Data Log Management for Cyber-Security Programmability of Cloud Services and Applications

ALESSANDRO CARREGA, **MATTEO REPETTO**

CNIT, NATIONAL LABORATORY OF SMART AND SECURE NETWORKS (S2N), GENOA (ITALY)

ALESSANDRO.CARREGA@CNIT.IT, MATTEO.REPETTO@CNIT.IT

Outline

- Cyber-security challenges
- New computing paradigms and software orchestration
- Concept and framework
- ASTRID architecture
- The Context Fabric and Context Abstraction
- Preliminary results
- Conclusions

Cyber-security challenges

Visibility
- dynamic and evolving systems
- partially unknown topologies

Efficiency
- limited computing resources
- overlapping appliances

Humans in the loop
- slow or ineffective reaction
- lack of skills/expertise/care
- manual and error-prone processes

Targets
- data
- service providers

Weak links in the chain
- social engineering, IoT
- outdated sw, weak configurations

Attacks
- stealthy and persistent
- multi-vector
- AI

New computing paradigms

Proliferation of cloud providers and services

openstack

aws

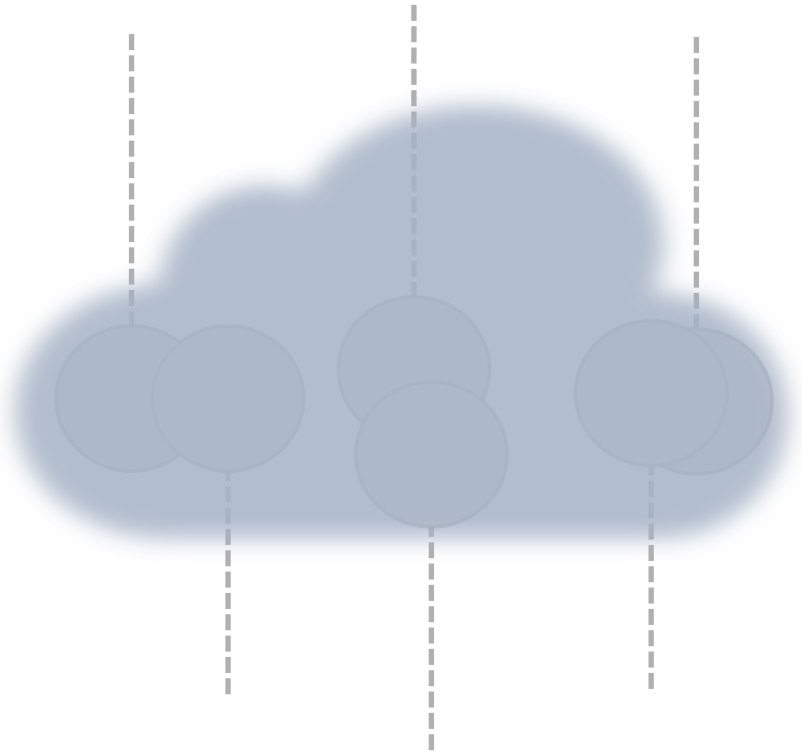
Google Cloud

IBM Cloud

Azure

vmware vSphere

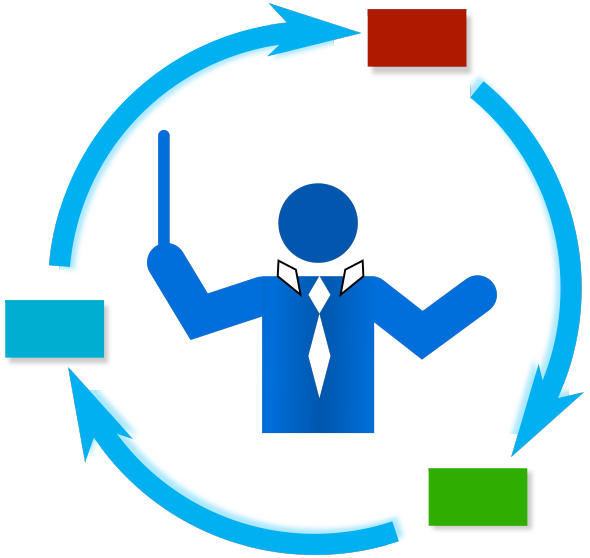
rackspace



New computing paradigms

Heterogeneity of infrastructures and services

- no common and uniform Security-as-a-Service APIs in CMS
- low cross-cloud interoperability



