Customizable Decay: How to Maximize Suricata Event Utility in Finite Space

Sascha Steinbiss, DCSO
Benno Evers, Tenzir
Matthias Vallentin, Tenzir

TENZIR

DCSO

Introduction

Sascha

- Senior Software Engineer
- >6 years at DCSO
- Former genome wrangler
- Suricata contributor
- Debian Developer

Benno

- C++ Developer at Tenzir
- Former PMC at Apache Mesos project
- Distributed Systems programming

Matthias

- Founder & CEO at Tenzir
- PhD @ UC Berkeley (with Zeek team)
- High-performance network monitoring
- SOC infrastructure and threat detection

Introduction

Sascha

- Senior Software Engineer
- >6 years at DCSO
- Former genome wrangler
- Suricata contributor
- Debian Developer

Benno

- C++ Developer at Tenzir
- Former PMC at Apache Mesos project
- Distributed Systems programming

Matthias

- Founder & CEO at Tenzir
- PhD @ UC Berkeley (with Zeek team)
- High-performance network monitoring
- SOC infrastructure and threat detection

Suricata: Not Just Alerts

- Suricata is an IDS/IPS *and* NSM tool
  - Output : alerts and extensive metadata per protocol transaction
- Metadata are useful
  - Other forms of detection (retro, pattern-based, data mining)
  - Analysis (network structure, communication patterns, …)
- Scenario: store metadata close to the Suricata sensor
  - Data residency requirements
  - Bandwidth constraints
  - Scaling/cost issue with centralized architecture

Too much data in confined space

Cumulative EVE-JSON volume of a real sensor with ~6Gbit/s of diverse traffic

Too much data in confined space?

- Extend space
  - Limited by deployment constraints (bare-metal appliances? cost?)
- Compress data
  - Adds additional overhead for compression/decompression
  - Still natural limits for space reduction
- Move data (tape storage, …)
  - Deployment constraints (rack space, network, …)
  - Huge impact on access latency/searchability
- Delete old data
  - Blunt tool with no awareness of data importance
  - We may still need it!

Can we do better?

Optimal Information Density?

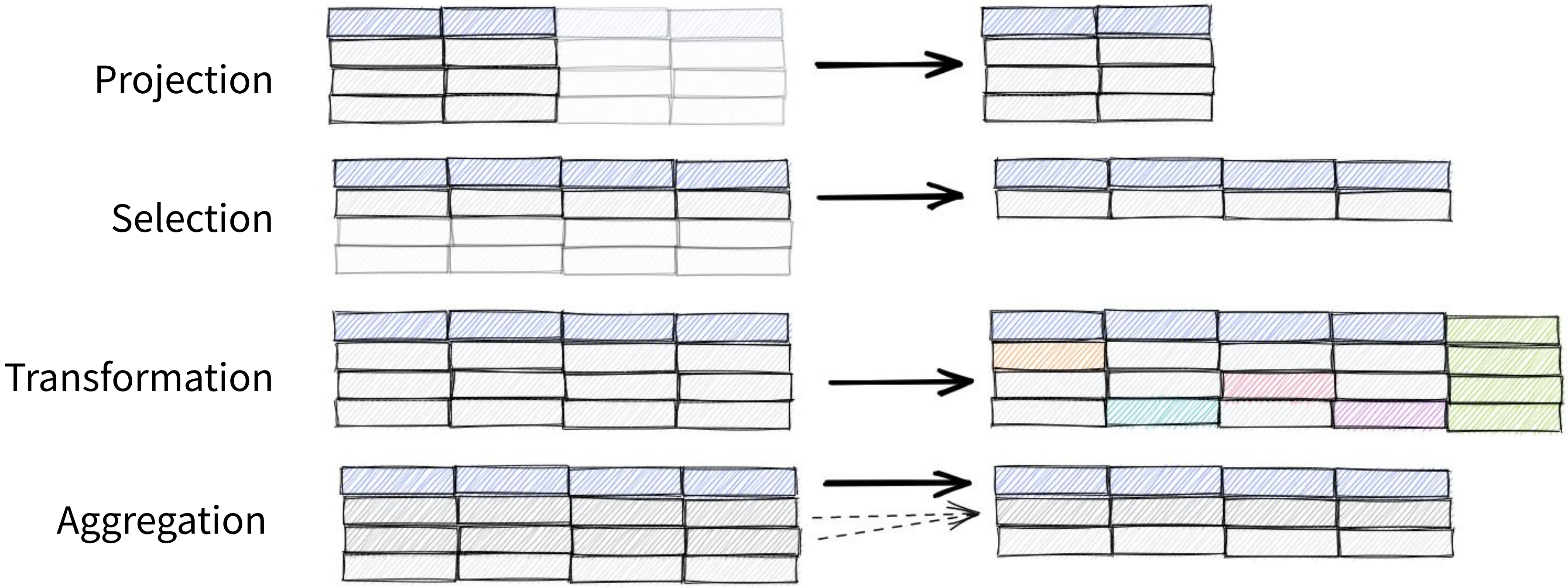
- Change data
  - Never delete relevant events, but make data range more coarse-grained
  - Maybe less data is already sufficient for the actual questions we have
  - Example: Who looked up domain X in a specific timeframe?


```
{"timestamp": "2020-11-06T10:32:23.723294+0000", "event_type": "dns", "src_ip": "10.0.0.207", "src_port": 32598,
"dest_ip": "148.0.250.134", "dest_port": 53, "proto": "UDP", "dns": {"rrname": "www.foobar.com", ...}}}
{"timestamp": "2020-11-06T10:39:39.325094+0000", "event_type": "dns", "src_ip": "10.0.0.207", "src_port": 29384,
"dest_ip": "148.0.250.134", "dest_port": 53, "proto": "UDP", "dns": {"rrname": "www.foobar.com", ...}}}
{"timestamp": "2020-11-06T10:42:57.000333+0000", "event_type": "dns", "src_ip": "10.0.0.207", "src_port": 44522,
"dest_ip": "148.0.250.134", "dest_port": 53, "proto": "UDP", "dns": {"rrname": "www.foobar.com", ...}}}
```


vs.


```
{"first_seen": "2020-11-06T10:32:23.723294+0000", "last_seen": "2020-11-06T10:42:57.000333+0000", "count": 3,
"event_type": "dns", "src_ip": "10.0.0.207", "dest_ip": "148.0.250.134", "proto": "UDP", "dns": {"rrname":
"www.foobar.com", ...}}}
```

Data Transformations



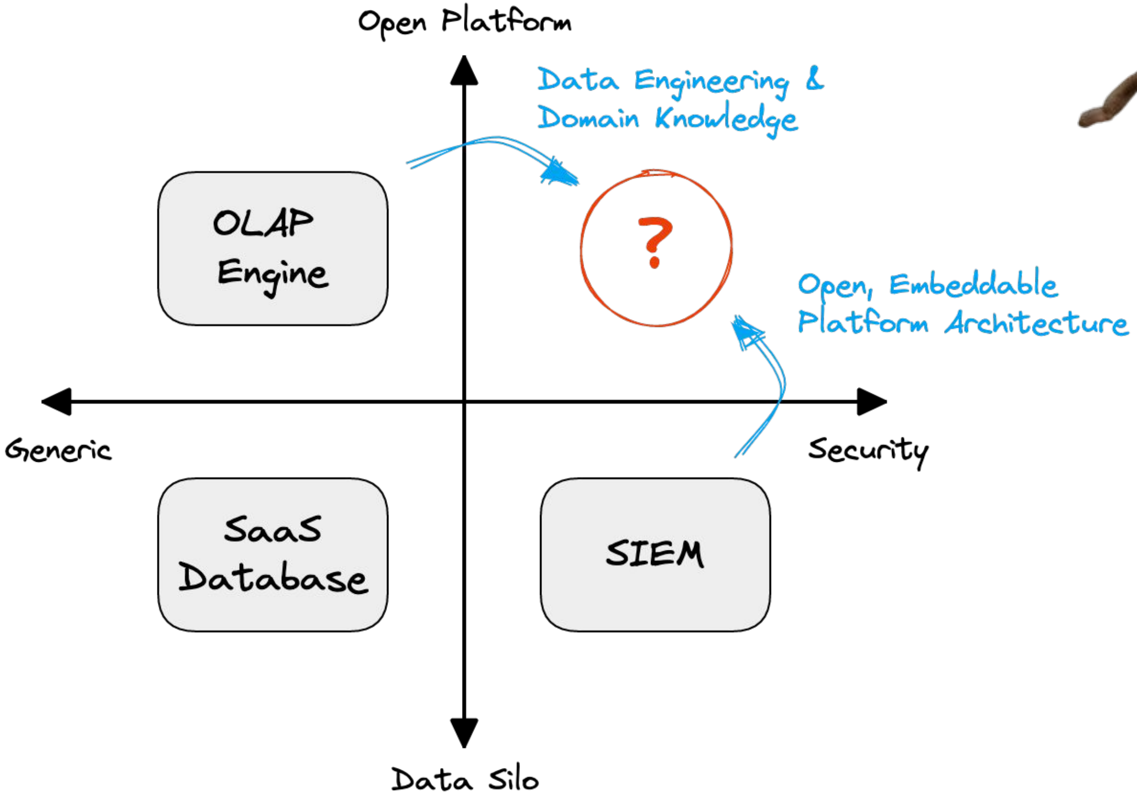
Contributions

1. Devise a **compaction** concept to trigger arbitrary dataflow pipelines
    - Composable, modular data processing **operations**
    - Trigger pipelines for spatial ("80% disk") and temporal ("after 42 days") conditions
2. Implement compaction in VAST (vast.io)
    - Easy-to-use declarative YAML configuration
    - Operationalize in a production deployment
3. Evaluate with real-world Suricata EVE logs

VAST

Recap

Why VAST?



