KheOps: Cost-effective Repeatability, Reproducibility, and Replicability of Edge-to-Cloud Experiments

Daniel Rosendo
daniel.rosendo@inria.fr
Univ Rennes, Inria, CNRS, , IRISA
Rennes, France

Kate Keahey
keahey@mcs.anl.gov
Argonne National Laboratory
Chicago, USA

Alexandru Costan
alexandru.costan@inria.fr
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

Matthieu Simonin
matthieu.simonin@inria.fr
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

Patrick Valduriez
patrick.valduriez@inria.fr
Univ Montpellier, Inria, CNRS, LIRMM
Montpellier, France

Gabriel Antoniu
gabriel.antoniu@inria.fr
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

ABSTRACT

Distributed infrastructures for computation and analytics are now evolving towards an interconnected ecosystem allowing complex scientific workflows to be executed across hybrid systems spanning from IoT Edge devices to Clouds, and sometimes to supercomputers (the Computing Continuum). Understanding the performance trade-offs of large-scale workflows deployed on such complex Edge-to-Cloud Continuum is challenging. To achieve this, one needs to systematically perform experiments, to enable their reproducibility and allow other researchers to replicate the study and the obtained conclusions on different infrastructures. This breaks down to the tedious process of reconciling the numerous experimental requirements and constraints with low-level infrastructure design choices.

To address the limitations of the main state-of-the-art approaches for distributed, collaborative experimentation, such as Google Colab, Kaggle, and Code Ocean, we propose KheOps, a collaborative environment specifically designed to enable cost-effective reproducibility and replicability of Edge-to-Cloud experiments. KheOps is composed of three core elements: (1) an experiment repository; (2) a notebook environment; and (3) a multi-platform experiment methodology.

We illustrate KheOps with a real-life Edge-to-Cloud application. The evaluations explore the point of view of the authors of an experiment described in an article (who aim to make their experiments reproducible) and the perspective of their readers (who aim to replicate the experiment). The results show how KheOps helps authors to systematically perform repeatable and reproducible experiments on the Grid5000 + FIT IoT LAB testbeds. Furthermore, KheOps helps readers to cost-effectively replicate authors experiments in different infrastructures such as Chameleon Cloud + CHI@Edge testbeds, and obtain the same conclusions with high accuracies (>88% for all performance metrics).

CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**; • **General and reference** → **Experimentation**; **Measurement**;

KEYWORDS

Reproducibility, Replicability, Repeatability, Computing Continuum, Workflows, Edge Computing, Cloud Computing

1 INTRODUCTION

Modern scientific workflows require hybrid infrastructures, combining resources and services executed on the IoT/Edge with other resources and services running on Clouds or on HPC systems (the *Computing Continuum* [39]) to enable their optimized execution. Due to the complexity of application deployments on such highly distributed and heterogeneous Edge-to-Cloud infrastructures, realizing the Computing Continuum vision in practice remains burdensome.

One challenge stems from systematically performing experiments on the continuum. In particular, the processes enabling their reproducibility, as well as the replication of the performance trade-offs are inherently difficult [41]. Figure 1 illustrates such processes. Let us consider the case of a group of researchers who execute their experiments on French scientific testbeds such as Grid'5000 [35] (providing Cloud/HPC servers) and FIT IoT LAB [32] (providing IoT/Edge devices), and want to publish their results in an article. Next, the readers want to replicate the experiments on American testbeds such as the Chameleon Cloud [44] and CHI@Edge [43].

These processes compel a lot of effort, are time-consuming, and bring many technical challenges for both sides. For instance, also depicted in Figure 1, they require: (1) following methodologies to systematically design the experiments and to reconcile many application requirements or constraints in terms of energy consumption, network efficiency, and hardware resource usage; (2) configuring systems and networks, and deploying applications on testbeds for large-scale evaluations; (3) analyzing, repeating experiments, and publishing results; and (4) finally, providing open access to the experiment artifacts in a public and safe repository.

Given such complexities, researchers end up not following rigorous methodologies for supporting the reproducibility of the experiments, as observed in our previous survey [52] and summarized in Figure 2. As a consequence, it makes it hard for other researchers to replicate the published studies [46].

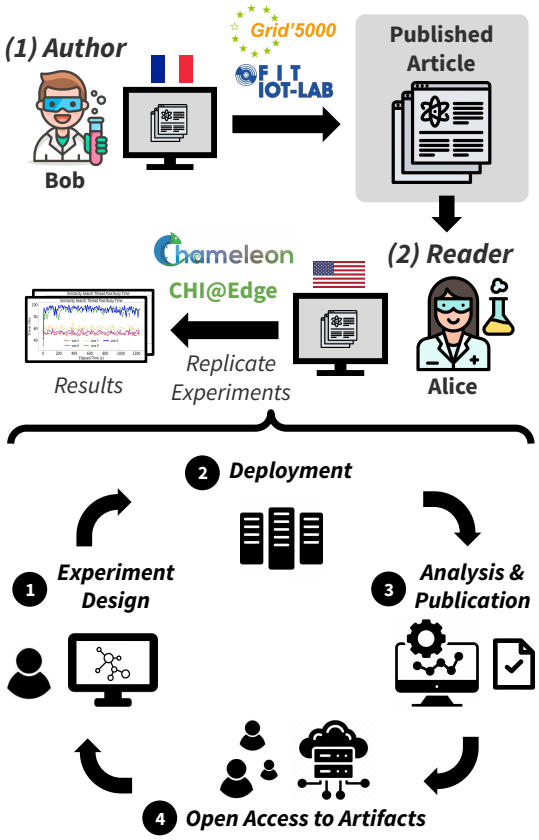


Figure 1: Processes for reproducing and replicating experiments regarding the authors and readers point of view.

Let us sum up the associated requirements in this context [40, 55]. To enable **reproducible** experiments on the Edge-to-Cloud continuum, the requirements (a-REQ) of the authors of the experiments can be described as follows:

a-REQ 1. Execute experiments on heterogeneous computing resources (*e.g.,* IoT/Edge and Cloud/HPC infrastructures).
a-REQ 2. Systematically describe and explain the experimental processes and their reasoning.
a-REQ 3. Efficiently configure the experimental infrastructure and express topologies in repeatable ways.
a-REQ 4. Easily share the experiment artifacts in a public and safe repository.

At the same time, to enable the **replicability** of the experiments, the readers of an article describing those experiments have the following requirements (r-REQ):

r-REQ 1. Find and access the experiment as simply as finding and reading its paper.
r-REQ 2. Perform the experiment, not just read about it.
r-REQ 3. Answer not just to the *"What"* question (What the experiment does?), but also the *"Why"* (Why did authors set up that way?) and *"How"* (How did authors connect machines/devices?)

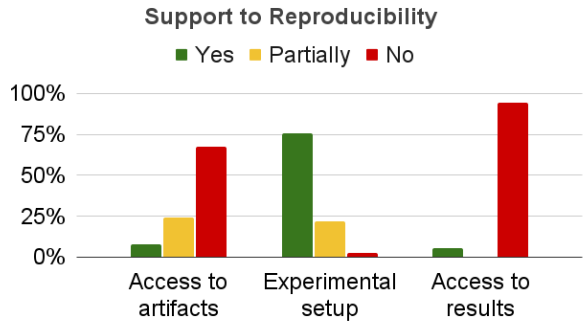


Figure 2: Support to the reproducibility of Edge-to-Cloud experiments provided by the 34 studies in our survey [52].

r-REQ 4. Efficiently configure the experimental infrastructure to reduce the time spent satisfying all the experiment requirements.

In this paper, we study the challenges of reproducing and replicating Edge-to-Cloud experiments in cost-effective ways. **Cost-effective means** to allow authors and readers to easily fulfill their experimental requirements as previously described. This calls for practical solutions beyond the state-of-the-art.

