ISSRE 2019

Tutorial Session

Building Applications for Trustworthy Data Analysis in the Cloud

Andrey Brito
André Martin
Lilia Sampaio
Fábio Silva

Security-aware data processing

Part 1

Why secure data processing?

1

Users want data to be processed in the cloud

In 2019, companies executed

79%

of their workload in the cloud

(RightScale 2019 - State of the Cloud Report)

Up to 2021,

94%

of this workload will be processed in the cloud

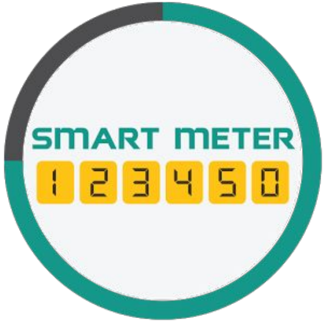
(Cisco Global Cloud Index: Forecast and Methodology, 2016-2021 White Paper)

Sensitive data requires increasing the level of security measures when processing and storing such data

Personal information

Health

Financial



Energy consumption

Company strategy related

1 Users want data to be processed in the cloud

2 Security is in the TOP 5 cloud challenges, being cited for over 81% of participants

During the first

6 MONTHS OF 2018

the equivalent of

291 RECORDS

was stolen or exposed

EVERY SECOND!

(Source: Article “2018: The year of the data breach tsunami” - MalwarebytesLABS, 2018)

1 Users want data to be processed in the cloud

2 Security is in the TOP 5 cloud challenges, being cited for over 81% of participants

Secure data processing is then very important!

How to securely process sensitive data?

Intel SGX

Software Guard eXtensions

- Trusted execution environments
- Hardware technology
- Guarantees of data integrity and confidentiality
- Use of isolated and protected memory areas called *enclaves*
- Supports remote attestation

SCONE

Secure CONtainer Environment

- Uses SGX to protect container processes
- Transparent to already existing Docker environments
- There are no changes to the application code being deployed
- Prepares the code to be SGX-compatible

1

Users want data to be processed in the cloud

2

Security is in the TOP 5 cloud challenges, being cited for over 81% of participants

Secure data processing is then very important!

3

Resources should be managed in order to attend users needs

Top Cloud Initiatives in 2019

64%
Optimize existing use of cloud

58%
Move more workloads to cloud

39%
Expand use of containers

(Source: RightScale 2019 - State of the Cloud Report)

1

Users want data to be processed in the cloud

2

Security is in the TOP 5 cloud challenges, being cited for over 81% of participants

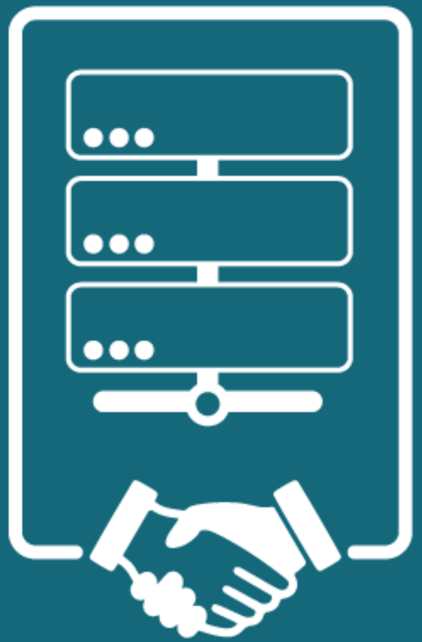
Secure data processing is then very important!

3

Resources should be managed in order to attend users needs

4

Quality of Service!



QoS and Reliability

Quality of Service as a reliability measure

- QoS management can be defined as *"the allocation of resources to an application in order to guarantee a service level along dimensions such as performance, availability and reliability"*

  (Ardagna et al. (2014) - Quality-of-service in cloud computing: modeling techniques and their applications)

- Cloud support
- Data processing
- Automatization
- Customization
- Secure executions