Open Platform
Data Engineering &
Domain Knowledge
OLAP
Engine
?
Open, Embeddable
Platform Architecture
Generic
Security
SaaS
Database
SIEM
Data Silo

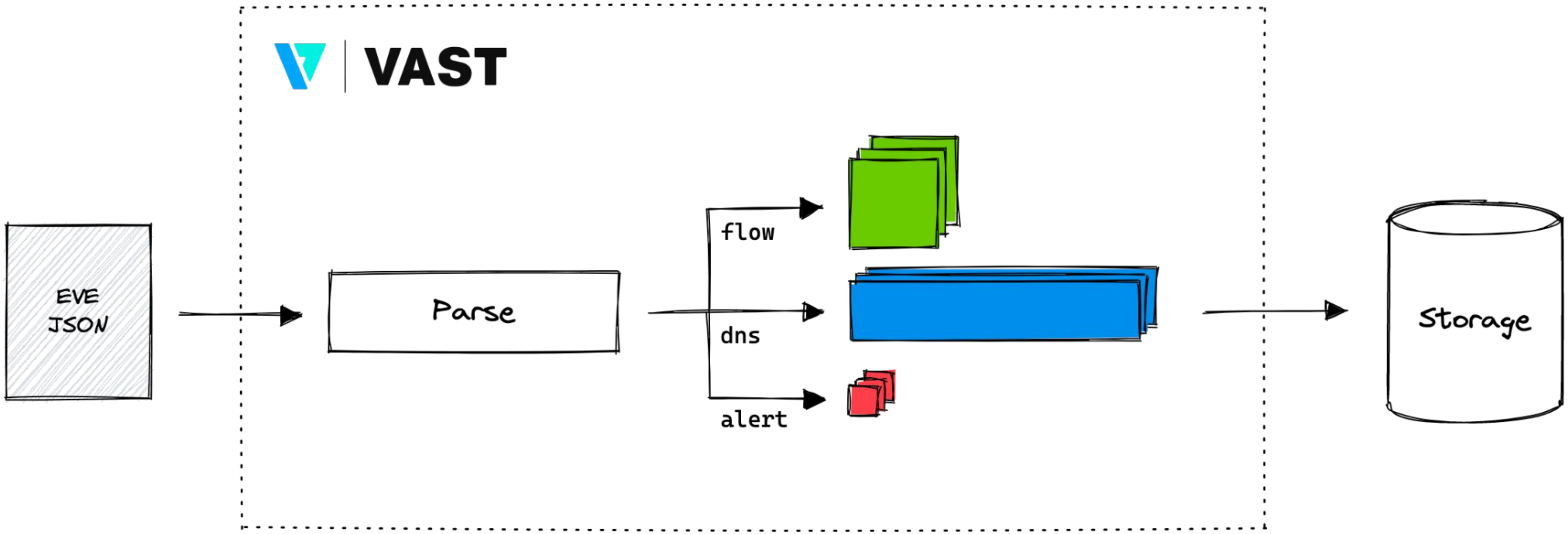
VAST: An Overview

- What is VAST?
  - Security-native high-performance telemetry database
  - Richly-typed, structured data
  - Open data plane via Apache Arrow (in-memory) and Parquet & Feather (disk)

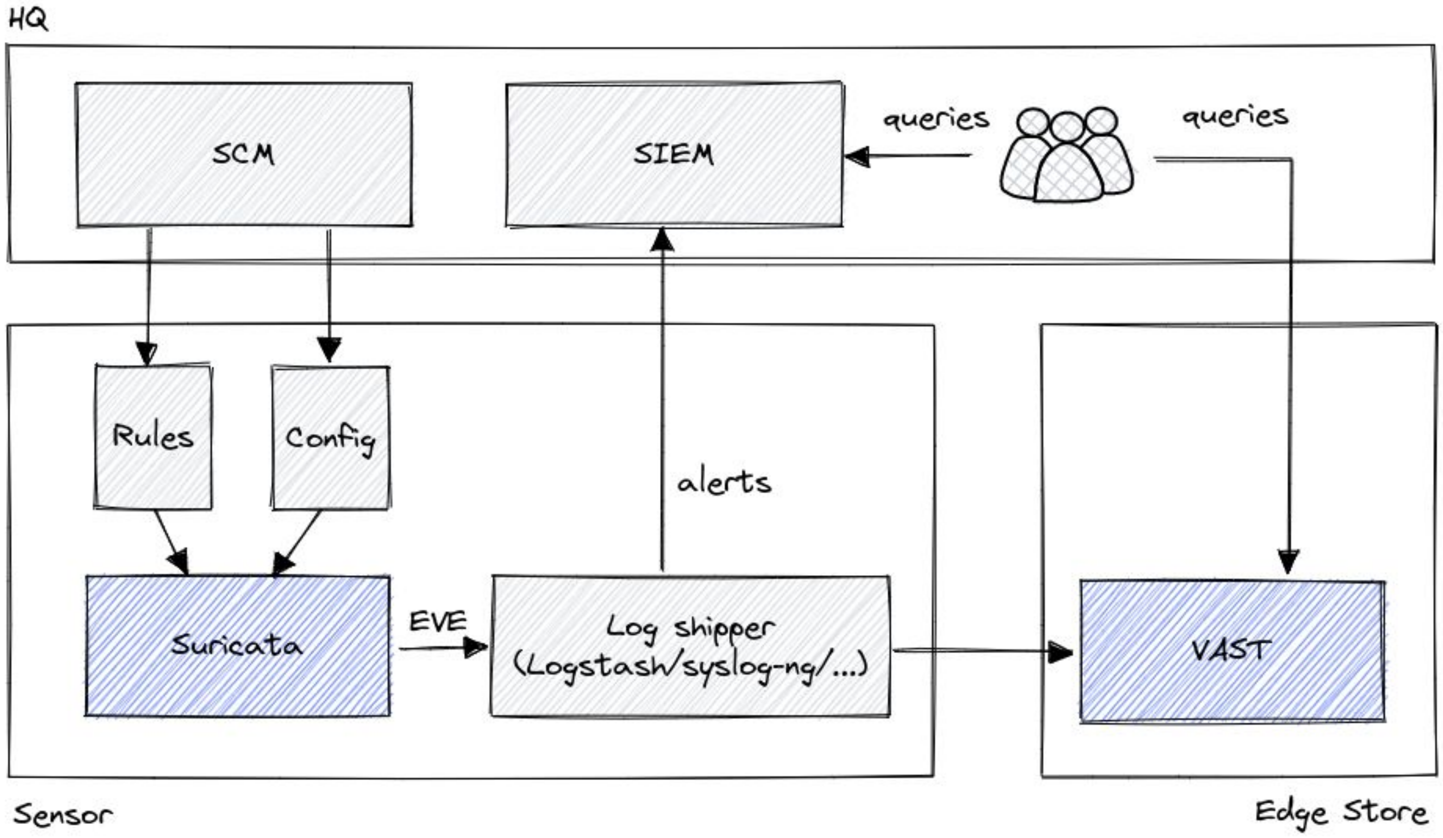
- Use Cases
  - Store alerts (and metadata) and pivot to PCAPs[1]
  - Automated querying (execute security content)[2]
  - Guided threat hunting via notebooks (Suricon 2023 ;-)[3]
  - SIEM offloading for pre-processing and cost savings

[1] Matthias Vallentin, "*Pivot like a Pro: Unified Threat Hunting in Network Security Data*", SuriCon 2019, Amsterdam
[2] Sascha Steinbiss, Matthias Vallentin, "*Distributing Security Content to Detect Threats Across Past, Present and Future*", SuriCon 2021, Boston
[3] https://xkcd.com/541/

Suricata Ingestion



Suricata and VAST at DCSO



EVE-JSON and VAST


```json
{
  "timestamp": "2018-02-16T13:38:56.245600+0100",
  "flow_id": 210421612967555,
  "event_type": "flow",
  "src_ip": "172.31.69.15",
  "src_port": 10897,
  "dest_ip": "131.202.242.193",
  "dest_port": 22,
  "proto": "TCP",
  "flow": {
    "pkts_toserver": 44,
    "pkts_toclient": 42,
    "bytes_toserver": 6968,
    "bytes_toclient": 3100,
    "start": "2018-02-16T13:38:56.245600+0100",
    "end": "2018-02-16T13:39:02.465647+0100",
    "age": 6,
    "state": "new",
    "reason": "timeout",
    "alerted": false
  },
}
```


data

```
type suricata.component.common = record {
  timestamp: timestamp,
  flow_id: count #index=hash,
  src_ip: addr,
  src_port: port,
  dest_ip: addr,
  dest_port: port,
  proto: string,
  event_type: string,
}
type suricata.component.flow = record {
  pkts_toserver: count,
  pkts_toclient: count,
  bytes_toserver: count,
  bytes_toclient: count,
  start: time,
  end: time,
  age: count,
  state: string,
  reason: string,
  alerted: bool
}
type suricata.flow = suricata.component.common + record {
  flow: suricata.component.flow,
  app_proto: string
}
```