Emergence of software orchestration paradigms

- more agility for security management

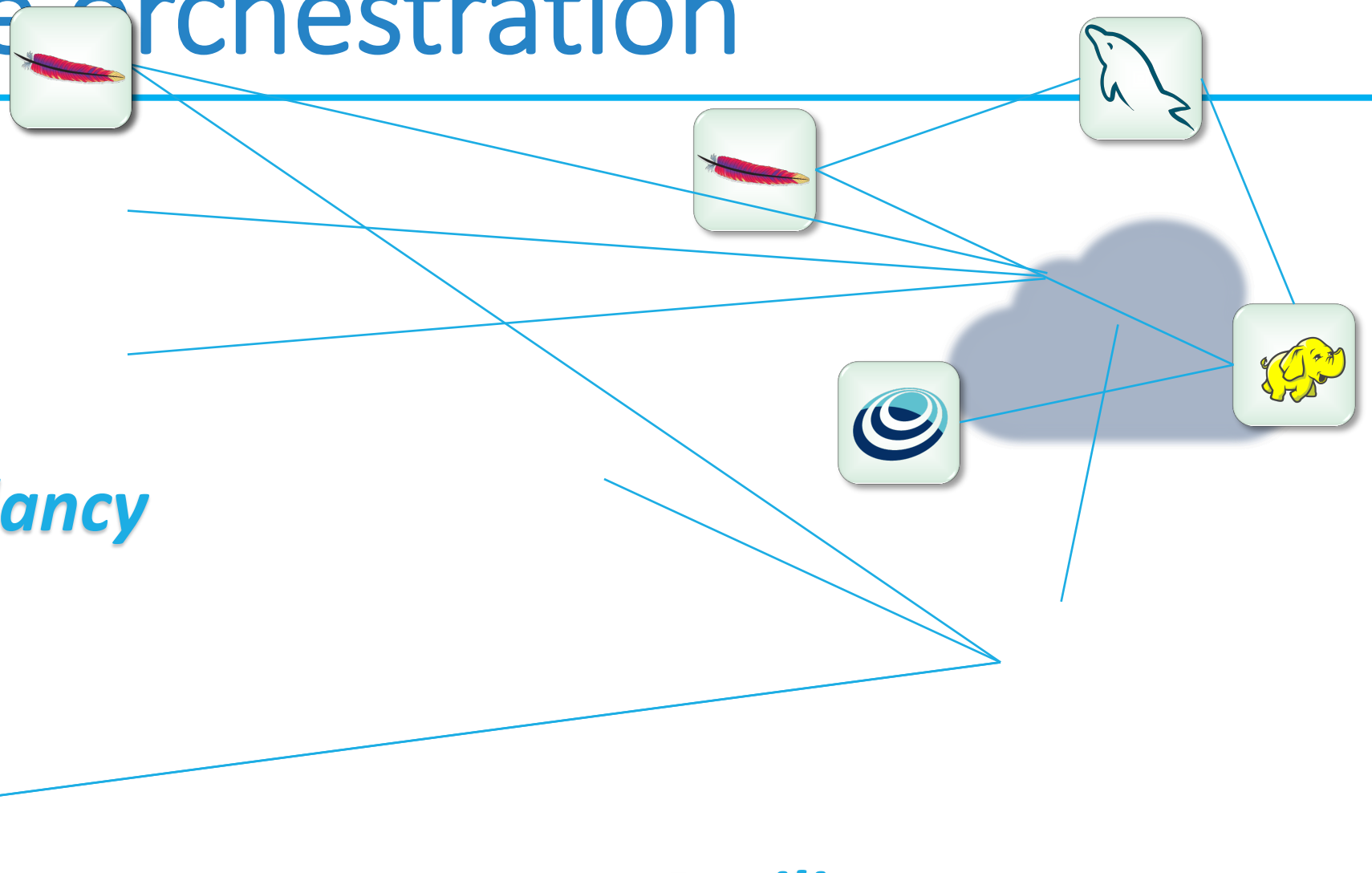
OPEN BATON

JUJU

kubernetes

CLOUDIATOR

Service orchestration

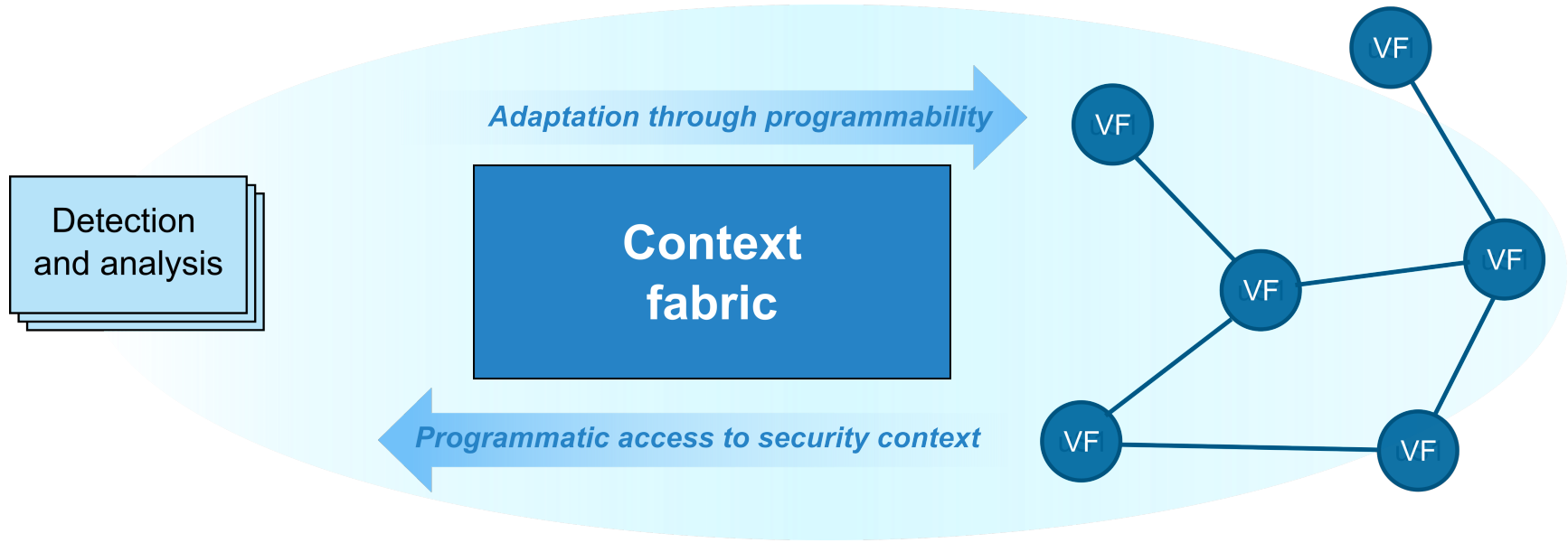


Elasticity

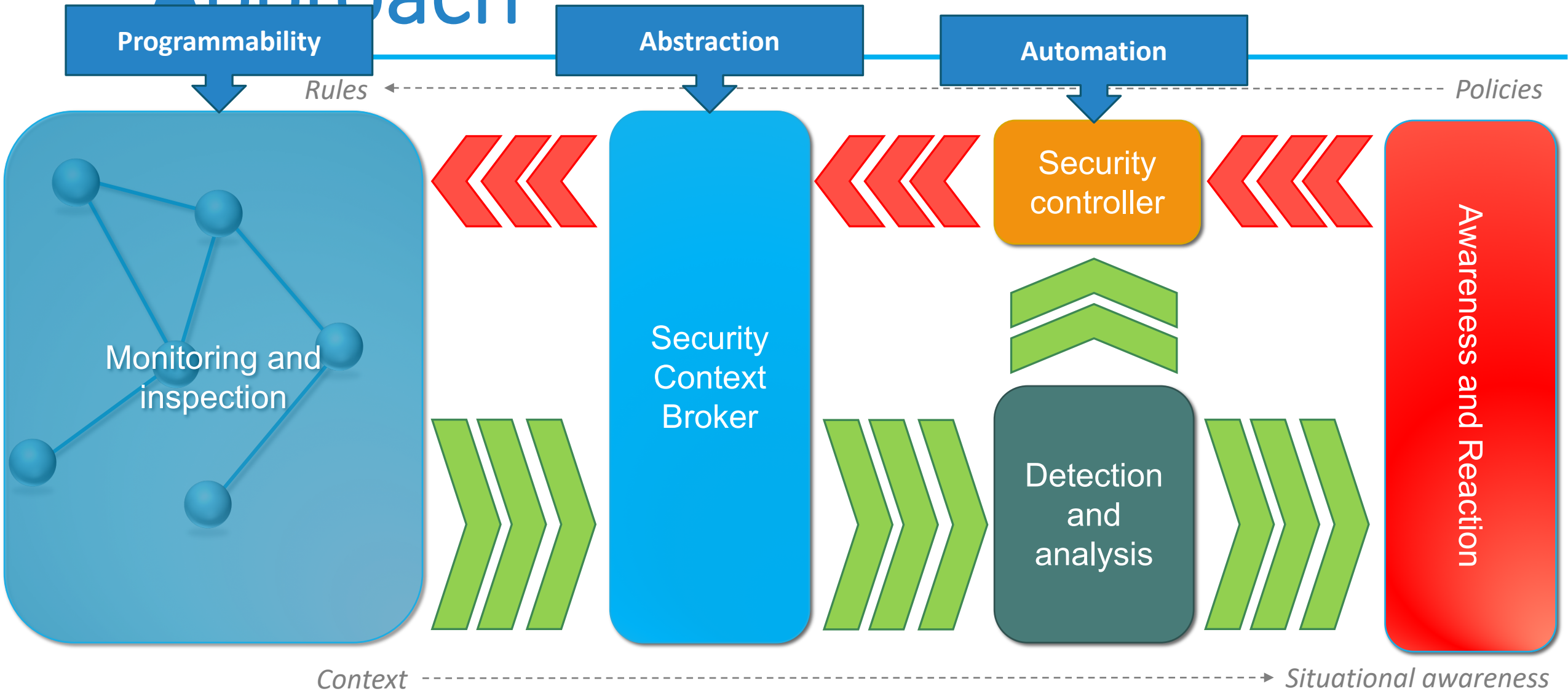
Redundancy

Resilience

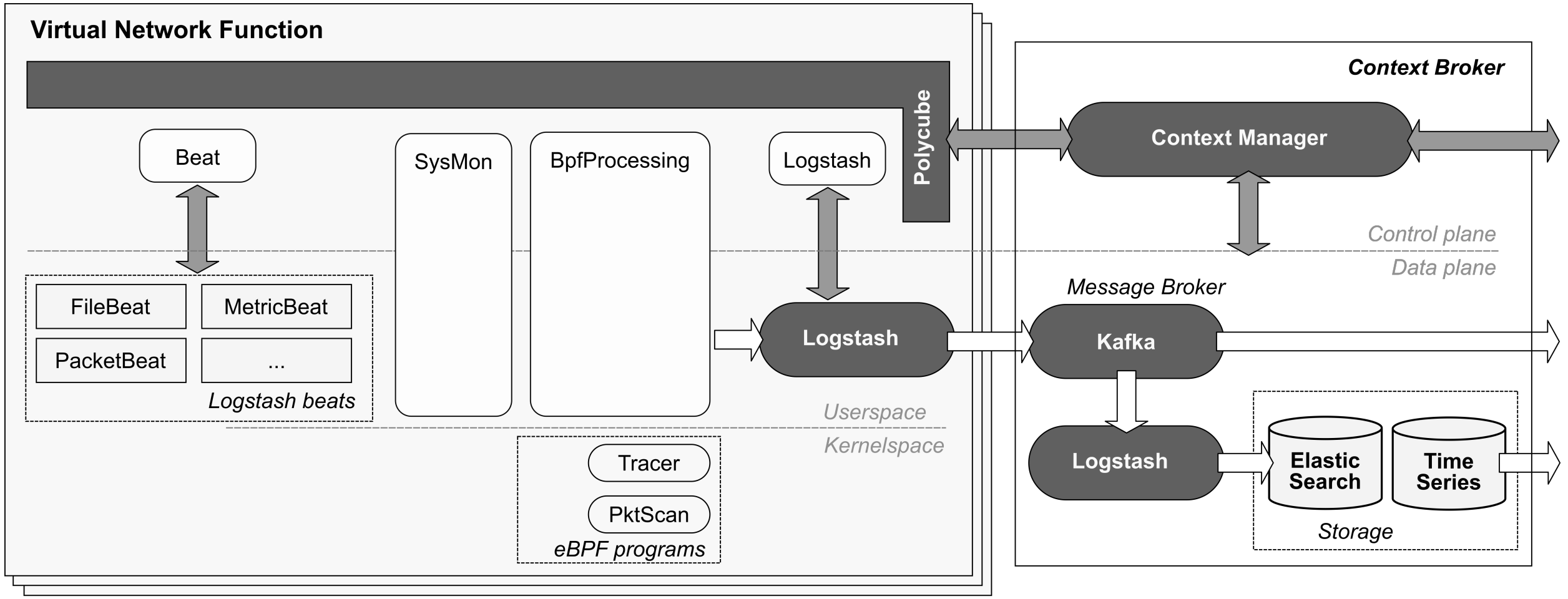
Concept


Detection and analysis
Adaptation through programmability
Context fabric
Programmatic access to security context
VF
VF
VF
VF
VF
VF

Approach


Programmability
Abstraction
Automation
Rules
Policies
Monitoring and inspection
Security Context Broker
Security controller
Detection and analysis
Awareness and Reaction
Context
Situational awareness

The Context Fabric



The Context Fabric - Implementation

Elastic Stack for collecting and storing data