Our main objective is to provide a collaborative environment and methodology that supports reproducible Edge-to-Coud experimentation between different open testbeds such as Grid'5000, FIT IoT LAB, Chameleon, *etc.*, equipped to deal with IoT/Edge and Cloud/HPC resources which are fundamental to reproducibility [42]. We propose the following main contributions:

(1) **A study of the characteristics of the main state-of-the-art collaborative environments** (*e.g.,* Google Colab, Kaggle, and Code Ocean) for enabling reproducible experiments. Their **main limitations in the context of Computing Continuum research** are discussed in Section 3.
(2) **A novel collaborative environment to enable reproducible Edge-to-Cloud experiments** (Section 4). This approach, named KheOps, allows researchers to reproduce and replicate Edge-to-Cloud workflows cost-effectively. KheOps core elements are: (1) a portal for sharing experiment artifacts; (2) a notebook environment for packaging code, data, environment, and results; and (3) a **multi-platform** experimental methodology for deploying experiments on heterogeneous resources from the IoT/Edge (FIT IoT LAB and CHI@Edge) to the Cloud/HPC Continuum (Grid5000 and Chameleon). We highlight that KheOps may be integrated with other large-scale scientific testbeds.
(3) An **experimental validation** of the proposed approach with a **real-world use case deployed on real-life IoT/Edge devices and Cloud/HPC systems**. The evaluations show that KheOps helps: (1) authors to perform reproducible experiments on the **Grid5000 + FIT IoT LAB** testbeds, and (2) readers to cost-effectively replicate authors experiments on the **Chameleon Cloud + CHI@Edge** testbeds, and **obtain the same conclusions with high accuracies**, >88% for all performance metrics (Section 5).

Table 1: ACM Digital Library Terminology Version 1.1 [1]

Repeatability	*Same team, same experimental setup: the measurement can be obtained with stated precision by the same team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same location on multiple trials. For computational experiments, this means that a researcher can reliably repeat their own computation.*
Reproducibility	*Different team, same experimental setup: the measurement can be obtained with stated precision by a different team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using the author's own artifacts.*
Replicability	*Different team, different experimental setup: the measurement can be obtained with stated precision by a different team, a different measuring system, in a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artifacts which they develop completely independently.*

2 BACKGROUND

In this section, we start by defining the terms repeatability, reproducibility, and replicability (Section 2.1). Next, we explore the following research question from the Computing Continuum perspective: ***What would a good collaborative system look like?*** In our vision, it should: (1) allow users to share research artifacts in a public and safe repository (Section 2.2); (2) provide an environment for setting up and describing experiments step-by-step (Section 2.3); (3) provide experimental methodologies to leverage heterogeneous Edge-to-Cloud computing resources from various scientific testbeds, at large-scale (Section 2.4).

2.1 Repeatability, Reproducibility, Replicability

An important requirement for researchers from various communities is that the scientific claims be verifiable by others (*i.e.,* building upon published results). As illustrated in Figure 2, such requirement is hardly satisfied in the context of Computing Continuum experiments. This can be achieved through repeatability, reproducibility, and replicability (3Rs) [34, 56]. There are many non-uniform definitions of the 3Rs in literature. In this work, we follow the terminology proposed by the ACM Digital Library [1] (Artifact Review and Badging version 1.1), as presented in Table 1.

Achieving **repeatability** means that one can reliably repeat the experiments and obtain precise measurements (*e.g.,* Edge to Cloud processing latency, memory consumption, among others) by using the same methodology and artifacts (*i.e.,* same testbed, same physical machines, same libraries/framework, same network configuration). Executing multiple experiments allows us to explore different scenario settings (*e.g.,* varying the number of Edge devices) and explore the impact of various parameters (*e.g.,* the network configuration between Edge devices and the Cloud server) on the performance metrics.

Reproducibility means that external researchers having access to the original methodology (*e.g.,* configuration of physical machines, network and systems, scenario descriptions) and using their own artifacts (*i.e.,* data sets, scripts, AI frameworks, *etc.*) can obtain precise measurements of the application processing latency and throughput, for instance.

Replicability refers to independent researchers (*i.e.,* the readers of an article that was published by a different team) having access to the original methodology and artifacts (*e.g.,* configuration of physical machines, processing steps, network setup, *etc.*) and performing the experiments in different testbeds. The goal is that independent researchers can obtain precise results and conclusions consistent with the original study.

2.2 Trovi sharing portal

Collaborative systems should be integrated with public and safe repositories providing open access to the research artifacts to enable the reproducibility of experiments. Repositories like Trovi [28], Kaggle [21], Code Ocean Explorer [9], AI Hub [7], GitHub [4], and Zenodo [5] allow users to store versioned and citeable (*e.g.,* through a DOI: Digital Object Identifier) artifacts such as code, datasets, or Jupyter notebooks, among others.

In this work, we leverage on the Trovi sharing portal because it provides a public REST API that facilitates integration with existing systems. Furthermore, Trovi provides a series of features to manage research artifacts such as: integration with GitHub and Zenodo; creating, packaging, and sharing artifacts as Jupyter notebooks with 500MB in total size by default; support for scientific testbeds like Chameleon, which allows users to re-launch the available artifacts on the testbed.

2.3 Jupyter environment

Another important aspect for reproducible and replicable experiments is that collaborative systems support executable research packages composed of code, data, environment configurations, and experiment results. The most popular open-source solutions are Jupyter notebooks [45] and Apache Zeppelin [8]. In this work, we use Jupyter notebooks for packaging research artifacts due to its wider compatibility with operating systems and programming languages, and the community support.

The Jupyter project consists of JupyterHub, JupyterLab, and notebooks. JupyterHub aims to serve Cloud-based Jupyter notebooks for multiple users. The goal is to provide users a ready-to-use computational environment with their own workspace on shared resources. JupyterHub servers are customizable, scalable, and portable on a variety of infrastructures. It is composed of a Hub that manages the following sub-services: a proxy that receives requests from clients; spawners to monitor notebook servers; and an authenticator to manage how users access the system.

JupyterLab refers to a web-based user interface providing mainly: notebook, terminal, text editor, file browser, and rich outputs. It allows users to configure and arrange their experimental workflows, as well as adding extensions to expand and enrich functionalities.

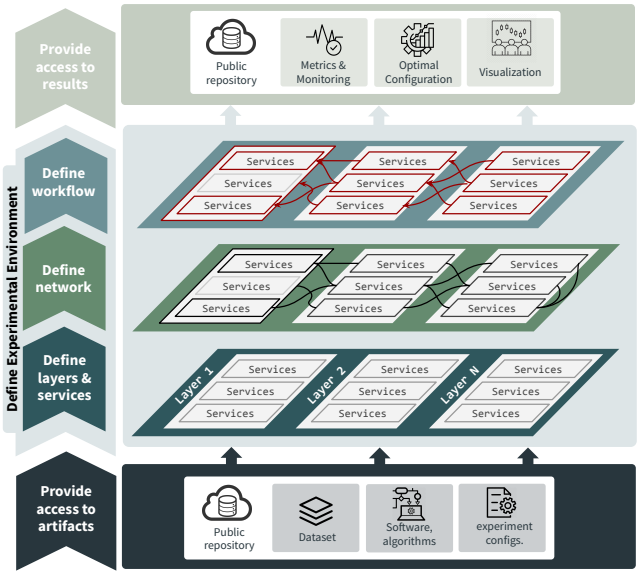


Figure 3: E2C*lab* experiment methodology [53].

Finally, notebooks allow users to create programming documents combining: (1) formatted text (*e.g., prospective data* that explains each step of an experiment workflow); (2) executable code with the respective outputs (*e.g., retrospective data derived by the execution*); and (3) experimental results with visualizations and various sorts of rich media, such as images and videos.

