HPC-BigData
Inria Project LAB

Enabling Reproducible Analysis of Complex Application Workflows on the Edge-to-Cloud Continuum

PhD candidate

Daniel Rosendo

Advisors

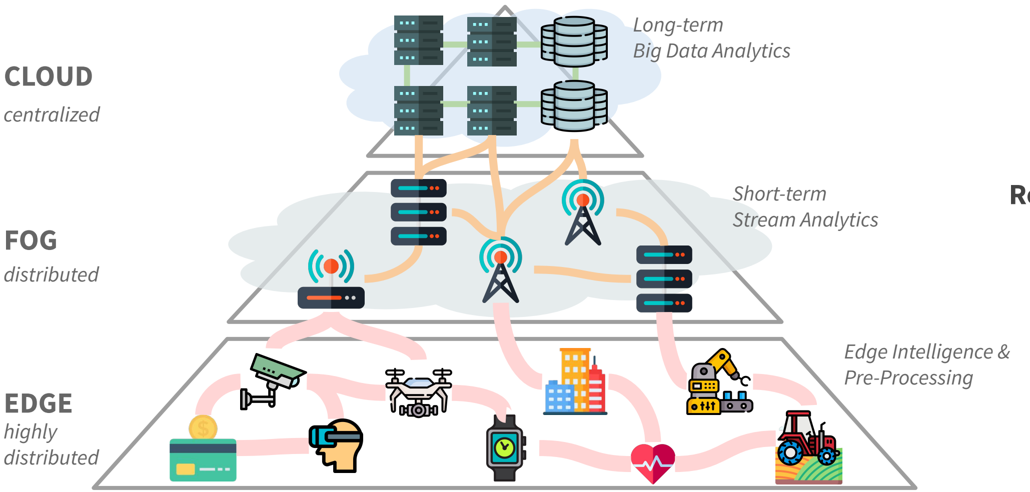
Gabriel Antoniu

Alexandru Costan

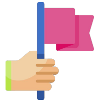
Patrick Valduriez

The Computing Continuum

Complex Application Workflows



Continuous dataflow** from IoT **Edge** devices to the **HPC/Cloud



Research Challenges & Opportunities

Understanding Performance of Application Workflows on the Edge-to-Cloud Continuum

Optimizing Performance of Edge-to-Cloud Application Workflows

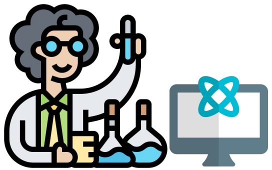
Supporting Reproducible Analysis of Complex Edge-to-Cloud Workflows

Research Challenge 1: **Understanding Performance**

Research Challenges & Opportunities ⟫ **Evaluation & Validation** ⟫ **Technical Challenges** ⟫ **Simple testbed setups**

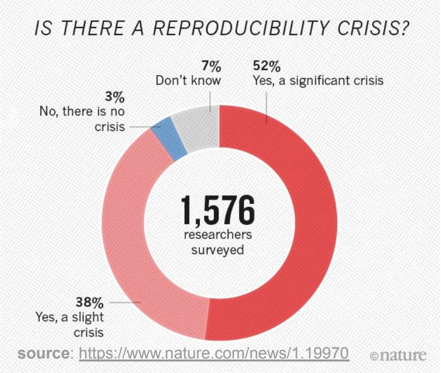
Which *system* ***parameters*** *and infrastructure* ***configurations impact performance*** *and* ***how?***

How to **systematically** perform large-scale **experiments** to enable the **reproducibility** of the **results?**



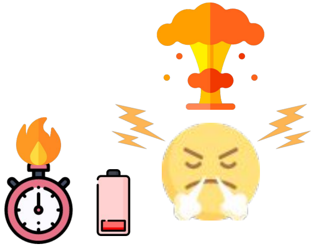
Representative setup to understand performance

Repeatable, Replicable & Reproducible experiments

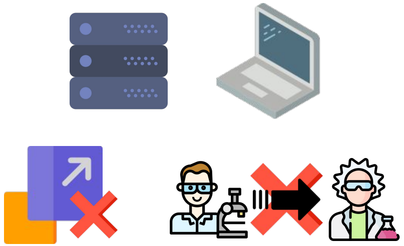


*"**+70%** failed to reproduce **another** scientist's experiments"*

*"**+50%** failed to reproduce their **own** experiments"*



- Install & configure services
- Configure network
- Map application parts with underlying infrastructure
- Define the execution workflow
- Establish procedures for reproducibility



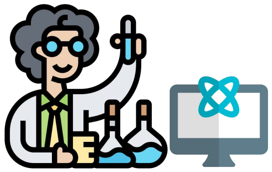
- Many abstractions
- Scalability issues
- Hard to reproduce

What Would Be an Ideal Solution?

Research Challenges & Opportunities

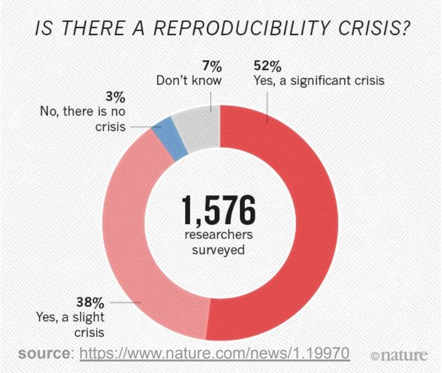
Which** system **parameters** and infrastructure **configurations impact performance** and **how?

How to **systematically** perform large-scale **experiments** to enable the **reproducibility** of the **results?**



Evaluation & Validation

- Representative setup to understand performance
- Repeatable, Replicable & Reproducible experiments

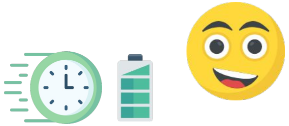


*"**+70%** failed to reproduce **another** scientist's experiments"*

*"**+50%** failed to reproduce their **own** experiments"*

Technical Challenges

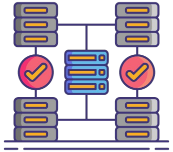
Focus on high-level aspects

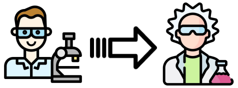


- Install & configure services
- Configure network
- Define the execution workflow
- Establish procedures for reproducibility