- Beats for **capturing the context**
  - standard beats: FileBeat, MetricBeat, PacketBeat
  - enhanced beats (work in progress): CubeBeat, SysBeat, NetdataBeat
- Logstash for **local processing pipelines**
  - aggregation, filtering, timestamp insertion, geolocalization, anonymization, removal of private and sensitive data, fingerprinting, serialization, name resolution, classification, encryption, parsing, and creation of structured data, …
- Logstash for **central processing pipelines**
  - timestamp insertion, fingerprinting, aggregation, …
- Elasticsearch for **metadata**
- Time series DB for **historical data**

The Context Fabric - Implementation

Kafka for **delivering real-time data** to multiple consumers

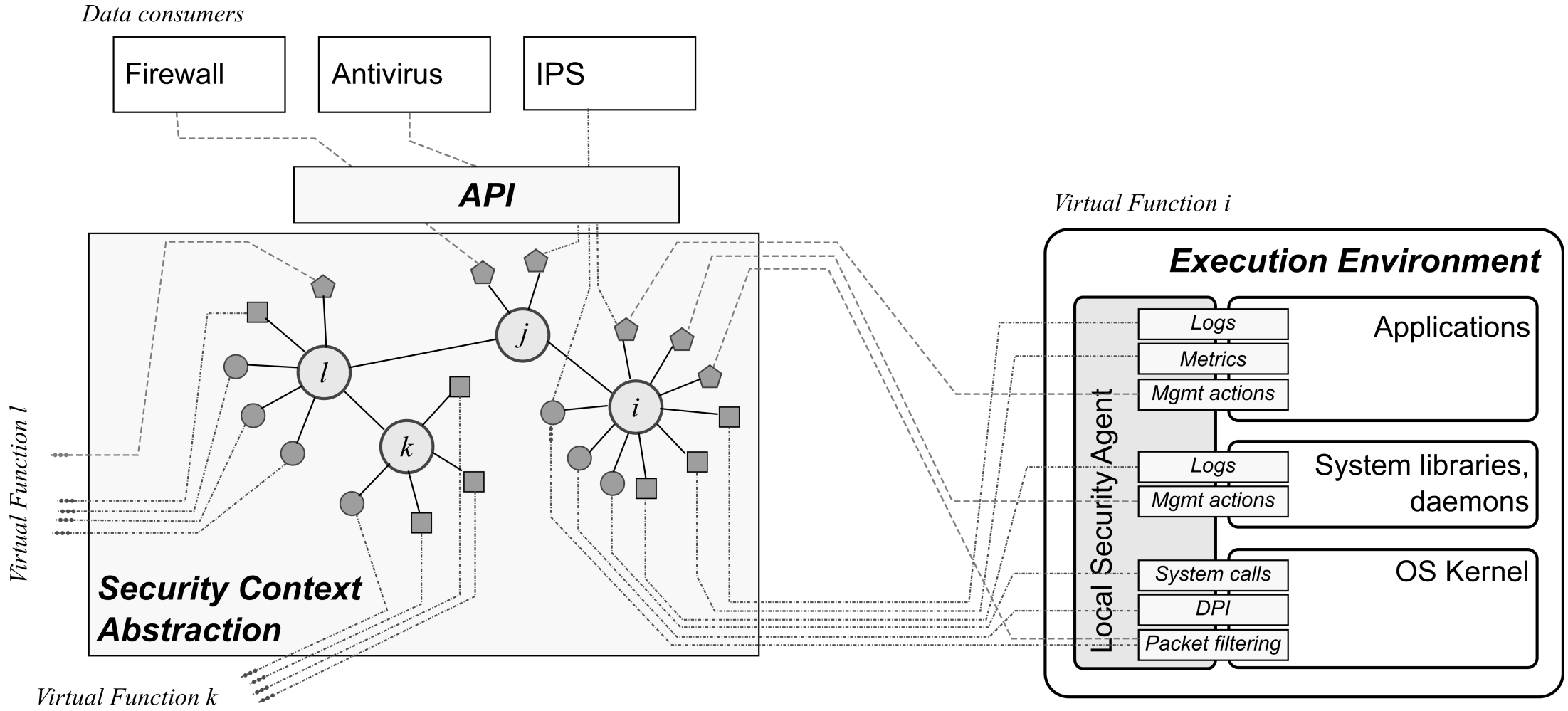
- detection algorithms, forensics DB, Elastictree, …