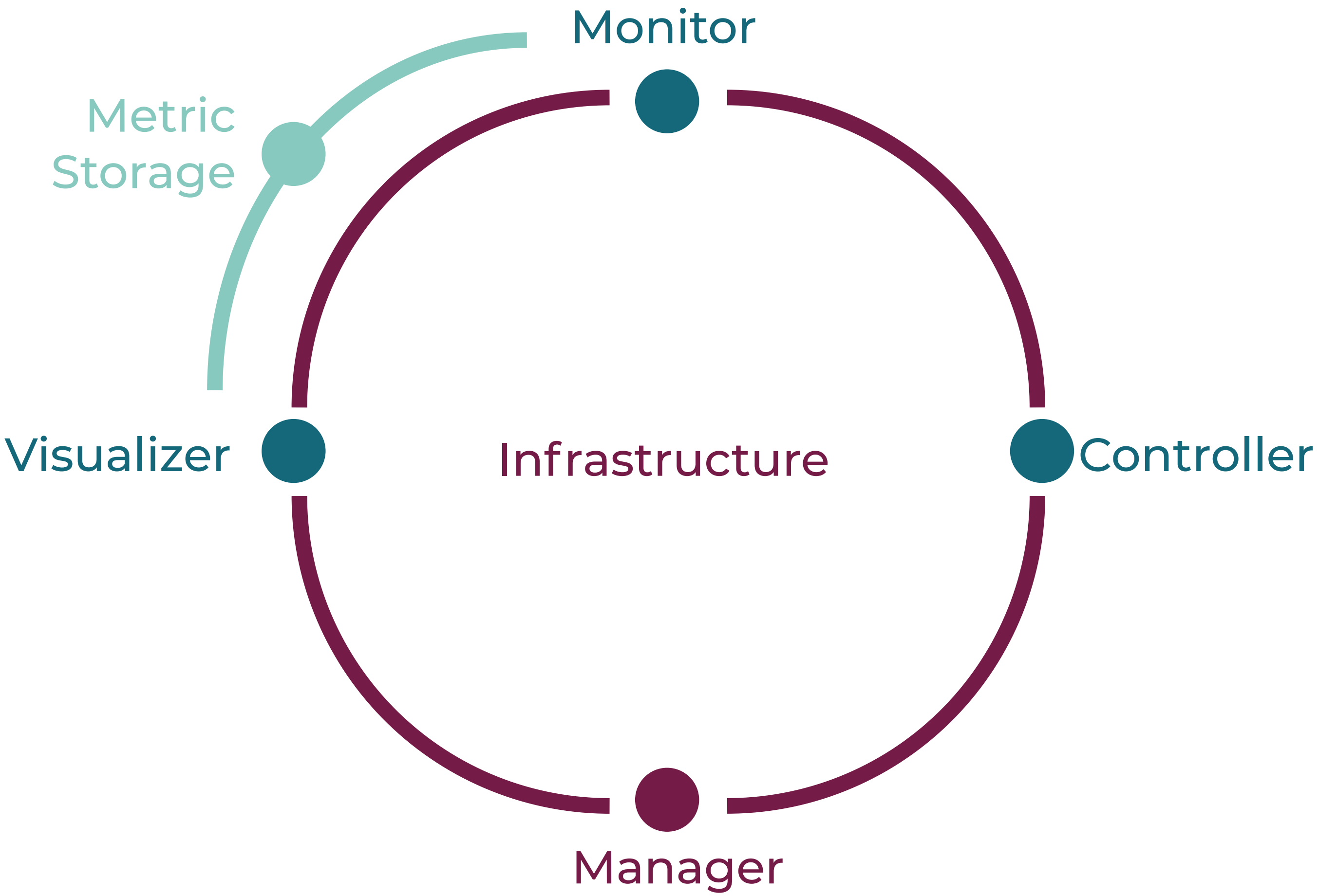


Figure 1. Asperathos architecture

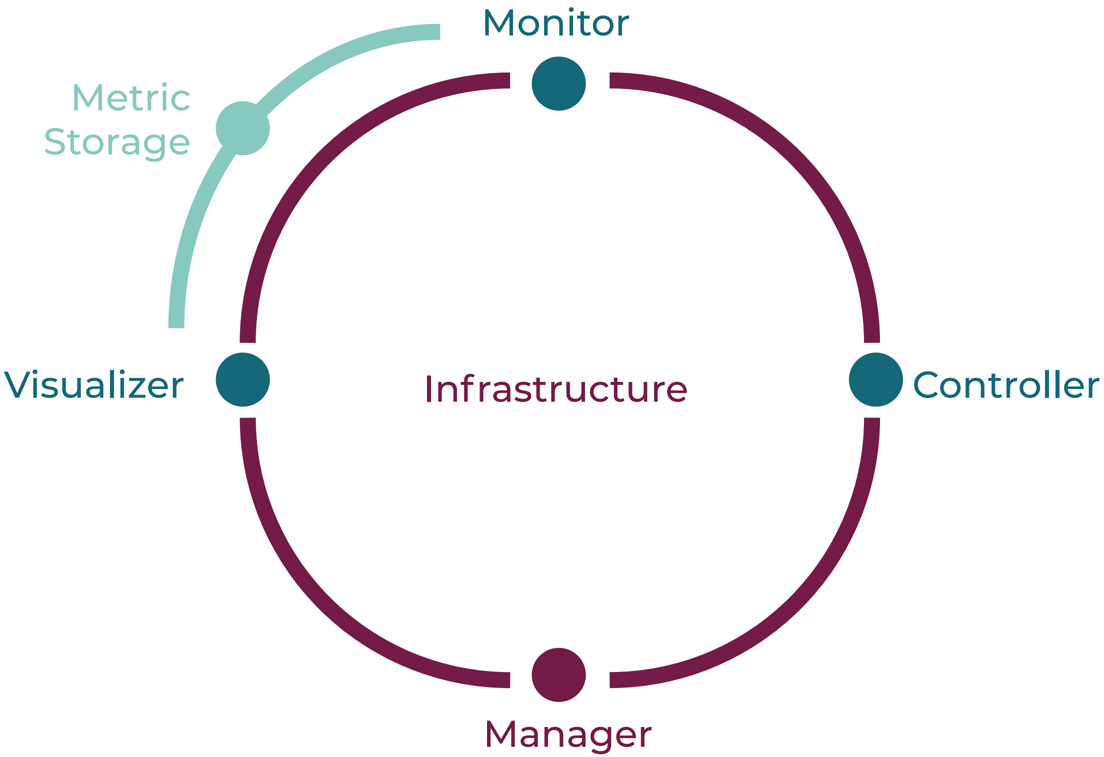


Figure 1. Asperathos architecture

Controlling the system in order to meet deadlines can be difficult

What can Asperathos do?

Confidential data processing

QoS-aware data processing

Confidential data processing

QoS-aware data processing

Using SCONE to build SGX applications

Intel SGX In Its Original Design

Intention: Only for very small functionality like generating secrets

Complicated usage: sgx_create_enclave

System call interface access through e-calls & o-calls

```
enclave_ref(enclave);
while (!enclave_terminated(enclave)) {
    enclave_enter(enclave, args->tcs_id, args->call_id, args, ret);
    switch (ret[0]) {
        case EXIT_TERMINATE:
            exit_code = ret[1];
            enclave_terminate(enclave);
            exit(exit_code);
        case EXIT_CPUID: {
            unsigned int* reg = (unsigned int*)ret[1];
            do_cpuid(reg);
            args->call_id = ECALL_SYSCALL_RESUME;
            break;
        }
        case EXIT_SYSCALL: {
            syscall_t* sc = (syscall_t*)ret[1];
            if (sc->syscallno == SYS_vfork){
                pid_t pid = vfork();
                sc->syscallno = pid;
            } else {
                scall(sc);
            }
            args->call_id = ECALL_SYSCALL_RESUME;
            break;
        }
        case EXIT_SLEEP: {
            __atomic_store_n(&E->ethread_state[args->id], ETHREAD_SLEEPI
```

SCONE'S Design Goals

Minimal developer effort: Compile w/ scone-gcc instead w/gcc

- Alternatively, use prebuilt scone docker images

Run entire application in enclave

Provide transparent attestation, encryption and secret injection (Palaemon)

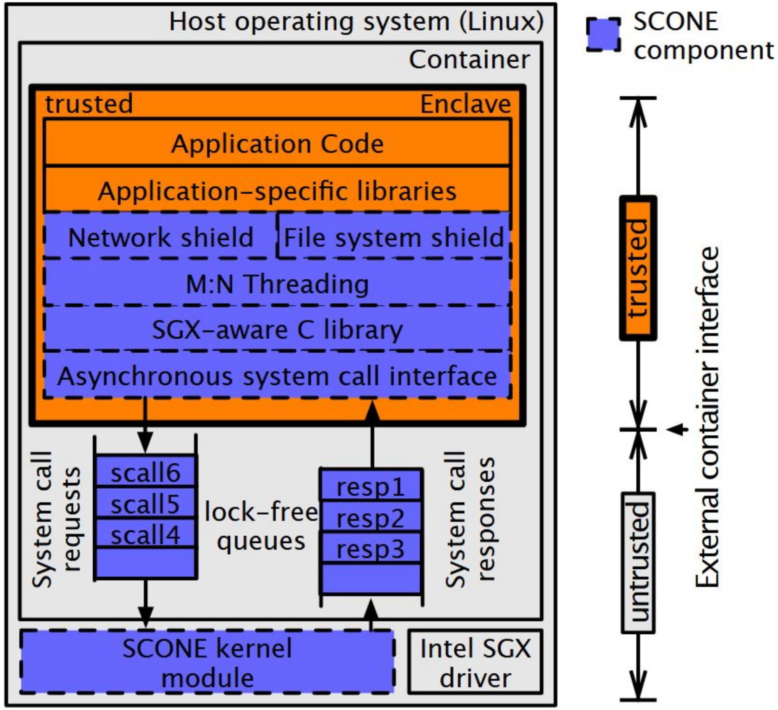
Tight integration in eco-systems, i.e., Docker & Swarm, Kubernetes

SCONE Under The Hood

Starter code

System call interface

User level scheduling



What is SCONE?