schema

Compaction

multiple levels of sophistication

Compaction

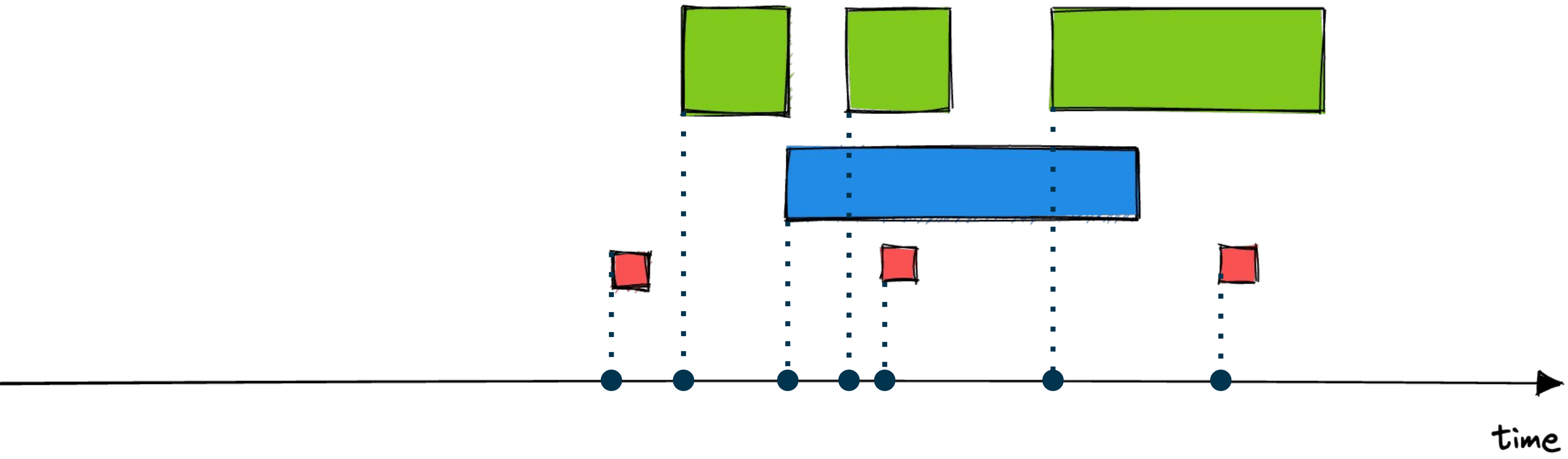
- **Lvl 1**: Keep space usage along target *capacity*
  - Delete data partitions randomly when hitting quota (e.g. 80–90%, < 7 TB)
- **Lvl 2**: Use *age* to delete data with respect to a total order
  - Delete from oldest to newest event timestamps
- **Lvl 3**: Add *event type* as another dimension
  - Delete proportionally to traffic mix (e.g., 40% DNS, 20% flow, 30% DCE/RPC, …)
- **Lvl 4**: Use per-event *weights* to express relative importance
  - Scale age by weight ("virtual age") before evaluating age
- **Lvl 5**: Do not delete, but *compact* data
  - Apply pipeline instead of deleting
- **Lvl 6**: Multi-level compaction
  - Compact compacted types again using different pipelines

Compaction

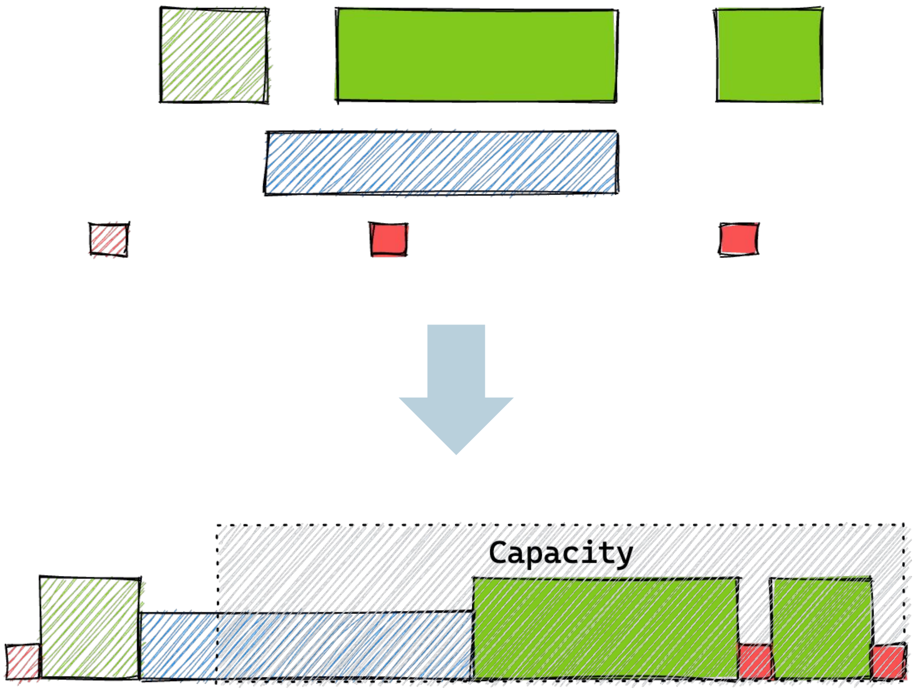
- **Lvl 1**: Keep space usage along target *capacity*
  - Delete data partitions randomly when hitting quota (e.g. 80–90%, < 7 TB)
- **Lvl 2**: Use *age* to delete data with respect to a total order
  - Delete from oldest to newest event timestamps
- **Lvl 3**: Add *event type* as another dimension
  - Delete proportionally to traffic mix (e.g., 40% DNS, 20% flow, 30% DCE/RPC, …)
- **Lvl 4**: Use per-event *weights* to express relative importance
  - Scale age by weight ("virtual age") before evaluating age
- **Lvl 5**: Do not delete, but *compact* data
  - Apply pipeline instead of deleting
- **Lvl 6**: Multi-level compaction
  - Compact compacted types again using different pipelines

TLP:CLEAR

Lvl 2: Age

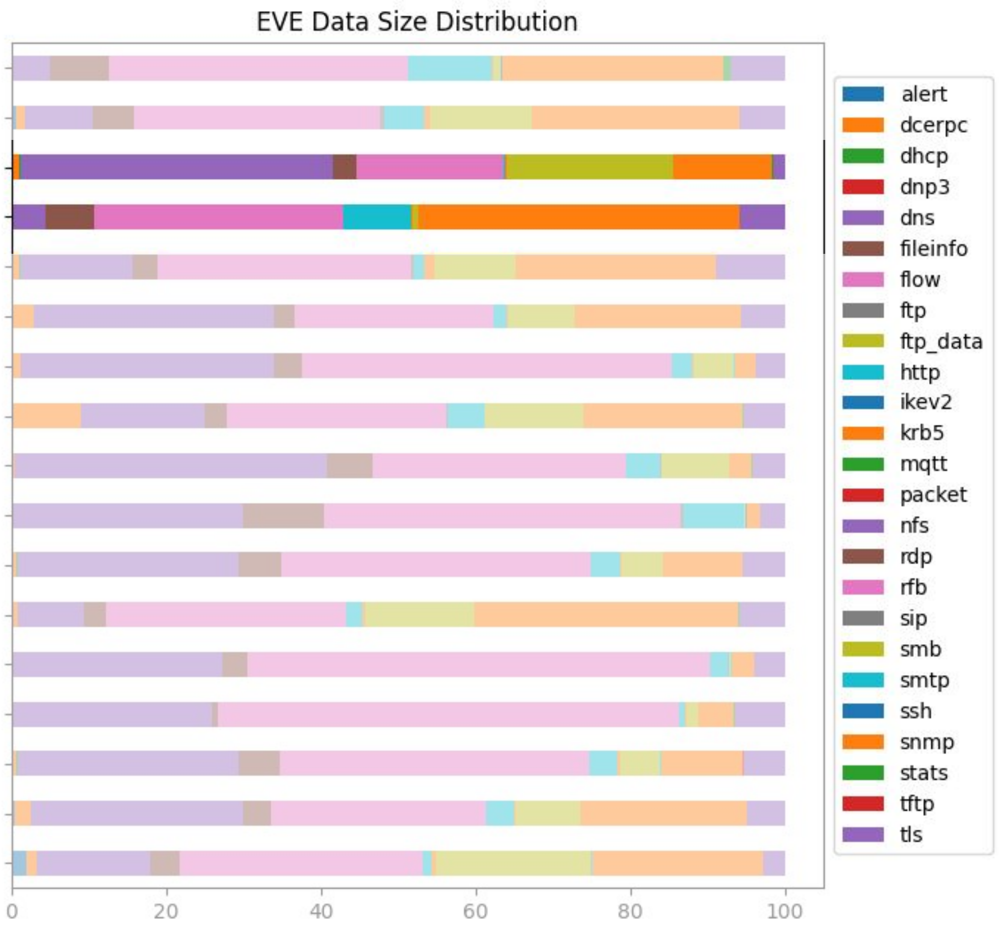


Lvl 2: Linearized View



Compaction

- **Lvl 1**: Keep space usage along target *capacity*
  - Delete data partitions randomly when hitting quota (e.g. 80–90%, < 7 TB)
- **Lvl 2**: Use *age* to delete data with respect to a total order
  - Delete from oldest to newest event timestamps
- **Lvl 3**: Add *event type* as another dimension
  - Delete proportionally to traffic mix (e.g., 40% DNS, 20% flow, 30% DCE/RPC, …)
- **Lvl 4**: Use per-event *weights* to express relative importance
  - Scale age by weight ("virtual age") before evaluating age
- **Lvl 5**: Do not delete, but *compact* data
  - Apply pipeline instead of deleting
- **Lvl 6**: Multi-level compaction
  - Compact compacted types again using different pipelines



Lvl 3: Event Type Distribution

- High variation!
  - Event type volume
  - Event type distribution
- More flexible approach needed to deal with heterogeneous data

Compaction

- **Lvl 1**: Keep space usage along target *capacity*
  - Delete data partitions randomly when hitting quota (e.g. 80–90%, < 7 TB)
- **Lvl 2**: Use *age* to delete data with respect to a total order
  - Delete from oldest to newest event timestamps
- **Lvl 3**: Add *event type* as another dimension
  - Delete proportionally to traffic mix (e.g., 40% DNS, 20% flow, 30% DCE/RPC, …)
- **Lvl 4**: Use per-event *weights* to express relative importance
  - Scale age by weight ("virtual age") before evaluating age
- **Lvl 5**: Do not delete, but *compact* data
  - Apply pipeline instead of deleting
- **Lvl 6**: Multi-level compaction
  - Compact compacted types again using different pipelines