2.4 E2Clab experimental methodology

Understanding and optimizing workflow performance requires executing and reproducing complex experiments at large scale. Several existing environments aid users to run such experiments. Their limitations are discussed in the next section and summarized in Table 2. Based on these findings and the specific Computing Continuum requirements, in this work we leverage the E2Clab methodology.

E2Clab [53] is an open-source framework (available at [6]) that implements a rigorous methodology (illustrated in Figure 3) for designing experiments with real-world workloads on the Edge-to-Cloud Continuum. It allows researchers to reproduce the application behavior in a controlled environment in order to understand and optimize performance [51]. E2Clab sits on top of EnOSlib [36] to enforce the experiment configurations on testbeds. High-level features provided by E2Clab are: *(i)* reproducible experiments; *(ii)* mapping application parts (Edge, Fog and Cloud/HPC) and physical testbeds; *(iii)* experiment variation and transparent scaling of scenarios; *(iv)* defining Edge-to-Cloud network constraints; *(v)* experiment deployment, execution and monitoring (*e.g.,* on Grid'5000, Chameleon, and FIT IoT LAB); and *(vi)* workflow optimization.

3 LIMITATIONS OF EXISTING COLLABORATIVE ENVIRONMENTS

We briefly discuss the limitations of state-of-the-art collaborative environments, with a focus on the specific challenges of the Computing Continuum.

Google Colab [18]. Mainly used by the AI community (more than 50K users), it is a ready-to-use Jupyter notebook service. Colab notebooks are stored in the *.ipynb* open-source Jupyter notebook format [19], and come with the most popular AI libraries and frameworks installed (*e.g.,* Scikit-Learn [50], TensorFlow [31], PyTorch [49], *etc.*) and allow users to run python code through the browser. It is typically used for machine learning, data analysis and education. Colab is popular because it allows users to share Jupyter notebooks without having to download, install, or run anything. Besdides, it provides free access to very expensive computing resources such as GPUs and TPUs. Colab permits multiple users to collaborate on the same notebook. Sharing datasets, ML models, pipelines, and notebooks on AI Hub [7] is also possible (more than 167 notebooks). Its GitHub integration allows users to quickly open GitHub-hosted Jupyter notebooks in Google Colab.

Kaggle [21]. This is a data science and AI platform that offers a customizable Jupyter notebook environment. Kaggle is a subsidiary of Google and, like Colab, it provides free access to GPUs as well as a repository of community-published (more than 10.3 million users) datasets (more than 50K public datasets) and code (*e.g.,* machine learning code) with more than 400K public notebooks. Kaggle is integrated with AI Hub and is popular in the data science and machine learning communities. Kaggle is also well-known for promoting Community Competitions in machine learning at no cost. The main differences [20] between Colab and Kaggle are: (1) Kaggle allows collaboration with other users on its Web site, while Colab allows collaboration with anyone using the notebook link; (2) Kaggle has a lot of data sets that users can use directly (*e.g.,* notebooks already set up with Kaggle databases [22]), while in Colab setting up notebooks with Google Drive [11] or managing files [10] (*e.g.,* to load data sets, files, and images) requires extra work; and (3) Kaggle creates a history of notebook commits that we can be reviewed.

Code Ocean [37]. Designed according to FAIR [57] (*i.e.,* Findable, Accessible, Interoperable, and Reusable), Code Ocean aims to make scientific work reproducible. It introduces the concept of Compute Capsule, which refers to Docker [14] containers composed of code, data, environments, and results. Capsules provide ready-to-use tools such as Git, Jupyter, RStudio, among others. Its integration with Git allows users to save changes on capsules and then commit them with just one click. Furthermore, users can easily share the link of a capsule and grant permissions. Code Ocean provides scalable compute and storage resources hosted on Amazon Web Services. Resources used by capsules are scaled out when the demand exceeds the machine capacity. Finally, Code Ocean provides a public Capsule Repository [9] with more than 1K research capsules. It allows authors of an article to incorporate capsules into the submission process via a Hub publishing API.

Despite these systems being widely used by the AI and data science communities, they present some limitations that hinder their adoption for Computing Continuum research. Table 2 summarizes these limitations in terms of:

(1) access to heterogeneous computing resources, from the IoT/Edge to the Cloud/HPC;
(2) support for large-scale experimental evaluations;

Table 2: Limitations of Existing Collaborative Environments.

Limitation	Google Colab	Code Ocean	Kaggle
Resource heterogeneity	CPU, disk, and memory limits; GPU types available; no access to IoT/Edge devices;	experiments run on AWS virtual machines; no access to IoT/Edge devices;	limits CPU, GPU, and TPU access; does not support IoT/Edge devices;
Large-scale experiments	limits sessions to 12 hours; paid access to multiple computing resources.	limits access to 10 compute hours; paid access to multiple computing resources.	limits execution time to 12 hours; paid access to Google Cloud Services.
Repeatability, Reproducibility, Replicability	hard to repeat and reproduce experiments on the same hardware: resource availability varies over time and usage limits fluctuate. Replicability in different infrastructures (*e.g.,* beyond Google machines) is not straightforward.	lacks support for the reproducibility of distributed experiments. Computing and storage resources are available in AWS virtual machines in the clients virtual private cloud. Hard to replicate experiments in different infrastructures.	lacks support for the repeatability and reproducibility of distributed experiments. Computing resources vary over time and hence between accesses. Replicability in different infrastructures is not easy to set up.

```yaml
environment:
  g5k: cluster: dahu
  iotlab: cluster: grenoble
layers:
- name: cloud
  services:
  - name: Server
    environment: g5k, quantity: 1
- name: edge
  services:
  - name: Client
    environment: iotlab, archi: rpi3, quantity: 5
```

Listing 1: E2Clab: layers and services configuration. Hardware details described in Section 5.1.

```python
from e2clab.services import Service
import enoslib as en

class Server(Service):
   def deploy(self):
      with en.actions(roles=self.roles) as a:
        a.shell("pip3 install torchvision torch")
        a.shell("pip3 install pillow paho-mqtt")
      return self.register_service(port=[1883])
```

Listing 2: E2Clab: user-defined service for the Cloud server.

(3) repeatability and reproducibility of experiments on the same hardware setup, and replicability on different infrastructures.

In summary, collaborative environments lack support for providing access to heterogeneous resources (*e.g.,* Edge-to-Cloud); performing experiments at large-scale; and achieving the repeatability, reproducibility, and the replicability of experiments in different testbeds. Hence, the need for novel approaches for reproducible evaluations of workflows targeting the characteristics of the Computing Continuum.

4 KHEOPS DESIGN

This section introduces KheOps, a collaborative environment for the cost-effective reproducibility and replicability of Edge-to-Cloud experiments. KheOps is designed to meet the experimental requirements of both authors and readers as presented in Section 1.

4.1 Architecture and implementation

Figure 4 presents the architecture of KheOps, which consists of three main components: *(i)* Trovi sharing portal; *(ii)* Jupyter environment (JupyterHub service and JupyterLab server); and *(iii)* E2Clab framework (multi-platform experiment methodology). Next, present the integration details of KheOps three components, and we briefly describe their main roles.

4.1.1 Experiment repository. KheOps uses Trovi to share research artifacts such as packaged experiments. These artifacts may be publicly available to allow others to recreate and rerun experiments. Trovi provides a REST API to manage experiment artifacts and integrate them with other systems. The JupyterHub in KheOps uses the Trovi REST API to download artifacts and launch them in the JupyterLab server.

Artifacts hosted in Trovi can also provide references to repositories like container registries (*e.g.,* DockerHub [2]), multipurpose repositories (*e.g.,* Zenodo [5]), code repositories (*e.g.,* Github [4]), and among others.

4.1.2 Notebook environment. Following our previous work [33] on integrating experiment workflows with Jupyter notebooks, we extend JupyterHub to authenticate users and to download (using the Trovi REST API) the experiment artifacts available at Trovi. We also extend JupyterLab to allow users to easily share their experiments in Trovi. Furthermore, JupyterLab is set up with the E2Clab framework as an experimental methodology.