Context Manager for **programming local agents**

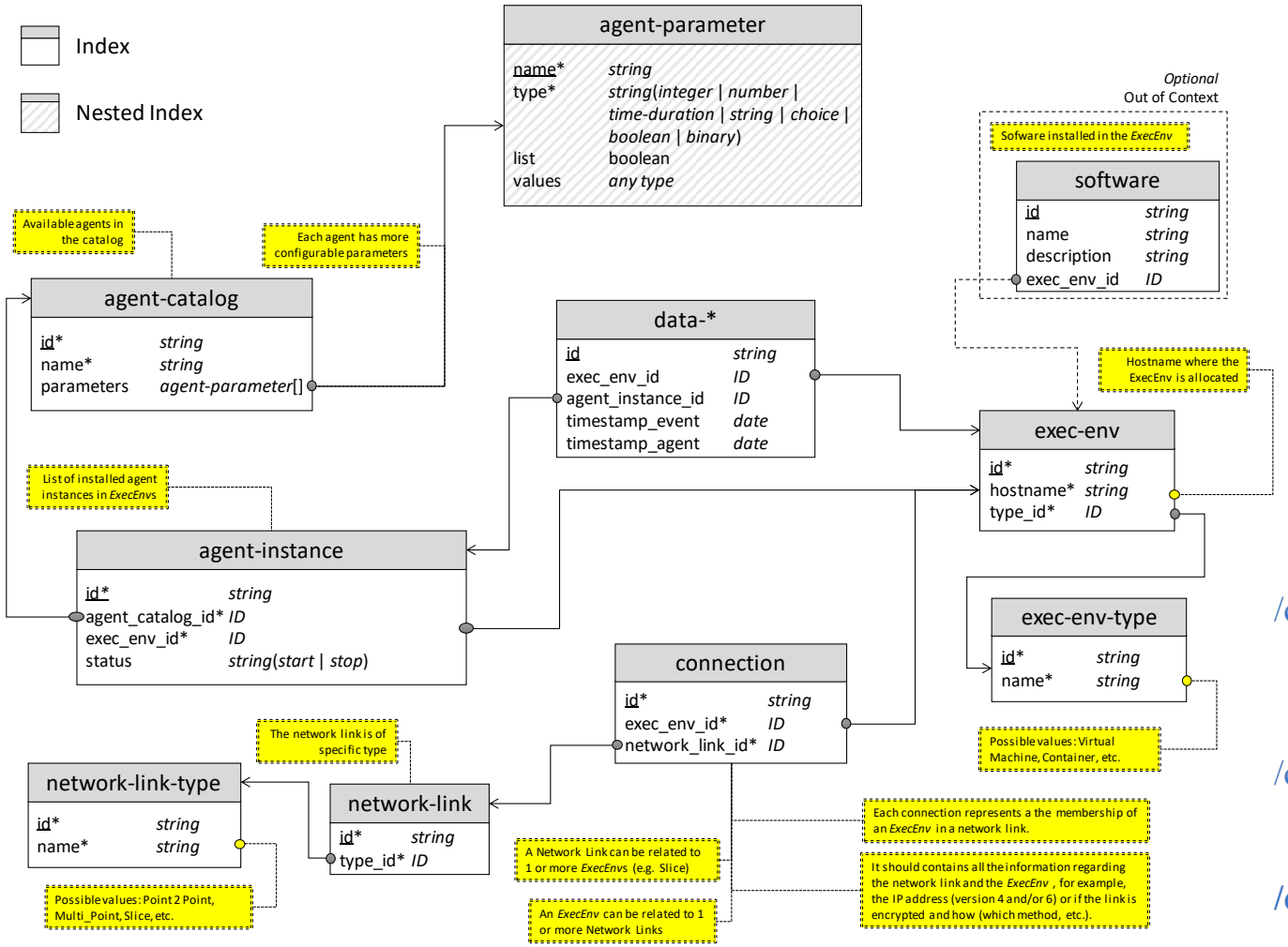
Polycube as **Local control framework**

- changes the configuration of inspection tools
  - beats, scripts,
- injects code
  - eBPF programs, Logstash pipelines

Context Abstraction

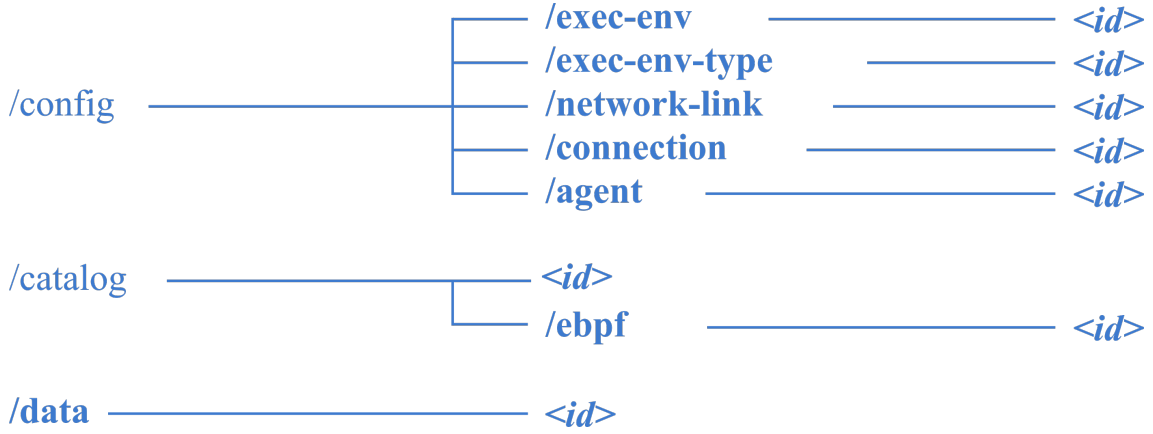


Context Abstraction – API



REST API to access the context:

- service topology
  - VMs, links, OS, libraries
- local agents
  - configurations, commands
- data
  - origin, timestamp, location