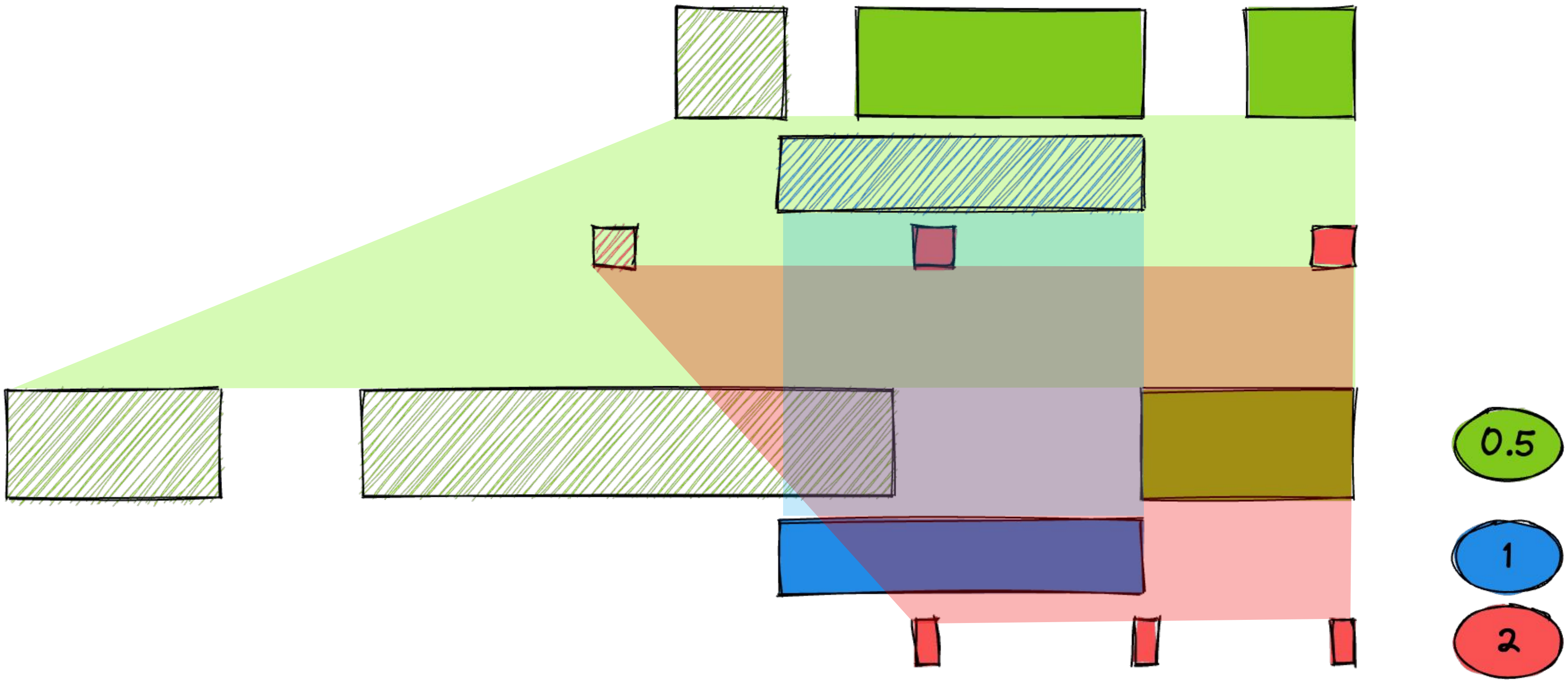
Lvl 4: Virtual Age

- Motivation: empirical event distribution != desired relative event priorities
- Want
  - "My alerts are more important than my DNS events"
  - "Prefer metadata events over mundane flow events, if they exist"
- Not
  - "I want 90% flows and 10% alerts"

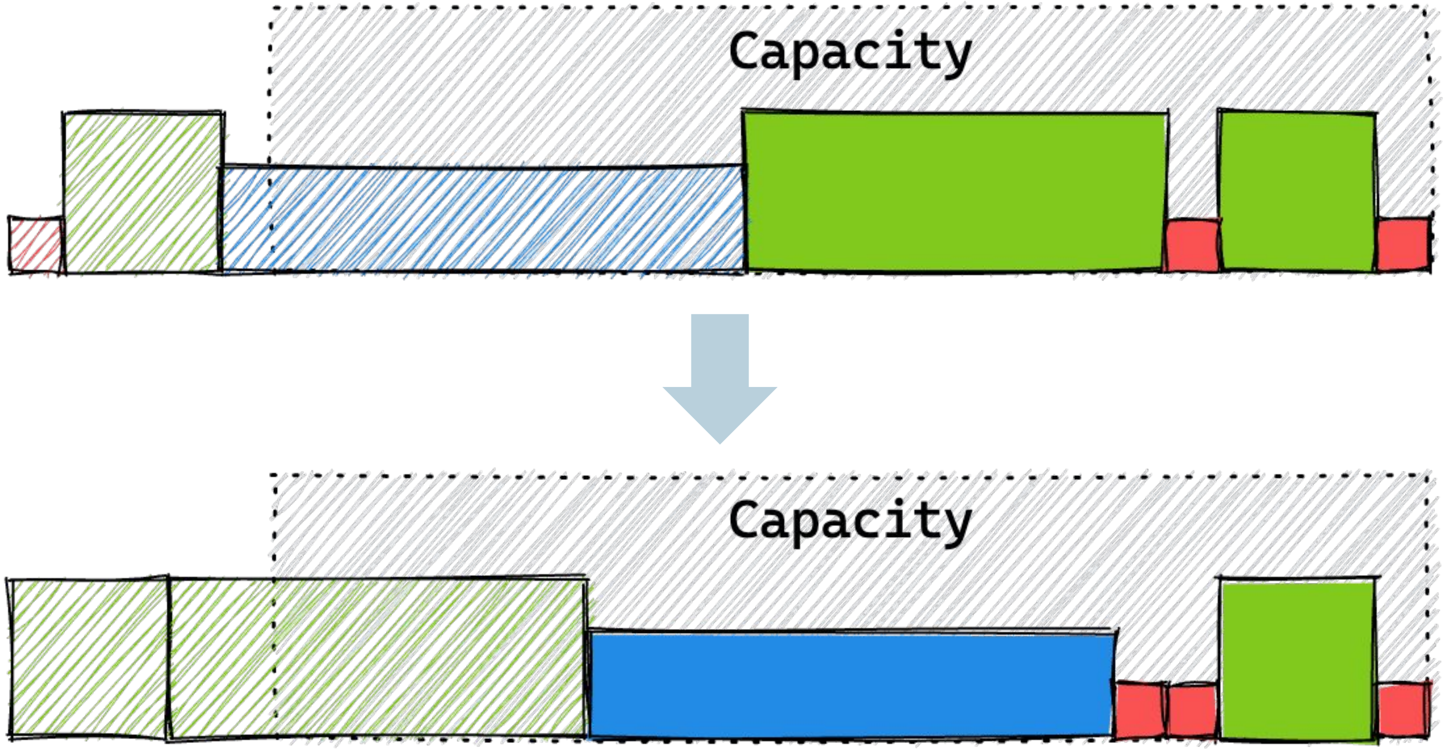
→ **Virtual Age**:

1. Attach a *weight* to each type
2. Adjust event age by projecting into a *virtual space*

Lvl 4: Virtual Age



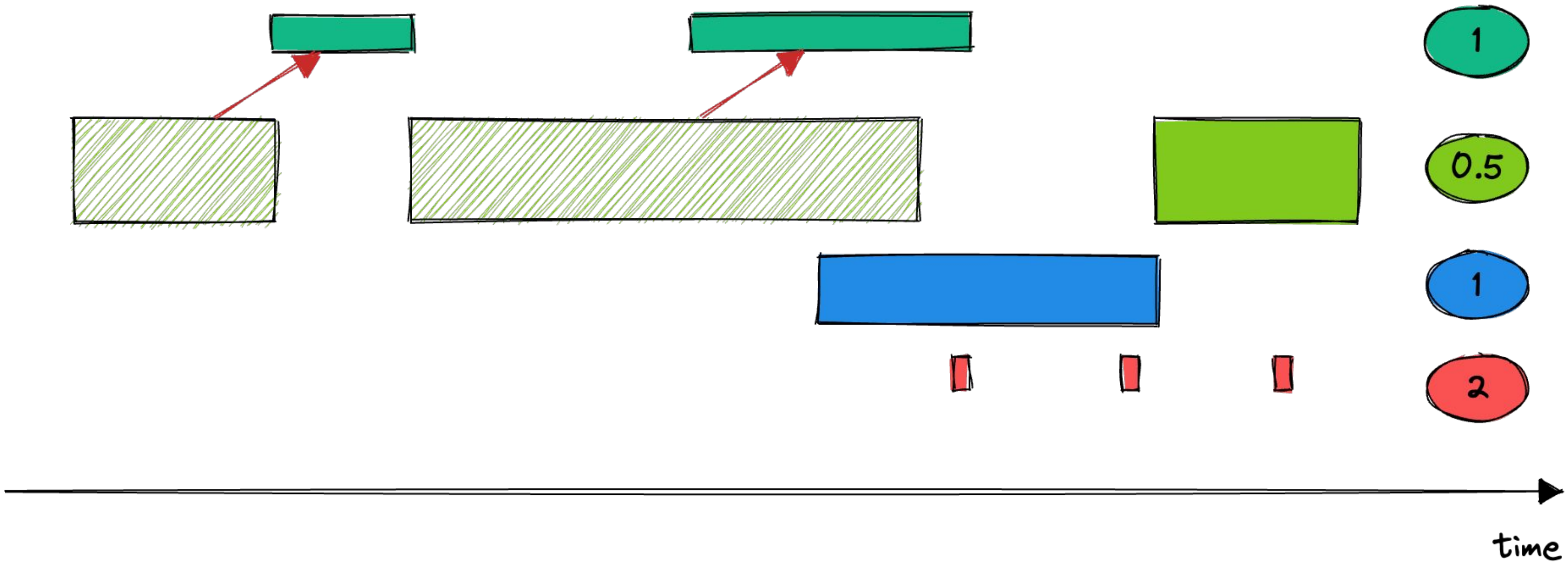
Lvl 4: Linearized View



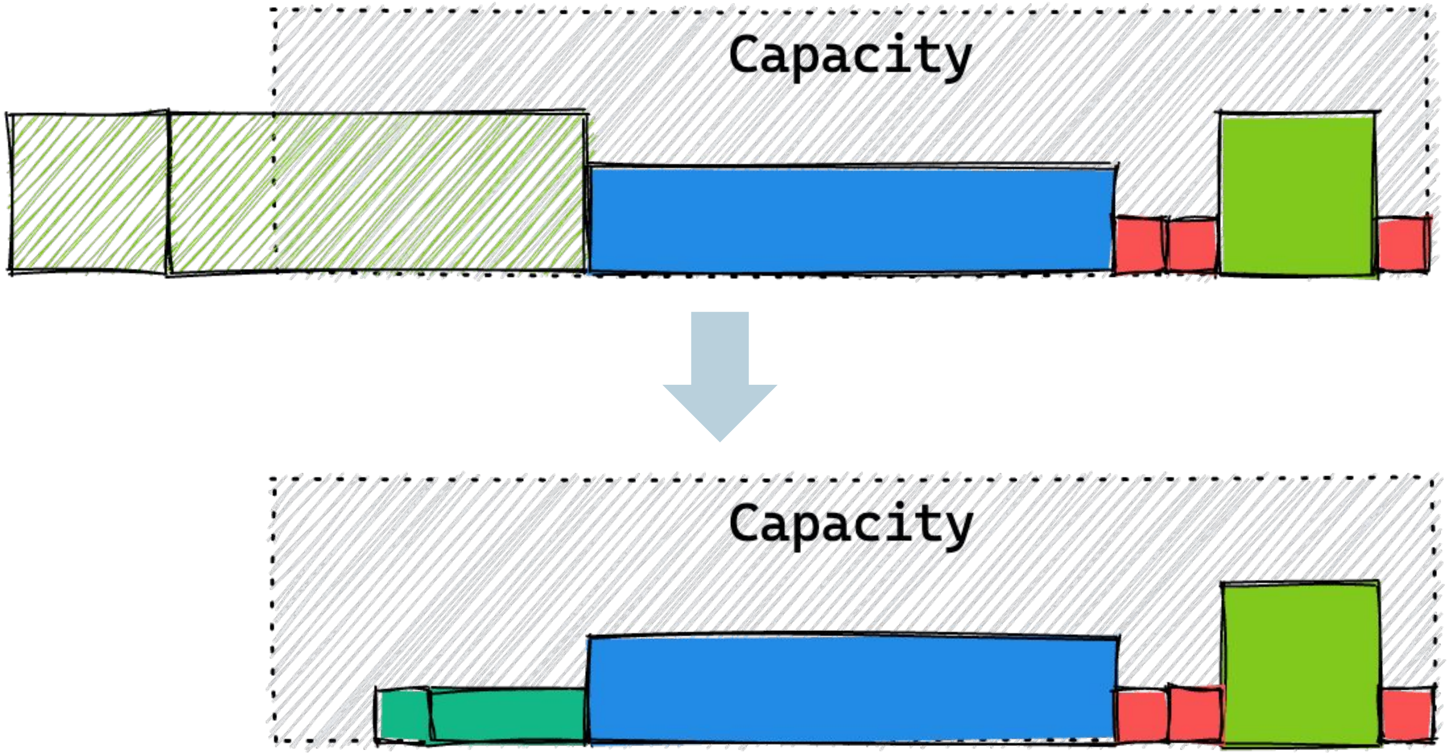
Compaction

- **Lvl 1**: Keep space usage along target *capacity*
  - Delete data partitions randomly when hitting quota (e.g. 80–90%, < 7 TB)
- **Lvl 2**: Use *age* to delete data with respect to a total order
  - Delete from oldest to newest event timestamps
- **Lvl 3**: Add *event type* as another dimension
  - Delete proportionally to traffic mix (e.g., 40% DNS, 20% flow, 30% DCE/RPC, …)
- **Lvl 4**: Use per-event *weights* to express relative importance
  - Scale age by weight ("virtual age") before evaluating age
- **Lvl 5**: Do not delete, but *compact* data
  - Apply pipeline instead of deleting
- **Lvl 6**: Multi-level compaction
  - Compact compacted types again using different pipelines

Lvl 5: Pipeline Transformation

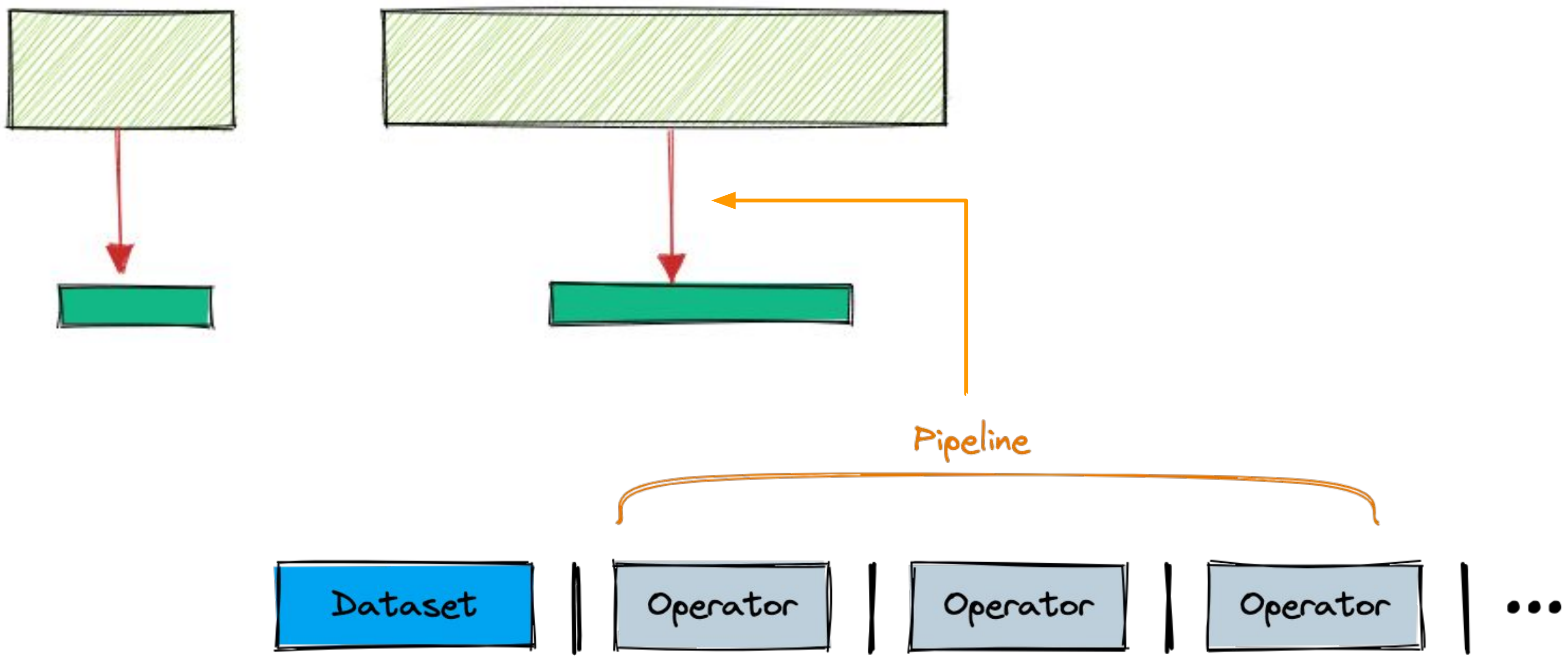


Lvl 5: Linearized View



Implementation

Lvl 5: Dataflow with Pipelines



VAST Docs Blog Changelog Roadmap

https://vast.io/docs/understand/query-language/operators

- About
  - Why VAST
  - Target Audience
  - Vision
  - Use Cases
- Try
- Setup
- Use
- Understand
  - Architecture
  - Data Model
  - Query Language
    - Expressions
    - Pipelines
    - Operators
      - drop
      - replace
      - hash
      - identity
      - rename
      - extend
      - select
      - summarize
      - where
    - Frontends
- Contribute
- Develop

Understand > Query Language > Operators

Operators

VAST ships with the following operators:

drop
Drops individual fields having the configured extractors from the...

replace
Replaces the fields matching the configured extractors with fixe...

hash
Computes a SHA256 hash digest of a given field.

identity
Does nothing with the input. (This operator primarily for testing ...

rename
Renames schemas and fields according to a configured mapping.

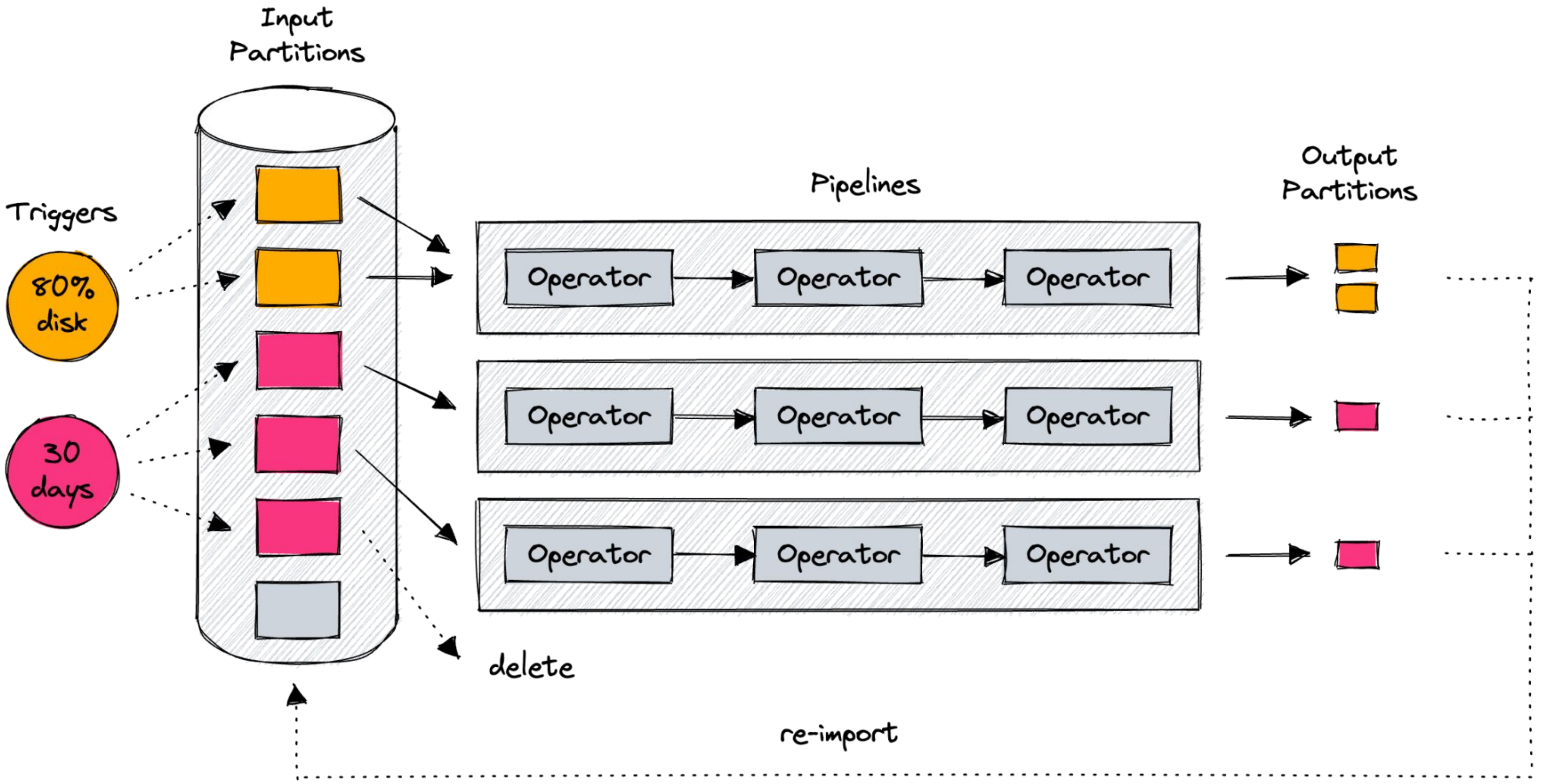
extend
Adds the configured fields with fixed values.