Abstract complexities

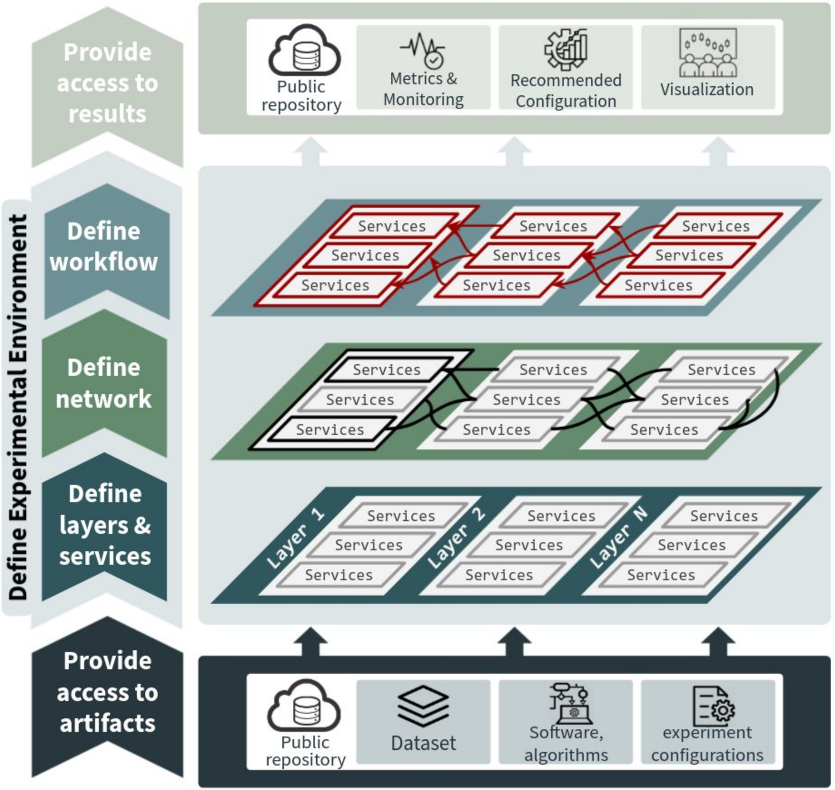
Large-scale testbed setups





- Representative setup
- Easy to scale
- Supports reproducible experiments

What is **E2C*lab***?

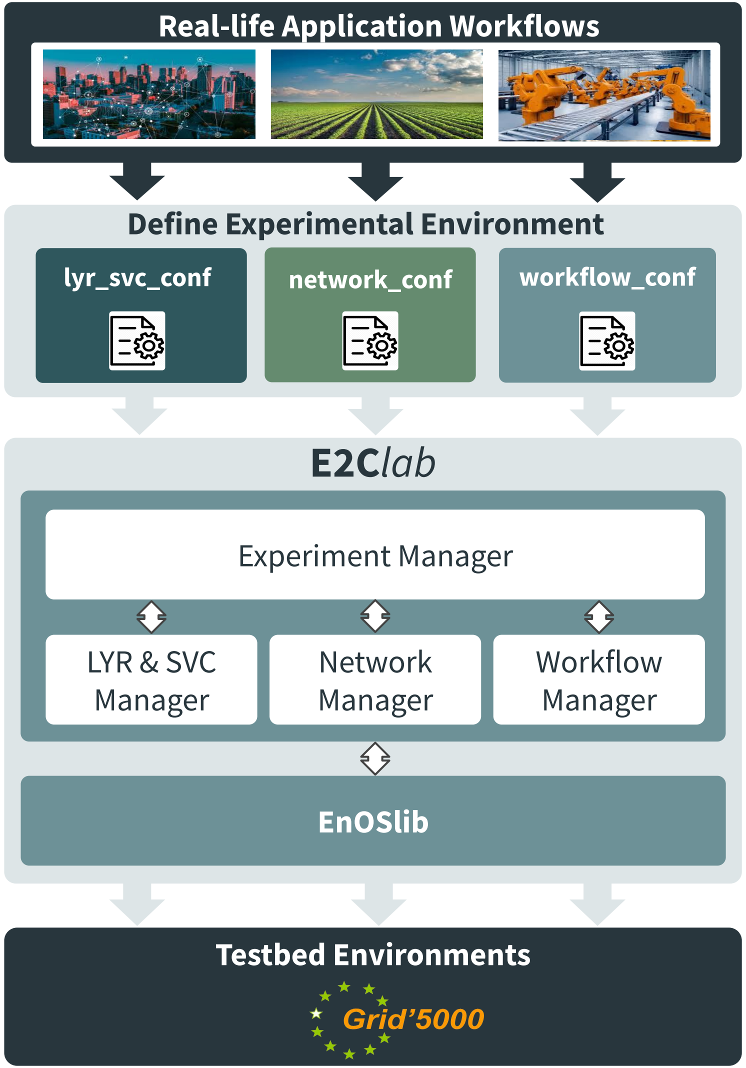


IEEE Cluster 2020
https://hal.archives-ouvertes.fr/hal-02916032

Methodology

- **Reproducible Experiments**
  Repeatability, Replicability & Reproducibility
- **Mapping**
  Application parts & physical testbed
- **Variation & Scaling**
  Experiment variation and transparent scaling
- **Network Emulation**
  Edge-to-Cloud communication constraints
- **Experiment Management**
  Deployment, Execution & Monitoring

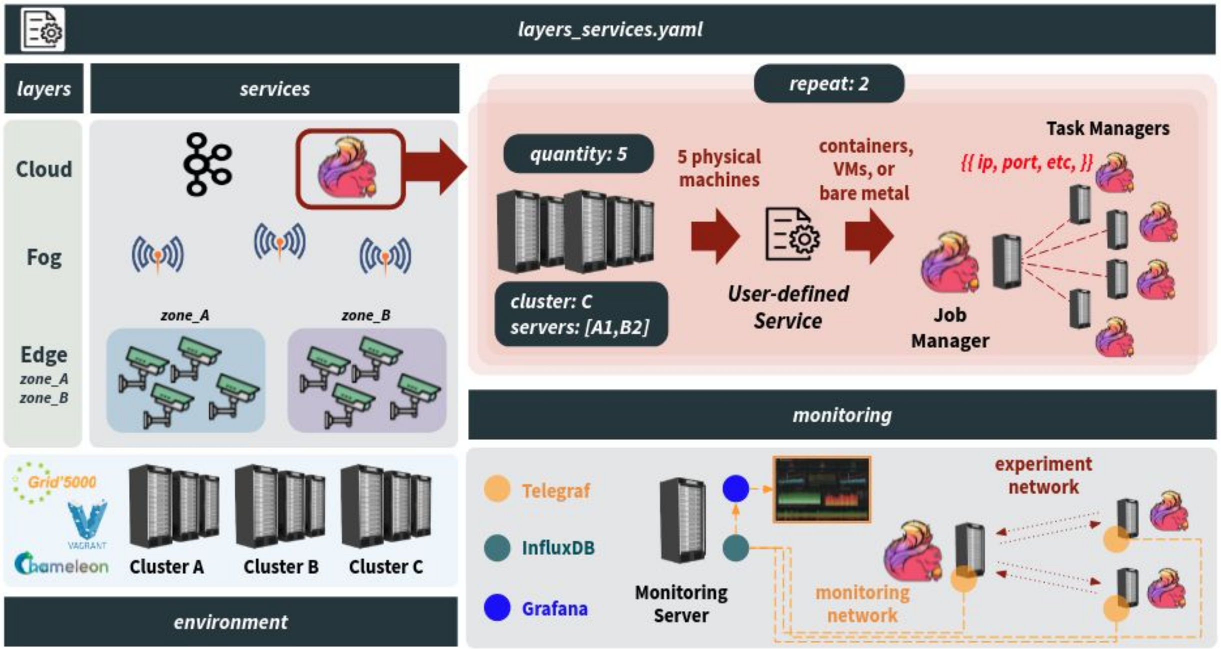
What is **E2C*lab***?