The JupyterLab is packaged with code, data, environment configurations, and experiment results. Its notebooks (file extension *.ipynb*) allow users to run experiments step-by-step by combining text (*e.g.,* explaining the reasoning of the experiments: *What* parameters? *Why* these parameters? and *How* it was set up?) with executable code. Such notebooks are ready to use (*e.g.,* installed with required library/software), executed through a browser, and shared as a Trovi artifact.

4.1.3 Multi-testbed experiment methodology. KheOps uses the E2Clab methodology to deploy experiments on large-scale scientific testbeds such as Grid'5000, Chameleon Cloud, CHI@Edge, and FIT IoT LAB. Notebooks come with three main template files (*e.g.,* executable code cells in the notebook, presented in Listings 1 to 4) that users can benefit from to easily configure and adapt the deployment logic

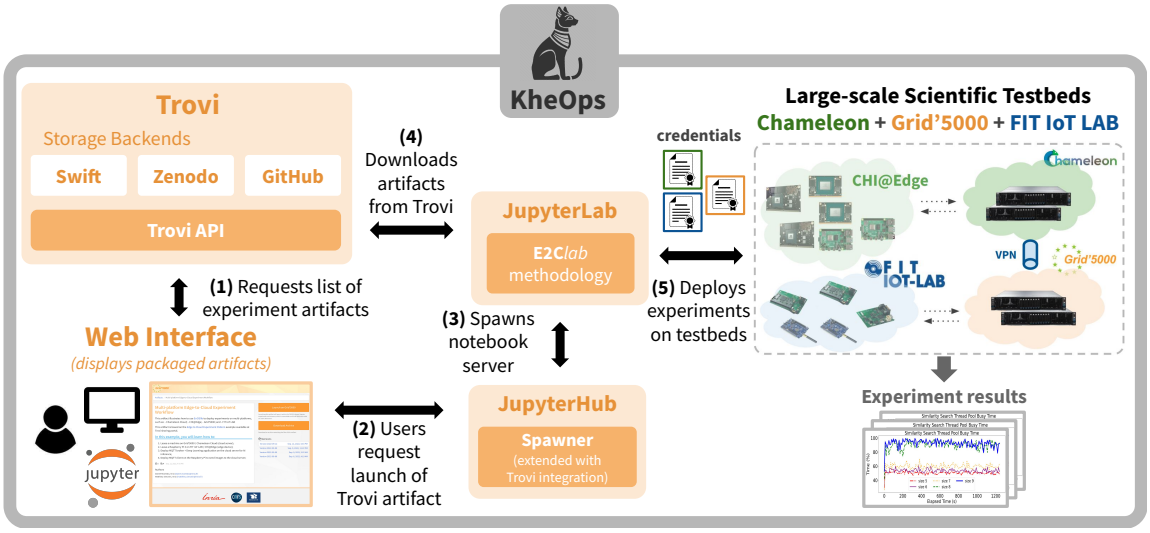


Figure 4: KheOps architecture and experimental workflow.

```
networks:
- src: cloud, dst: edge
  delay: "150ms", rate: "25kbit", loss: "0.02"
```

Listing 3: E2Clab: network configuration.

(*e.g.,* computing resources, network, and application execution) according to their experimental needs.

The first file, named *layers_services.yaml* and presented in Listing 1, allows users to lease IoT/Edge and Cloud/HPC resources. Through this file, users may also set up their applications and services as presented in Listing 2. Next, the *network.yaml* file (Listing 3) allows users to define delay, loss, and bandwidth between computing resources. Finally, the *workflow.yaml* file (Listing 4) guides users to define the experiment workflow through three main steps: *prepare* (*e.g.,* copy artifacts to remote nodes, install libraries, *etc.*), *launch* (*e.g.,* execute the application parts), and *finalize* (*e.g.,* backup results from remote nodes to the JupyterLab server).

E2Clab abstracts all the complexities of deploying and executing experiments across various testbeds. To do so, users need to add the credential files of the respective testbeds to their notebooks. Setting up a VPN is also supported as this may be required to enable the communication between different geographically distributed tesbeds (*e.g.,* Chameleon in the USA and Grid'5000 in France).

4.2 Experimental workflow

In summary, the workflow for launching an experiment artifact on large-scale testbeds consists of 5 main steps. First, through a web interface, users can browse the list of experimental artifacts publicly available in Trovi (step 1). Selecting an artifact displays details such as the experiment description, the authors and contact information, and the artifact versions.

A *launch* button allows users to execute the artifact (step 2). This button redirects users to the JupyterHub service. After authentication, the request to launch the artifact is sent to the JupyterHub Spawner. Next, the Spawner spawns the JupyterLab server (step 3)

```
- hosts: edge.*
  depends_on:
    conf_selector: cloud.server.*
    grouping: round_robin, prefix: "server"
  prepare:
  - copy:
      src:{{working_dir}}/artifacts, dest: /
  launch:
  - shell: bash /edge_worker.sh edge_data 100 "{{
    server.ip}}" False
  finalize:
  - fetch:
      src:/tmp/predict.log, dest:{{working_dir}}/
```

Listing 4: E2Clab: workflow configuration.

and then it downloads experimental artifacts such as notebooks, code, and datasets, among others (step 4). The JupyterLab service is set up with the E2Clab framework as the experimental methodology. Finally, users can execute the code cells from the notebook to lease IoT/Edge and Cloud/HPC computing resources available on the testbeds, deploy and execute the application, and gather the experiment results (step 5).

Steps 2 to 4 are automatically executed. This is a one-click feature that allows users to have a ready-to-use environment for reproducing and replicating complex Edge-to-Cloud experiments in a cost-effective manner. Note that the whole workflow requires only three clicks: selecting the experiment artifact (step 1); then launching it (steps 2 to 4); and executing it on the testbeds (step 5).

5 EVALUATION

In this section, we show how KheOps can be used to analyze the performance of a real-life Edge-to-Cloud application deployed in the African savanna (illustrated in Figure 5). This application is composed of distributed Edge devices monitoring animal migration in the Serengeti region. Devices at the Edge collect and compress wildlife images, then the image is sent to the Cloud where the animal classification happens using a pre-trained Neural Network

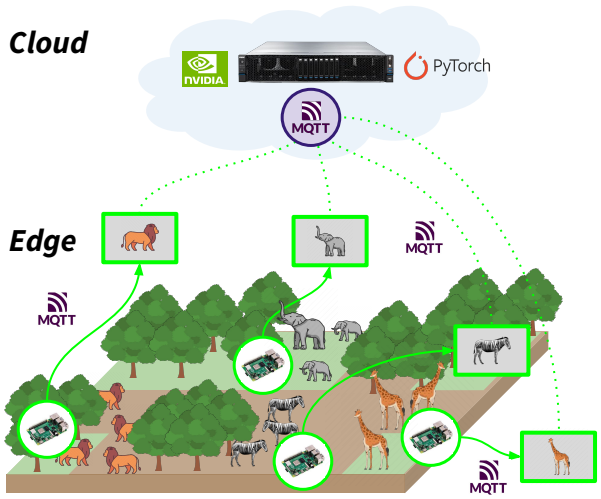


Figure 5: Edge-to-Cloud application: monitoring animals migration in the African savanna.

model. Finally, classified data helps conservationists to learn what management strategies work best to protect species.

The goals of these experiments are:

- to understand the impact on performance of **Cloud-centric** and **Hybrid (Edge+Cloud)** processing;
- to show how authors of an article can benefit from KheOps to make their **experiments reproducible**;
- to show how readers of an article can leverage KheOps to **replicate the experiments** in an article (published using KheOps).

To reproduce the evaluations in this section, refer to [16].