select
Keeps the fields having the configured extractors and removes t...

summarize
The summarize operator bundles input records according to a gr...

where
Keeps rows matching the configured expression and removes th...

Compaction Pipelines



Lvl 2: Rotation Example

```yaml
plugins:
  compaction:
    space:
      interval: 1 hour
      disk-budget-high: 95
      disk-budget-low: 90
      scan-binary: /usr/local/bin/vastdiskbudget
      step-size: 3
    time:
      interval: 1 hour
      rules:
        - after: 90 days
          types: [suricata.dns, suricata.alert, ...]
```

Lvl 5: Flow Aggregation Example

```yaml
plugins:
 compaction:
   space:
     mode: weighted-age
     interval: 1h
     disk-budget-high: 95
     disk-budget-low: 90
     scan-binary:  /usr/local/bin/vastdiskbudget
     step-size: 3
     weights:
       - weight: 1
         pipeline: aggregate-flows
         types:
           - suricata.flow
       - weight: 1
         pipeline: aggregate-dns
         types:
           - suricata.dns
       - weight: 1
         pipeline: aggregate-snmp
         types:
           - suricata.snmp
       - weight: 1
         pipeline: aggregate-smb
         types:
           - suricata.smb
       - weight:
         pipeline: aggregate-http
         types:
           - suricata.http
       - weight: 1
         pipeline: aggregate-tls
         types:
           - suricata.tls
       - weight: 1.5
         types:
           - suricata.flow_agg
           - suricata.dns_agg
           - suricata.snmp_agg
           - suricata.smb_agg
           - suricata.http_agg
           - suricata.tls_agg
```

Lvl 5: Flow Aggregation Example

```yaml
pipelines:
  aggregate-flows:
    - summarize:
        group-by:
          - timestamp
          - src_ip
          - dest_ip
          - dest_port
          - proto
        time-resolution: 1 minute
        aggregate:
          timestamp_min:
            min: timestamp
          timestamp_max:
            max: timestamp
          flow.pkts_toserver: sum
          flow.start: min
          flow.end: max
          flow.alerted: any
          src_port: distinct
          community_id: distinct
          count:
            count: event_type
    - extend:
        fields:
          event_type: flow_agg
    - rename:
        schemas:
          - from: suricata.flow
            to: suricata.flow_agg
```

grouping variables (group-by: timestamp, src_ip, dest_ip, dest_port, proto)

new min/max fields (timestamp_min, timestamp_max)

collect distinct values (src_port, community_id)

new count field (count)

rename event type (extend)

rename schema (rename)

Lvl 5: Flow Aggregation Example


```json
{
  "timestamp": "2020-11-06T07:34:23.000074+0000",
  "flow_id": 1618483912196429,
  "in_iface": "enp175s0f1",
  "event_type": "flow",
  "src_ip": "10.0.0.12",
  "src_port": 53905,
  "dest_ip": "148.0.51.138",
  "dest_port": 53,
  "proto": "UDP",
  "app_proto": "dns",
  "flow": {
    "pkts_toserver": 1,
    "pkts_toclient": 1,
    "bytes_toserver": 88,
    "bytes_toclient": 136,
    "start": "2022-06-29T07:31:23.458630+0000",
    "end": "2022-06-29T07:31:23.394739+0000",
    "age": 0,
    "state": "established",
    "reason": "timeout",
    "alerted": false
  },
  "host": "53c5bbd7849b48e1a458347df93c0dc5",
}
```

Lvl 5: Flow Aggregation Example


```json
{
  "timestamp": "2020-11-06T07:34:23.000074+0000",
  "flow_id": 1618483912196429,
  "in_iface": "enp175s0f1",
  "event_type": "flow",
  "src_ip": "10.0.0.12",
  "src_port": 53905,
  "dest_ip": "148.0.51.138",
  "dest_port": 53,
  "proto": "UDP",
  "app_proto": "dns",
  "flow": {
    "pkts_toserver": 1,
    "pkts_toclient": 1,
    "bytes_toserver": 88,
    "bytes_toclient": 136,
    "start": "2022-06-29T07:31:23.458630+0000",
    "end": "2022-06-29T07:31:23.394739+0000",
    "age": 0,
    "state": "established",
    "reason": "timeout",
    "alerted": false
  },
  "host": "53c5bbd7849b48e1a458347df93c0dc5",
}
```


min/max

distinct

sum

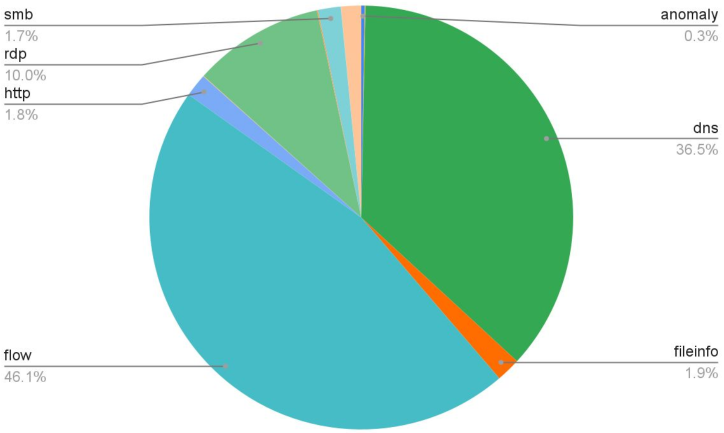
min

max


```json
{
  "timestamp": "2022-06-29T07:34:00.000000",
  "timestamp_max": "2022-06-29T07:34:57.373881",
  "timestamp_min": "2022-06-29T07:34:00.348758",
  "src_ip": "10.0.0.12",
  "dest_ip": "148.0.51.138",
  "src_port": [
    53905,
    …
    61966,
    62814,
    63753
  ],
  "dest_port": 53,
  "community_id": [],
  "count": 14,
  "flow.alerted": false,
  "flow.pkts_toclient": 14,
  "flow.pkts_toserver": 14,
  "flow.bytes_toclient": 2833,
  "flow.bytes_toserver": 1158,
  "flow.start": "2022-06-29T07:31:23.458630",
  "flow.end": "2022-06-29T07:33:16.222173",
  "event_type": "flow_agg"
}
```