Defining the experimental environment: *Layers and Services*

```yaml
environment:
  name: g5k
  job_name: example
  env_name: "debian10-x64-big"
  walltime: "01:30:00"
monitoring:
  type: tig
  network: private
layers:
- name: cloud
  services:
  - name: Flink
    cluster: chiclet
    quantity: 5
    repeat: 2
    roles: [monitoring]
  - name: Kafka
    servers: ["chifflot-8.lille.grid5000.fr"]
    roles: [monitoring]
- name: fog
  services:
  - name: Mosquitto
    quantity: 3
    roles: [monitoring]
- name: edge.zone_A
  services:
  - name: Cameras
    quantity: 4
- name: edge.zone_B
  services:
  - name: Cameras
    quantity: 4
```

Selecting, reserving, monitoring, and configuring resources



Defining the experimental environment: *User-defined Service*

"Our methodology can be generalized to other applications!"

Reusable services

GitLab

- ClientKerA.py
- Default.py
- Flink.py
- Horovod.py
- Kafka.py
- KerA.py
- KerAF.py
- KerFlink.py
- Mosquitto.py
- PlantNetEngine.py

```python
from e2clab.services import Service

class Flink(Service):

    def deploy(self):
        # service logic
        self.deploy_docker(registry_opts=registry_opts)

        JM_PORT = "8081"
        job_manager_service = "job_manager"
        task_manager_service = "task_manager"

        roles_job_manager = Roles({job_manager_service: [self.hosts[0]]})
        roles_task_manager = Roles({task_manager_service: self.hosts[1:len(self.hosts)]})

        JOB_MANAGER_ADDRESS = roles_job_manager[job_manager_service][0].address
        self.env.update({"JOB_MANAGER_RPC_ADDRESS": JOB_MANAGER_ADDRESS})

        with actions(roles=roles_job_manager) as p:
            p.docker_container(name=job_manager_service,
                               image="flink:1.9-scala_2.11",
                               restart="yes",
                               restart_policy="always",
                               network_mode="host",
                               env=self.env,
                               command="jobmanager")

        with actions(roles=roles_task_manager) as p:
            p.docker_container(name=task_manager_service,
                               image="flink:1.9-scala_2.11",
                               restart="yes",
                               restart_policy="always",
                               network_mode="host",
                               env=self.env,
                               command="taskmanager")

        # register service
        self.register_service(_roles=roles_job_manager, service_port=JM_PORT, sub_service=job_manager_service)
        self.register_service(_roles=roles_task_manager, sub_service=task_manager_service)

        return self.service_extra_info, self.service_roles
```

Big Data frameworks | **AI frameworks** | **User's applications**

mxnet Pl@ntNet

Apache Spark HOROVOD

kafka streams TensorFlow

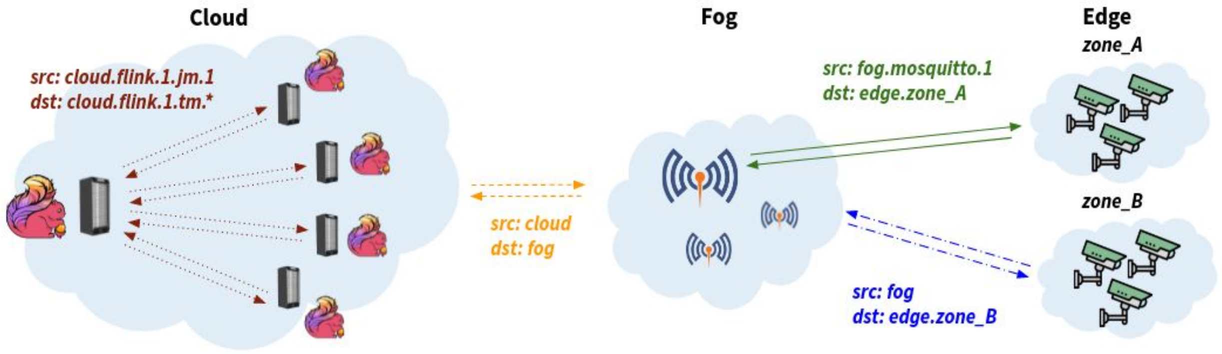
among others!!!

Defining the experimental environment: ***Network***

```yaml
networks:
- src: cloud
  dst: fog
  delay: "2ms"
  rate: "10gbit"
  loss: 2
- src: fog.mosquitto.1
  dst: edge.zone_A
  delay: "2ms"
  rate: "1gbit"
  loss: 5
- src: fog
  dst: edge.zone_B
  delay: "20ms"
  rate: "150mbit"
  loss: 7
```

