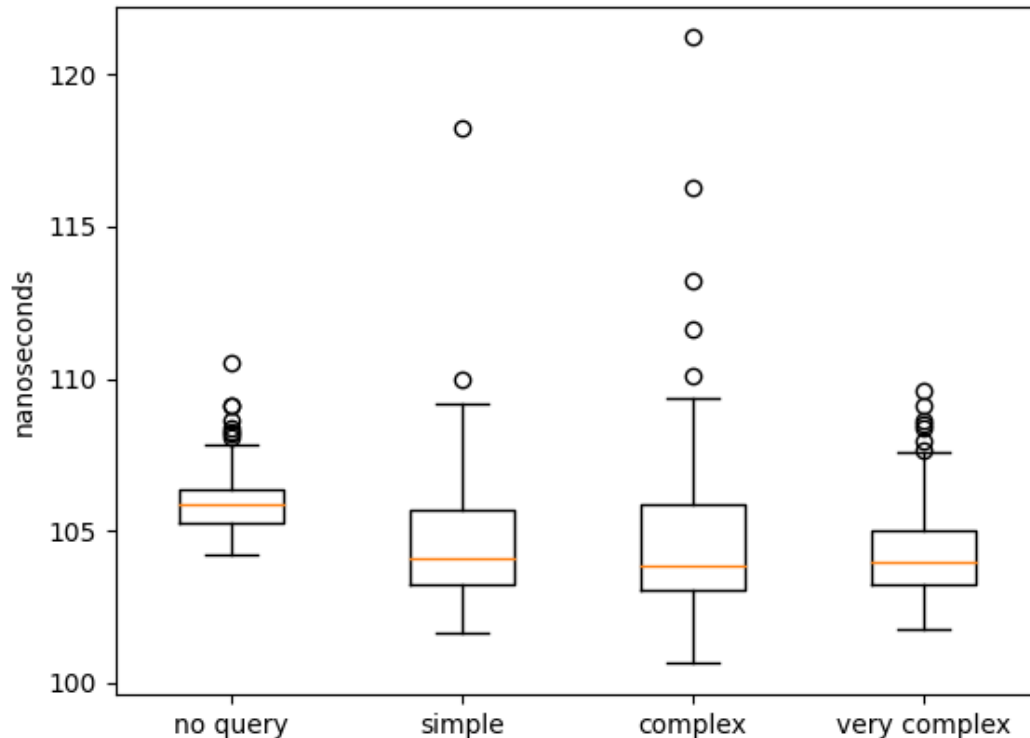
Query handling runtime analysis of equinix-nyc.dirB.20180517-134900



- Packet handling time 45ns without query
- 5 times faster than writing
- Query complexity only minor impact
 - Packet handling time 47ns with complex query (less than 10 additional clock cycles at 4.2GHz)

Implementation Evaluation – Query Resolving

Query total runtime analysis of equinix-nyc.dirB.20180517-134900



- Query complexity has no negative impact on overall performance
- Performance tends to be better since less data is written/processed after query is applied*

* I used a query which prevents reducing the data volume

Implementation Evaluation – Query Resolving

- *Very complex query* I used (6 filters chained by && operator):
- `frame.len > 0 && frame.len <= 999999999 && frame.time < Jun 15, 2021 12:00:00 && frame.time > Jun 15, 2000 12:00:00 && port >= 0 && ip.addr != 0.0.0.0`
- Tries not to reduce the number of packets queried
- Only chained by && as `||` would possibly evaluate faster (as the rest of the query is not evaluated if true)

* I used a query which prevents reducing the data volume

Conclusio

- Ingestion rate up to 4.3m packets/s
- Effective compression ratio up to 2.9 : 1
- Efficient query resolving & expressive parametric query interface

* I used a query which prevents reducing the data volume

Bibliography

- [1] Michel, O., Sonchack, J., Keller, E., & Smith, J. M. (2019, March). PIQ: Persistent interactive queries for network security analytics. In Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (pp. 17-22).
- [2] Caida.org. Passive Monitor: equinix-nyc. http://www.caida.org/data/passive/passive_sampler_dataset.xml (accessed on 19.06.2021)
sampler for performance experiments: equinix-nyc.dirB.20180517-134900
- [3] Suel, T. (2019). Delta Compression Techniques. Encyclopedia of Big Data Technologies, 63.
- [4] Larsson, N. J., & Moffat, A. (2000). Off-line dictionary-based compression. Proceedings of the IEEE, 88(11), 1722-1732.

Feature Set – Writer Module

- *-file* specifies the path to an input pcap file
- *-live* capture traffic from a live network interface
- *-output* specifies the path for binary data dump files
- *-sequential* executes each step in the data pipeline sequentially
 - used to determine bottlenecks
- *-benchmark* prints statistics for further analysis

Feature Set – Query Module

- *-i* specifies the path for binary data dump files
- *-o* specifies the output directory
- *-f* allows to set a filter
- *-pcap* redirects output to a pcap file
- *-aggregate* performs statistical operations on a field of the ip-tuple
 - Aggregation operator and field can be further specified
- *-benchmark* prints statistics for further analysis