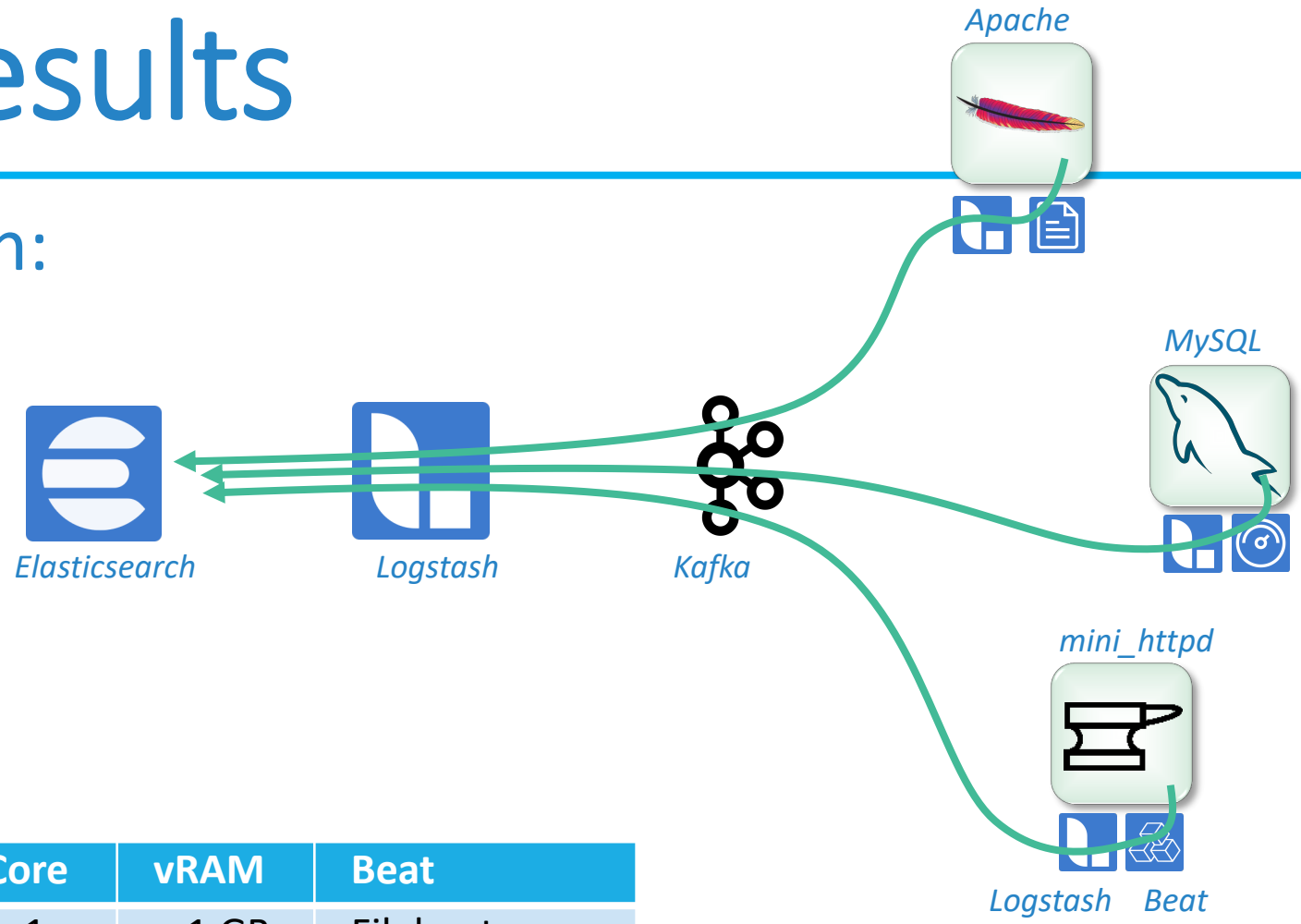
Preliminary results

Overhead on service execution:

- CPU, memory, network

Parameters:

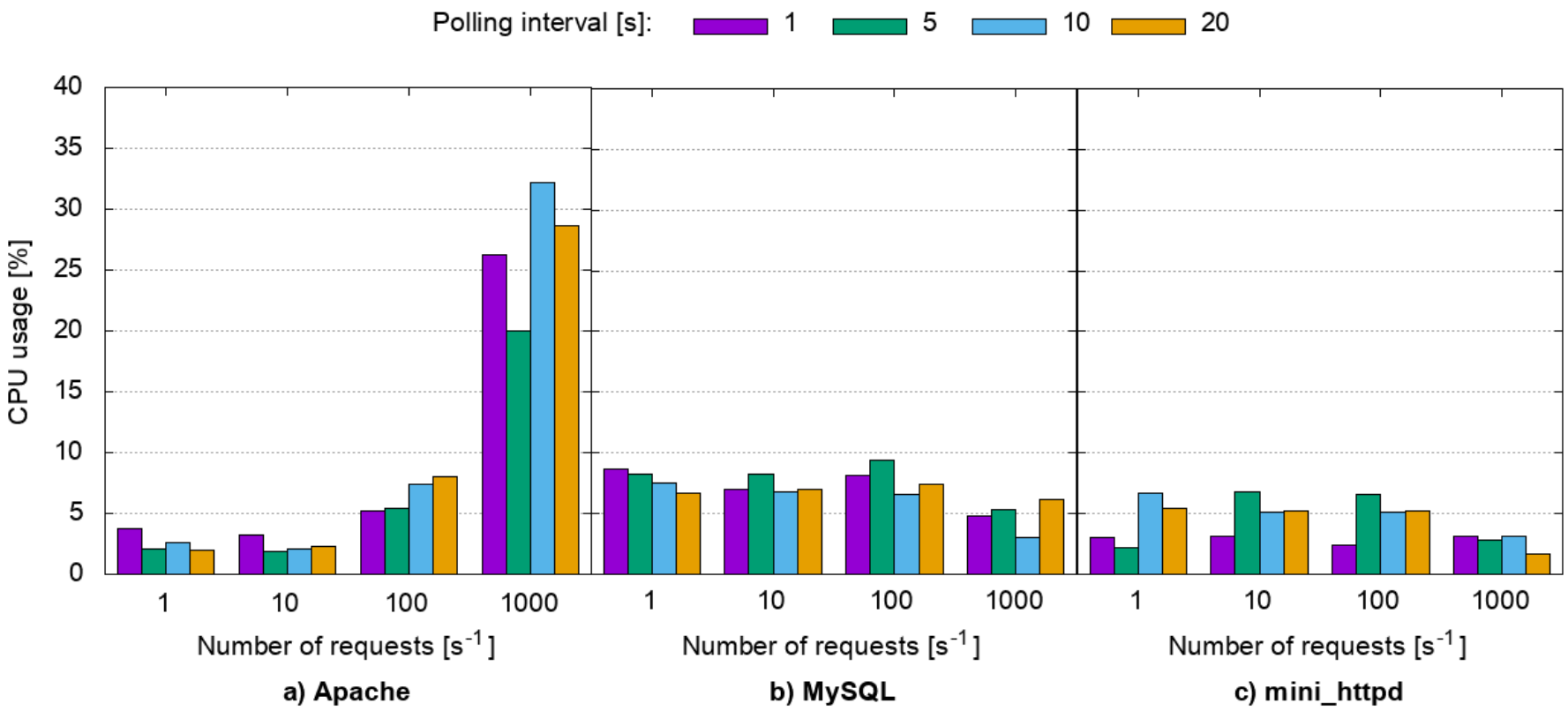
- workload
  - 10, 100, 500, 1000 1/s
- polling time
  - 1, 50, 500, 1000 s



Function	Operating System	vCore	vRAM	Beat
Apache	Debian 4.9	1	1 GB	Filebeat
MySQL	Debian 4.9	1	1 GB	Metricbeat
mini httpd	Ubuntu 18.04.2 LTS	2	2 GB	Cubebeat
CB	Debian 4.9	4	4 GB	—