1) **Cross Compiler** to “sconify” applications, i.e., run them in Intel SGX enclaves

2) A **System Library** to provide system call support to talk to the external world, provides transparent file and network encryption, remote attestation and secret management

How To Use SCONE? 5 Easy Steps

1) **Enable** SGX in Bios (if not done already)
2) Install **Intel SGX Drivers**
3) Download/pull **cross compiler docker image**
4) **Compile** your favorite application
5) **Run** you application

How To Use SCONE? Step #1 - Enable Intel SGX in Bios

Under **Security** -> **Intel SGX**

Usually three options:

1. **Disabled**
2. **Enabled <- to choose**
3. **Software controlled**

How To Use SCONE? Step #2 - Install Intel SGX Drivers

Use the following one liner:

$ curl -fssl https://tinyurl.com/y2byyh4h | bash

Or follow official steps:

https://github.com/intel/linux-sgx-driver#install-the-intel-sgx-driver

How To Use SCONE? Step #3 - Download cross compiler docker image

Use the following two one liners:

$ docker pull sconecuratedimages/issre2019:crosscompilers

(This is the SCONE cross-compiler image for scone-based compilation based on the Alpine Linux docker imager)

$ docker pull alpine

(This is the bare bone Alpine Linux docker image for native compilation)

How To Use SCONE? Step #5 - Compile your favorite application

```
#include <iostream>
#include <cmath>

using namespace std;

int main() {
   char* secret = (char*)"Karate";
   int x = 0;
   while(x < 10) {
      double y = sqrt((double)x);
      cout << "The square root of " << x << " is " << y << endl;
      x++;
   }
   cout << secret << endl;
   do cout << '\n' << "Press a key to continue...";
   while (cin.get() != '\n');
   return 0;
}
```

How To Use SCONE? Step #5 - Compile your favorite application

$ wget -O sqrt.cc https://tinyurl.com/y6nyt4ly

$ docker run -v $(pwd):/myApp --device=/dev/isgx -it sconecuratedimages/issre2019:crosscompilers

$ cd /myApp

$ g++ -o sqrt-scone sqrt.cc

How To Use SCONE? Step #5 - Run your favorite application

$ SCONE_VERSION=1 ./sqrt-scone

That’s it!

Now We Do A Memory Dump (in a second terminal)

```
$ wget -O dump-memory.py https://tinyurl.com/y2x4nnyx
$ wget -O memory-dump.sh https://tinyurl.com/y3c6ucmw
$ chmod +x *.sh *.py
$ sudo ./memory-dump.sh

$ cat content-memory | grep Karate
```

Now The Same Without SCONE And Compare

```
$ docker run -v $(pwd):/myApp -it alpine

$ cd /myApp && apk add g++

$ g++ -o sqrt-native sqrt.cc

$ ./sqrt-native
```

Use case analysis: anonymization of sensitive echocardio

The Radiomics application

Anonymizing sensitive echocardio data

- Sensitive information is removed from video frames
- 2 types of input
  - Default video by video
  - Video archives