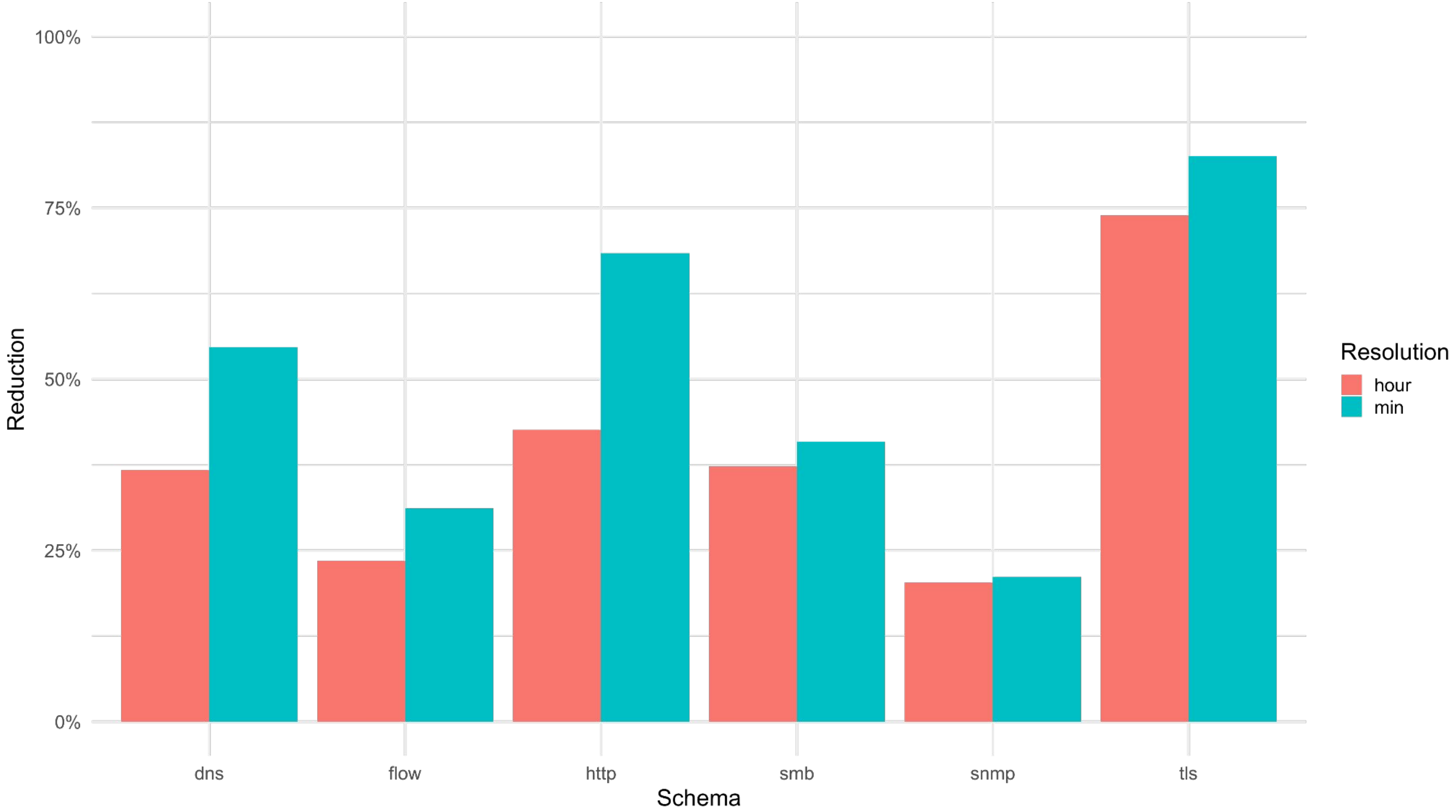
Space Reduction

- What space reduction can we achieve if we compact a typical dataset?
- Example
  - 3 days of "Realistic Cyber Defense Dataset" CSE-CIC-IDS2018[1]
  - Suricata 7, all EVE output options enabled
  - ~150GB pcaps, ~42M events
- Parameters
  - Event types compacted:
    - smb, dns, http, flow, snmp, tls
  - Two resolutions evaluated
    - 1 hour
    - 1 minute



[1] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani: "*Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, 2018

Space Reduction by Aggregation



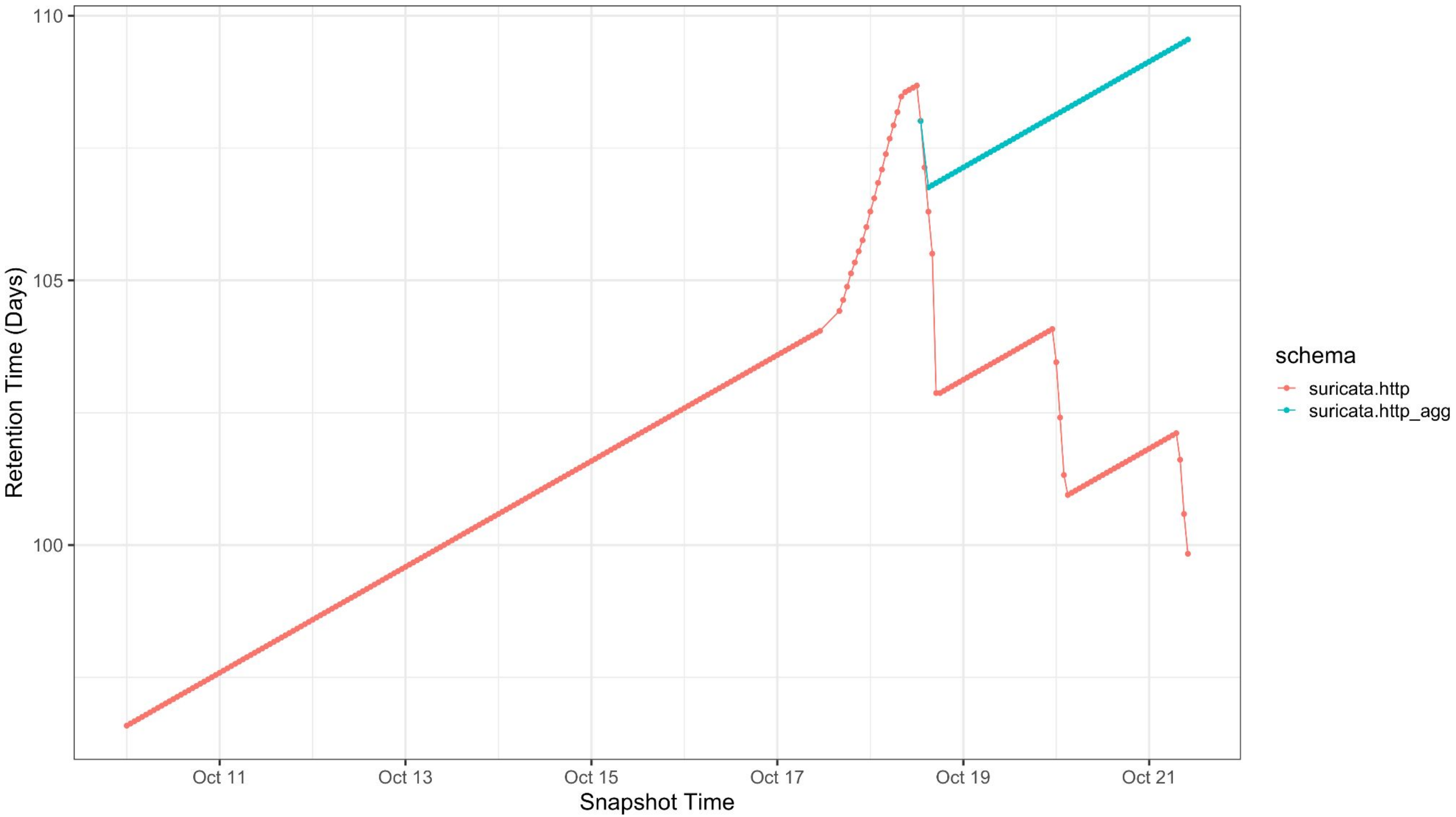
Space Reduction by Projection


```
{
	"timestamp":
"2018-02-16T18:17:17.044699",
	"flow_id": 1599357715314279,
	"pcap_cnt": 55512743,
	"vlan": null,
	"in_iface": null,
	"src_ip": "196.52.43.92",
	"src_port": 6712,
	"dest_ip": "172.31.64.32",
	"dest_port": 161,
	"proto": "UDP",
	"event_type": "snmp",
	"community_id": null,
	"tx_id": null,
	"snmp": {
	"version": 1,
	"pdu_type": "get_request",
	"vars": [
		"1.3.6.1.2.1.1.1.0"
	],
	"community": "public"
	}
}
```

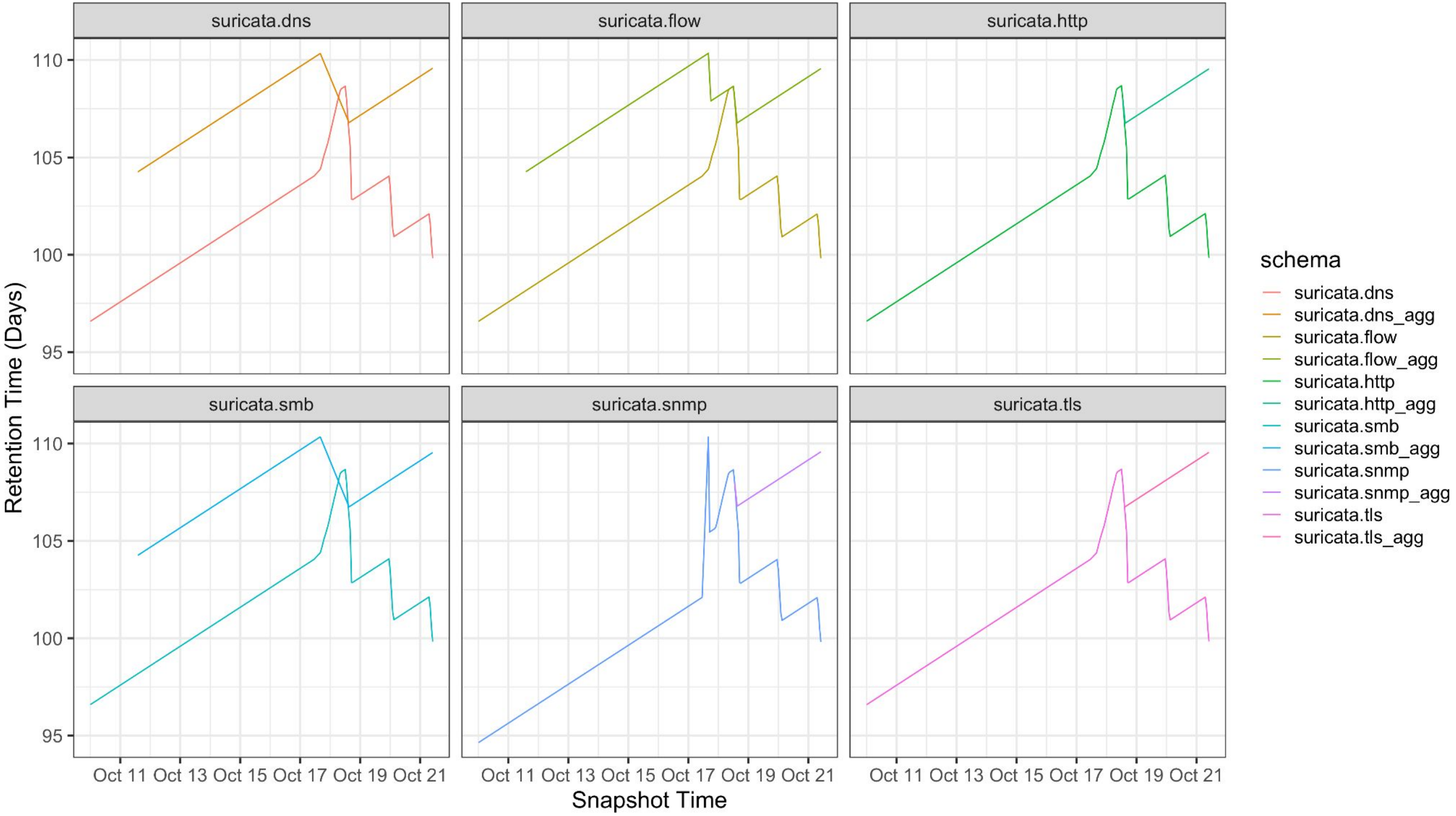

→


```
{
	"timestamp":
"2018-02-16T18:17:00.000000",
	"src_ip": "196.52.43.92",
	"dest_ip": "172.31.64.32",
	"dest_port": 161,
	"proto": "UDP",
	"community_id": [],
	"count": 1,
	"timestamp_max":
"2018-02-16T18:17:17.044699",
	"timestamp_min":
"2018-02-16T18:17:17.044751",
	"event_type": "snmp_agg"
}
```

Real-world measurements



Effect on Retention Time



Compaction

- **Lvl 1**: Keep space usage along target *capacity*
  - Delete data partitions randomly when hitting quota (e.g. 80–90%, < 7 TB)
- **Lvl 2**: Use *age* to delete data with respect to a total order
  - Delete from oldest to newest event timestamps
- **Lvl 3**: Add *event type* as another dimension
  - Delete proportionally to traffic mix (e.g., 40% DNS, 20% flow, 30% DCE/RPC, …)
- **Lvl 4**: Use per-event *weights* to express relative importance
  - Scale age by weight ("virtual age") before evaluating age
- **Lvl 5**: Do not delete, but *compact* data
  - Apply pipeline instead of deleting
- **Lvl 6**: Multi-level compaction
  - Compact compacted types again using different pipelines

Thank you!