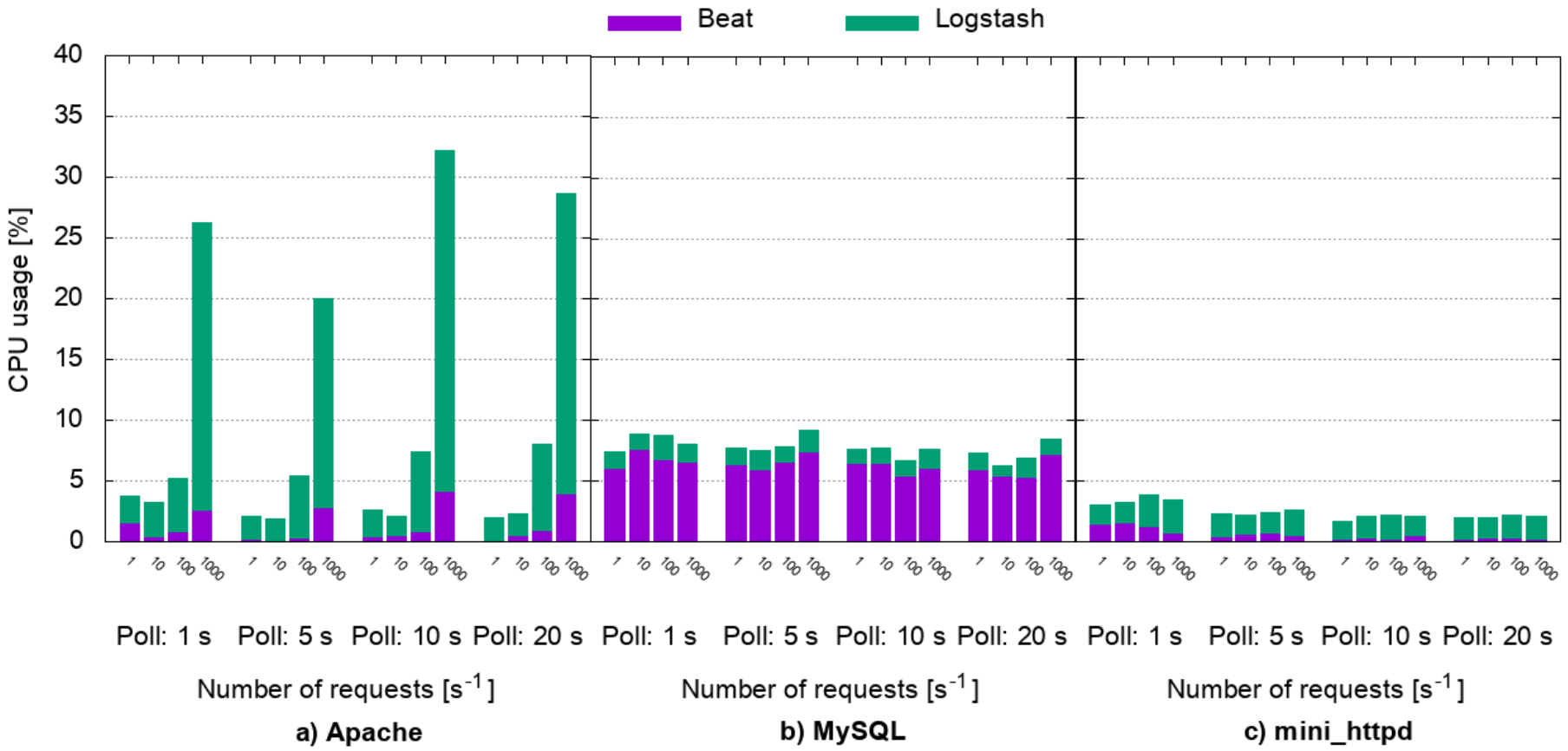
Preliminary results – CPU usage

Total CPU usage by local agents (Beat + Logstash)



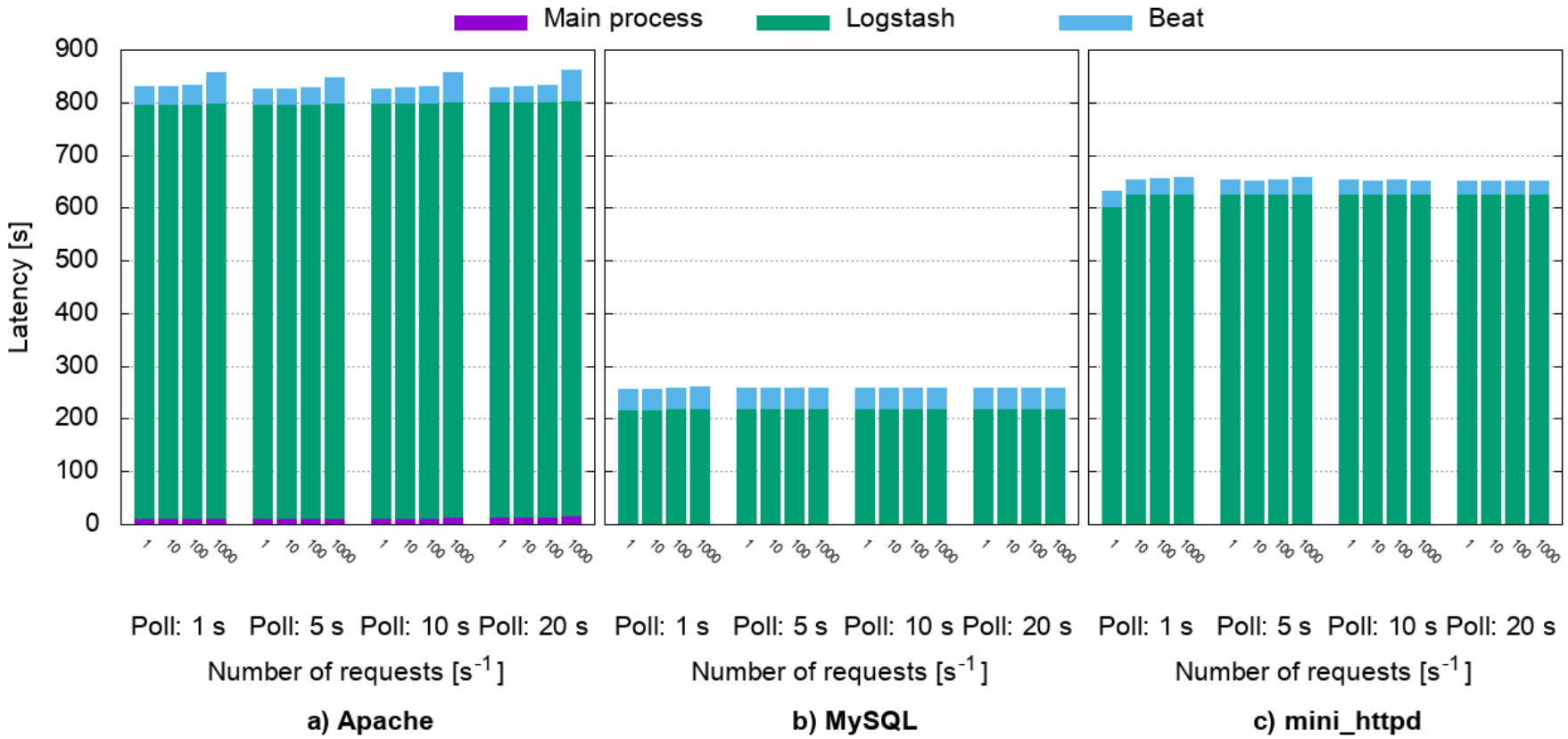
Preliminary results – CPU usage

Breakdown of CPU usage by local agents (Beat, Logstash)