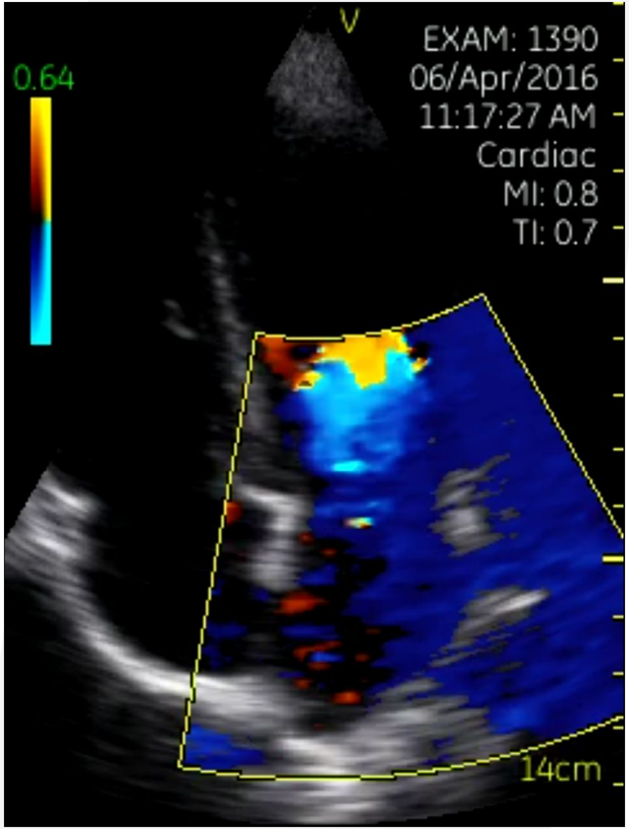


Figure 2. Radiomics video entry

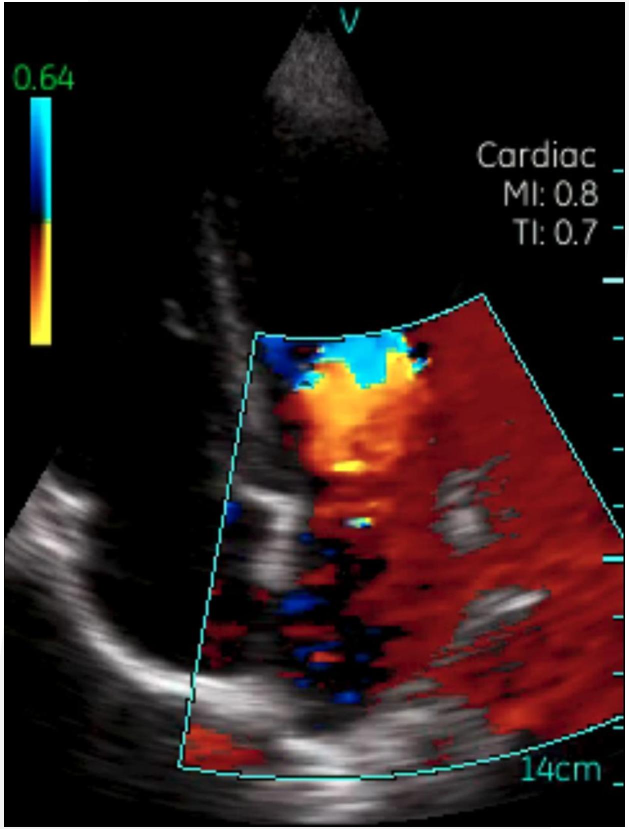


Figure 3. Radiomics anonymized result

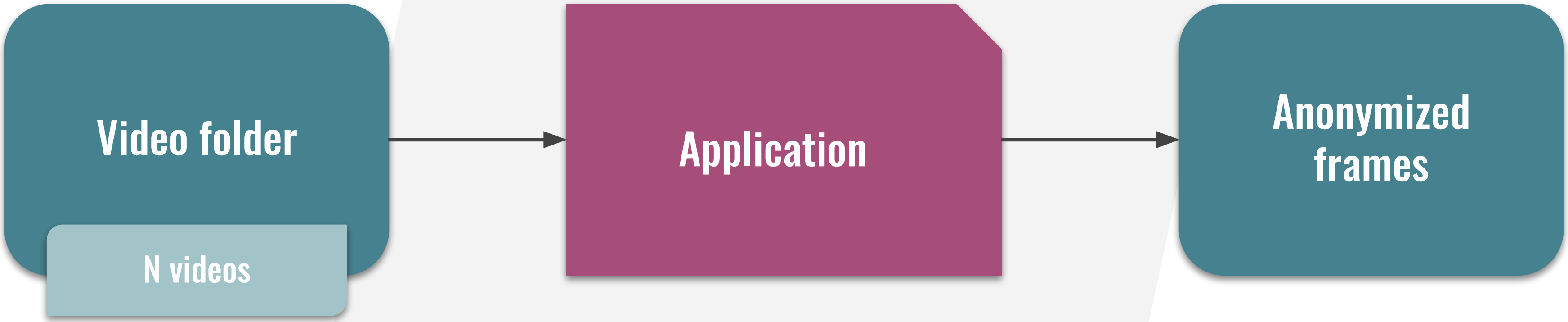


Figure 4. Radiomics simple architecture

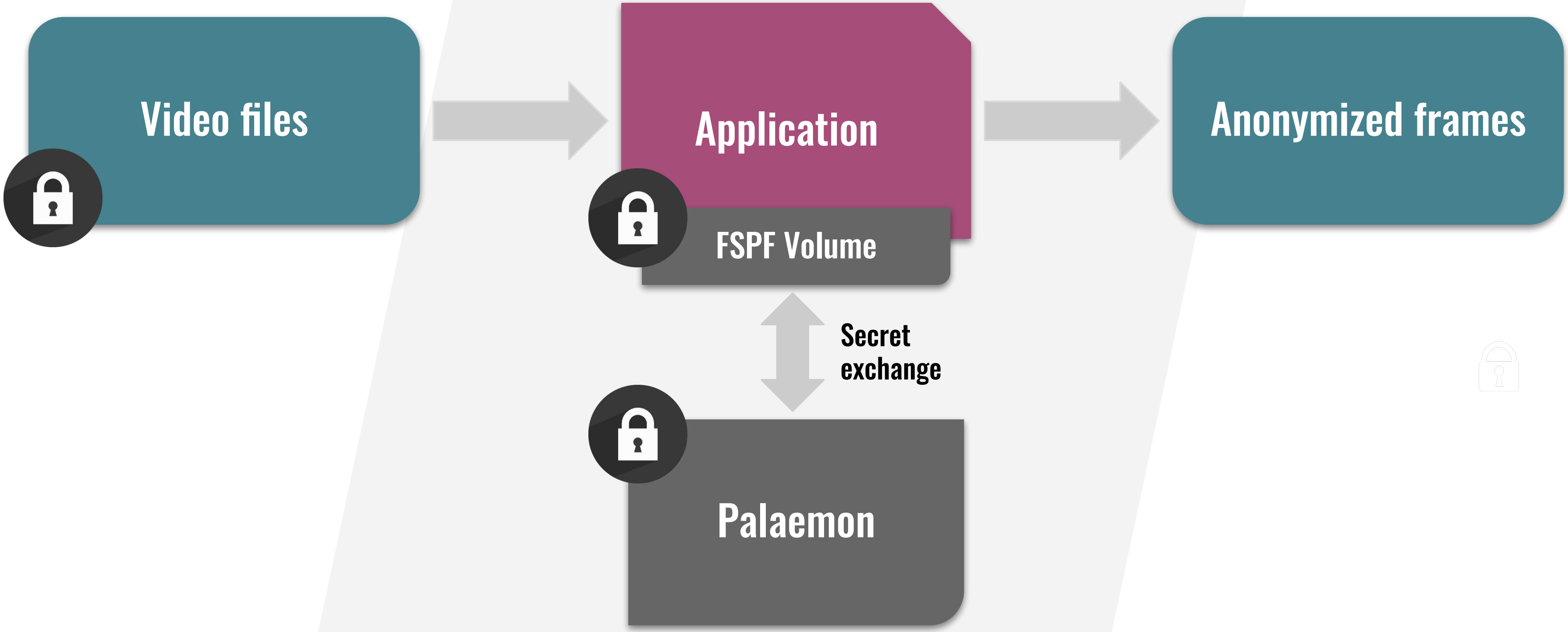


Figure 5. Radiomics architecture using SCONE and FSPF

Performance Overheads - 1

Understanding the performance of the use case: Execution time for Radiomics using SCONE and FSPF

- **Scenarios**
  - Unprotected
  - Protected execution
  - Protected execution and FSPF
- **Factors**
  - Sample size
- **EPC size: 90MB**
- **Machine used**
  - Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
  - 16GB RAM

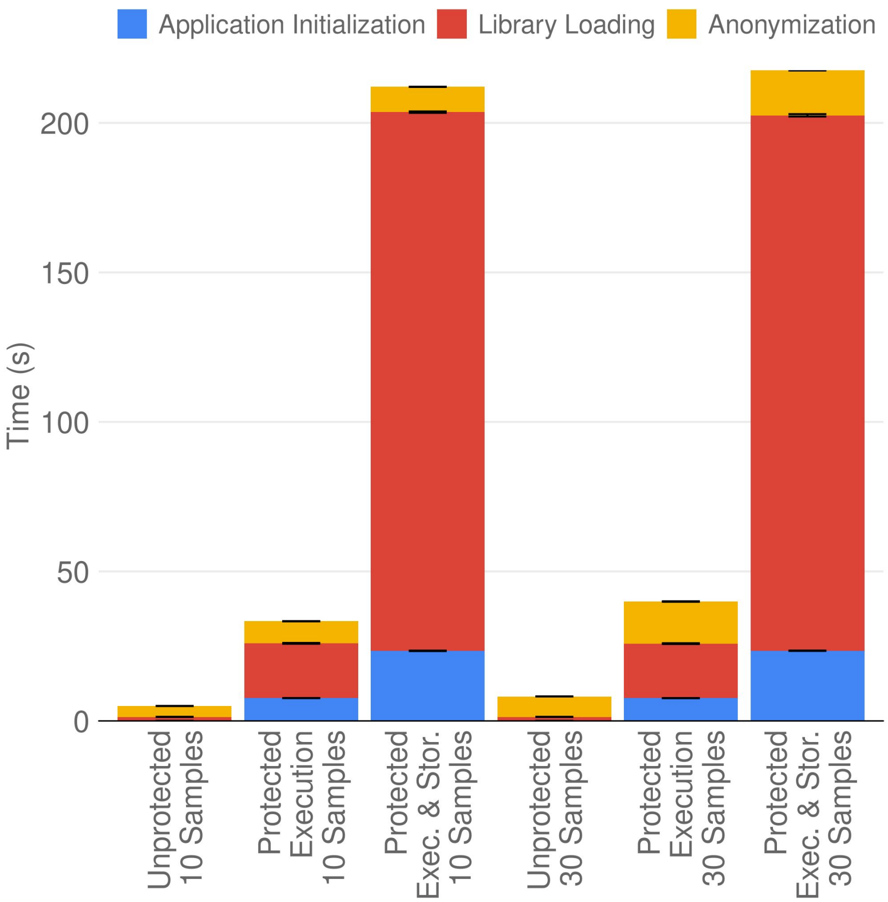


Figure 7. Experiment results considering execution time for SCONE executions

Performance Overheads - 2

Understanding the performance of the use case: Execution time for Radiomics using SCONE and FSPF