5.1 Experimental setup

Application performance metrics. The main tracked metric is the *processing time*, which refers to the time required to: pre-process the image captured (*e.g.,* image compression on the Edge device); transmit the image to the Cloud server; and finally decompress the image and predict the animal through an AI model. In addition, we analyze the *amount of data transmitted* to the Cloud and the *resource consumption* (*e.g.,* CPU and memory) on the Edge device.

To increase the accuracy of the results, we measure the processing duration 100 times for each experiment, each time with a different image and an interval of 30 seconds (*i.e.,* Edge devices transmit images to the Cloud server every 30 seconds). The remaining metrics are captured using Dool (Dstat) [3] at application runtime. All results are presented as the mean followed by their respective 95% confidence interval.

KheOps replicability metric. To measure how close/precise readers experiments are from authors experiments, we define the Replicability Accuracy ($Rep_{accuracy}$) metric. For assessing variability and error in results [48], a recommendation is to repeat the experiments multiple times to achieve narrower inferential error bars (i.e., confidence interval, standard deviation, *etc.*) [38]. The Replicability Accuracy metric is calculated as Equation 1:

$$Rep_{accuracy} = 1 - \left| \frac{min(x_{1A}, x_{2A})}{max(x_{1A}, x_{2A})} - \frac{min(x_{1R}, x_{2R})}{max(x_{1R}, x_{2R})} \right| \quad (1)$$

Ideally, $Rep_{accuracy}$ would be close to 1. x_{iA} and x_{iR} refer to the application performance metric value obtained from authors and readers experiments, respectively. For instance, in Figure 6a, x_{1A} refers to the Cloud-centric bar and x_{2A} to the Edge+Cloud bar.

Workload. Devices at the Edge transmit images (from the Snapshot Serengeti dataset [30] composed of millions of wildlife images collected annually) to the Cloud server that predicts animals using a trained MobileNetV3 Convolutional Neural Network model. We evaluate this workload considering two network configurations, 25Kbit and 15Kbit bandwidth with a round-trip delay of 150ms.

Software. On the Edge devices, we use the zlib [24] Python library to compress images. MQTT [23] protocol is used to transmit images to the Cloud server. On the Cloud server, we use an MQTT broker to receive images, then zlib to decompress images, and finally PyTorch to predict animals.

Hardware. The **authors** perform experiments on the following testbeds in France: Grid'5000 and FIT IoT LAB. On Grid'5000 (Cloud server), they use the dahu [13] machine equipped with an Intel Xeon Gold 6130 CPU 2.10GHz, 16 cores/CPU, 192GB of RAM, and Ethernet network. On FIT IoT LAB (Edge device), they use a Raspberry Pi 3 Model B [25] with four ARM Cortex-A53 processing cores running at 1.2GHz, 1GB LPDDR2 memory, and 2.4GHz 802.11ac wireless LAN.

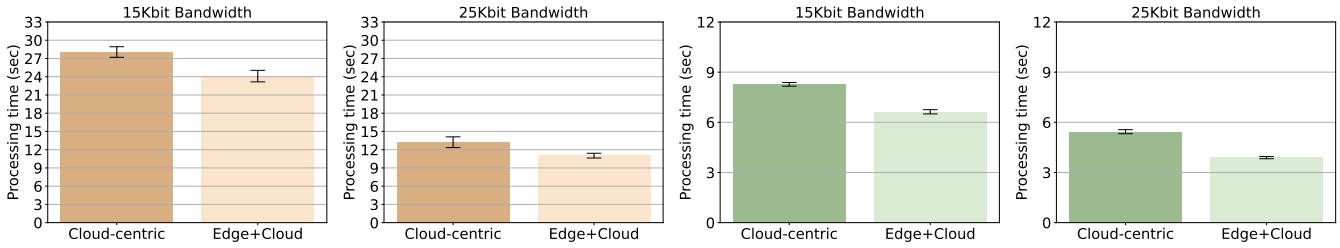
The **readers** replicate authors experiments on the following testbeds in USA: Chameleon Cloud and CHI@Edge. On Chameleon CHI@TACC (Cloud server), they use the Skylake [12] machine equipped with an Intel Xeon Gold 6126 CPU 2.60GHz, 12 cores/CPU, 192GB of RAM, and Ethernet network. On CHI@Edge (Edge device), they use a Raspberry Pi 4 [26], with four BCM2711 Cortex-A72 processing cores running at 1.5GHz, 8GB LPDDR4 memory, and 2.4GHz and 5GHz 802.11ac wireless LAN.

5.2 How KheOps helps experiment authors

Let us consider the requirements of the experiment authors (a-REQ) as introduced in Section 1.

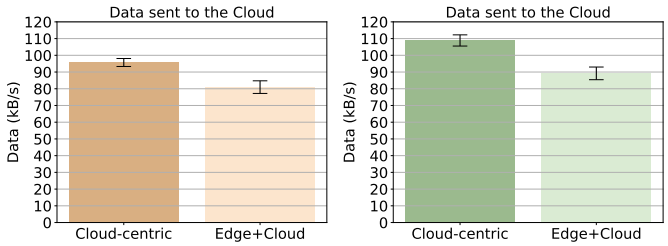
a-REQ 1. Execute experiments on heterogeneous computing resources. KheOps provides access to IoT/Edge devices and Cloud/HPC resources at large-scale, using the E2Clab methodology. Supported testbeds include (but are not limited to, as explained in Section 6.3): Grid'5000, FIT IoT LAB.

a-REQ 2. Systematically describe and explain the experimental processes and their reasoning. Through Jupyter notebooks and the E2Clab configuration files, the authors describe and explain the experiment design choices such as the layers (*e.g.,* Edge and Cloud), the services (*e.g.,* the Edge client and the Cloud server), the network constraints, and the application workflow execution. This is done in Jupyter notebooks by combining text (explaining the configurations) followed by executable code (E2Clab files).



(a) Grid'5000 + FIT IoT LAB. **(b) Grid'5000 + FIT IoT LAB.** **(c) Chameleon + CHI@Edge.** **(d) Chameleon + CHI@Edge.**

Figure 6: Cloud-centric vs Edge+Cloud processing: (a, b) executed by authors on Grid'5000 and FIT IoT LAB testbeds; and (c, d) replicated by readers on Chameleon Cloud and CHI@Edge testbeds.



(a) Grid'5000 + FIT IoT LAB. **(b) Chameleon + CHI@Edge.**

Figure 7: Amount of data sent to the Cloud regarding the Cloud-centric and Edge+Cloud processing approaches.

a-REQ 3. Efficiently configure the experimental infrastructure and repeat the experiments. All the complexities of configuring the Edge-to-Cloud infrastructure, such as leasing computing resources, mapping the application parts (*e.g.,* Edge and Cloud services), enforcing the network constraints, and executing the workflow are transparently handled by KheOps. The authors just need to define their experimental needs in the E2Clab configuration files. Repeating and adapting the experiments (*e.g.,* changing the network constraints) is easily done through E2Clab instrumentation.

a-REQ 4. Easily share the experiment artifacts in a public and safe repository. Through the Trovi and JupyterLab integration, authors can upload their artifacts to the Trovi sharing portal with a few clicks.

We discuss the experimental results from the authors perspective, using the three application performance metrics mentioned earlier.

5.2.1 ***Impact of the network on the processing time.*** Authors define two sets of experiments. In the first one (Figure 6a), they fix the network bandwidth at 15Kbit and vary the processing approach between Cloud-centric and Hybrid (Edge+Cloud). In the second one (Figure 6b), they fix the bandwidth at 25Kbit for both processing approaches.