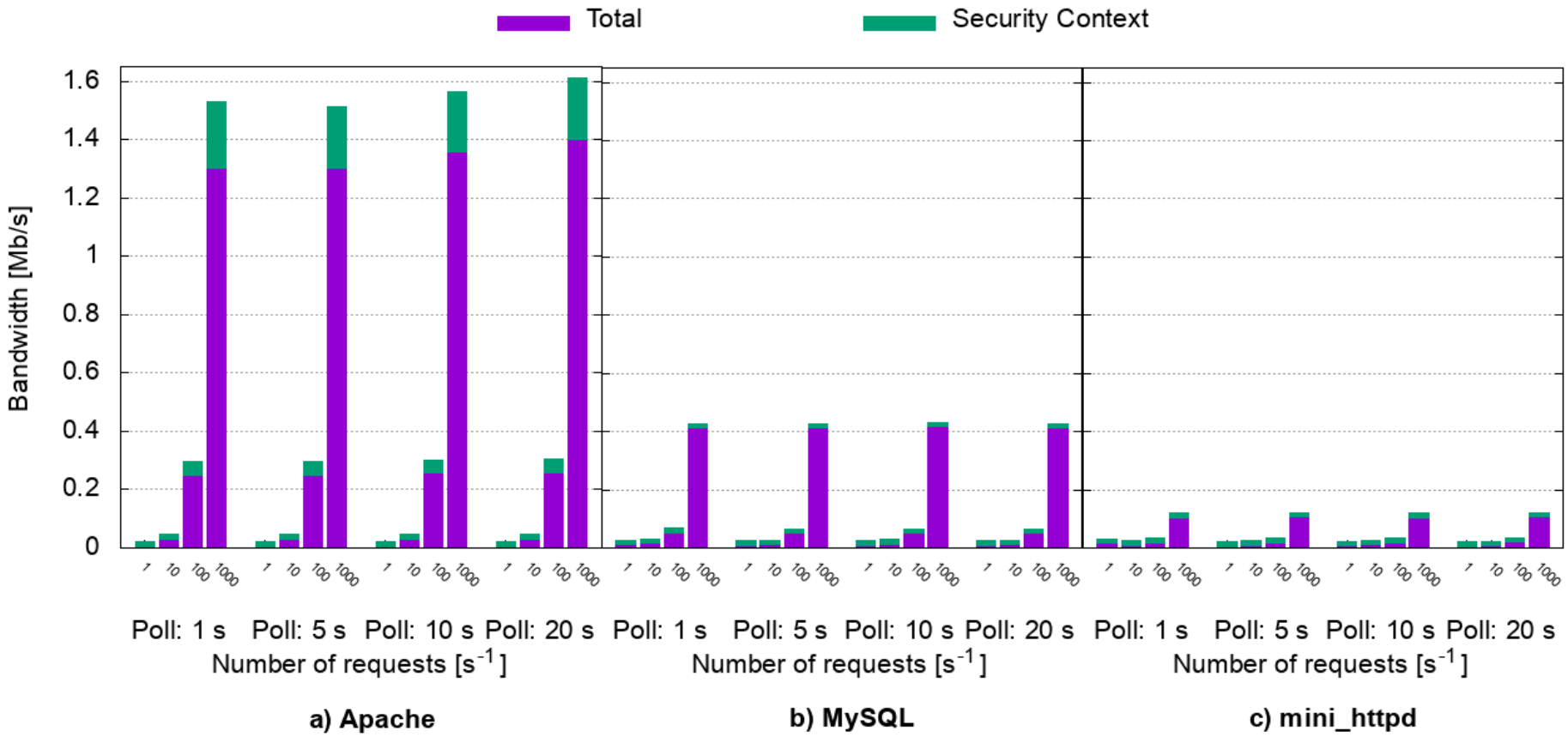
Preliminary results – Memory usage

Breakdown of memory usage by main process (Apache, MySQL, mini_httpd), Logstash and Beats