VAST

vast.io

Join our Community Slack!

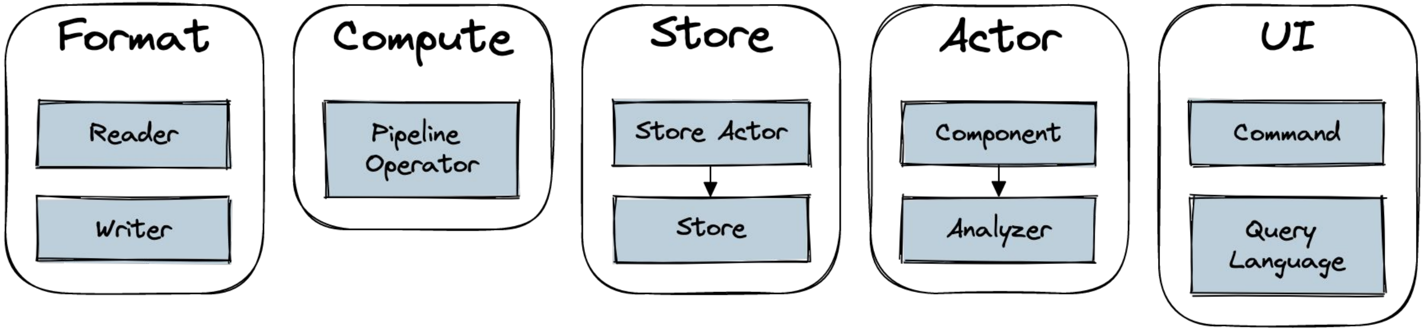
slack.tenzir.com

Meerkats will prevail!

Backup

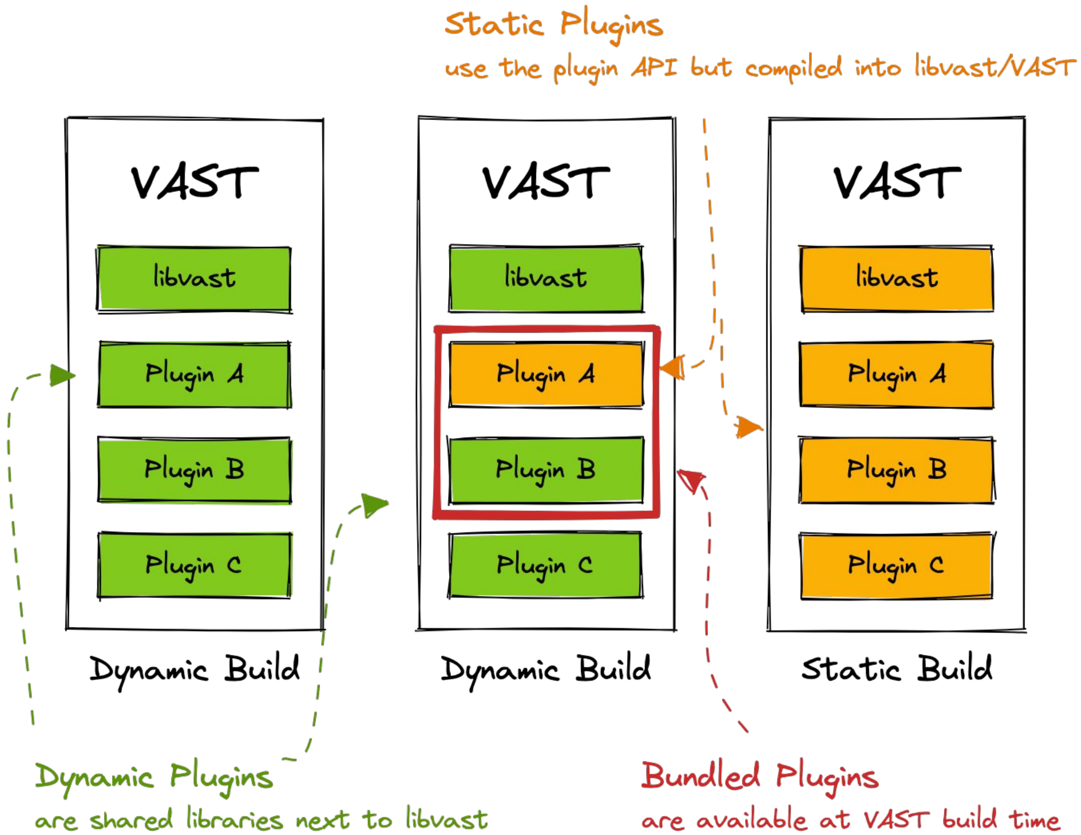
VAST Plugins

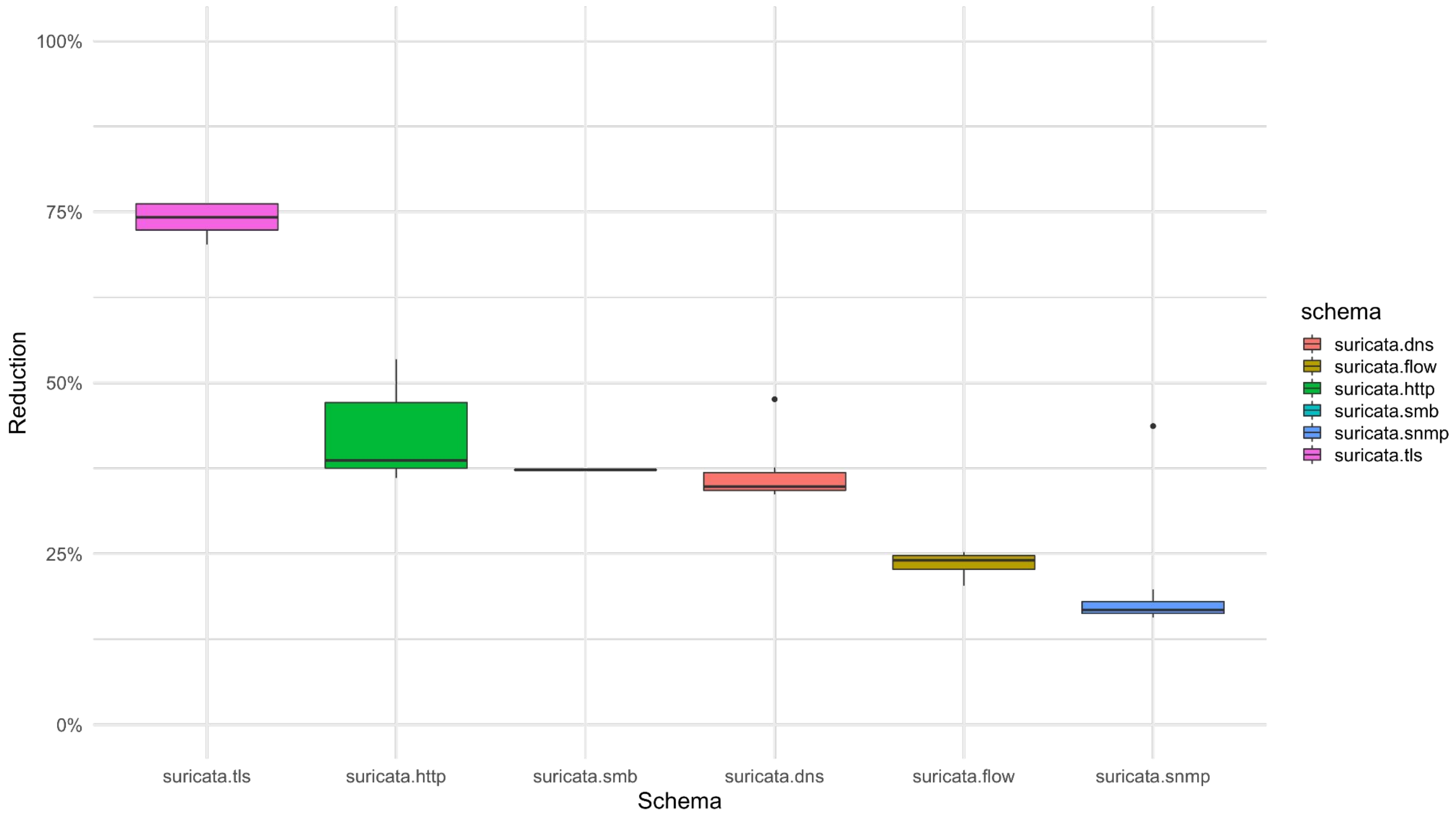
- Plugin architecture



- Easy to implement missing transformation functionality
  - Pipeline operators
  - Aggregation functions

VAST Plugins




100%
75%
50%
25%
0%
Reduction
suricata.tls
suricata.http
suricata.smb
suricata.dns
suricata.flow
suricata.snmp
Schema
schema
suricata.dns
suricata.flow
suricata.http
suricata.smb
suricata.snmp
suricata.tls