- **Scenarios**
  - Unprotected
  - Protected execution
  - Protected execution and FSPF
- **Factors**
  - Sample size
  - **EPC size**
  - **Number of vCPUs**
- **Machine used**
  - Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
  - 16GB RAM

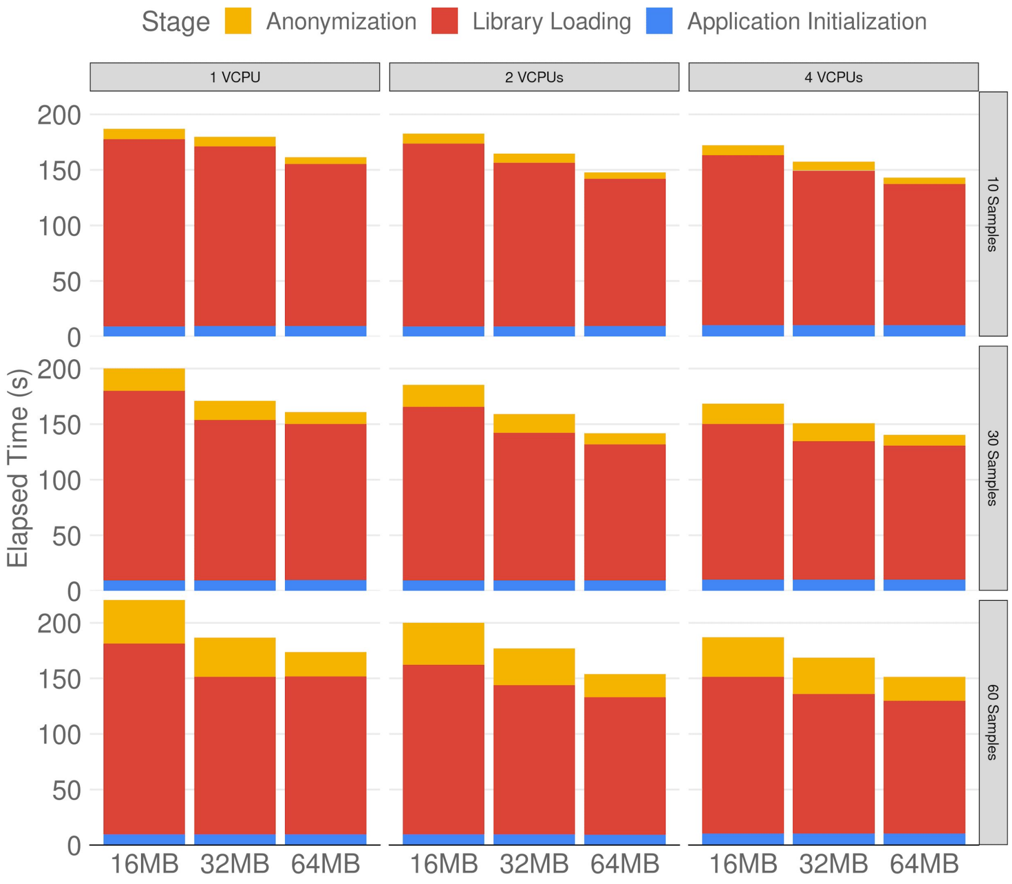


Figure 8. Experiment results considering execution time for SCONE executions varying EPC