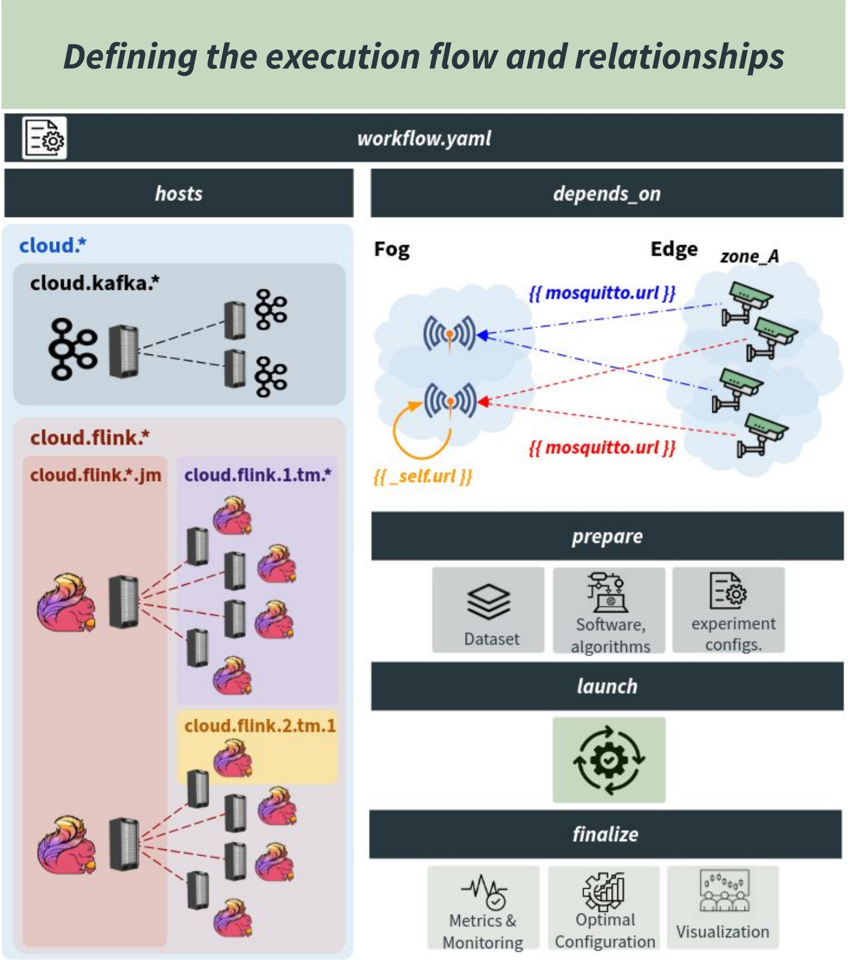
Defining network constraints

network.yaml

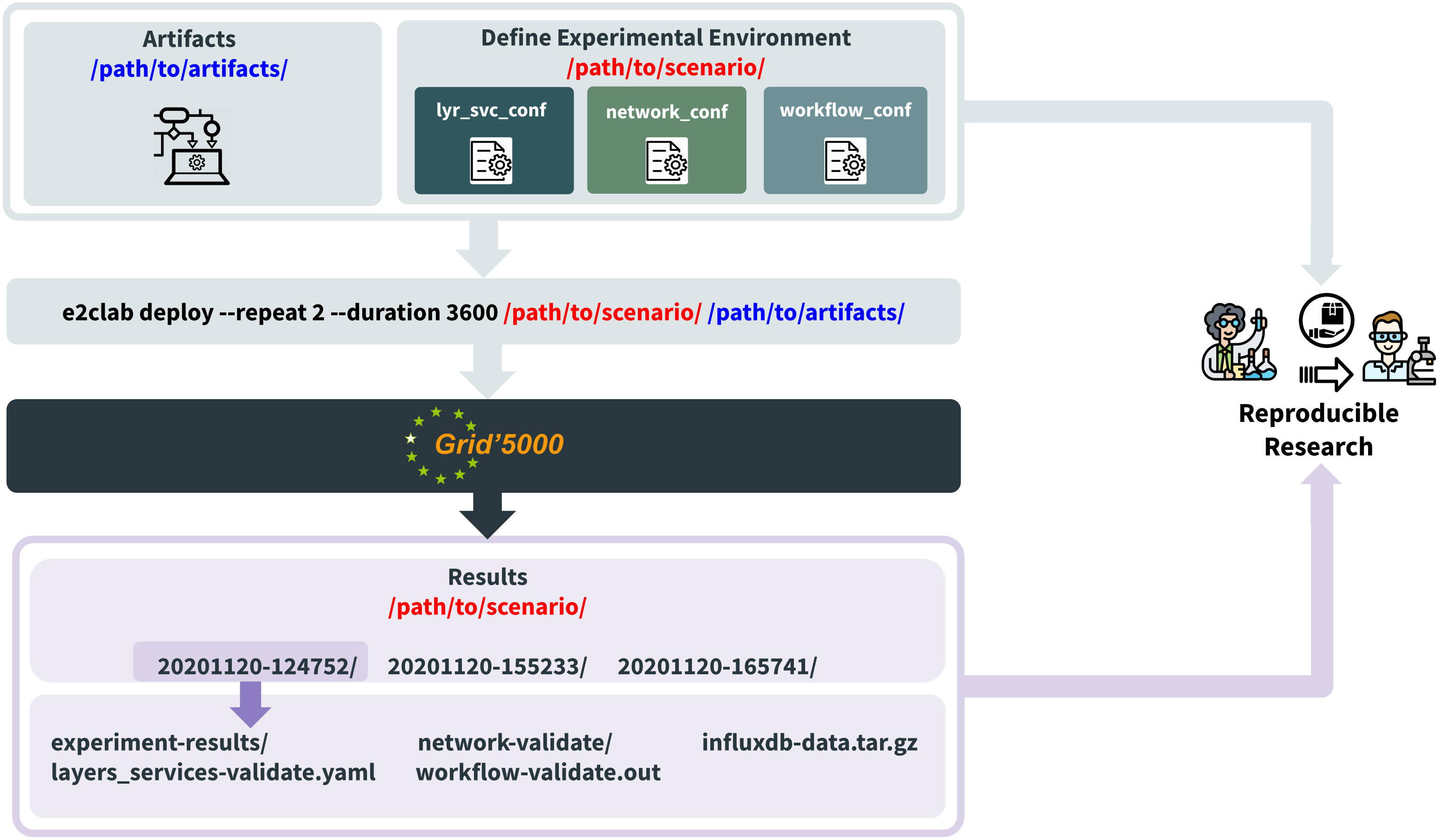


Defining the experimental environment: ***Workflow***

```yaml
hosts: edge.zone_A.*
  depends_on:
    service_selector: "fog.mosquitto.*"
    grouping: "round_robin"
    prefix: "mosquitto"
  prepare:
    - copy:
        src: "{{ working_dir }}/libs/"
        dest: "/opt/lib/"
  launch:
    - shell: "java my-app.jar tcp://{{ mosquitto.url }}
                              /opt/metrics/throughput"
  finalize:
    - fetch:
        src: "/opt/metrics/throughput"
        dest: "{{ working_dir }}/experiment-results/producers/"
hosts: ...
  ...
hosts: ...
```



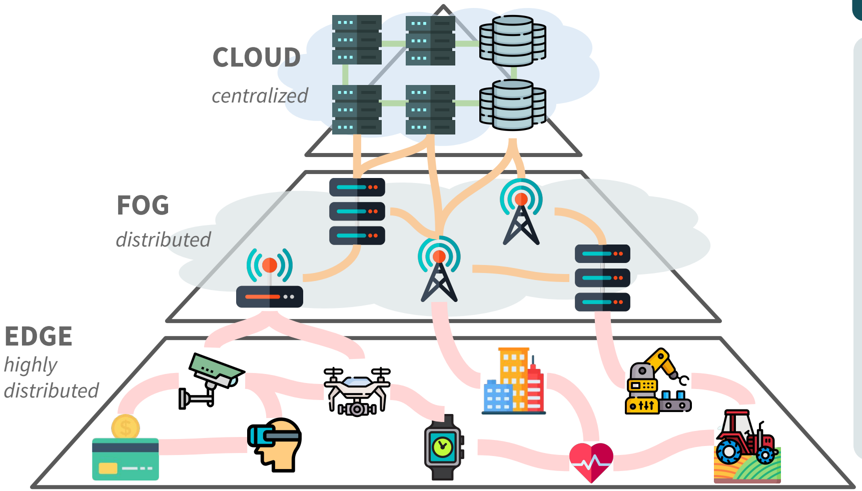
Experimental cycle in **E2C*lab***



Research Challenge 2: **Optimizing Performance**

How to configure the system components to minimize the processing latency?

Metrics
energy, cost, resource usage & latency
Objective
minimize processing time
Constraints
energy consumption limits &
equipment cost < budget
Variables
config. params. of Edge, Fog, and Cloud systems



Where should application parts be executed to minimize communication costs and end-to-end latency?

Metrics
costs, latency, & resource usage
Objectives
minimize communication costs &
minimize end-to-end latency
Constraints
budget & hardware resource limits
Variables
Edge-to-Cloud network communication links (bandwidth and delay) & number of Fog nodes per Edge device

Problem Statement

The performance optimization of application workflows on highly heterogeneous resources is challenging!