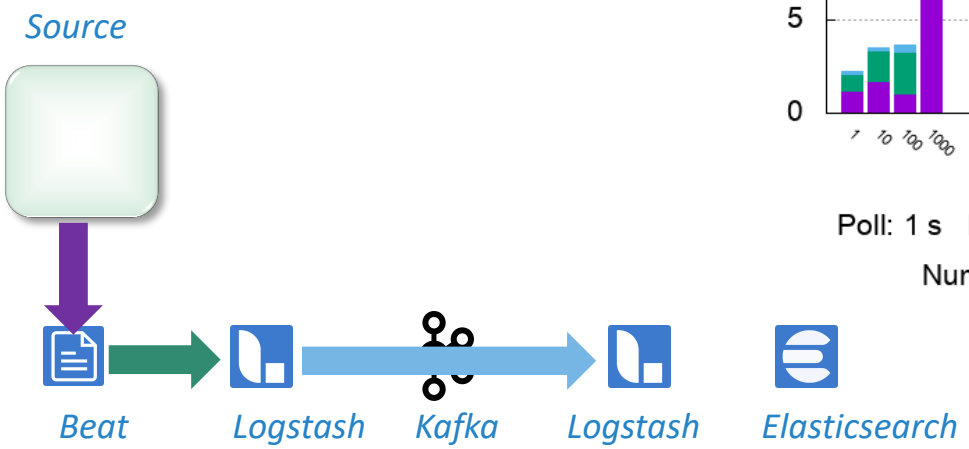
Preliminary results – Network

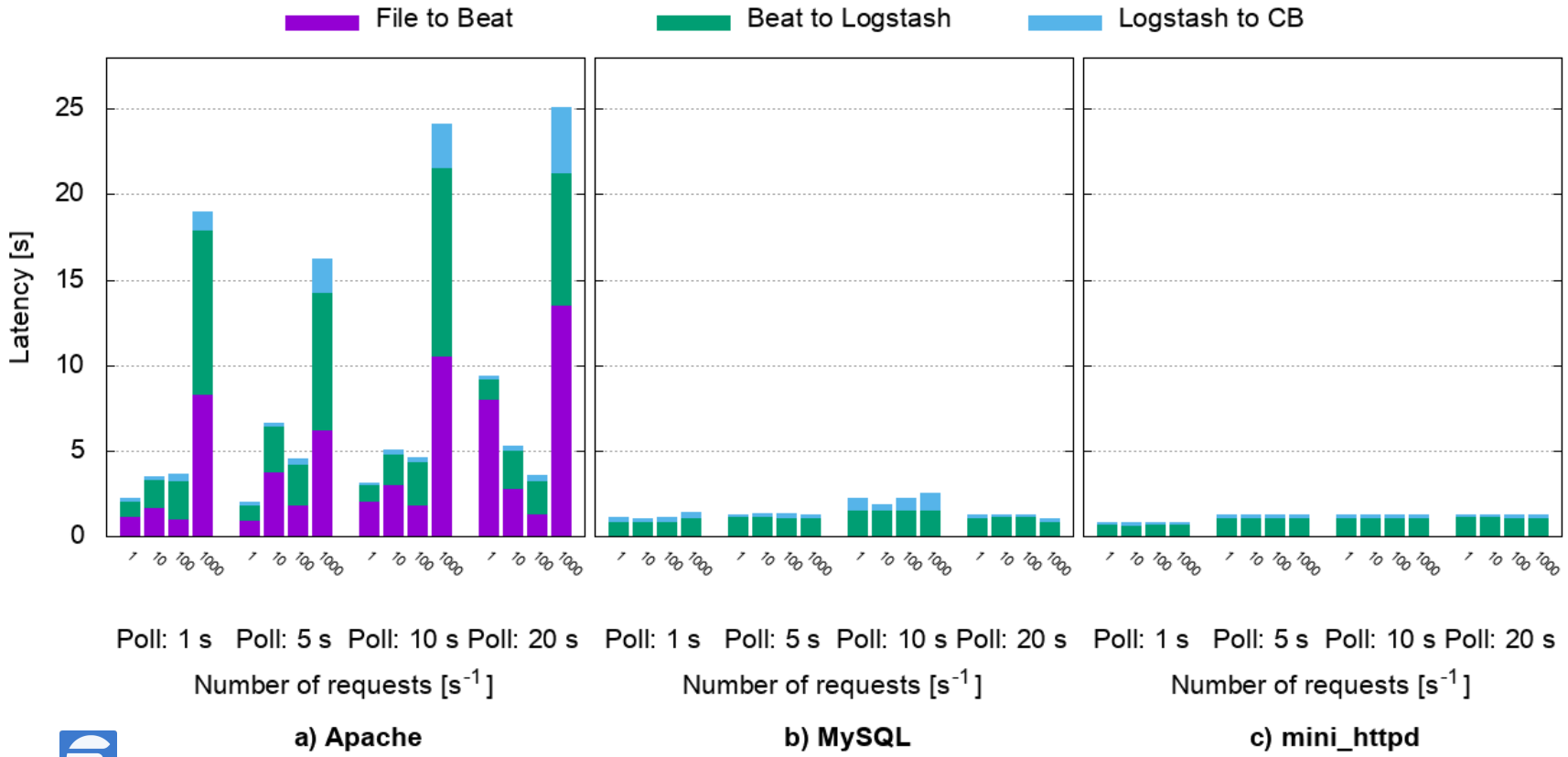
Bandwidth usage (Context collection/ total)



Preliminary results – Network

Breakdown of latency





Conclusions

Large workloads may lead to excessive overhead

- File collection is the most critical operation (frequent disk access)
- Logstash operation should be optimized (especially memory)
- eBPF filters have very low overhead

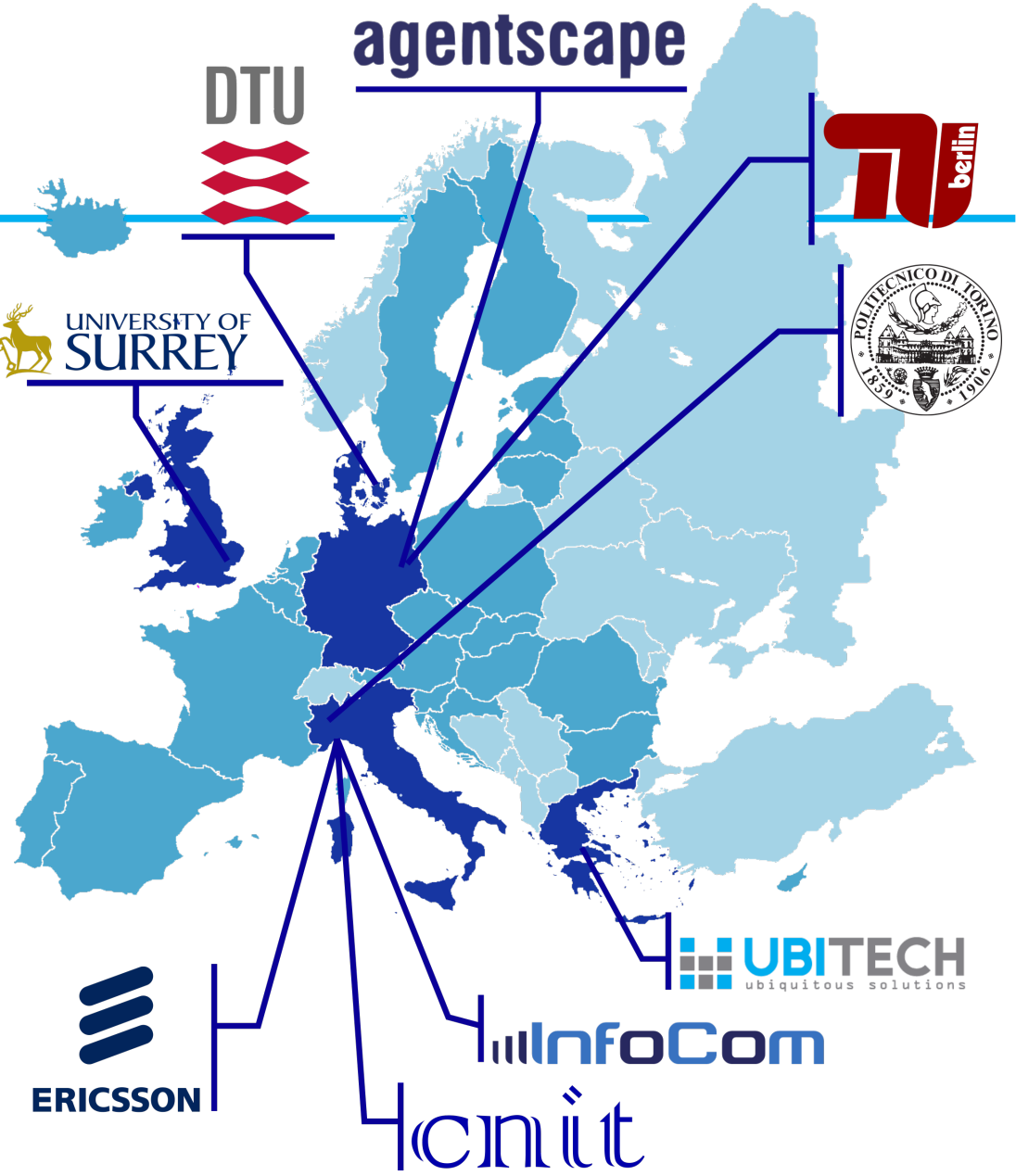
Low dependency on the polling interval

Next steps:

- development of the control part
- evaluate the delay of control commands
- integration in the ASTRID Use Cases

ASTRID factsheet

Project acronym	ASTRID
Project ID	786922
Starting Date	1st May 2018
Ending Date	30th April 2021
Call	H2020-DS-2016-2017
Topic	H2020-DS-07-2017 – Cybersecurity PPP: Addressing Advanced Cyber Security Threats and Threat Actors
Total Cost	EUR 2,932,297.50
EC Contribution	EUR 2,932,297.50
Funding scheme	RIA



Thank you!

M. Repetto:
matteo.repetto@cnit.it