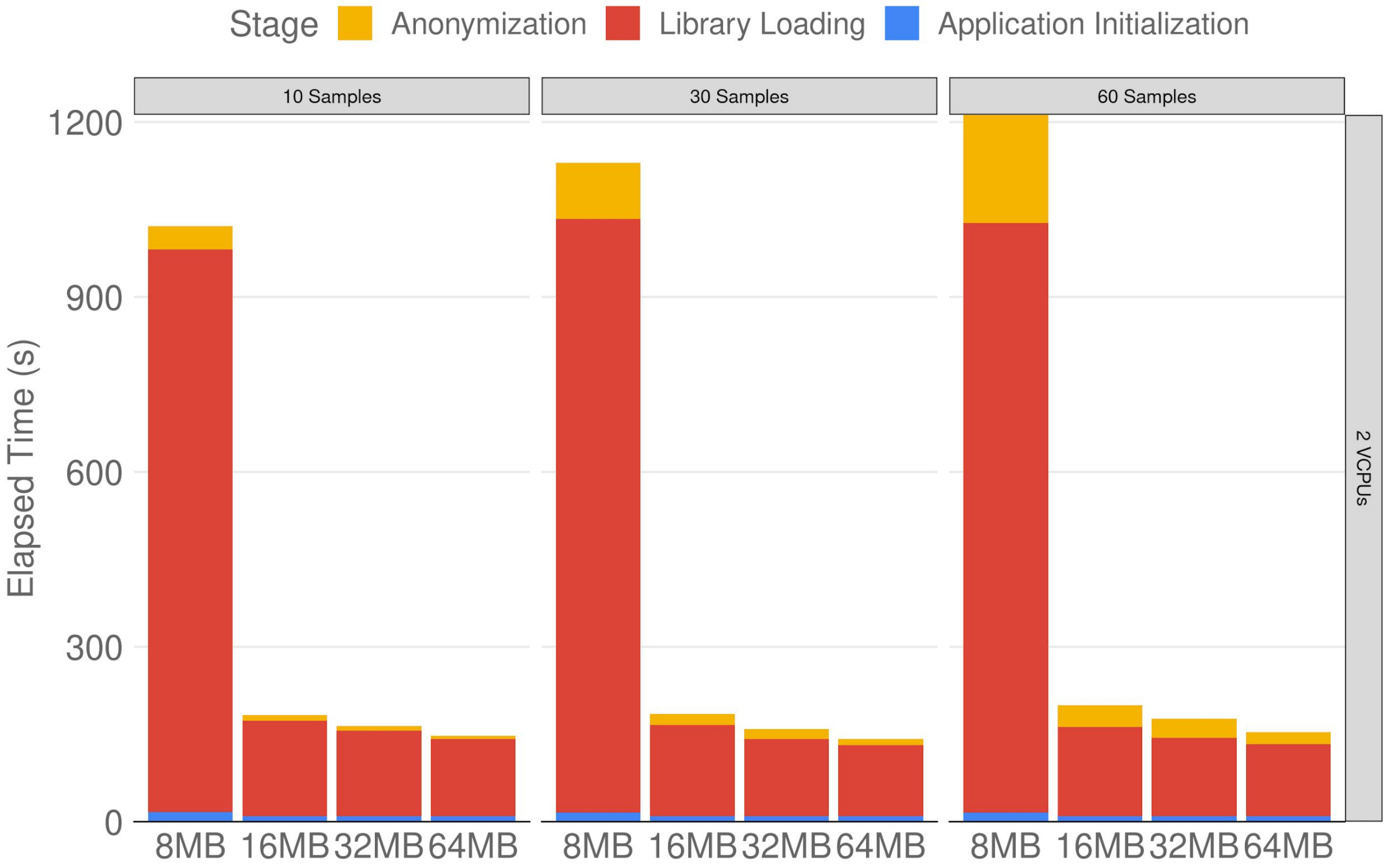


Figure 9. Experiment results show that for 8MB of EPC, elapsed time is much higher

Lessons learned

What can we learn from these experiments?

- Requiring many third party libraries is expensive because they need to be integrity checked
- For batch processing multiple items at once may be desirable
- EPC can make a huge difference in some cases, (our 8 MB EPC example)
- It can be better to use four 1-CPU, 16MB EPC machines than one four-CPU, 64MB EPC machine

Action! What do you need to perform an execution?

- The repositories used in this tutorial are available on GitHub
  - https://github.com/ufcg-lsd/issre-tutorial
- We are now going to perform 2 example executions:
  - SCONE + Radiomics
  - SCONE + FSPF + Radiomics
  - Reference to the guide:
    - https://github.com/ufcg-lsd/radiomics-scone

QoS and security-aware data processing

Part 2

Confidential data processing

QoS-aware data processing

Kubernetes 101

Nodes, clusters and volumes

The hardware

- **Nodes**
  - Representation of a single machine in a cluster
  - Physical machines or virtual machines
- **Clusters**
  - Composition of a set of nodes
  - It shouldn't matter to the program which individual machines are actually running the code
- **Volumes**
  - Data can't be saved to any arbitrary place in the file system
  - Can be mounted to the cluster and accessed by containers in a given pod

Containers, pods and deployments

The software

- **Containers**
  - Programs running in Kubernetes
  - Self-contained Linux execution environments
- **Pods**
  - Composition of a set of containers
  - Pods are used as the unit of replication in Kubernetes
- **Deployments**
  - Manages and monitor a set of pods
  - Declares how many replicas of a pod should be running at a time

Why Kubernetes?

- Allow for the **deployment of jobs**!
- Jobs are good for batch processing
  - Processes that run for a certain time to completion
- Applications are monitored and taken care of
- Large and active community
- Supported by all major cloud providers

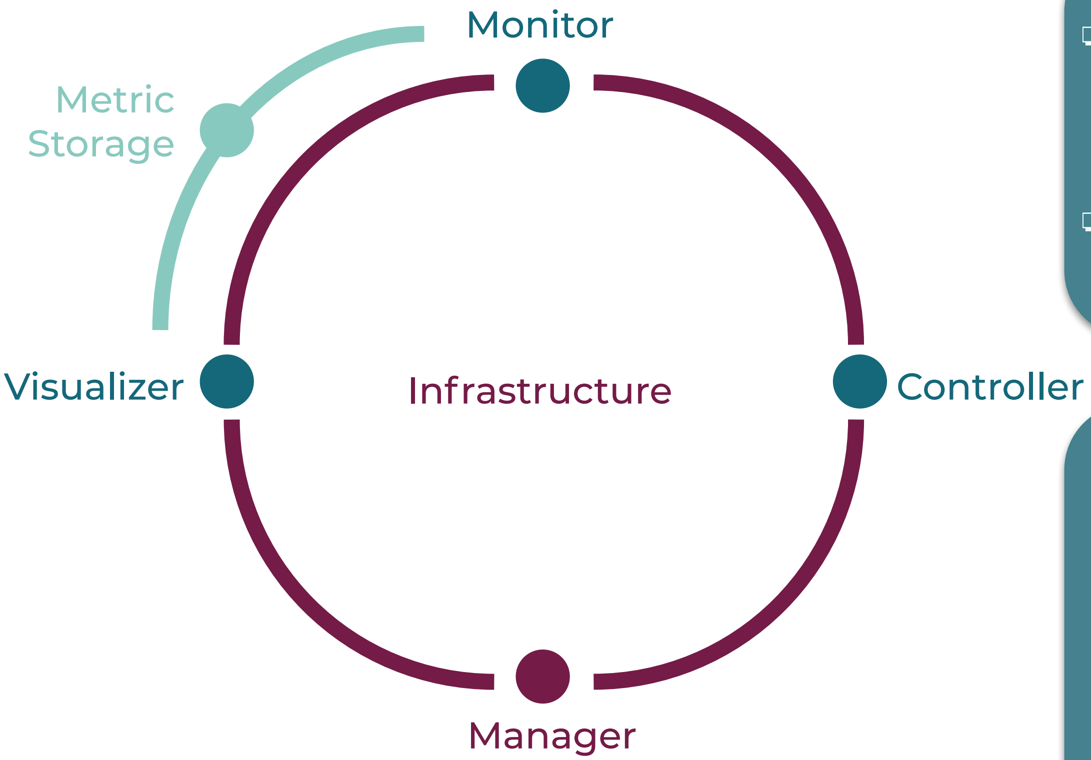
Asperathos 101

MANAGER

- Entry point to the users
- Receives a job submission and prepares its execution
- Handles cluster configuration
- Sends requests to the Monitor and Controller components

CONTROLLER

- Adjusts the amount of resources allocated to an application in order to guarantee QoS



MONITOR

- Calculates and publishes application metrics (M1) or resource metrics (M2)
- M1. Application progress; M2. CPU, memory, etc.

VISUALIZER

- Allows the visualization of a job's progress
- Consumes metrics from the Monitor and generates graphs
- Grafana
- Influxdb and Monasca

What is a job?

- "**plugin**": "kubejobs"
- "**cmd**": ["python", "app.py"]
- "**img**": "image/name"
- "**init_size**": 1
- "**redis_workload**": "https://gist.githubusercontent.com/raw/43eaeffe10"
- "**job_resources_lifetime**": 30
- "**control_plugin**": "kubejobs"
- "**control_parameters**":
  - "**schedule_strategy**":"default"
  - "**actuator**": "k8s_replicas"
  - "**check_interval**": 10
  - "**trigger_down**": 1
  - "**trigger_up**": 1
  - "**min_rep**": 1
  - "**max_rep**": 10
  - "**actuation_size**": 1
  - "**metric_source**": "redis"
- "**monitor_plugin**": "kubejobs"
- "**expected_time**": 130
- "**enable_visualizer**": true
- "**visualizer_plugin**": "k8s-grafana"
- "**env_vars**": {}

What you need to use Asperathos?