Heterogeneous constraints
computing resources;
network communication;
application requirements;

+

Search space complexity
myriad of combination of possibilities;
multi-objective problems;
NP-hard complex;

+

Selecting the most accurate optimization technique
multiple methods that perform differently;
hyperparameter search;

Our Optimization Methodology

IEEE Cluster 2021
https://hal.archives-ouvertes.fr/hal-03310540

Phase I

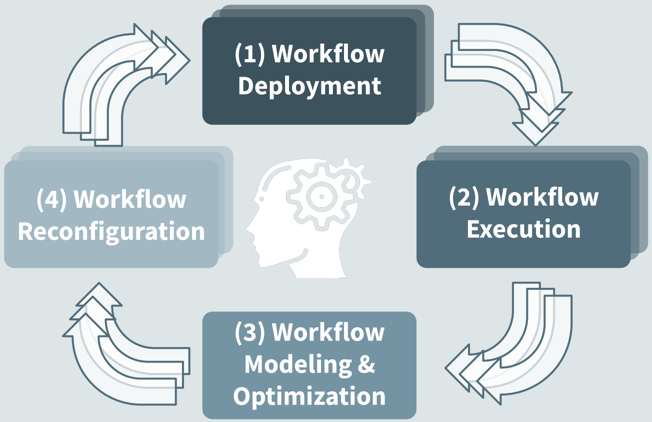
Define Optimization Problem

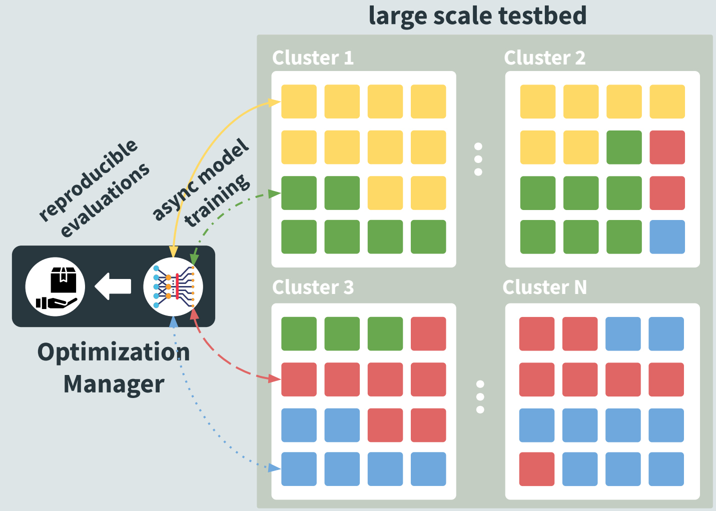
$$
\begin{array}{llll}
\underset{x}{\min/\max} & f_m(x), & m = 1, 2, \ldots, M & \\
\text{subject to} & g_j(x) \leq 0, & j = 1, 2, \ldots, J & \textit{Inequality constraints.} \\
 & h_k(x) = 0, & k = 1, 2, \ldots, K & \textit{Equality constraints.} \\
 & x_i^L \leq x_i \leq x_i^U, & i = 1, 2, \ldots, I & \textit{Bounds on variables.}
\end{array}
$$

Phase II

Parallel + Scalable + Reproducible

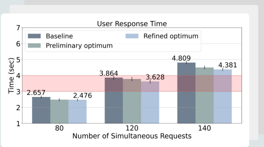
application optimization on large scale testbeds

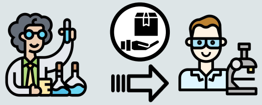




Phase III

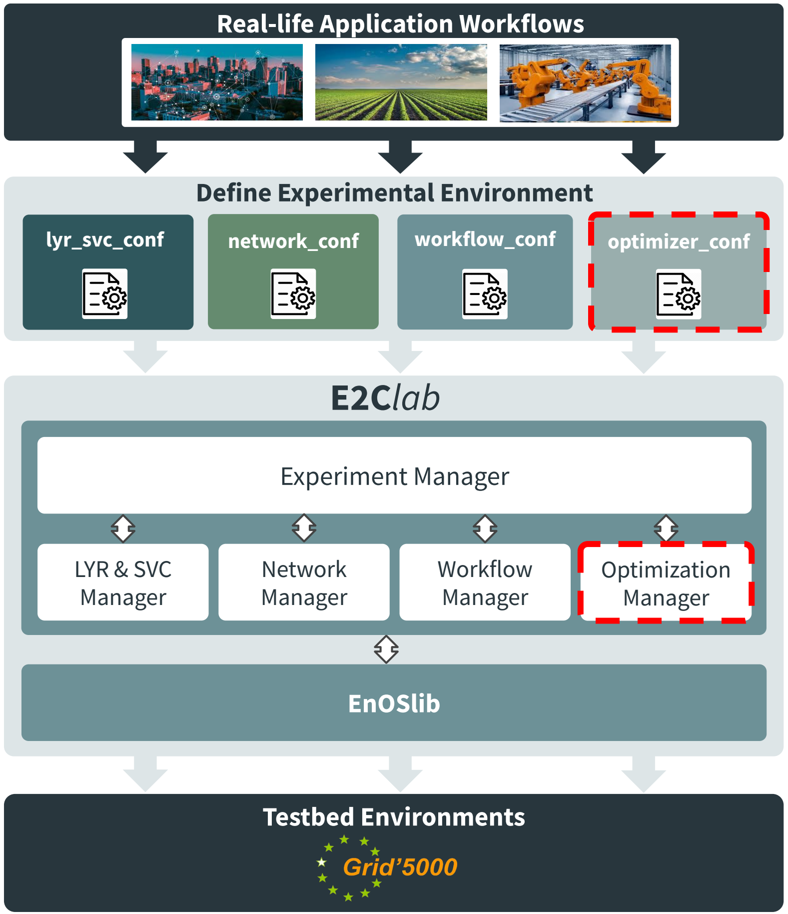
Summary of Computations





Reproducible Research

Implementation in **E2C*lab***



How to setup an optimization?

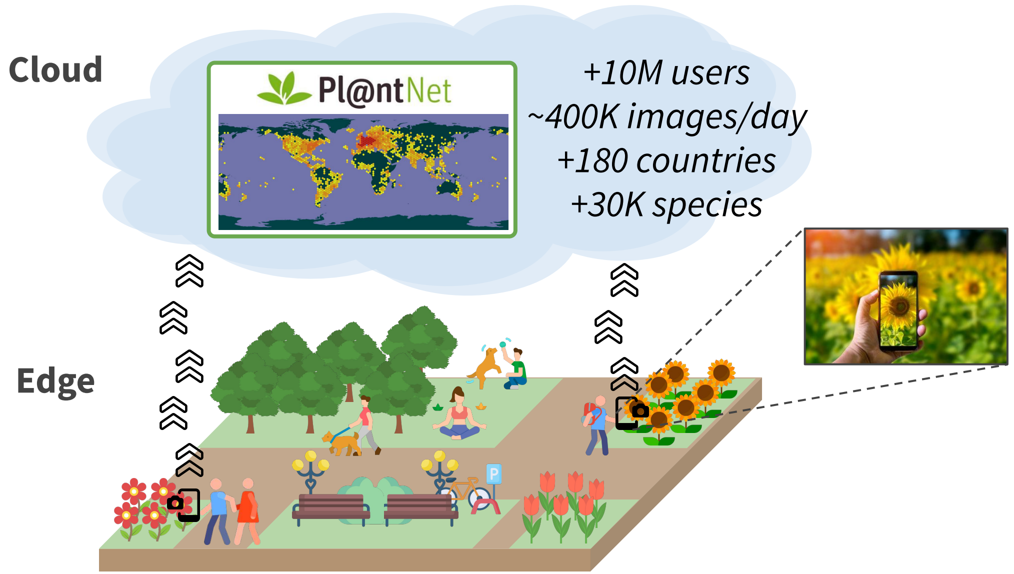
User-defined optimization

```python
from e2clab.optimizer import Optimization

class UserDefinedOptimization(Optimization):

    def run(self):
        algo = SkOptSearch(
            optimizer=Optimizer(
                base_estimator='ET',
                n_initial_points=45,
                initial_point_generator="lhs",
                acq_func="gp_hedge"))
        algo = ConcurrencyLimiter(algo,
    max_concurrent=2)
        scheduler = AsyncHyperBandScheduler()
        objective = tune.run(
            self.run_objective,
            metric="user_resp_time",
            mode="min",
            name="plantnet_engine",
            search_alg=algo,
            scheduler=scheduler,
            num_samples=10,
            config={
                "http": tune.randint(20, 60),
                "download": tune.randint(20, 60),
                "simsearch": tune.randint(20, 60),
                "extrac": tune.randint(3, 9)})

    def run_objective(self, _config):
        # create an optimization directory
        self.prepare()
        # deploy the configs on the testbed
        self.launch()
        # backup the optimization computations
        self.finalize()
        # report the metric value to Ray Tune
        tune.report(user_resp_time=user_resp_time)
```