From the results, the authors observe that the Hybrid (Edge+Cloud) approach outperforms the Cloud-centric one for both network configurations. In the 15Kbit bandwidth setup, the processing time for the Cloud-centric is about 27 seconds on average, against 24 seconds for the hybrid processing. In the 25Kbit bandwidth configuration, this difference is lower, 13 seconds and 11 seconds for the Cloud-centric and Hybrid, respectively. The higher the bandwidth, the lower will be the difference between the two processing approaches. This is because image transmission is the most time-consuming task among the other tasks (*i.e.,* compressing/decompressing images and model inference).

5.2.2 ***Amount of data sent to the Cloud.*** According to the results presented in Figure 7a, authors observe that the Hybrid (Edge+Cloud) approach transmits less data (81kB/s on average) to the Cloud compared to the Cloud-centric approach (96kB/s on average). This is because, in Hybrid processing, Edge devices compress images before transmitting them to the Cloud.

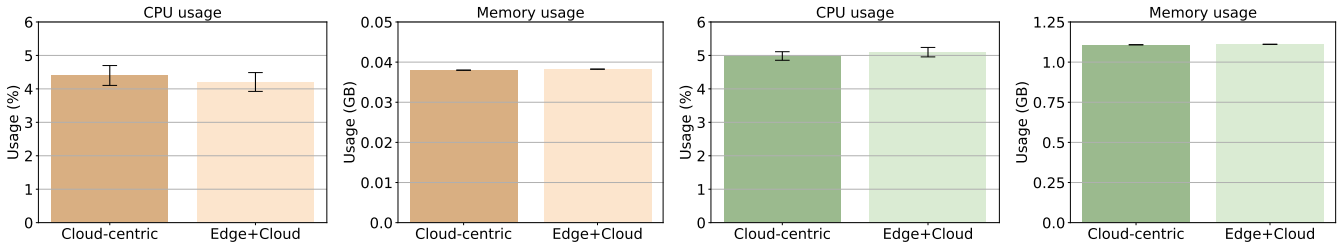
5.2.3 ***Resource consumption on the Edge device.*** Results in Figures 8a and 8b show that there is no significant difference in the CPU and memory usage in the Edge device when changing between the Cloud-centric and Hybrid processing approaches. CPU usage is around 4.2% and 4.4% for Hybrid and Cloud-centric processing, respectively. Memory usage is around 0.38GB for both.

5.3 How KheOps helps readers

After the authors publish their results, other researchers from a different lab download the article from a scientific database and decide to replicate the study on their own premises (*e.g.,* on a different testbed). Following the same logic, we present how KheOps helps the readers to replicate the experiments cost-effectively, that is, according to the readers requirements (r-REQ) in Section 1.

r-REQ 1. Find and access the experiment as simply as finding and reading the paper. Through the KheOps web interface (step 1 in Figure 4) the readers obtain access to all the public experiments shared by the community and available in Trovi. Then, they select the experiment shared by the authors of the article to get more details.

r-REQ 2. Perform the experiment, not just read about it. Next, in the experiment details web page, readers can launch a JupyterLab server with artifacts in just a single click (steps 2, 3, and 4 in Figure 4). Finally, following the experiment instructions described in the Jupyter notebook, the readers deploy and execute the experiments on their testbeds, such as (but not limited to): the Chameleon Cloud and CHI@Edge (step 5 in Figure 4).



(a) Grid'5000 + FIT IoT LAB. **(b) Grid'5000 + FIT IoT LAB.** **(c) Chameleon + CHI@Edge.** **(d) Chameleon + CHI@Edge.**

Figure 8: Resource consumption on the Edge device: CPU and Memory usage.

Table 3: Accuracy of replicated experiments.

Metric	Replicability accuracy	Experiment result
Processing time 15Kbit	0.943	Figure 6a and 6c
Processing time 25Kbit	0.882	Figure 6b and 6d
Data sent to the cloud	0.973	Figure 7a and 7b
CPU usage	0.978	Figure 8a and 8c
Memory usage	0.996	Figure 8b and 8d

***r-REQ 3. Experiment reasoning:* "What", "Why", *and* "How".** Before running the experiments, the readers can go through the Jupyter notebook to understand *What* the experiment does (*e.g.,* capture and compress images on Edge devices and then decompress the images and predict the animals on the Cloud server). The readers can also discover *Why* the authors set up the experiment with a 25kbit and 15kbit network bandwidth. Finally, KheOps allows to understand *How* the authors interconnect the Edge devices with the Cloud server (*e.g.,* assigning a public IP to the Cloud server, or opening firewall rules; using the MQTT protocol; among others).

r-REQ 4. Efficiently configure the experimental infrastructure. To achieve this, the readers just have to adapt the *layers_services* configuration file (presented in Listing 1) to the Chameleon Cloud and CHI@Edge testbeds. Configuring the network bandwidth to 25kbit and then changing it to 15kbit is as simple as changing the *rate* parameter in the *network* file (Listing 3). Finally, copying data to the Edge device, interconnecting it with the Cloud server, launching the application, and finally collecting the results is as simple as defining the *workflow* configuration file (Listing 4). The *network* and *workflow* configuration files are testbed agnostic, meaning that users do not need to update these files when changing the deployment from Grid'5000 + FIT IoT LAB to Chameleon + CHI@Edge.

Next, we report on the replicated experiments.

5.3.1 ***Impact of the network on the processing time.*** From the results in Figures 6c and 6d, readers conclude that the Hybrid (Edge+Cloud) processing approach outperforms the Cloud-centric one for both network configurations. This conclusion is consistent with the results observed in the published article.

Following the analysis, readers observe that in the 15Kbit bandwidth network configuration, the processing time for the Cloud-centric is about 8 seconds on average, against 6.5 seconds for the hybrid processing. In the 25Kbit bandwidth setup, this difference is lower, 5.5 seconds and 4 seconds for the Cloud-centric and Hybrid, respectively. Similarly to the authors results, readers also observe that the higher the bandwidth, the lower will be the difference between the two processing approaches.

Furthermore, as presented in Table 3, we highlight that readers obtained a replicability accuracy of 88.2% and 94.3% for 15Kbit and 25Kbit network configurations, respectively.

5.3.2 ***Amount of data sent to the Cloud.*** According to the results presented in Figure 7b, readers observe that the Hybrid approach transmits less data than the Cloud-centric. The former transmits around 89.2kB/s and the latter 108.8kB/s. Compressing images on the Edge helps to reduce the amount of data sent to the Cloud server. This conclusion is also consistent with the published article and presents a replicability accuracy of 97.3%.

5.3.3 ***Resource consumption on the Edge device.*** Results in Figures 8c and 8d show that there is no significant difference in the CPU and memory usage between the Cloud-centric and the Hybrid processing approaches. CPU usage is around 5.1% and 5% for Hybrid and Cloud-centric processing, respectively. Memory usage is around 1.1GB for both. We highlight that these conclusions are consistent with the published article and present a replicability accuracy of 97.8% and 99.6% for CPU and memory usage, respectively.

Despite readers observing a lower processing time compared to the authors, they could verify that their experiment conclusions are consistent with the original study, and their results present a high replicability accuracy (see Table 3).

This time difference is expected since readers used a more powerful Edge device (Raspberry Pi4 against Raspberry Pi3) for processing the most time-consuming task (*e.g.,* image compression and then transmission). The Raspberry Pi4 has more RAM memory (8GB *vs.* 1GB in Raspberry Pi3), a better CPU (1.5GHz *vs.* 1.2GHz), network (5GHz *vs.* 2.4GHz).

Furthermore, regarding the remaining metrics such as the amount of data sent to the Cloud and the CPU and memory usage on the Edge device, readers observe small differences when replicating the original study in different testbeds. This is due to the different deployment approaches used by each testbed, for instance, in FIT IoT LAB the Raspberry Pi 3 board runs an embedded Linux that is built with Yocto [29], while

CHI@Edge is based on Docker [14] containers. Despite that, the conclusions observed by authors and readers are the same and present high accuracies.