Have a Kubernetes cluster ready to receive applications

+

Have an application image capable of consuming items from a redis workload

Consuming items from redis

```
import redis
import requests
import os

# Asperathos will return the redis host as an environment variable named REDIS_HOST.
# The default port of redis is 6379.
r = redis.StrictRedis(host=os.environ['REDIS_HOST'],
                      port=6379,
                      db=0)

# r.llen("job") returns the length of the queue "job" on redis.
while r.llen("job") > 0:

    # `rpoplpush` moves one item from our work queue
    # to an auxiliary queue for items being processed,
    # returning its value
    item_url = r.rpoplpush('job', 'job:processing')

    # downloading the content of the items
    content = requests.get(item_url).text

    # do the actual processing
   do_something(content)
```

Action! How to deploy an Asperathos instance?

- The repository for Asperathos and its components is available on GitHub
  - https://github.com/ufcg-lsd/asperathos
- We are now going to deploy an instance of Asperathos
  - Reference to the QuickStart guide in the GitHub page linked above (Section 4)

Confidential data processing

➕

QoS-aware data processing

Use case analysis: combining security and QoS

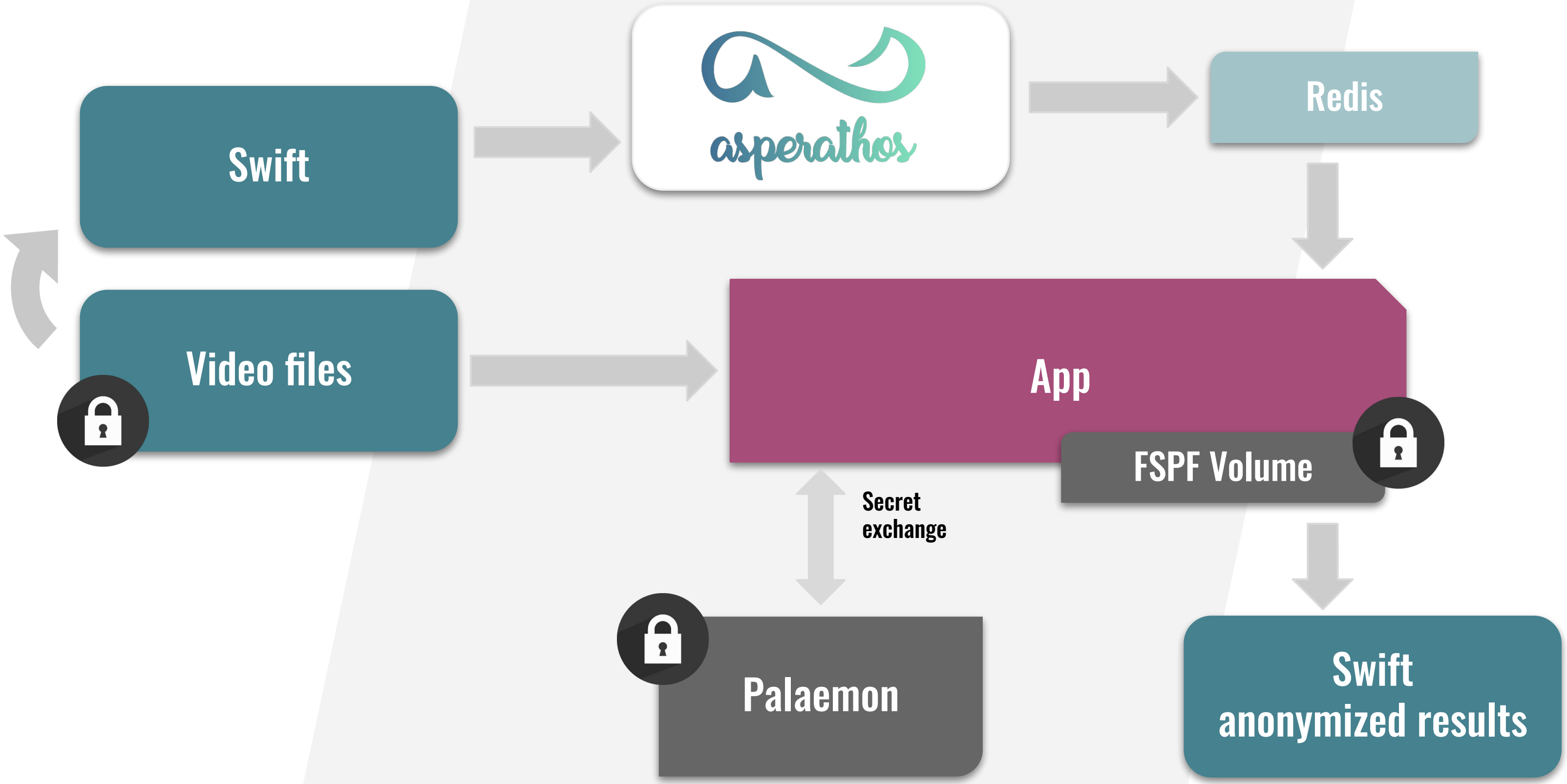
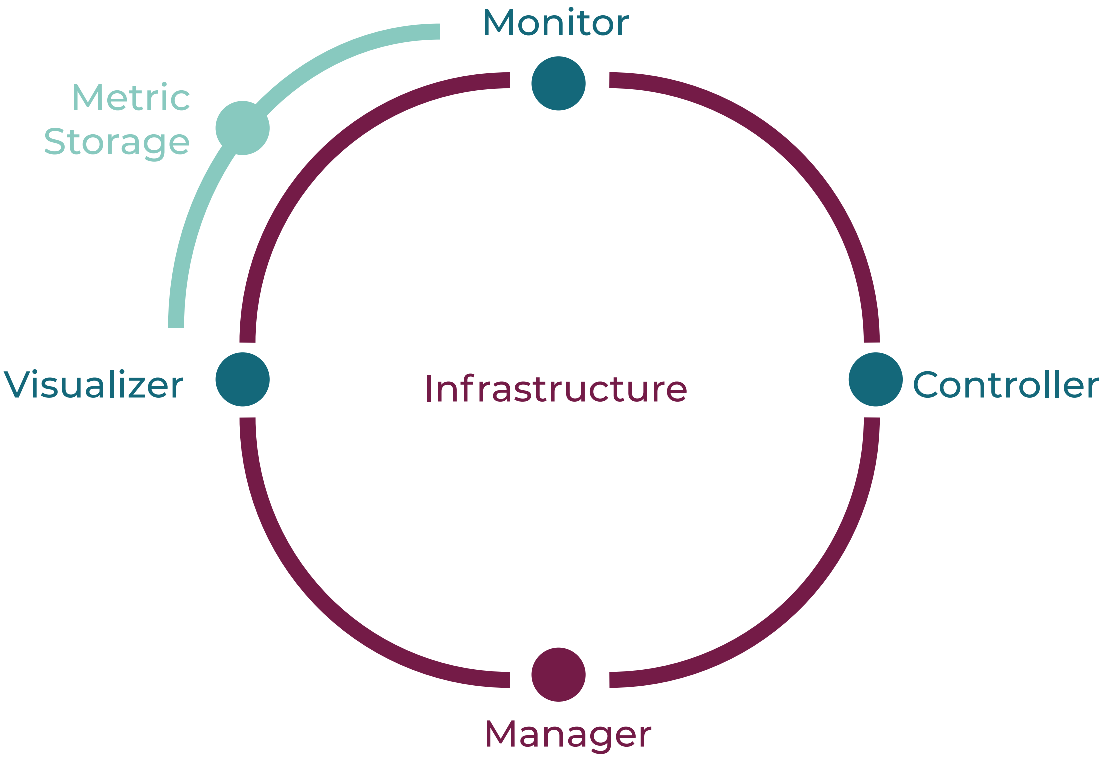