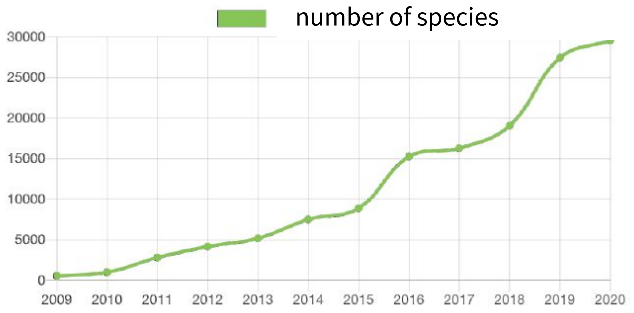
Supports state-of -the-art Bayesian Optimization libraries

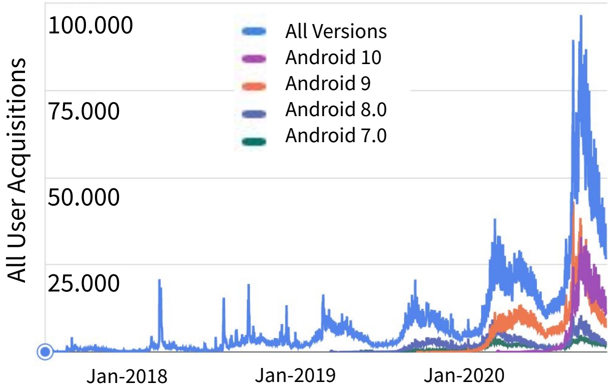
among others!!!

Validation with a Large-scale Real-life Application: Pl@ntNet



Download on the App Store | GET IT ON Google Play | Pl@ntNet Online Version





*Pl@ntNet has an **exponential grow** of **new users acquisition** every spring (peaks in May-June)*

Understanding and Optimizing the Performance of Pl@ntNet

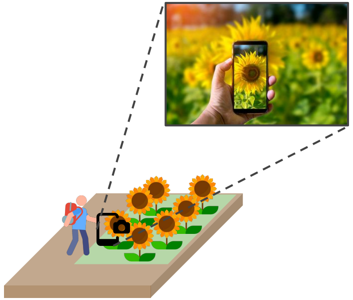
Validation with a Large-scale Real-life Application: Pl@ntNet

Pl@ntNet Identification Engine

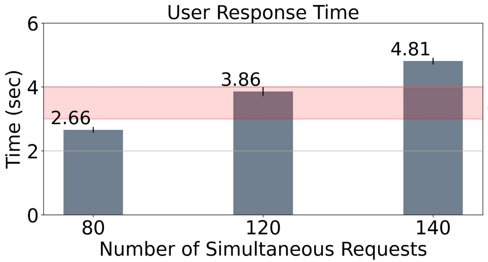
Thread pool	baseline (# threads)	Description	Hardware
HTTP	40	# simultaneous requests being processed.	CPU
Download	40	# simultaneous images being downloaded.	CPU
Extract	7	# simultaneous inferences in a single GPU.	**GPU**
Simsearch	40	# simultaneous similarity search.	CPU

Main performance metric: user response time

Pl@ntNet



“Over **3-4 seconds** more than **60%** of users **abandon** the transaction and may even **delete** the application”

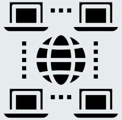


Research Questions

What is the **software configuration** that **minimizes the user response time**?

How do the **Extraction** and **Similarity Search** thread pool configurations impact the **user response time**?

How does the **number of simultaneous users** accessing the application **impact on the user response time**?



Large scale experimental evaluations

Grid’5000

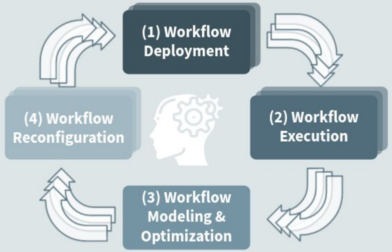
42 nodes

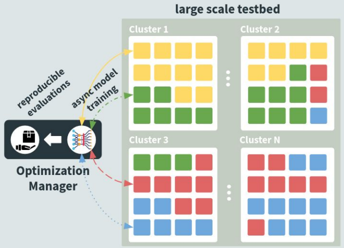
What is the **software configuration** that **minimizes the user response time**?

Phase I
Define Optimization Problem

$$
\begin{array}{ll}
\text{Find} & (http, download, simsearch, extract),\ \ \textit{in order to} \\
\text{Minimize} & UserResponseTime \\
\text{Subject to} & 20 \leq (http, download, simsearch) \leq 60,\ \ \textit{Pool Size.} \\
 & \ 3 \leq (extract) \leq 9, \qquad\qquad\qquad\qquad \textit{Pool Size.}
\end{array}
$$

Phase II
Parallel, Scalable, Reproducible Workflow Optimization





Bayesian Optimization
Extra Trees regressor

Phase III
Summary of Computations

Thread pool	baseline	preliminary optimum
HTTP	40	**54**
Download	40	**54**
Extract	7	7
Simsearch	40	**53**
User response time (sec)	**2.65**	**2.48**

preliminary optimum

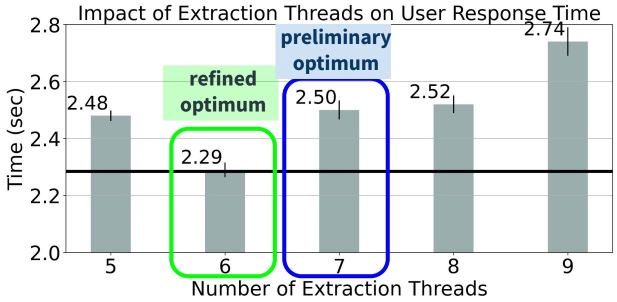
serves 35% more simultaneous users

reduces the user response time by 7%

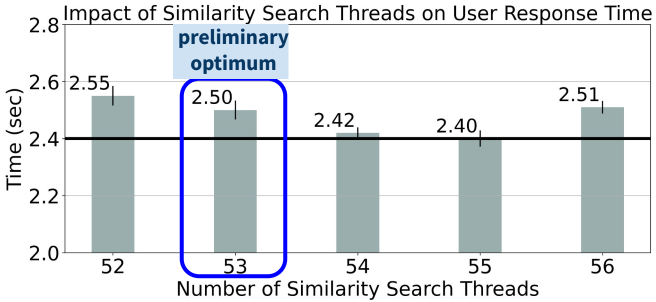
How do the **Extraction** and **Similarity Search** thread pool configurations impact the **user response time**?