6 DISCUSSION

KheOps core elements (*i.e.,* Trovi, JupyterLab, E2Clab) exhibit several features that make it a promising environment for advancing Computing Continuum research through reproducible and replicable experiments. We briefly discuss them here.

6.1 Usability and reusability

KheOps targets **usability** by allowing users to easily find experiment artifacts shared in Trovi and then to launch experiments in a JupyterLab server in just a few clicks. KheOps abstracts all the low-level details of defining and configuring the experimental environment. It provides a high-level abstraction for mapping application parts with the Edge and Cloud infrastructures. Besides, the configuration files used to define the whole experimental environment are designed to be easy to use and understand.

KheOps also targets **reusability** of the experiment artifacts. For instance, readers of an article can reuse the authors artifacts to replicate the study or build upon the existing artifacts to generate new results. In addition, through E2Clab *User-Defined Services*, users can define their own services (*e.g.,* the Edge client and the Cloud server) with the desired deployment logic (*e.g.,* mapping the services to the physical machines/devices; installing required software and packages; *etc.*). Such services can be shared in this repository [15].

6.2 Analyzing other real-life applications

The KheOps approach is **generic** in terms of deployment and analysis of **other applications**. We highlight that, despite our evaluations focusing on the African savanna use-case, KheOps can be easily used in other contexts. Supporting new applications can be achieved by describing and implementing their logic in the *User-Defined Services* configuration file.

6.3 Integration with other scientific testbeds

The KheOps approach is **generic** with respect to the **deployment testbeds**. KheOps allows users to analyze application workflows on various large-scale scientific testbeds, beyond the four testbeds used in this work. The definition of the experimental environment through E2Clab configuration files (*e.g., layer_services.yaml, network.yaml*, and *workflow.yaml*) is tesbed agnostic, meaning that a deployment on the Grid'5000 testbed can be easily replicated in Chameleon (if the required computing resources are available).

6.4 Reproducibility and artifact availability

The experimental evaluations presented in this work follow a rigorous methodology [53] to support reproducible Edge-to-Cloud experiments on large-scale scientific testbeds. All the experiment artifacts are publicly available [16] at the Trovi sharing portal and the results are also publicly available [17] in our GitLab repository.

6.5 KheOps limitations

Next, we discuss future research under the KheOps approach to help with experiment reproducibility.

Provenance data capture. It may assist in the processes of reproducing complex Edge-to-Cloud workflows [47]. Typically, users have to execute and repeat various experiments. The output of this process generates hundreds of data related to the experimental setup (*e.g.,* hardware, software, code, data set, *etc.*) and application workflow execution. Analyzing such data is only possible with the help of provenance data capture [54].

Abstract hardware description. The hardware configuration is a significant barrier to reproducibility [27], especially in complex Edge-to-Cloud deployments comprising heterogeneous computing resources. The description of resources should be in terms of hardware requirements to execute the experiments (*e.g.,* CPU, GPU, memory, disk, and network). The goal is to abstract the hardware resource description among various testbeds, preventing independent researchers from knowing about the infrastructure of the original experimental environment.

7 RELATED WORK

We have not found in the literature related work proposing collaborative environments with a focus on the Edge-to-Cloud Continuum. Closer solutions to KheOps, but without focusing on the Computing Continuum, are Google Colab [18], Kaggle [21], and Code Ocean [37] as presented in Section 3.

KheOps differs from Google Colab, Kaggle, and Code Ocean mainly regarding the features presented in Table 2 (limitations) such as the access to heterogeneous Edge-to-Cloud resources; access to large-scale infrastructures; and supporting the experiment repeatability and reproducibility on the same hardware setup and replicability in different infrastructures. In addition, KheOps relies on open scientific testbeds (*e.g.,* Grid5000, FIT IoT LAB, Chameleon, and CHI@Edge) that are highly reconfigurable and controllable and designed to support reproducible experiments.

8 CONCLUSION

KheOps is, to the best of our knowledge, the first collaborative environment supporting the cost-effective reproducibility of applications on the Edge-to-Cloud Continuum. It provides simplified abstractions for systematically defining and explaining the experimental environment through Jupyter notebooks (*e.g.,* infrastructures, services, network, and workflow execution); provides access to heterogeneous computing resources from the IoT/Edge to the Cloud/HPC; and allows researchers to easily find and share the experiment artifacts in the Trovi portal.

The experimental validation shows that KheOps helps authors to make their experiments repeatable and reproducible on the Grid5000 and FIT IoT LAB testbeds. Furthermore, KheOps helps readers to cost-effectively replicate authors experiments in different infrastructures such as Chameleon Cloud + CHI@Edge testbeds, and obtain the same conclusions with accuracies >88% for all performance metrics.

ACKNOWLEDGMENTS

This work was funded by Inria through the HPC-BigData Inria Challenge (IPL) and through the UNIFY Associate Team joint in the framework of the JLESC international lab and the HPDaSc associate team with Brazil. It was co-funded by the French ANR OverFlow project (ANR-15- CE25-0003). Experiments presented in this paper were carried out using the Chameleon Cloud, CHI@Edge, Grid'5000, and FIT IoT LAB testbeds, supported by a scientific interest group hosted by several Universities. We also would like to thank Argonne National Laboratory for supporting this work. This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357 as well as by the NSF award 2130889 and NIFA award 2021-67021-33775.