Figure 9. Architecture of the combination of Asperathos and Radiomics

Action! What do you need to perform an execution?

- The repositories used in this tutorial are available on GitHub
  - https://github.com/ufcg-lsd/issre-tutorial
- We are now going to perform an execution of Radiomics using Asperathos
  - Reference to the guide:
    - https://github.com/ufcg-lsd/radiomics-asperathos

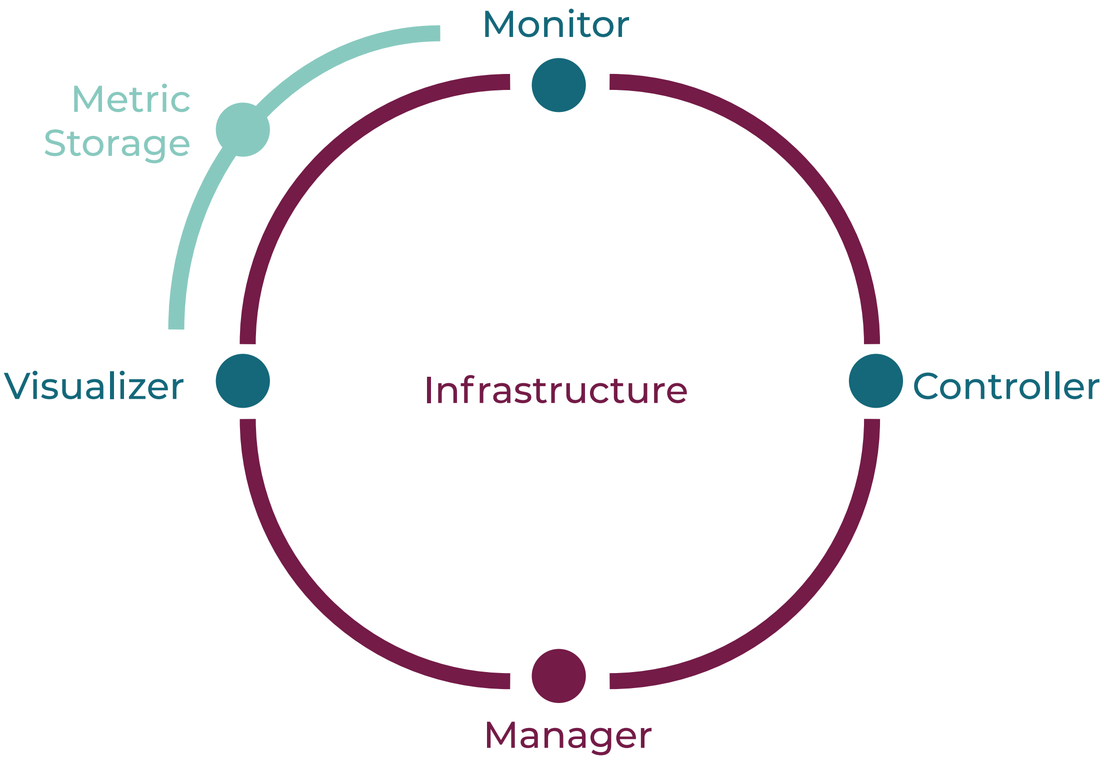
Customizing Asperathos components



Asperathos architecture allows the addition of customized plugins

What can you customize?

Controlling the resources used by a job



Controller

- ❑ Responsible for adjusting the quantity of resources allocated to the execution of one application in a way that deadlines are met and QoS guaranteed

Customizable strategies

- ❑ Default: fixed actuation size
- ❑ PID: actuation size depends on the error magnitude and trend
- ❑ Your own strategy!

Creating a plugin for the Controller component

Adaptive KubeJobs

- Based on cluster utilization of CPU and RAM
- User defines the threshold for these metrics
- If the utilization is lower than the threshold, the controller uses the control strategy defined
- If higher, the controller decreases job replicas until the utilization returns to an acceptable value

How to install

Send a POST request to <your_asperathos_url:port>/plugins with this json body:


```
{
	"plugin_source":
"https://git.lsd.ufcg.edu.br/asperathos-custom/adaptive-kubejobs",
	"install_source": "git",
	"plugin_module": "adaptive_kubejobs",
	"component": "controller",
	"plugin_name": "adaptive_kubejobs"
}
```

How to use

Requirements: metric-server by Kubernetes

Json must contain the parameters bellow:

```
"control_plugin": "adaptive_kubejobs",
"control_parameters": {
    "schedule_strategy":"default",
    "actuator": "k8s_replicas",
    "check_interval": 10,
    "trigger_down": 1,
    "trigger_up": 1,
    "min_rep": 1,
    "max_rep": 10,
    "actuation_size": 1,
    "metric_source": "redis",
    "max_ram": 0.7,
    "max_cpu": 0.5
}
```

Conclusions

Conclusions

- SGX is a promising technology that enables sensitive data to be processed with confidentiality and integrity guarantees in untrusted clouds
- SCONE provides transparency to some non-trivial aspects of SGX programming, such as remote attestation and file/network encryption
- If application’s working memory fits in the EPC available in your VM/machine, processing overhead is small
- But it is also important to minimize the number of files that need to be authenticated for the application
- If lots of data needs to be confidentially processed, Asperathos is a good alternative for orchestrating batch executions with QoS

Thank you for attending!

ISSRE 2019
Tutorial Session

Building Applications for Trustworthy Data Analysis in the Cloud

Andrey Brito
andrey@computacao.ufcg.edu.br
André Martin
andre.martin@tu-dresden.de
Lilia Sampaio
liliars@lsd.ufcg.edu.br
Fábio Silva
fabiosilva@lsd.ufcg.edu.br

ATMOSPHERE
Adaptive, Trustworthy, Manageable, Orchestrated, Secure Privacy-assuring Hybrid, Ecosystem for REsilient Cloud Computing

More information:
www.atmosphere-eubrazil.eu