Sensitivity Analysis: One-at-a-time (OAT) method

Extraction



Similarity Search



Thread pool	baseline	preliminary optimum	refined optimum
HTTP	40	54	54
Download	40	54	54
Extract	7	7 **[5, 6, 8, 9]**	6
Simsearch	40	53 **[50, 51, 52, 54, 55, 56]**	53
User response time (sec)	**2.65 (±0.0914)**	**2.48 (±0.0912)**	**2.47 (±0.0826)**

preliminary optimum

serves 35% more simultaneous users

reduces the user response time by 7%

refined optimum

reduces the GPU memory usage by 12%

How does the **number of simultaneous users** accessing the application **impact on the user response time**?

Thread pool	baseline	preliminary optimum	refined optimum
HTTP	40	54	54
Download	40	54	54
Extract	7	7	6
Simsearch	40	53	53

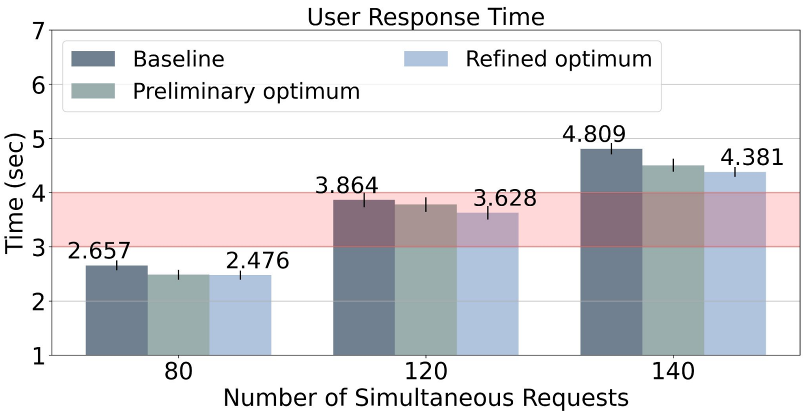
refined optimum

serves 35% more simultaneous users

reduces the user response time by 7%

reduces the GPU memory usage by 12%

"Over **3-4 seconds** more than **60%** of users **abandon** the transaction and may even **delete** the application"



Main takeaways

Our methodology has proved useful for **understanding** and **improving** the **performance** of a **real life application** used at very large-scale.

The **Pl@ntNet** configuration found **using our methodology** serves **35% more simultaneous users** and **reduces the user response time by 7%** compared to the baseline.

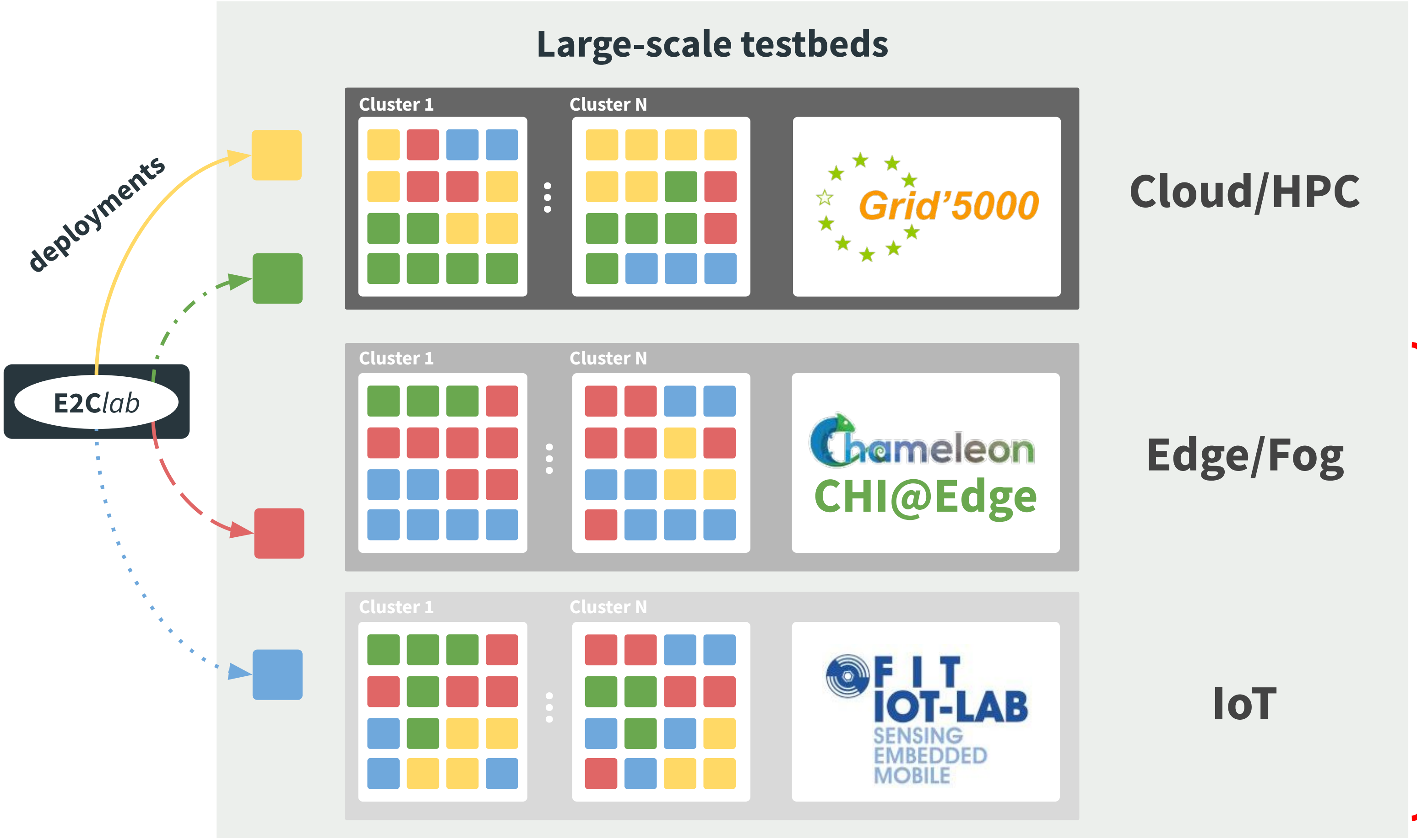
Feedback to G5K staff: **availability of GPU nodes was the main limitation**