REFERENCES

[1] [n. d.]. *Artifact Review and Badging Version 1.1.* https://www.acm.org/publications/policies/artifact-review-and-badging-current
[2] [n. d.]. *What is Docker Hub?* Retrieved Jun 1, 2023 from https://www.docker.com/products/docker-hub/
[3] 2018. *Dool (Dstat) monitoring.* Retrieved Jan 14, 2023 from https://github.com/scottchiefbaker/dool
[4] 2018. *GitHub.* Retrieved Jan 14, 2023 from https://github.com/
[5] 2018. *Zenodo.* Retrieved Jan 14, 2023 from https://zenodo.org/
[6] 2019. *E2Clab source code.* Retrieved Jan 14, 2023 from https://gitlab.inria.fr/E2Clab/e2clab
[7] 2023. *AI Hub.* Retrieved Jan 14, 2023 from https://aihub.cloud.google.com/
[8] 2023. *Apache Zeppelin.* Retrieved Jan 15, 2023 from https://zeppelin.apache.org/
[9] 2023. *Code Ocean Explore: Open Science Library.* Retrieved Jan 19, 2023 from https://codeocean.com/explore
[10] 2023. *Colab: Cloud Storage from the command line.* Retrieved Jan 18, 2023 from https://cloud.google.com/storage/docs/gsutil
[11] 2023. *Colab: Google Spreadsheets.* Retrieved Jan 18, 2023 from https://github.com/burnash/gspread#more-examples
[12] 2023. *Compute skylake cluster at CHI@TACC.* Retrieved Feb 16, 2023 from https://www.chameleoncloud.org/hardware/node/sites/tacc/clusters/chameleon/nodes/0b0bceb9-14bf-423e-890f-3ef187511d71/
[13] 2023. *Dahu cluster.* Retrieved Feb 16, 2023 from https://www.grid5000.fr/w/Grenoble:Hardware#dahu
[14] 2023. *Docker.* Retrieved Jan 18, 2023 from https://www.docker.com/
[15] 2023. *E2Clab User Defined Services.* Retrieved Feb 8, 2023 from https://gitlab.inria.fr/E2Clab/user-defined-services
[16] 2023. *Experiment artifacts.* Retrieved Feb 8, 2023 from https://www.chameleoncloud.org/experiment/share/347adbf3-7c14-4834-b802-b45fdd0d9564
[17] 2023. *Experiment results.* Retrieved Jan 14, 2023 from https://gitlab.inria.fr/E2Clab/Paper-Artifacts
[18] 2023. *Google Colab.* Retrieved Jan 17, 2023 from https://colab.research.google.com/
[19] 2023. *Google Colab: Frequently Asked Questions.* Retrieved Jan 18, 2023 from https://research.google.com/colaboratory/faq.html
[20] 2023. *Google Colab vs Kaggle.* Retrieved Jan 20, 2023 from https://datasciencenotebook.org/compare/colab/kaggle
[21] 2023. *Kaggle community.* Retrieved Jan 19, 2023 from https://www.kaggle.com/
[22] 2023. *Kaggle datasets.* Retrieved Jan 20, 2023 from https://www.kaggle.com/datasets
[23] 2023. *MQTT: The Standard for IoT Messaging.* Retrieved Feb 16, 2023 from https://mqtt.org/
[24] 2023. *Python zlib.* Retrieved Feb 16, 2023 from https://docs.python.org/3/library/zlib.html
[25] 2023. *Raspberry Pi 3 Model B.* Retrieved Feb 16, 2023 from https://www.iot-lab.info/docs/boards/raspberry-pi-3/
[26] 2023. *Raspberry Pi 4.* Retrieved Feb 16, 2023 from https://chameleoncloud.org/experiment/chiedge/hardware-info/
[27] 2023. *SC: The largest Reproducibility Laboratory.* Retrieved Feb 8, 2023 from https://www.chameleoncloud.org/blog/2023/02/20/sc-the-largest-reproducibility-laboratory/
[28] 2023. *Trovi: Practical Open Reproducibility.* Retrieved Jan 20, 2023 from https://chameleoncloud.gitbook.io/trovi/
[29] 2023. *Yocto Project.* Retrieved Jan 14, 2023 from https://www.yoctoproject.org/
[30] 2023. *Zooniverse dataset.* Retrieved Feb 16, 2023 from https://www.zooniverse.org/organizations/meredithspalmer/snapshot-safari
[31] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 265–283.
[32] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele, et al. 2015. FIT IoT-LAB: A large scale open experimental IoT testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 459–464.
[33] Jason Anderson and Kate Keahey. 2019. A case for integrating experimental containers with notebooks. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 151–158.
[34] L. A. Barba and G. K. Thiruvathukal. 2017. Reproducible Research for Computing in Science Engineering. *Computing in Science Engineering* 19, 6 (2017), 85–87.
[35] Raphaël Bolze, Franck Cappello, Eddy Caron, Michel Dayde, Frédéric Desprez, Emmanuel Jeannot, Yvon Jégou, Stephane Lanteri, Julien Leduc, Nouredine Melab, Guillaume Mornet, Raymond Namyst, Pascale Primet, Benjamin Quétier, Olivier Richard, El-Ghazali Talbi, and Iréa Touche. 2006. Grid'5000: A Large Scale And Highly Reconfigurable Experimental Grid Testbed. *International Journal of High Performance Computing Applications* 20, 4 (2006), 481–494. https://doi.org/10.1177/1094342006070078
[36] Ronan-Alexandre Cherrueau, Marie Delavergne, Alexandre Van Kempen, Adrien Lebre, Dimitri Pertin, Javier Rojas Balderrama, Anthony Simonet, and Matthieu Simonin. 2021. Enoslib: A library for experiment-driven research in distributed computing. *IEEE Transactions on Parallel and Distributed Systems* 33, 6 (2021), 1464–1477.
[37] April Clyburne-Sherin, Xu Fei, and Seth Ariel Green. 2019. Computational reproducibility via containers in psychology. *Meta-psychology* 3 (2019).
[38] Geoff Cumming, Fiona Fidler, and David L Vaux. 2007. Error bars in experimental biology. *The Journal of cell biology* 177, 1 (2007), 7–11.
[39] ETP4HPC. April 29, 2020. ETP4HPC Strategic Research Agenda. https://www.etp4hpc.eu/sra.html.
[40] Odd Erik Gundersen, Yolanda Gil, and David W Aha. 2018. On reproducible AI: Towards reproducible research, open science, and digital scholarship in AI publications. *AI magazine* 39, 3 (2018), 56–68.
[41] Benjamin Haibe-Kains, George Alexandru Adam, Ahmed Hosny, Farnoosh Khodakarami, Massive Analysis Quality Control (MAQC) Society Board of Directors Shraddha Thakkar 35 Kusko Rebecca 36 Sansone Susanna-Assunta 37 Tong Weida 35 Wolfinger Russ D. 38 Mason Christopher E. 39 Jones Wendell 40 Dopazo Joaquin 41 Furlanello Cesare 42, Levi Waldron, Bo Wang, Chris McIntosh, Anna Goldenberg, Anshul Kundaje, et al. 2020. Transparency and reproducibility in artificial intelligence. *Nature* 586, 7829 (2020), E14–E16.
[42] Kate Keahey. 2020. The Silver Lining. *IEEE Internet Computing* 24, 4 (2020), 55–59.
[43] Kate Keahey, Jason Anderson, Michael Sherman, Zhuo Zhen, Mark Powers, Isabel Brunkan, and Adam Cooper. 2021. Chameleon@Edge Community Workshop Report.
[44] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S Gunawi, Cody Hammock, et al. 2020. Lessons learned from the chameleon testbed. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. 219–233.
[45] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. 2016. *Jupyter Notebooks-a publishing format for reproducible computational workflows.* Vol. 2016.
[46] Matthew S Krafczyk, A Shi, Adhithya Bhaskar, D Marinov, and Victoria Stodden. 2021. Learning from reproducing computational results: introducing three principles and the Reproduction Package. *Philosophical Transactions of the Royal Society A* 379, 2197 (2021), 20200069.
[47] Ling Liu and M Tamer Özsu. 2009. *Encyclopedia of database systems.* Vol. 6. Springer.
[48] Engineering National Academies of Sciences, Medicine, et al. 2019. *Reproducibility and replicability in science.* National Academies Press.
[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.
[50] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.
[51] Daniel Rosendo, Alexandru Costan, Gabriel Antoniu, Matthieu Simonin, Jean-Christophe Lombardo, Alexis Joly, and Patrick Valduriez. 2021. Reproducible Performance Optimization of Complex Applications on the Edge-to-Cloud Continuum. In *Cluster 2021 - IEEE International Conference on Cluster Computing*. Portland, OR, United States, 23–34. https://doi.org/10.1109/Cluster48925.2021.00043
[52] Daniel Rosendo, Alexandru Costan, Patrick Valduriez, and Gabriel Antoniu. 2022. Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review. *Journal of Parallel and Distributed Computing* 166 (Aug. 2022), 71–94.

https://doi.org/10.1016/j.jpdc.2022.04.004

[53] Daniel Rosendo, Pedro Silva, Matthieu Simonin, Alexandru Costan, and Gabriel Antoniu. 2020. E2Clab: Exploring the Computing Continuum through Repeatable, Replicable and Reproducible Edge-to-Cloud Experiments. In *Cluster 2020 - IEEE International Conference on Cluster Computing*. Kobe, Japan, 1–11. https://doi.org/10.1109/CLUSTER49012.2020.00028

[54] Renan Souza, Vítor Silva, Jose J. Camata, Alvaro L. G. A. Coutinho, Patrick Valduriez, and Marta Mattoso. 2019. Keeping Track of User Steering Actions in Dynamic Workflows. *Future Generation Computer Systems* 99 (2019), 624–643. https://doi.org/10.1016/j.future.2019.05.011

[55] Victoria Stodden, Marcia McNutt, David H Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, Michael A Heroux, John PA Ioannidis, and Michela Taufer. 2016. Enhancing reproducibility for computational methods. *Science* 354, 6317 (2016), 1240–1241.

[56] Victoria Stodden and Sheila Miguez. 2014. Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research. *Journal of Open Research Software* (Jul 2014). https://openresearchsoftware.metajnl.com/articles/10.5334/jors.ay

[57] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* 3, 1 (2016), 1–9.