Site	Cluster	# Nodes	GPU	# GPUs	RAM	NIC
Lille	chifflet	8	Nvidia GTX 1080 Ti	2	11GB	eth0/eno1, eth1/eno2
Lille	*chifflot*	*6* ***2***	*[1-6] Nvidia Tesla P100* ***[7-8] Nvidia Tesla V100-PCIE***	*2* *2*	*16GB* ***32GB***	*eth0/ens5f0, eth1/ens5f1*
Lyon	*gemini*	***2***	***Nvidia Tesla V100-SXM2***	*8*	***32GB***	*eth0/enp1s0f0*
Lyon	orion	4	Nvidia Tesla M2075	1	5GB	eth0/enp68s0f0
Lyon	*neowise*	***10***	***AMD Radeon Instinct MI50***	*8*	***32GB***	*eth0/eno1, eth1/eno2*
Grenoble	*drac*	*12*	*Nvidia Tesla P100* **(CPU=Power POWER8NVL 1.0)**	*4*	*16GB*	*eth0/enP1p1s0f0*
Nancy	*grouille*	***2***	***Nvidia A100-PCIE***	*2*	***40GB***	*eth2/eno33*
Nancy	*gruss*	***4***	***Nvidia A40***	*2*	***45GB***	*eth2/eno33*
Nancy	graffiti	12 1	[1-12] Nvidia RTX 2080 Ti [13] Nvidia Quadro RTX 6000	4 4	11 GB 22 GB	eth2/ens4f0
Nancy	graphique	5	Nvidia GTX 980	2	4GB	eth0/eno1
Nancy	grele	14	Nvidia GTX 1080 Ti	2	11GB	eth0/eno1
Nancy	grimani	6	Nvidia Tesla K40M	2	12GB	eth0/eno1
Nancy	grue	5	Nvidia Tesla T4	4	15GB	eth0/eno1

Pl@ntNet requires ~27GB of GPU RAM

Source (October 1st 2020 / December 2nd, 2021): https://www.grid5000.fr/w/Hardware

Ongoing work: **from IoT/Edge to Cloud/HPC environments**



work in progress

supporting FIT IoT-LAB + CHI@Edge

Ongoing work: **Provenance capture in E2C*lab***

Provenance capture for Edge-to-Cloud experiments as a **support to the analysis and debugging** of experiment results

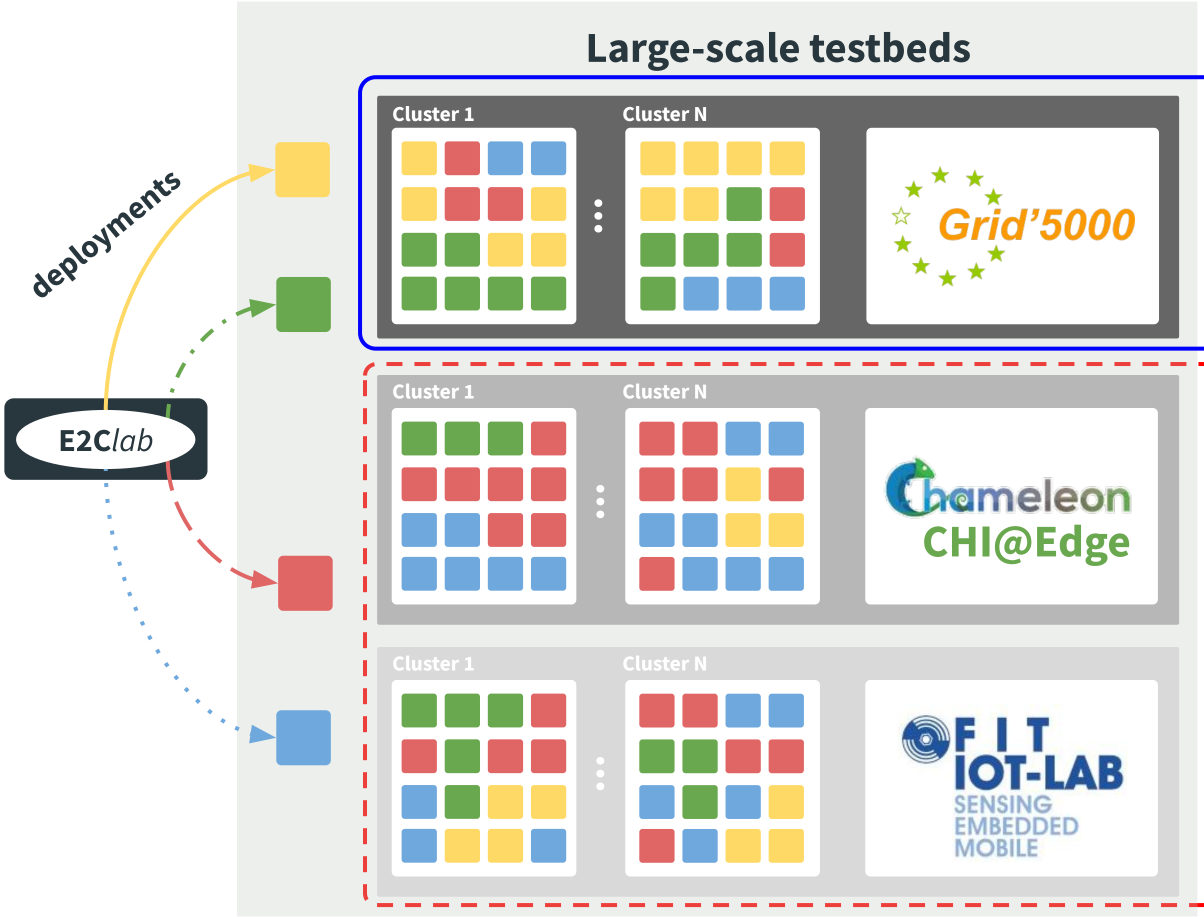
Goal: understand how experiment results have been produced

- What **parameters produced** these **results**?
- What **steps** did I **invoke** during **workflow execution**?

In collaboration with:

- Marta Mattoso (Federal University of Rio de Janeiro, Brazil)

Ongoing work: **Provenance capture**



Cloud-like
computing resources

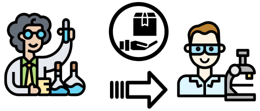
Komadu,
DfAnalyzer,
among others.

Edge-like
computing resources
?

How do the existing provenance systems **perform** in **resource-constrained** environments?

How to **minimize** the provenance **capture overhead** in **IoT/Edge** environments?

Final Considerations



Reproducible Research

- Access to the experiment **artifacts**, **results**, and **definition of the experimental environment**
    - https://gitlab.inria.fr/E2Clab/Paper-Artifacts/plantnet

- **E2C***lab* is **open source!** https://gitlab.inria.fr/E2Clab/e2clab
- **Documentation**: https://e2clab.gitlabpages.inria.fr/e2clab/

Acknowledgments

- **Inria**
    - Matthieu Simonin, Alexandru Costan, Gabriel Antoniu, Patrick Valduriez
- **Hasso-Plattner-Institut**
    - Pedro Silva
- **Pl@ntNet team**
    - Jean-Christophe Lombardo, Alexis Joly
- **Argonne National Laboratory**
    - Romain Egele, Jaehoon Koo, Prasanna Balaprakash, Orcun Yildiz
- **PhD funded by:** HPC-BigData Inria Project LAB

HPC-BigData
Inria Project LAB

Thank you!